Artificial Intelligence and Machine Learning in Financial Modeling

Ren-Raw Chen, Ph.D.

Chen, Ren-Raw Artificial Intelligence and Machine Learning in Asset Pricing / Ren-Raw Chen ISBN

Contents

A Few Words

It is quite an interesting journey for this book. It all began in a thanksgiving dinner in 2017 when my kids (and kid-in-laws) came home for the holiday...

Before then, I had heard about terms like machine learning and I certainly had known (I thought) about artificial intelligence (of course, which turned out to be a very incorrect perception of AI, more to say about that later). And yet, the information about ML I had received was mainly from my equally ignorant colleagues, so you can imagine I had been completely misled. The only thing that had grabbed my attention at the time was that my skills (which is quant finance and I had written a book Mathematical Finance about it) would have become outdated.

As anyone would have reacted, denial was my immediate reaction. I was trying to find all sorts of excuses (only in my mind) to refute such a possibility. Now, looking back, I am quite embarrassed to realize that I had been going through the five stages of grief – denial, anger, bargaining, depression and acceptance. Except that I would add a sixth state: *embracing*.

So, let's get back to the dinner of 2017 thanksgiving. My son had been just two years into his job. He graduated in 2015 with a computer science degree from UPenn. He was all excited about his job and the work he was doing and the challenges he was facing at work. As any father, I was very happy for (and with) him and would have loved to hear his story. Of course, as any father, I had no idea what he was saying (about AI and ML) – until " *...maximum likelihood...*". I stopped him right away! "Wait... maximum likelihood?" "What does maximum likelihood has anything to do with ML?" Maximum likelihood is something I know so well. Isn't it a standard econometric tool? Are we talking about the same thing?

Now you know, it is the exact same maximum likelihood in both ML and econometrics. My son was focusing on probabilistic modeling in ML so he had used Monte Carlo simulations a lot, which is also something I am totally familiar with. From then on, what he had said made total sense to me...

As we all know, knowing how to correctly and cleverly search on Google is valuable knowledge. Now we have a term for it, called *prompt engineering*. After

that dinner, I then was able to gather so much correct information about machine learning and was also able quickly link what I know with this new technology.

After a few days of "prompt engineering", I had quickly and pleasantly discovered machine learning is nothing more than applied statistics. The history of these two branches of probability theory is also quite interesting. Apparently when probability theory had become quite well-developed, applied fields had started to develop. Two very different approaches had emerged and they had not seemed to agree with each other. One approach had been data-driven (known as frequencists) and as you can imagine this approach had grown into machine learning. The other approach (known as statisticians) had continued the more mathematical route to derive various applied theories in statistics.

Because of that, the same exact techniques had been named differently. For example, learning = estimation; prediction = out-of-sample test;

This book is very narrowly focused upon AI in financial modeling, or more specifically in asset pricing. I had taught continuous time finance or advance financial theory [1]. (both in the ph.d. programs and the master programs in quant finance) for a very long time (till recently). In advanced finance theory which is based upon the Black-Scholes model, the pricing of assets is based upon the Q (risk-neutral) measure. Prior to the Black-Scholes, economists had used the P (physical) measure with risk-adjusted discounting. Now in AI, we go back to the P measure.

[1] Introductory financial theory is often referred to discrete time finance.

Chapter 1

Introduction

Big data (ML, AI) is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it

Dan Ariely, Duke University

1.1 What Is AI? What is ML? What is BD?

They are all called AI now, which is probably the best, since it is really hard to distinguish them. But are they really the same? How are they different? In what way are they different? If you ask 10 different people, you probably get 12 different answers. Here is how I think about them.

1.2 Artificial Intelligence versus Natural Intelligence

From its name, it is not hard to think that any artificial intelligence should stem from a natural (biological) intelligence. Models that are not coming from natural intelligence (i.e. those that are purely mathematical like graph theory and purely statistical like Bayesian analysis) shouldn't be qualified as AI models.

By this narrow definition, I only allow four areas of models to be qualified as AI models:

- reinforcement learning
- swarm intelligence
- neural networks
- genetic algorithms

This is because each of the above items has a natural intelligence that AI copies from. Reinforcement learning is about mouse passing mazes. Swarm intelligence is about group behaviors from bees, ants, birds, and fish. Neural networks copy from neurons in brains. Finally, genetic algorithms follow biological evolutions.

This often reminds me of Chinese martial arts. If you like Jackie Chan, you should remember one of his famous early movies Drunken Master where he invents his moves from monkeys.[1] There are numerous examples in Chinese martial arts where those kung-fu moves are insipred from animals. Some probably know Shaolin monks in China to be famous for their tiger-crane kung-fu which obviously is originated from tigers and cranes.

Today, learning from animals has moved to a digital level. Computer algorithms are created to mimic how animals behave, or even more fundamentally how biology works. It is truly to my absolute amazement that simple algorithms can do what natural intelligence can do. It is worth noting that the above mentioned AI models are so easy (only a few lines of code) and yet so powerful.

As you can see, the above mentioned AI models are behavioral models. That is, they mimic how biology works. From this perspective, they are not at all similar to machine learning models which are mostly used to solve optimization problems. That said, we can use AI models to solve optimization problems as well. For example, PSO, or particle swarm optimization, is used for finding an optimum of a hyper-plane. Reinforcement learning can be used to find the shortest path. Not to mention that various neural networks are discovered to solve very complex optimization problems, including the ChatGPT. Hopefully this book can demonstrate some of the amazing works AI models have done. I hope you enjoy as much as I do in the world of artificial intelligence.

1.3 Machine Learning versus Statistics

There is very little difference between machine learning and applied statistics. Both can be traced back to the same parent in the nineteenth century when the probability

[1] https://en.wikipedia.org/wiki/Drunken Monkey

theory was discovered. Since then, two branches parted their ways. One branch continued the theoretical development and later became what we know as statistics today. The other branch moved towards a more practical direction and focused more on data (also known as frequentists), which later combined with computer technology and became what we know as machine learning today.[2]

We can almost find one-to-one terminology translation between statistics and machine learning. The term "learning" is "estimation" in statistics. The term "prediction" is "out-of-sample testing" in statistics. The term "supervised learning" corresponds to "parametric modeling" (and "unsupervised learning" refers to "non-parametric modeling". The last similarity deserves a special attention. In statistics, we know that when a non-parametric model is used, it is extremely difficult to have significant estimates (i.e. the null hypothesis is extremely hard to reject). This is because the type-I error is big. Furthermore, we also know that 1 minus the type-II error measures the power of the model. With a large amount of data, we can reduce the type-II error and hence increase the power of the model. This is main reason why machine learning (unsupervised) requires big data.

Not only did terminologies start to diverge as the two fields went on their own ways to evolve, focuses of the same theories also started to differ. Traditional statisticians continued to emphasize analytical solutions as they, following the theoretical path of probability theory, value mathematical elegance. Machine learning scholars, on the other hand, turned their focus on efficient algorithms. The natural consequence of such a divergence is that machine learning can focus on non-parametric models while statistics continues to focus on parametric models.

As we now witness, as computing powers grow and costs shrink exponentially, the machine learning branch seems to win the current battle. This has already started decades ago in finance. For example, in the term structure (of interest rates) literature, solving differential equations numerically has been very costly and also quite intimidating to those who could not program. Hence in the early days (probably till early 2000s), closed-form models such as the Vasicek model dominated the interest rate world. And yet, gradually various expansions of the Vasicek model have emerged (most notably the Duffie-Kan model in 1996). These models require numerical solutions to differential equations but they can be solved just as fast as closed-form solutions.

[2]https://en.wikipedia.org/wiki/Frequentist probability

1.4 Big Data versus Small Data

In Finance, there hasn't really been any use of big data, not by today's standards anyway. Also in finance, there has been no use of textual data (let alone image or video data). Basically, there are only numeral data and a large portion of the literature use highly infrequent data (such as monthly or quarterly, or even annually).

So why AI/ML models matter?

For one thing, finance can start adopting (already happening) textual, image, and video data. Such data can in a great way remedy the shortcomings of infrequent data such as financial statements (quarterly) or macro economic data such as GDP (annual).

However, it requires skills to process non-numeral data. Researchers must be very familiar with natural language processing (NLP). Unfortunately, such skills are not easy to learn. Not only do researchers need to be familiar with languages like Python or R, they must also know the basics of NLP such as tokenization, embedding, and packages like `word2vec`, as a few examples.

Nevertheless, this is similar to in the 80s when finance were transforming from only time value of money and annuity calculation to stochastic calculus and differential equations. At the time, only a very few could adapt to the change. But those who could have later become the leaders in the field. I remember when I was in grad school ($1985 \sim 1990$), I needed to take math classes in the math department with those math graduate students. Gradually and inevitably, such courses began to be offered in the business school, partly because math professors understood by then such math is essential in researching in the field of finance.

We are facing the exact same situation here. Taking NLP courses in the computer science department is not only inefficient, but unnecessary. Such knowledge needs to be introduced in the business school with business applications. Big data knowledge is so important now that it must be made accessible to everybody, not just computer scientists and engineers. I guess this is what "prompt engineering" is all about.

1.5 A Few Words of My Own

As I mentioned above, when a new piece of technology is introduced to finance, there is always some resistance and yet inevitably those who cannot adapt will disappear and a new generation of scholars will enter and shine. I must admit that when I first heard of machine learning, it was accompanied by comments like "black box", "make

no sense". So I was accepting this notion and in "denial" . When I forced myself to start to understand it, it is very frustrating (similar to "bargaining/depression"?), mainly in that information is either inaccurate or irrelevant. But after a very long and flat learning curve, I began to "accept" it. Now I am really enjoying it and fully "embrace" it. I wish the audience of this book enjoys it as much as I do.

1.6 Appendix

1.6.1 Type-I and Type-II Errors

We say reject the null, not accept the null.

5% significance (t > 2) means we can reject the null with 95% confidence. So, 5% is the type-I error. If we cannot reject the null, we cannot say that we accept the null because we need to then look at the type-II error.

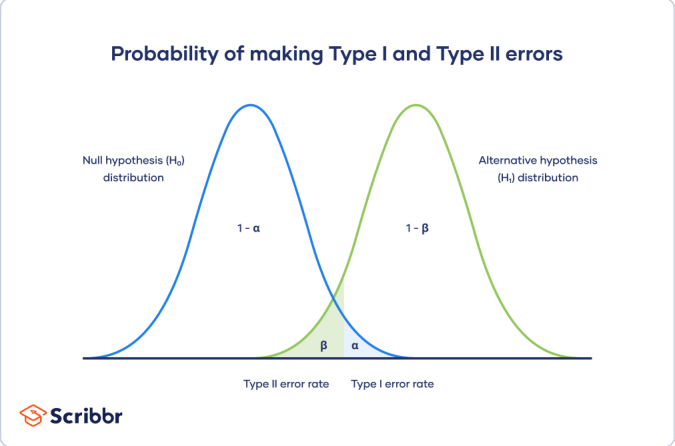


Figure 1.1: Type-I and Type-II Errors

As we can see from the figure, if the null is true, then we only have very few samples from the α area. Hence, if there are enough samples there, then we can "confidently" (at 5% level) reject the null. On the contrary, if we have not enough samples in the α area, but still quite far from the center of the null distribution, we cannot say the null is true. That is the reason why we cannot accept the null.

However, if the alternative is true, then we will have a lot of samples falling in the α area.

If type-II error is small, then we can accept the null confidently. This, $1 - \beta$, is known as the power of the model.

Chapter 2

Bayesian Inference

Thomas Bayes was an English statistician, philosopher and Presbyterian minister who is known for formulating a specific case of the theorem that bears his name: Bayes' theorem. Bayes never published what would become his most famous accomplishment; his notes were edited and published posthumously by Richard Price

Wikipedia

2.1 Introduction

Bayesian inference introduced in this chapter is also useful in many other AI/ML topics in later chapters.

2.2 Bayes Rule

Given a parameter θ and a feature (which is known as explanatory or independent variable in finance) x, define the following probabilities:

- prior $p(\theta)$ where θ is a vector of parameters and x is a vector of random variables $x_1 \cdots x_n$
- posterior $p(\theta|x)$
- likelihood is to use data to estimate parameters (joint distribution of parameters)

- marginal distribution (data) $p(x)$

When there is a set of parameters, we write $\underline{\theta}_{M\times 1}$ (underlined) as a vector, and also $\underline{x}_{K\times 1}$. The data collected for $\underline{x}$ is represented as $X_{N\times K}$ (capital).

Our objective is to infer θ given the data we collect. That is, we are interested in the posterior distribution. The simple Bayesian rule can be applied:

$$p(\theta|x) = \frac{p(\theta, x)}{p(x)} = \frac{p(x|\theta)p(\theta)}{p(x)} \propto p(x|\theta)p(\theta) \tag{2.1}$$

The final result is obtained because $p(x)$ is just data which are not related to θ and hence can be treated as a constant.

2.3 Maximum Likelihood Estimation of a Bayesian model

2.3.1 Bernulli-Beta Bayesian Analysis – Get a Flavor!

The data generating function is a Bernulli: $p(x|\theta) = \theta^x(1-\theta)^{1-x}$. Hence the likelihood function is

$$\begin{aligned} \mathscr{L} = p(x_1 \cdots x_n|\theta) &= \textstyle\prod_{i=1}^{n} p(x_i|\theta) \\ &= \textstyle\prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} \\ &= \theta^{\sum x_i}(1-\theta)^{n-\sum x_i} \end{aligned} \tag{2.2}$$

where $x_i = 1$ when the event is true and 0 if it is false.

Let us first do an MLE assuming θ is constant.

$$\begin{aligned} &\ln \mathscr{L} = \textstyle\sum x_i \ln\theta + \left(n - \sum x_i\right)\ln(1-\theta) \\ &\frac{\partial}{\partial\theta}\ln\mathscr{L} = \textstyle\sum x_i \frac{1}{\theta} - \left(n - \sum x_i\right)\frac{1}{1-\theta} = 0 \\ &\frac{\sum x_i}{\theta} = \frac{n - \sum x_i}{1-\theta} \\ &(1-\theta)\textstyle\sum x_i = \theta\left(n - \sum x_i\right) \\ &\hat{\theta} = \frac{\sum x_i}{n} \end{aligned} \tag{2.3}$$

A dangerous prediction: If we see 3 consecutive white swans $x_1 = 0, x_2 = 0, x_3 = 0$, then what is the probability θ that the fourth swan is black? Since $\hat{\theta} = \frac{\sum x_i}{n}$, next swan has 0% probability of appearing.

How does Bayesian inference help?

Let the prior be beta:

$$p(\theta) = \mathrm{B}(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\theta^{a-1}(1-\theta)^{b-1} \tag{2.4}$$

where its mean is $\frac{a}{a+b}$, its mode is $\frac{a-1}{a+b-2}$, and its variance is $\frac{ab}{(a+b+1)(a+b)^2}$.

Now we derive the posterior:

$$\begin{aligned} &p(\theta|x) \propto p(x|\theta)p(\theta) \\ &= \left(\theta^{\sum x_i}(1-\theta)^{n-\sum x_i}\right)\left(\tfrac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\theta^{a-1}(1-\theta)^{b-1}\right) \\ &= k\tfrac{\Gamma(a^*)\Gamma(b^*)}{\Gamma(a^*+b^*)}\theta^{\left(a+\sum x_i\right)-1}(1-\theta)^{\left(b+n-\sum x_i\right)-1} \\ &\propto \mathrm{B}(a^*, b^*) \end{aligned} \tag{2.5}$$

which is proportional to a beta distribution with $a^* = a+\sum x_i$ and $b^* = b+n-\sum x_i$.

Now the mean is

$$\frac{a^*}{a^*+b^*} = \frac{a+\sum x_i}{a+b+n}$$

As we can see now, that 0's from the first three draws will not give the prediction of the next draw 0. Rather, it will give the prior mean. In this case, it is $\frac{a}{a+b+n}$.

Let $a = b = 1$. Then the mean estimate is $\frac{1+\sum x_i}{n+2}$. Sampling 3 consecutive white swans will then predict the next black swan probability is $\frac{1}{5} = 0.2$. As we can see, continually having white swans will slowly reduce the probability of predicting a black swan: $\frac{1}{6}, \frac{1}{7}, \cdots$, until 0.

There is cute way to re-express the above result:

$$\begin{aligned} \mathbb{E}[p(\theta|x)] &= \frac{a+\sum x_i}{a+b+n} \\ &= w\frac{a}{a+b} + (1-w)\frac{\sum x_i}{n} \end{aligned} \tag{2.6}$$

where $w = \frac{a+b}{a+b+n}$ is a weighted average of prior mean and likelihood mean.

2.3.2 Point Estimation

The posterior $p(\theta|X)$ is our belief state. To convert it to a single best guess (point estimate), we pick the value that minimizes some loss function, e.g.,

$$\hat{\theta} = \arg\min_{\theta'} \int L(\theta', \theta) p(\theta|X) d\theta \tag{2.7}$$

† $L(\theta', \theta) = (\theta' - \theta)^2$

$$\begin{aligned} &\arg\min_{\theta'} \int L(\theta', \theta) p(\theta|X) d\theta = \arg\min_{\theta'} \int (\theta' - \theta)^2 p(\theta|X) d\theta \\ &\frac{\partial}{\partial \theta'} = \int 2(\theta' - \theta) p(\theta|X) d\theta = 0 \\ &\theta' = \int \theta p(\theta|X) d\theta = \mathbb{E}[\theta|X] \end{aligned} \tag{2.8}$$

which is minimized when $\theta' = \mathbb{E}[\theta|X]$.

† $L(\theta', \theta) = \begin{cases} 1 & \theta = \theta' \\ 0 & \theta \neq \theta' \end{cases}$

or another way to writing it (more rigorous) $L(\theta', \theta) = \delta(\theta' \to \theta)$ where δ is the Dirac delta function.

$$\begin{aligned} &\arg\max_{\theta'} \int L(\theta', \theta) p(\theta|X) d\theta = \arg\max_{\theta'} \int \delta(\theta \to \theta') p(\theta|X) d\theta \\ &= \arg\max_{\theta'} \mathbb{E}[\delta(\theta \to \theta')|X] \\ &= \arg\max_{\theta'} p(\theta|X) \end{aligned} \tag{2.9}$$

This is the definition of the mode.

† $L(\theta', \theta) = |\theta' - \theta|$

$$\begin{aligned} &\arg\min_{\theta'} \int L(\theta', \theta) p(\theta|X) d\theta = \arg\min_{\theta'} \int |\theta' - \theta| p(\theta|X) d\theta \\ &= \arg\min_{\theta'} \left\{ \int_{\theta' > \theta} (\theta' - \theta) p(\theta|X) d\theta + \int_{\theta' \leqslant \theta} (\theta - \theta') p(\theta|X) d\theta \right\} \end{aligned} \tag{2.10}$$

Differentiate w.r.t. θ' and set it to 0:

$$\begin{aligned} &\int_{-\infty}^{\theta'} p(\theta|X) d\theta - \int_{\theta'}^{\infty} p(\theta|X) d\theta = 0 \\ &\int_{-\infty}^{\theta'} p(\theta|X) d\theta = \int_{\theta'}^{\infty} p(\theta|X) d\theta = \frac{1}{2} \end{aligned} \tag{2.11}$$

2.3.3 Gaussian Case

Under (multi-variate) normality, we have:

$$p(\underline{r}|\underline{\theta}) \ = (2\pi)^{-n/2} \left|\Omega\right|^{-1/2} \exp\left(\frac{-1}{2}(\underline{r}-\underline{\theta})'\Omega^{-1}(\underline{r}-\underline{\theta})\right) \tag{2.12}$$

and

$$p(\underline{\theta}) = (2\pi)^{-n/2} \left|\Sigma\right|^{-1/2} \exp\left(\frac{-1}{2}(\underline{\theta}-\underline{\mu})'\Sigma^{-1}(\underline{\theta}-\underline{\mu})\right) \tag{2.13}$$

and then the posterior:

$$\begin{aligned} p(\underline{\theta}|\underline{r}) \propto & p(\underline{r}|\underline{\theta})p(\underline{\theta}) \\ = & (2\pi)^{-n/2} \left|\Omega\right|^{-1/2} + (2\pi)^{-n/2} \left|\Sigma\right|^{-1/2} \\ & \exp\left(\frac{-1}{2}(\underline{r}-\underline{\theta})'\Omega^{-1}(\underline{r}-\underline{\theta})\right) \exp\left(\frac{-1}{2}(\underline{\theta}-\underline{\mu})'\Sigma^{-1}(\underline{\theta}-\underline{\mu})\right) \end{aligned} \tag{2.14}$$

Taking log:

$$\begin{aligned} \ln p(\underline{\theta}|\underline{r}) &= -\frac{1}{2}(\underline{r}-\underline{\theta})'\Omega^{-1}(\underline{r}-\underline{\theta}) - \frac{1}{2}(\underline{\theta}-\underline{\mu})'\Sigma^{-1}(\underline{\theta}-\underline{\mu}) + c_1 \\ &= -\frac{1}{2}\underline{\theta}'\Omega^{-1}\underline{\theta} + \underline{\theta}'\Omega^{-1}\underline{r} - \frac{1}{2}\underline{\theta}'\Sigma^{-1}\underline{\theta} + \underline{\theta}'\Sigma^{-1}\underline{\mu} + c_2 \\ &= -\frac{1}{2}\underline{\theta}'\left(\Omega^{-1}+\Sigma^{-1}\right)\underline{\theta} + \underline{\theta}'\left(\Omega^{-1}\underline{r}+\Sigma^{-1}\underline{\mu}\right) + c_2 \\ &= -\frac{1}{2}\underline{x}'\left(\Omega^{-1}+\Sigma^{-1}\right)\underline{x} + c_3 \end{aligned} \tag{2.15}$$

where $\underline{x} = \underline{\theta} - (\Omega^{-1}+\Sigma^{-1})^{-1}\left(\Sigma^{-1}\underline{\mu}+\Omega^{-1}\underline{r}\right)$ and (see Appendix)

$c_1 = \ln\left(\frac{1}{\{2\pi\}^{n/2}|\Omega|^{1/2}} + \frac{1}{\{2\pi\}^{n/2}|\Sigma|^{1/2}}\right)$

$c_2 = c_1 - \frac{1}{2}\left\{\underline{r}'\Omega^{-1}\underline{r} + \underline{\mu}'\Sigma^{-1}\underline{\mu}\right\}$

$c_3 = c_2 + \left\{\left(\Omega^{-1}+\Sigma^{-1}\right)^{-1}\left(\Sigma^{-1}\underline{\mu}+\Omega^{-1}\underline{r}\right)\right\}'\left(\Omega^{-1}+\Sigma^{-1}\right)\left\{\left(\Omega^{-1}+\Sigma^{-1}\right)^{-1}\left(\Sigma^{-1}\underline{\mu}+\Omega^{-1}\underline{r}\right)\right\}$

Since c_3 is a constant (unrelated to θ), we can drop it and simply write equation (2.15) as:

$$\ln p(\underline{\theta}|\underline{r}) \propto -\frac{1}{2}\underline{x}'\left(\Omega^{-1}+\Sigma^{-1}\right)\underline{x}$$

which is the log density of a Gaussian:

$$\underline{\theta}|\underline{r} \sim N\left(\left(\Omega^{-1}+\Sigma^{-1}\right)^{-1}\left(\Sigma^{-1}\underline{\mu}+\Omega^{-1}\underline{r}\right), \left(\Omega^{-1}+\Sigma^{-1}\right)^{-1}\right) \tag{2.16}$$

2.3.4 With Data

Maximum likelihood function:

$$p(\underline{r}|\underline{\theta}) = \prod_{t=1}^{T} (2\pi)^{-n/2} |\Omega|^{-1/2} \exp\left(\frac{-1}{2}(\underline{r}_t - \underline{\theta})'\Omega^{-1}(\underline{r}_t - \underline{\theta})\right) \tag{2.17}$$

and (applying the prior of equation (**??**))

$$\ln p(\underline{\theta}|\underline{r}) \propto \frac{-1}{2} \sum_{t=1}^{T} (\underline{r}_t - \underline{\theta})'\Omega^{-1}(\underline{r}_t - \underline{\theta}) - \frac{1}{2}(\underline{\theta} - \underline{\mu})'\Sigma^{-1}(\underline{\theta} - \underline{\mu}) \tag{2.18}$$

Partially differentiate the log posterior function with respect to $\underline{\theta}$ and get:

$$\frac{\partial}{\partial \underline{\theta}} \ln p(\underline{\theta}|\underline{r}) \propto \sum_{t=1}^{T} \Omega^{-1}(\underline{r}_t - \underline{\theta}) - \Sigma^{-1}(\underline{\theta} - \underline{\mu}) = 0$$

and solve for $\underline{\theta}$:

$$\begin{aligned} &\sum_{t=1}^{T} \Omega^{-1}(\underline{r}_t - \underline{\theta}) - \Sigma^{-1}(\underline{\theta} - \underline{\mu}) = 0 \\ &\sum_{t=1}^{T} \Omega^{-1}\underline{r}_t + \Sigma^{-1}(\underline{\theta} - \underline{\mu}) = (T\Omega^{-1} + \Sigma^{-1})\underline{\theta} \end{aligned} \tag{2.19}$$

Then

$$\hat{\underline{\theta}} = \left(T\Omega^{-1} + \Sigma^{-1}\right)^{-1} \left(\sum_{t=1}^{T} \Omega^{-1}\underline{r}_t + \Sigma^{-1}\underline{\mu}\right) \tag{2.20}$$

where $\underline{\mu}$ is taken as the unconditional (historical average) mean from the data and Σ is taken as the unconditional variance-covariance matrix from data, given that Ω could be set to be equal to the identity matrix.

The standard error of the Bayesian estimates $\hat{\underline{\theta}}$ is the inverse of the information matrix which can be shown to be

$$\left(T\Omega^{-1} + \Sigma^{-1}\right)^{-1}$$

2.4 MLE for the Black-Litterman Model

Now, we add a view. Let P be the matrix of views (absolute or relative). Without rigor, Black and Litterman (1991, 1992) replace $\underline{r}$ by $P\underline{r}$ but with the same variance-covariance . Then equation (2.14) becomes:

$$\begin{aligned} f(P\underline{r}|\underline{\theta}) &\propto \exp\left\{-\frac{1}{2}[P\underline{r} - P\underline{\theta})]'\Omega^{-1}[P\underline{r} - P\underline{\theta})]\right\} \\ &= \exp\left\{-\frac{1}{2}(\underline{r} - \underline{\theta})'(P'\Omega^{-1}P)(\underline{r} - \underline{\theta})\right\} \end{aligned} \tag{2.21}$$

As a result, equation (2.16) becomes:

$$\underline{\theta}|\underline{r}, P \sim N\left[\left(P'\Omega^{-1}P+\Sigma^{-1}\right)^{-1}\left(\Sigma^{-1}\underline{\mu}+P'\Omega^{-1}P\underline{r}\right), \left(P'\Omega^{-1}P+\Sigma^{-1}\right)^{-1}\right] \tag{2.22}$$

Black and Litterman (1991, 1992) set[1]

$$\underline{q} = P\underline{r} \tag{2.23}$$

as a user-defined return vector. This return vector needs to be consistent with the view specified in the P matrix. This is the key to the Black-Litterman model. As we shall demonstrate later in our empirical work, equation (2.23) dictates the performance (reward as well as risk) of the portfolio. The user of the Black-Litterman model must specify P matrix and $\underline{q}$ return vector simultaneously.

In the empirical work where raw returns are collected for each variable in $\underline{r}$, we then need to adjust them by P using equation (2.23). [2] Then equation (2.22) is used in the Sharpe ratio calculation to solve for the portfolio weights.

2.5 Naive Bayes Classification (see #7)

Bayes rule is easily used for classification, which will be discussed in Chapter 9. One can regard Naive Bayes classification as supervised learning as opposed to other classification methods (such as k-means) which are unsupervised. As mentioned in Chapter 12, supervised learning has the advantage that it is faster and does not require a lot of data.

In this section, I only use examples from Hull and Brownlee who have done excellent jobs in explaining naive Bayes classification. As the Greek philosopher Aristotle once said: "If you can't [compete with] them, join them."

[1] See footnote 2.

[2] In various papers in the literature, the notation is quite inconsistent and confusing. We hope that we provide consistency and clarity of the notation. In their original paper: Fischer Black and Robert Litterman (1991) uses $P \times \mathbb{E}[R] = Q + \varepsilon$ where $\varepsilon \sim N(0, \Omega)$. Then they postulate that the mean is $\overline{E[R]} = \left(P'\Omega^{-1}P+\Sigma^{-1}\right)^{-1}\left(\Sigma^{-1}\mu+P'\Omega^{-1}Q\right)$. Note that we replace R by $\underline{r}$ and Q by $\underline{q}$ as capital letters in the paper represent matrices and small letters with a lower bar represent vectors.

2.5.1 Hull

Now we apply Bayesian rule differently on classification.

$$\begin{aligned} p(C|x_1 \cdots x_n) &= \frac{p(x_1 \cdots x_n|C)p(C)}{p(x_1 \cdots x_n)} \\ &= \frac{\prod p(x_i|C)}{p(x_1 \cdots x_n)} p(C) \end{aligned} \tag{2.24}$$

In Hull, several examples are provided. The first example evaluates how good a customer is based upon his/er debt-to-income ratio d, and FICO score c.

† Example 1

Hull begins with just the debt-to-income ratio, d. From experience, we know the probability of a good loan and the debt-to-income ratio as:

$$\begin{aligned} p(L = g) &= 0.7917 \\ p(d > 19.85) &= 0.3951 \end{aligned}$$

Also, let $p(d > 19.85|L = g) = 0.3614$.

Then, applying equation (2.24), we get:

$$\begin{aligned} p(L = g|d > 19.85) &= \frac{p(d > 19.85|L = g)p(L = g)}{p(d > 19.85)} \\ &= \frac{0.3614 \times 0.7917}{0.3951} \\ &= 0.7242 \end{aligned}$$

Now we include FICO, c. We would like to know how good a client is if his/er debt-to-income ratio and FICO are both bad:

$$p(L = g|d > 19.85, c < 687.5)$$

We first have to assume that d and c are independent conditional on the loan quality, as described in Figure 2.1.

Then, we can easily calculate the quality of the borrower:

$$p(L|d, c) = \frac{p(d, c|L)p(L)}{p(d, c)} \quad = \frac{p(d|L)p(c|L)p(L)}{p(d, c)}$$

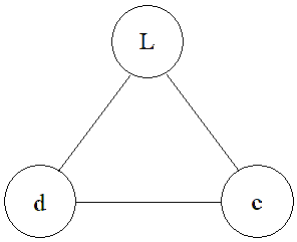


Figure 2.1: A Graph of Hull's Example

Undirected graphs and directed graphs will be introduced in Chapter 6 and Chapter 7 respectively. A general graph theory will first be presented and undirected as well as directed graphs will be explained and analyzed. While graph theory is a large branch in mathematics (equivalent to calculus), Chapters xx and yy will only cover relevant information regarding finance applications.

† Example 2

The second example in Hull takes three features:

- H – whether the applicant owns a house
- E – years of employment; and
- T – number of co-applicants

We are given the following:

$$\begin{aligned}
&p(H = \text{yes}|L = \text{good}) = 0.6\\
&p(H = \text{yes}|L = \text{bad}) = 0.5\\
&p(E > 1\text{y}|L = \text{good}) = 0.7\\
&p(E > 1\text{y}|L = \text{bad}) = 0.6\\
&p(T = 2|L = \text{good}) = 0.2\\
&p(T = 2|L = \text{bad}) = 0.1
\end{aligned}$$

To evaluate the quality of the borrower,

$$\begin{aligned}
p(L = \text{good}|H = y, E > 1, T = 2) &= \frac{p(H|L)p(E|L)p(T|L)p(L)}{p(H, E, T)}\\
&= \frac{0.6 \times 0.7 \times 0.2}{p(H, E, T)} \times 0.85\\
&= \frac{0.0714}{p(H, E, T)}
\end{aligned}$$

and

$$\begin{aligned} p(L = \text{bad}|H = \text{yes}, E > 1, T = 2) &= \frac{p(H|L)p(E|L)p(T|L)p(L)}{p(H, E, T)} \\ &= \frac{0.5 \times 0.6 \times 0.1}{p(H, E, T)} \times 0.85 \\ &= \frac{0.0045}{p(H, E, T)} \end{aligned}$$

Note that the denominator $p(H, E, T)$ is data so irrelevant. We know that they sum to 1, hence $p(H, E, T) = 0.0714 + 0.0045 = 0.0759$, and the actual probabilities are:

$$\begin{aligned} p(L = \text{good}|H = \text{yes}, E > 1, T = 2) &= 0.941 = \tfrac{0.0714}{0.0759} \\ p(L = \text{bad}|H = \text{yes}, E > 1, T = 2) &= 0.059 = \tfrac{0.0045}{0.0759} \end{aligned}$$

† Example 3

In this example, Hull shows us how to use raw data to perform Naive Bayes classification (by assuming Gaussian distributions). Going back to Example 1 where we use FICO c and debt-to-income ratio d to evaluate a borrower. Recall that the $p(L = g) = 0.7917$. Now, we are given the data of the past borrowers – their FICO scores, their debt-to-income ratios, and finally if they pay back their loans.

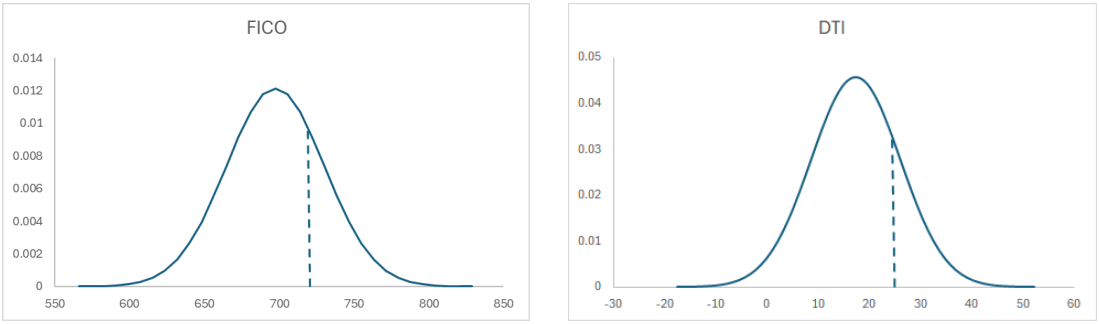
To build the Naive Bayes model, we divide the data into two groups: good and bad. And in each group, we calculate the means and standard deviations of c and d. It should be noted that since the means and standard deviations are calculated within each group, these means and standard deviations are conditional means and standard deviations (i.e. given a loan being good or bad). The result is in the following table.

Table 2.1: Conditional Means and Standard Deviations

	FICO c		DTI d	
	mean	std.dev.	mean	std.dev.
Good loan	697.38	32.85	17.37	8.72
Bad loan	686.73	24.26	20.41	9.11

Given the above table, we can draw (so to visualize more easily) the two Gaussian densities as in Figure 2.2. Now we want to evaluate the credit quality of a borrower who has 720 FICO and 25% debt-to-income ratio. Since the distributions are continuous, we cannot calculate the probabilities of exactly at such values.

Instead, we can only calculate densities. Figure 2.2 depicts how to obtain the two density values (see dotted lines).



(a) FICO (b) DTI

Figure 2.2: Distributions of FICO and DTI

Plug these values to the above two densities and obtain the following values:

Table 2.2: Density Values

	FICO c		DTI d	
Good loan	720	0.009581	25	0.031199
Bad loan	720	0.006421	25	0.038572

Now we can compute $p(L = \text{good}|c = 720, d = 25)$ and $p(L = \text{bad}|c = 720, d = 25)$:

$$\begin{aligned} & p(L = \text{good}|c = 720, d = 25) \\ &= \frac{p(c = 720|L = \text{good})p(d = 25|L = \text{good})p(L = \text{good})}{Q} \\ &= \frac{0.009581 \times 0.031199 \times 0.7917}{Q} \\ &= \frac{0.0002366}{Q} \end{aligned}$$

and

$$\begin{aligned} & p(L = \text{bad}|c = 720, d = 25) \\ &= \frac{p(c = 720|L = \text{bad})p(d = 25|L = \text{bad})p(L = \text{bad})}{Q} \\ &= \frac{0.006421 \times 0.038572 \times 0.7917}{Q} \\ &= \frac{0.0000516}{Q} \end{aligned}$$

Given this result, we can easily calculate probability of success and failure:

$$p(L = g|x) = \frac{0.0002366}{0.000237 + 0.0000516} = 0.821$$
$$p(L = b|x) = \frac{0.0000516}{0.000237 + 0.0000516} = 0.179$$

By some given threshold, we can say that this application is "APPROVED".

2.5.2 Jason Brownlee

Use the given diabetes data and assume normality. Perform all the calculations as in Example 3 of Hull. Make a prediction. Find accuracy ratio.

- diabeties case
- iris case

2.6 Bayesian Network

(Wiki) A Bayesian network (also known as a Bayes network, Bayes net, belief network, or decision network) is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG) which is covered in Chapter 7.

See the example in Figure 2.3. Each variable follows a Bernulli distribution. The DAG shows that rain will stop sprinkler from watering the lawn and wet the grass (notice that, given raining, the probability of sprinkler on is very low – 1%). Sprinkler will wet the grass.

The probabilistic model can be written as:

$$p(G, S, R) = p(G|S, R)p(S|R)p(R)$$

The top-right table of Figure 2.3 is $p(R)$. The top-left table is $p(S|R)$. Finally the bottom table is $p(G|S, R)$.

Not only graph theory is related to Bayesian networks, it also broadly overlaps with textual analysis (NLP, or natural language processing) which will be introduced in Chapter 4.

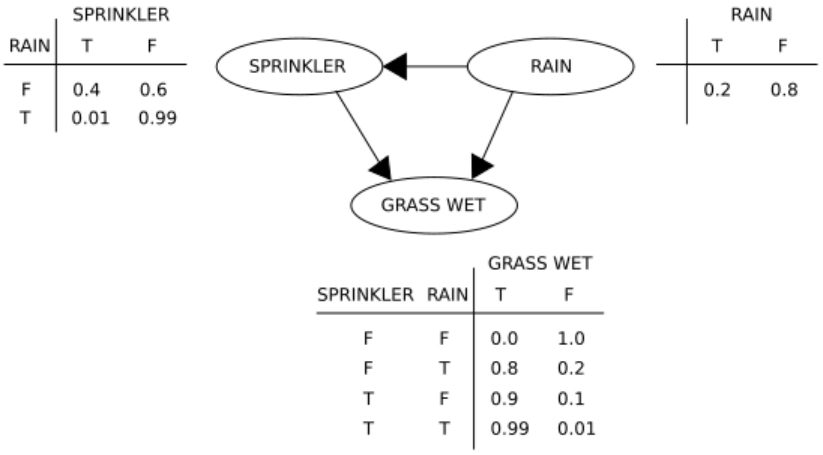


Figure 2.3: A Bayesian Network (DAG)

2.7 Homework and Project

† Homework

Do the pima or iris diabetes example.

† Project

Collect data and run the Black-Litterman model.

2.8 Appendix

2.8.1 Maximum Likelihood Estimation under Uni-variate Gaussian

Take a regression model:

$$y = a + bx + e$$

where $\mu = a + bx$ and $\mathbb{V}[e] = \sigma^2$. Therefore, $y \sim N(\mu, \sigma^2)$ and

$$\begin{aligned} f(y) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-a-bx)^2} \end{aligned}$$

The likelihood function is:

$$\mathscr{L}(y_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y_i-a-bx_i)^2}$$

Collect n observations of y, that is $y_1, \cdots, y_n$:

$$\begin{aligned}\mathscr{L}(y_1,\cdots,y_n|\mu,\sigma) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y_i-a-bx_i)^2} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-a-bx_i)^2\right)\end{aligned}$$

Taking logs,

$$\begin{aligned}\ln\mathscr{L}(y_1,\cdots,y_n|\mu,\sigma) &= -\frac{n}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-a-bx_i)^2 \\ &= -\frac{n}{2}\left(\ln 2\pi + \ln\sigma^2\right) - \frac{1}{2\sigma^2}(\underline{y}-\mathbf{X}\underline{\beta})'(\underline{y}-\mathbf{X}\underline{\beta}) \\ &= -\frac{n}{2}\left(\ln 2\pi + \ln\sigma^2\right) - \frac{1}{2\sigma^2}(\underline{y}'\underline{y} - \underline{\beta}'\mathbf{X}'\underline{y} - \underline{y}'\mathbf{X}\underline{\beta} + \underline{\beta}'\mathbf{X}'\mathbf{X}\underline{\beta})\end{aligned}$$

where

$$\underline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \quad \underline{\beta} = \begin{pmatrix} a \\ b \end{pmatrix}$$

Note that $\underline{\beta}'\mathbf{X}'\underline{y}$ equals $\underline{y}'\mathbf{X}\underline{\beta}$. Differentiate the log likelihood function with respect to the parameters:

$$\frac{\partial \ln\mathscr{L}}{\partial\underline{\beta}} = \frac{\partial \ln\mathscr{L}}{\partial(a,b)'} = -\frac{1}{2\sigma^2}(-2\mathbf{X}'\underline{y} + 2\mathbf{X}'\mathbf{X}\underline{\beta})$$

Setting it to 0 and solving for the parameters:[3]

$$\begin{aligned}-\frac{1}{2\sigma^2}(-2\mathbf{X}'\underline{y} + 2\mathbf{X}'\mathbf{X}\hat{\underline{\beta}}) &= 0 \\ \hat{\underline{\beta}} = \begin{pmatrix} \hat{a} \\ \hat{b} \end{pmatrix} &= (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\underline{y})\end{aligned}$$

It is easy to verify that $\hat{a} = \bar{y} - \hat{b}\bar{x}$ and $\hat{b} = \mathbb{K}[x,y]/\mathbb{V}[x]$. Continue to take derivative with respect to σ^2:

$$\frac{\partial \ln\mathscr{L}}{\partial\sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(\underline{y}-\mathbf{X}\underline{\beta})'(\underline{y}-\mathbf{X}\underline{\beta})$$

[3]Note that hat represents estimates of the parameters.

Setting it to 0 and:

$$\hat{\sigma}^2 = \frac{1}{n}(\underline{y} - \mathbf{X}\underline{\beta})'(\underline{y} - \mathbf{X}\underline{\beta})$$

Substituting in $\hat{\beta}$ for β and we get:

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{n}(\underline{y} - \mathbf{X}\underline{\hat{\beta}})'(\underline{y} - \mathbf{X}\underline{\hat{\beta}}) \\ &= \frac{1}{n}(\underline{y} - \underline{\hat{y}})'(\underline{y} - \underline{\hat{y}}) \\ &= \frac{1}{n}\underline{e}'\underline{e} \\ &= \frac{1}{n}\sum_{i=1}^{n} e_i^2\end{aligned}$$

Note that this is NOT the OLS result (which is $n - k$). In other words, the MLE result for the variance estimate is not an unbiased estimator.

Note that OLS can only give us point estimates and does not provide inference. That is, we do not know the standard errors of these estimates. To get the standard errors of the estimates, we need to know the distribution of each estimator (an estimator is a formula which calculates the estimate). For this, we need to know the distribution of the each (dependent and independent) variable. Hence, we need a parametric method. When the variables follow a joint Gaussian, then the distributions of the estimators can be derived. This is the maximum likelihood estimation through which we use the information matrix (i.e. Hessian matrix) which is derived below.

The Gradient matrix is:

$$\begin{pmatrix} \frac{\partial \ln \mathscr{L}}{\partial \underline{\beta}} \\ \frac{\partial \ln \mathscr{L}}{\partial \sigma^2} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2\sigma^2}(-2\mathbf{X}'\underline{y} + 2\mathbf{X}'\mathbf{X}\underline{\beta}) \\ -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(\underline{y} - X\underline{\beta})'(\underline{y} - X\underline{\beta}) \end{pmatrix}$$

To derive the Hessian matrix, we take the second order derivatives:

$$\frac{\partial^2 \ln \mathscr{L}}{\partial \underline{\beta}\underline{\beta}'} = \frac{\partial\left[\frac{1}{\sigma^2}(\mathbf{X}'\underline{y} - \mathbf{X}'\mathbf{X}\underline{\beta})\right]}{\partial \underline{\beta}'} = -\frac{1}{\sigma^2}\mathbf{X}'\mathbf{X}$$

$$\frac{\partial^2 \ln \mathscr{L}}{\partial \underline{\beta}\partial\sigma^2} = \frac{\partial\left[\frac{1}{\sigma^2}(\mathbf{X}'\underline{y} - \mathbf{X}'\mathbf{X}\underline{\beta})\right]}{\partial\sigma^2} = -\frac{1}{\sigma^4}(\mathbf{X}'\underline{y} - \mathbf{X}'\mathbf{X}\underline{\beta}) = -\frac{1}{\sigma^4}\mathbf{X}'(\underline{y} - \mathbf{X}\underline{\beta}) = -\frac{1}{\sigma^4}\mathbf{X}'\underline{e}$$

$$\frac{\partial^2 \ln \mathscr{L}}{\partial\sigma^2\partial \underline{\beta}} = \frac{\partial\left[-\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(\underline{y} - \mathbf{X}\underline{\beta})'(\underline{y} - \mathbf{X}\underline{\beta})\right]}{\partial \underline{\beta}} = \frac{1}{2\sigma^4}(-2\mathbf{X}'\underline{y} + 2\mathbf{X}'\mathbf{X}\underline{\beta}) = -\frac{1}{\sigma^4}\mathbf{X}'\underline{e}$$

$$\frac{\partial^2 \ln \mathscr{L}}{\partial\sigma^2\partial\sigma^2} = \frac{\partial\left[-\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(\underline{y} - \mathbf{X}\underline{\beta})'(\underline{y} - \mathbf{X}\underline{\beta})\right]}{\partial\sigma^2} = \frac{n}{2\sigma^4} - \frac{2}{2\sigma^6}(\underline{y} - \mathbf{X}\underline{\beta})'(\underline{y} - \mathbf{X}\underline{\beta}) = \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}\underline{e}'\underline{e}$$

The Hessian matrix is:

$$H(\Theta) = \begin{bmatrix} -\frac{1}{\sigma^2}\mathbf{X}'\mathbf{X} & -\frac{1}{\sigma^4}\mathbf{X}'\underline{e} \\ -\frac{1}{\sigma^4}\mathbf{X}'\underline{\varepsilon} & \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}\underline{e}'\underline{e} \end{bmatrix}$$

Taking the expected value:

$$\begin{aligned} \mathbb{E}[H(\Theta)] &= \begin{bmatrix} -\frac{1}{\sigma^2}X'X & 0 \\ 0 & -\frac{n}{2\sigma^4} - \frac{1}{\sigma^6}n\mathbb{E}[s^2] \end{bmatrix} \\ &= \begin{bmatrix} -\frac{1}{\sigma^2}X'X & 0 \\ 0 & \frac{n}{2\sigma^4} - \frac{n\sigma^2}{2\sigma^6} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{1}{\sigma^2}X'X & 0 \\ 0 & -\frac{n}{2\sigma^4} \end{bmatrix} \end{aligned}$$

Information matrix is the negative of the expected Hessian matrix as follows:

$$I(\Theta) = -E[H(\Theta)] = \begin{bmatrix} \frac{1}{\sigma^2}X'X & 0 \\ 0 & \frac{n}{2\sigma^4} \end{bmatrix}$$

and the variance is the reciprocal of the information matrix:

$$\mathbb{V}[\Theta] = I(\Theta)^{-1} = \begin{bmatrix} \frac{1}{\sigma^2}X'X & 0 \\ 0 & \frac{n}{2\sigma^4} \end{bmatrix}^{-1}$$

Using the simple regression shown above, we have:

$$\frac{1}{\sigma^2}\mathbf{X}'\mathbf{X} = \begin{bmatrix} \frac{n}{\sigma^2} & \frac{1}{\sigma^2}\sum_{i=1}^n x_i \\ \frac{1}{\sigma^2}\sum_{i=1}^n x_i & \frac{1}{\sigma^2}\sum_{i=1}^n x_i^2 \end{bmatrix}$$

and hence (information matrix)

$$I(\Theta) = \begin{bmatrix} \frac{n}{\sigma^2} & \frac{1}{\sigma^2}\sum_{i=1}^n x_i & 0 \\ \frac{1}{\sigma^2}\sum_{i=1}^n x_i & \frac{1}{\sigma^2}\sum_{i=1}^n x_i^2 & 0 \\ 0 & 0 & \frac{n}{\sigma^2} \end{bmatrix}$$

Taking inverse:

$$\begin{aligned} I^{-1} &= \frac{1}{\frac{n^2}{\sigma^6}\sum_{i=1}^n x_i^2 - \frac{n^2}{\sigma^6}\left(\sum_{i=1}^n x_i\right)^2} \begin{bmatrix} \frac{n}{\sigma^4}\sum_{i=1}^n x_i^2 & -\frac{n}{\sigma^4}\sum_{i=1}^n x_i & 0 \\ -\frac{n}{\sigma^4}\sum_{i=1}^n x_i & \frac{n^2}{\sigma^4} & 0 \\ 0 & 0 & \frac{n}{\sigma^4}\sum_{i=1}^n x_i^2 - \frac{1}{\sigma^4}\left(\sum_{i=1}^n x_i\right)^2 \end{bmatrix} \\ &= \frac{1}{\frac{n^3}{\sigma^6}s^2} \begin{bmatrix} \frac{n^2}{\sigma^4}(s^2 + \bar{x}^2) & -\frac{n^2}{\sigma^4}\bar{x} & 0 \\ -\frac{n^2}{\sigma^4}\bar{x} & \frac{n^2}{\sigma^4} & 0 \\ 0 & 0 & \frac{n^2}{\sigma^4}s^2 \end{bmatrix} \\ &= \sigma^2 \begin{bmatrix} \frac{1}{ns^2}(s^2 + \bar{x}^2) & -\frac{1}{ns^2}\bar{x} & 0 \\ -\frac{1}{ns^2}\bar{x} & \frac{1}{ns^2} & 0 \\ 0 & 0 & \frac{1}{n} \end{bmatrix} \end{aligned}$$

2.8.2 OLS

To compare with MLE, one can estimate regression coefficients with ordinary least squares (OLS). Since the data are assumed to be i.i.d., each observation is assumed to have the following error term:

$$\varepsilon_i = y_i - (\alpha + \beta x_i) \tag{2.25}$$

where $\varepsilon_i \sim N(0, \sigma^2)$.

The sum of squared errors is:

$$\varepsilon'\varepsilon = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \{y_i - (\alpha + \beta x_i)\}^2 \tag{2.26}$$

To minimize the sum of squares, we take partial derivatives w.r.t. the two parameters: α and β.

First, taking the partial derivative w.r.t. α and setting it to 0:

$$\frac{\partial}{\partial \alpha} \sum_{i=1}^{n} \varepsilon_i^2 = 2 \sum_{i=1}^{n} \{y_i - (\alpha + \beta x_i)\} (-1) = 0$$

and solving for α

$$\begin{aligned} \sum_{i=1}^{n} y_i &= \sum_{i=1}^{n} \hat{\alpha} + \hat{\beta} \sum_{i=1}^{n} x_i \\ \hat{\alpha} &= \bar{y} - \hat{\beta}\bar{x} \end{aligned}$$

So we know the estimate of α (i.e. $\hat{\alpha}$) relies on the estimate of β (i.e. $\hat{\beta}$).

Now, we take the partial derivative of equation (2.26) w.r.t. β and set it to 0:

$$\frac{\partial}{\partial \beta} \sum_{i=1}^{n} \varepsilon_i^2 = 2 \sum_{i=1}^{n} \{y_i - (\alpha + \beta x_i)\} (-x_i) = 0$$

Solving for β:

$$\begin{aligned} \sum_{i=1}^{n} x_i y_i + \alpha \sum_{i=1}^{n} x_i - \beta \sum_{i=1}^{n} x_i^2 &= 0 \\ \sum_{i=1}^{n} x_i y_i + (\bar{y} - \beta\bar{x}) \sum_{i=1}^{n} x_i - \beta \sum_{i=1}^{n} x_i^2 &= 0 \\ \sum_{i=1}^{n} x_i y_i + \bar{y} \sum_{i=1}^{n} x_i - \beta\bar{x} \sum_{i=1}^{n} x_i - \beta \sum_{i=1}^{n} x_i^2 &= 0 \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} - \beta \sum_{i=1}^{n} x_i^2 - \beta n\bar{x}^2 &= 0 \\ \sum_{i=1}^{n} x_i y_i - 2n\bar{x}\bar{y} + n\bar{x}\bar{y}) - \beta \sum_{i=1}^{n} x_i^2 - 2n\bar{x}^2 + n\bar{x}^2 &= 0 \\ \sum_{i=1}^{n} (x_i y_i - x_i\bar{y} - \bar{x}y_i + \bar{x}\bar{y}) - \beta \sum_{i=1}^{n} (x_i^2 - 2x_i\bar{x} + \bar{x}^2) &= 0 \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) - \beta \sum_{i=1}^{n} (x_i - \bar{x})^2 &= 0 \end{aligned}$$

we obtain its estimator as:

$$\hat{\beta} = \frac{s_{xy}}{s_x^2}$$

Lastly, we need to find the estimator for σ^2. To do that, let's review some very basic statistical knowledge. Let x be i.i.d. and each observation $x_i \sim N(\mu, \sigma^2)$. First, the sample mean $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ is an unbiased estimator of the population mean μ:

$$\mathbb{E}[\bar{x}] = \frac{1}{n}\sum\nolimits_{i=1}^n \mathbb{E}[x_i] = \frac{1}{n}\sum\nolimits_{i=1}^n \mu = \frac{1}{n} n\mu = \mu \tag{2.27}$$

with the variance of the sample mean being:

$$\mathbb{V}[\bar{x}] = \mathbb{V}\left[\frac{1}{n}\sum\nolimits_{i=1}^n x_i\right] = \frac{1}{n^2}\mathbb{V}\left[\sum\nolimits_{i=1}^n x_i\right] = \frac{1}{n^2}\sum\nolimits_{i=1}^n \sigma^2 = \frac{1}{n^2} n\sigma^2 = \frac{1}{n}\sigma^2 \tag{2.28}$$

where the fourth term is achieved because x_i's are independent of each other.

Unfortunately the sample variance $s^2 = \frac{1}{n}\sum_{i=1}^n (x_i - \bar{x})^2$ is a biased one:

$$\begin{aligned}
\mathbb{E}[s^2] &= \frac{1}{n}\sum\nolimits_{i=1}^n \mathbb{E}\left[(x_i - \bar{x})^2\right] \\
&= \frac{1}{n}\mathbb{E}\left[\sum\nolimits_{i=1}^n x_i^2 - n\bar{x}^2\right] \\
&= \frac{1}{n}\left\{\sum\nolimits_{i=1}^n \mathbb{E}\left[x_i^2\right] - n\mathbb{E}\left[\bar{x}^2\right]\right\} \\
&= \frac{1}{n}\left\{\sum\nolimits_{i=1}^n \mathbb{E}\left[x_i^2\right] - n\left(\mathbb{V}[\bar{x}] + \mathbb{E}[\bar{x}]^2\right)\right\} \\
&= \frac{1}{n}\left\{\sum\nolimits_{i=1}^n \left(\sigma^2 + \mu^2\right) - n\left(\frac{\sigma^2}{n} + \mu^2\right)\right\} \\
&= \frac{1}{n}\left\{(n-1)\sigma^2\right\} \\
&= \frac{n-1}{n}\sigma^2
\end{aligned}$$

A new estimator defined as follows:

$$\hat{\sigma}^2 = \frac{n}{n-1}s^2 = \frac{1}{n-1}\sum\nolimits_{i=1}^n (x_i - \bar{x})^2$$

which is then unbiased.

Now simply let $\mu = 0$ for each x_i (i.e. $\varepsilon_i = x_i \sim N(0, \sigma^2)$) and the same result applies.

2.8.3 Another Example – Poisson and Gamma

Assume:

$x|\theta \sim P(\theta)$ a Poisson

$p(\theta) \sim G(a, b)$ a Gamma which has mean $\frac{a}{b}$, mode $\frac{a-1}{b}$ (valid only for $a \geqslant 1$ otherwise 0) and variance $\frac{a}{b^2}$.

Then we can show that

$$\theta|x \sim G(n\bar{x} + a, n + b)$$

and we can let $a^* = n\bar{x} + a$ and $b^* = n + b$.

Hence $\mathbb{E}[\theta|X] = \frac{a^*}{b^*}$.

2.8.4 Simple MLE under Normality (Review)

Assume normality for $p(x|\theta)$ when $\theta = \{\mu, \sigma\}$ is fixed. To estimate $\theta = \{\mu, \sigma\}$, we collect independent observations of . Each x_i is a Gaussian and $x_i \perp x_j$.

Since $\theta = \{\mu, \sigma\}$ is fixed, $p(\theta) = 1$, and we can directly apply (1) $p(\theta|x) \propto p(x|\theta)$. This is known as the likelihood function and the log likelihood function as follows:

$$\begin{aligned} \mathscr{L} &= p(x_1 \cdots x_n|\mu, \sigma) = \textstyle\prod_{i=1}^{n} p(x_i|\mu, \sigma) \\ \ln \mathscr{L} &= \textstyle\sum_{i=1}^{n} \ln p(x_i|\mu, \sigma) \\ &= \textstyle\sum_{i=1}^{n} \left\{ -\frac{1}{2}\ln(2\pi) - \ln\sigma - \frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2 \right\} \\ &= -\frac{n}{2}\ln(2\pi) - n\ln\sigma - \frac{1}{2}\textstyle\sum_{i=1}^{n}\left(\frac{x_i-\mu}{\sigma}\right)^2 \end{aligned} \tag{2.29}$$

Take partial derivatives w.r.t. μ and σ and solve the simultaneous equations:

$$\begin{aligned} \frac{\partial}{\partial\mu}\ln\mathscr{L} = -\frac{1}{\sigma}\textstyle\sum_{i=1}^{n}\left(\frac{x_i-\mu}{\sigma}\right) &= 0 \\ \textstyle\sum_{i=1}^{n}(x_i - \mu) &= 0 \\ \mu &= \frac{1}{n}\textstyle\sum_{i=1}^{n} x_i \end{aligned} \tag{2.30}$$

which actually does not depend upon σ.

To derive the estimator for σ (or more precisely σ^2, we take the corresponding

partial derivative:

$$\begin{aligned} \ln \mathscr{L}(\theta = \sigma^2) &= -\tfrac{n}{2}\ln(2\pi) - \tfrac{n}{2}\ln\theta - \tfrac{1}{2\theta}\sum_{i=1}^{n}(x_i-\mu)^2 \\ \frac{\partial}{\partial\theta}\ln\mathscr{L}(\theta) &= -\tfrac{n}{2\theta} + \tfrac{1}{2\theta^2}\sum_{i=1}^{n}(x_i-\mu)^2 = 0 \\ n\theta &= \sum_{i=1}^{n}(x_i-\mu)^2 \\ \theta &= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2 \end{aligned} \tag{2.31}$$

Note that we can find MLE for $\theta = \sigma$ (as a self-exercise), but it is clear that the estimate is quite complex. This also indicates that $\hat{\sigma} \neq \sqrt{\hat{\sigma}^2}$ (the estimate of σ is not the square root of the estimate of σ^2)!

2.8.5 MLE for the Hyper Parameters (LOS)

$$p(x|z,\sigma) \sim N(z,\sigma^2)$$

where $z \sim N(\mu, a^2)$. Now the objective is to estimate μ given σ and a fixed (say $\sigma = 0.3$ and $a = 0.3$).

The posterior

$$\begin{aligned} p(\mu|x) &\propto p(x|z)p(z|\mu)p(\mu) \\ &= \exp\left\{-\tfrac{1}{2}\left(\tfrac{x-z}{\sigma}\right)^2 - \tfrac{1}{2}\left(\tfrac{z-\mu}{a}\right)^2\right\} \end{aligned} \tag{2.32}$$

where $p(\mu) = 1$.

Now we need to integrate over z so that $p(\mu|x) \propto p(x|\mu)$:

$$p(x|\mu) = \int_{-\infty}^{\infty} \exp\left\{-\tfrac{1}{2}\left(\tfrac{x-z}{\sigma}\right)^2 - \tfrac{1}{2}\left(\tfrac{z-\mu}{a}\right)^2\right\} dz \tag{2.33}$$

Then we arrive a probability function that connects x with μ. Collecting data $\{x_1 \cdots x_n\}$ and the likelihood function is:

$$\begin{aligned} \ln\mathscr{L} &= \sum_{i=1}^{n} \ln p(x_i|\mu) \\ &= \sum_{i=1}^{n} \ln \int_{-\infty}^{\infty} \exp\left\{-\tfrac{1}{2}\left(\tfrac{x-z}{\sigma}\right)^2 - \tfrac{1}{2}\left(\tfrac{z-\mu}{a}\right)^2\right\} dz \end{aligned} \tag{2.34}$$

This is not solvable and hence we will keep it as is. Taking partial derivative w.r.t. μ and setting it to 0, we can solve for its estimator:

$$\begin{aligned}\frac{\partial}{\partial\mu}\ln\mathscr{L} &= \frac{\partial}{\partial\mu}\sum_{i=1}^{n}\ln\int_{-\infty}^{\infty}\exp\left\{-\tfrac{1}{2}\left(\tfrac{x-z}{\sigma}\right)^2-\tfrac{1}{2}\left(\tfrac{z-\mu}{a}\right)^2\right\}dz\\ &= \frac{1}{X}\sum_{i=1}^{n}\int_{-\infty}^{\infty}\frac{\partial}{\partial\mu}\exp\left\{-\tfrac{1}{2}\left(\tfrac{x-z}{\sigma}\right)^2-\tfrac{1}{2}\left(\tfrac{z-\mu}{a}\right)^2\right\}dz\\ &= \frac{1}{X}\sum_{i=1}^{n}\int_{-\infty}^{\infty}\exp\left\{-\tfrac{1}{2}\left(\tfrac{x-z}{\sigma}\right)^2-\tfrac{1}{2}\left(\tfrac{z-\mu}{a}\right)^2\right\}\left(-\tfrac{z-\mu}{a}\right)dz\\ &= 0\end{aligned} \tag{2.35}$$

This can be numerically solved.

2.8.6 Detailed Derivation of (2.15)

We work backwards. First, for the sake of easy notation, we name $(\Omega^{-1}+\Sigma^{-1})$ as A and $\left(\Sigma^{-1}\underline{\mu}+\Omega^{-1}\underline{r}\right)$ as $\underline{b}$. Then

$$\begin{aligned}c_3 &= c_2+\frac{1}{2}(A^{-1}\underline{b})'A(A^{-1}\underline{b})\\ &= c_2+\frac{1}{2}\underline{b}'A^{-1}\underline{b}\end{aligned} \tag{2.36}$$

and

$$\begin{aligned}-\frac{1}{2}\underline{x}'\left(\Omega^{-1}+\Sigma^{-1}\right)\underline{x} &= -\frac{1}{2}(\underline{\theta}-A^{-1}\underline{b})'(A)(\underline{\theta}-A^{-1}\underline{b})\\ &= -\frac{1}{2}(\underline{\theta}'-\underline{b}'A^{-1})(A)(\underline{\theta}-A^{-1}\underline{b})\\ &= -\frac{1}{2}(\underline{\theta}'A-\underline{b}')(\underline{\theta}-A^{-1}\underline{b})\\ &= -\frac{1}{2}(\underline{\theta}'A\underline{\theta}-2\underline{\theta}'\underline{b}+\underline{b}'A^{-1}\underline{b})\end{aligned} \tag{2.37}$$

given $\underline{b}'\underline{\theta}=\underline{\theta}'\underline{b}$.

Adding the two terms together (i.e. last line, line #4, of equation (2.15)):

$$\#4 = -\frac{1}{2}\underline{x}'\left(\Omega^{-1}+\Sigma^{-1}\right)\underline{x}+c_3 = c_2-\frac{1}{2}\underline{\theta}'A\underline{\theta}+\underline{\theta}'\underline{b} \tag{2.38}$$

Now we begin with line 1 of equation (2.15):

$$\begin{aligned}
\#1 &= -\frac{1}{2}(\underline{r}-\underline{\theta})'\Omega^{-1}(\underline{r}-\underline{\theta}) - \frac{1}{2}(\underline{\theta}-\underline{\mu})'\Sigma^{-1}(\underline{\theta}-\underline{\mu}) + c_1 \\
&= -\frac{1}{2}(\underline{r}'\Omega^{-1}\underline{r} - 2\underline{\theta}'\Omega^{-1}\underline{r} + \underline{\theta}'\Omega^{-1}\underline{\theta}) - \frac{1}{2}(\underline{\theta}'\Sigma^{-1}\underline{\theta} - 2\underline{\theta}'\Sigma^{-1}\underline{\mu} + \underline{\mu}'\Sigma^{-1}\underline{\mu}) + c_1 \\
(\#2) &= c_2 + \underline{\theta}'\Omega^{-1}\underline{r} - \frac{1}{2}\underline{\theta}'\Omega^{-1}\underline{\theta} - \frac{1}{2}\underline{\theta}'\Sigma^{-1}\underline{\theta} + \underline{\theta}'\Sigma^{-1}\underline{\mu} \\
(\#3) &= c_2 - \frac{1}{2}(\underline{\theta}'(\Omega^{-1}+\Sigma^{-1})\underline{\theta}) + \underline{\theta}'(\Omega^{-1}\underline{r} + \Sigma^{-1}\underline{\mu}) \\
&= c_2 - \frac{1}{2}\underline{\theta}'A\underline{\theta} + \underline{\theta}'\underline{b}
\end{aligned} \tag{2.39}$$

which is equal to equation (2.38).

$$\underline{r}'\Omega^{-1}\underline{r} + \boxed{\underline{\mu}'\Omega^{-1}\underline{\mu}}$$

Chapter 3

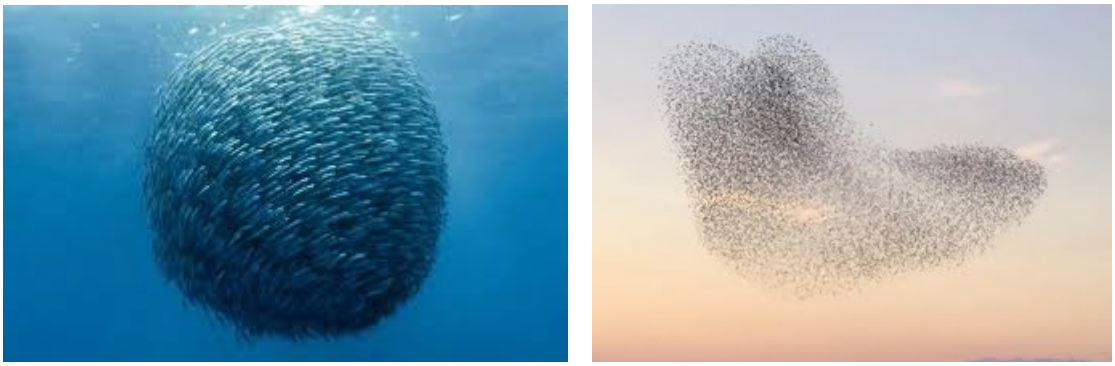
Swarm Intelligence

What is not good for the swarm is not good for the bee.

Marcus Aurelius, 121-180

3.1 Introduction

Swarm intelligence is the collective behavior of decentralized, self-organized systems, natural or artificial. The concept is employed in work on artificial intelligence. The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems.



(a) Swarm of Fish (b) Swarm of Birds

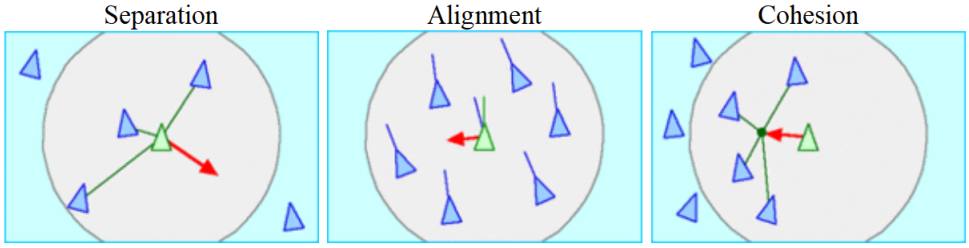
Figure 3.1: Swarms in Nature

3.2 Theory and Algorithm

Reynold (1987)[1] was the first to "artificialize" such natural intelligence and create a computer algorithm named Boids (for bird-oid object). Reynold's algorithm is amazingly simple. For any given bird, Reynold devises a set of linear equations (vectors) combining which determines how the bird should fly to its next destination.

- separation
- cohesion
- alignment
- add others...

There are countless versions of Boids.[2] One can add obstacles. One can add an objective destination (swim to target). One can do Boids in a maze. The basic Boids can be described by the following algorithm.



Sources: https://en.wikipedia.org/wiki/Boids

Figure 3.2: Reynold's Algorithm

Formally, let there be m birds flying in an n-dimensional space. Also let:

- $\vec{f}_t^{(i)}$ be the i-th fish at time t
- $\vec{v}_t^{(i)}$ be a vector in the $\mathbb{R}^n$ space representing the velocity of the i-th fish
- $\vec{x}_t^{(i)}$ be a vector in $\mathbb{R}^n$ space representing the position (coordinates) of the i-th bird and $\vec{x}_t^{(i)} = \vec{x}_{t-1}^{(i)} + \vec{v}_t^{(i)}$

[1] According to Wikipedia, Reynold created Boids in 1986: "Boids is an artificial life program, developed by Craig Reynolds in 1986, which simulates the flocking behaviour of birds."

[2] For example, see Google Scholar: https://scholar.google.com/scholar?q=boids+flocking+algorithm&hl=en&as_sdt=0

The vectors above have n dimensions, $j = 1, \cdots, n$. For example, $\vec{x}_t^{(i)} =< x_{j,t}^{(i)} >$.

Finally let $F = \{f_t^{(i)} | i = 1, \cdots, m\}$ be the collection of all fish. Define a mapping function $X_t^{(i)} = \wp(f_t^{(i)})$ which returns all $f_t^{(j \neq i)} \in F - f_t^{(i)}$ where a radius d and an angle a are predetermined such that

$$\left\| \vec{x}_t^{(i)}, \vec{x}_t^{(j \neq i)} \right\| < d \quad \text{and} \quad \angle\{\vec{x}_t^{(j \neq i)}, \vec{x}_t^{(i)}\} = a^\circ \tag{3.1}$$

are satisfied where

$$\angle\{\vec{a}, \vec{b}\} = \cos^{-1}\left[\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right]$$

is the angle between two vectors $\vec{a}$ and $\vec{b}$ and furthermore (each vector has n-dimensional coordinates $< x_1, \cdots, x_n >$, ignoring t)

$$\left|\vec{a}\right| = \sqrt{\sum_{j=1}^{n} (x_j^{(a)})^2}$$

which is length (from origin) of vector $\vec{a}$

$$\left\|\vec{a}, \vec{b}\right\| = \sqrt{\sum_{j=1}^{n} (x_j^{(a)} - x_{j,t}^{(b)})^2}$$

is the distance and

$$\vec{a} \cdot \vec{b} = \sum_{j=1}^{n} x_{j,t}^{(a)} x_j^{(b)}$$

is the inner product (sum product).

In other words, $X_t^{(i)}$ can be recognized as the neighborhood of fish $f_t^{(i)}$ under prespecificed parameter values of d and a at a given time t.

3.2.1 Alignment and Cohesion

In words, for any given bird i, where it is heading depends on a reference group of birds "nearby", described by a set of birds $X_t^{(i)} = \wp(f_t^{(i)})$. These reference birds must be "nearby" in the following sense – they must be within a distance (specified by the radius d) and within an angle (specified by a°), as depicted graphically as:[3]

[3]These reference birds are like "my leaders" for a given bird.

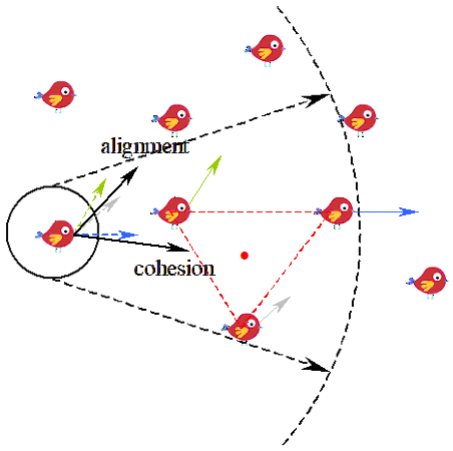


Figure 3.3: Reynold's Algorithm

where the circled bird is referencing to three nearby birds by the angle and the radius. The alignment and cohesion (we ignore separation parameter for the moment) parameters are calculated as follows:[4]

$$\begin{aligned} \vec{v}_{A,t}^{(i)} &= \text{avg}\left(\vec{v}_{t-1}^{(j\neq i)} | f_{t-1}^{(j)} \in X_{t-1}^{(i)}\right) - \vec{v}_{t-1}^{(i)} \\ \vec{v}_{C,t}^{(i)} &= \text{avg}\left(\vec{x}_{t-1}^{(j\neq i)} | f_{t-1}^{(j)} \in X_{t-1}^{(i)}\right) - \vec{x}_{t-1}^{(i)} \end{aligned} \tag{3.2}$$

and then an average velocity is calculated as follows:

$$\vec{v}_t^{(i)} = w_A \vec{v}_{A,t}^{(i)} + w_C \vec{v}_{C,t}^{(i)} \tag{3.3}$$

where $w_A + w_C = 1$ and each is positive. Finally, the position of each fish is updated as follows:

$$\vec{x}_t^{(i)} = \vec{x}_{t-1}^{(i)} + \vec{v}_t^{(i)} \tag{3.4}$$

As emphasized earlier, a swarm is a behavioral model which describes how fish (ants, bees, birds) move and an artificial swarm is a mathematical (linear algebraical) algorithm that imitates this natural behavior by animals. One can use an artificial swarm to solve a number of complex problems.[5]

[4]We ignore separation in our model because in our applications particles can take the same coordinates (i.e. collision is allowed).

[5]While this is out of the scope of this paper, we encourage the readers to view a popular YouTube clip on how drones use an artificial swarm: "'Skynet'Drones Work Together for 'Homeland Security'" (https://www.youtube.com/watch?v=oDyfGM35ekc).

3.2.2 Separation

Separation is to prevent any two fish from colliding. To do that, any two fish are not allowed to be closer than :

$$\left\| \vec{x}_{t-1}^{(j\neq i)} - \vec{x}_{t-1}^{(i)} \right\| > \varepsilon \tag{3.5}$$

where ε is a prespecified value.

There are an infinite number of ways to satisfy this constraint and yet none of them is easy. We will discuss this later. Luckily, we do not need this condition in finance (as we are studying real animals). If only separation is applied, then the flock will diverge.

3.2.3 Follow a Leader

Define the leader position at the previous iteration as $\vec{x}_{L,t-1}^{(i)}$. This can be a fixed location or the location (*) of a chosen leader ℓ. That is:

$$\vec{x}_{L,t-1}^{(i)} = \begin{cases} \vec{x}_{t-1}^{*} \\ \vec{x}_{t-1}^{(\ell)} \end{cases} \tag{3.6}$$

where ℓ represents the leader fish. Then we can define the velocity for a fish to swam towards the leader as:

$$\vec{v}_{L,t}^{(i)} = \vec{x}_{t-1}^{(i)} - \vec{x}_{L,t-1}^{(i)} \tag{3.7}$$

Expanding equation (3.3) to include the leader as follows:

$$\vec{v}_{t}^{(i)} = w_A \vec{v}_{A,t}^{(i)} + w_C \vec{v}_{C,t}^{(i)} + w_L \vec{v}_{L,t}^{(i)} \tag{3.8}$$

3.2.4 Memory

In many situations, we need to include a personal experience. A particle can have a tendency not to move away from its own personal position (comfort zone).

This addition is the let each particle try to stay in its own personal history

$$\vec{v}_{t}^{(i)} = w_A \vec{v}_{A,t}^{(i)} + w_C \vec{v}_{C,t}^{(i)} + w_L \vec{v}_{L,t}^{(i)} + w_P \vec{v}_{P,t}^{(i)} \tag{3.9}$$

For $i = 1, \cdots, m$ particles and each particle is a vector of $j = 1, \cdots, n$ dimensions, we have:

$$\vec{v}_{P,t}^{(i)} = \alpha_i \vec{u}_{t}^{(i)} + (1 - \alpha_i)(\vec{p}_{t-1}^{(i)} - \vec{x}_{t-1}^{(i)}) \tag{3.10}$$

where $\vec{u}_t^{(i)}$ is a uniform random variable and $0 \leq \alpha_i \leq 1$.

At each position there is "fitness function" $\phi(\vec{x}_t^{(i)})$ (sometimes called distance function) at which a "cost" or "benefit" is computed. This fitness function is the objective function to be minimized (or maximized).

$$\vec{p}_t^{(i)} = \max_t \{\phi(\vec{x}_t^{(i)})\} \tag{3.11}$$

(looping through history to record the personal best)

Data of patents

Animation (Fish)

3.3 Particle Swarm Optimization

In theory, swarm intelligence is effective for optimization problems in a high-dimensional space. PSO is such an application. The original version of PSO was first proposed by Eberhart and Kennedy (1995) who modify the behavioral model of swarm into an objective-seeking algorithm. Similar to Renold's, their model "artificializes" the group behavior of a flock of birds seeking food. Via bird-to-bird chirping (peer-to-peer communication), all birds fly to the loudest sound of chirping. Subsequently, Eberhart and Shi (1998) improve the model by adding an inertia term (symbolized as w later as we introduce the model) and it has become the standard PSO algorithm used today. Setting a proper value of the inertia term is to seek the balance between *exploitation* and *exploration*. A larger value of the inertia term gives more weight to exploration (as the bird is more likely to fly on its own) and a smaller value of the inertia term gives more weight to exploitation (as the bird intends more to fly toward other birds).[6]

One can compare PSO to the grid search. A grid search can find the global optimum and yet it takes an exploding amount of time to reach such a solution, especially in a high-dimensional space. PSO can be regarded as a "smart grid search" where each particle performs a "stupid search" and yet by communicating with other particles and by having a large number of such particles we can reach the global optimum quickly.

The PSO algorithm can be formally defined as follows. For $i = 1, \cdots, m$

[6]Similar to PSO, an ACO (ant colony optimization) by Dorigo, Bonabeau, and Theraulaz (2000) and ACS (ant colony system) by Dorigo and Gambardella (1997) are both based upon swarm intelligence. The first ant system is first developed by Dorigo, Maniezzo, and Colorni (1991) and then popularized by Dorigo, Maniezzo, and Colorni (2000).

particles and each particle is a vector of $j = 1, \cdots, n$ dimensions, we have:

$$\begin{cases} v_{i,j}(t+1) = w(t)v_{i,j}(t) + r_1c_1(p_{i,j}(t) - x_i(t)) + r_2c_2(g_j(t) - x_{i,j}(t)) \\ x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \end{cases} \tag{3.12}$$

where $v_{i,j}(t)$ is velocity of the i-th particle in the j-th dimension at time t; $x_{i,j}(t)$ is position of the i-th particle in the j-th dimension at time t; $w(t)$ is a "weight" (less than 1) which decides how the current velocity will be carried over to the next period (and usually it is set as $w(t) = \alpha w(t-1)$ and $\alpha < 1$ to introduce diminishing velocity);[7] and finally $r_1, r_2 \sim u(0,1)$ follow a uniform distribution.

In the swam literature, $w(t)\vec{v}_i(t)$ is called inertia; $r_1c_1(\vec{p}_i(t) - \vec{x}_i(t))$ is called the cognitive component and $r_2c_2(\vec{g}(t) - \vec{x}_i(t))$ is called the social component. Coefficients and are known as acceleration coefficients.

At each position there is "cost function" $f(\cdot)$ (sometimes called distance function) at which a "cost" (or penalty) is computed. This cost function is the objective function to be minimized (or maximized).

The global best at any given time is either the maximum or minimum value of the objective function generated by all particles at the time:

$$\vec{g}(t) = \min_i\{f(\vec{p}_i(t))\} \tag{3.13}$$

and the personal best at the time is:

$$\vec{p}_i(t) = \min_t\{f(\vec{x}_i(t))\} \tag{3.14}$$

and $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the "fitness function". The usual fitness function is:

$$f(\vec{x}_i(t)) = \left\|x_i - \underline{\chi}\right\| = \sum_{j=1}^{J} (x_{ij} - \chi_j)^2 \tag{3.15}$$

where $\underline{\chi} =< \chi_1, \cdots, \chi_J >$ is a coordinate in an n-dimensional space.

Later, we illustrate via a very simple example how the process is so easily implemented.

As we can see, the algorithm (at least the standard one presented here) of PSO is quite different from that of a generic swarm by Reynolds (1978). Yet they both share the same behavioral pattern of a natural swarm. In other words, (1) both PSO and the generic swarm are based upon peer-to-peer communication in order to

[7] The reason is that as a particle is approaching the global best, the velocity should approach 0 (i.e. the particle should no longer move at the global optimum.)

achieve the objective and (2) the particles in both PSO and the generic swarm are identical (like birds or ants) and each particle follows its neighbor particles. The difference is just how each particle weighs its neighbors. In PSO, each particle only cares of the global best discovered by its neighbors and in the generic swarm each neighbor's position is important.

3.3.1 A Demonstration

Let's begin with an exercise of bottom-seeking of a lake. A school of fish is thrown into the lake to seek the bottom of the lake. If the shape of the lake is smooth (as in Figure 3.4). Then a quick Newton's method can lead to the bottom quicly and accurately. However, if the lake is shaped as in Figure 3.7, then the efficient Newton's method won't work. But a grid search can still do the job. Yet imagine the "lake" is an n-dimensional hypercube, then grid searches are not feasible. Here is how PSO can help.

A school of fish is thrown into the lake and try to find the bottom of the lake. Each fish is communicate to other fish via the global best. Each fish is also trying to stick to its own personal (local) best. Finally, each fish, while considering to move toward the global best and not move too far away from its personal best, also explores the neighborhood of its current position. The final movement is the weighted average of the three forces. See Figures **??** and **??**.

PSO is proven to be highly efficient in high dimensional searches. We can think of PSO as a smart grid search. It takes advantage of fish's "intelligence" (i.e. exploration and exploitation) to locate the global optimum.

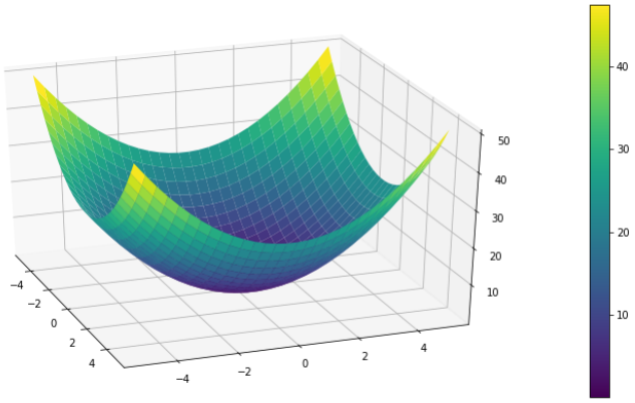


Figure 3.4: No Local Optima

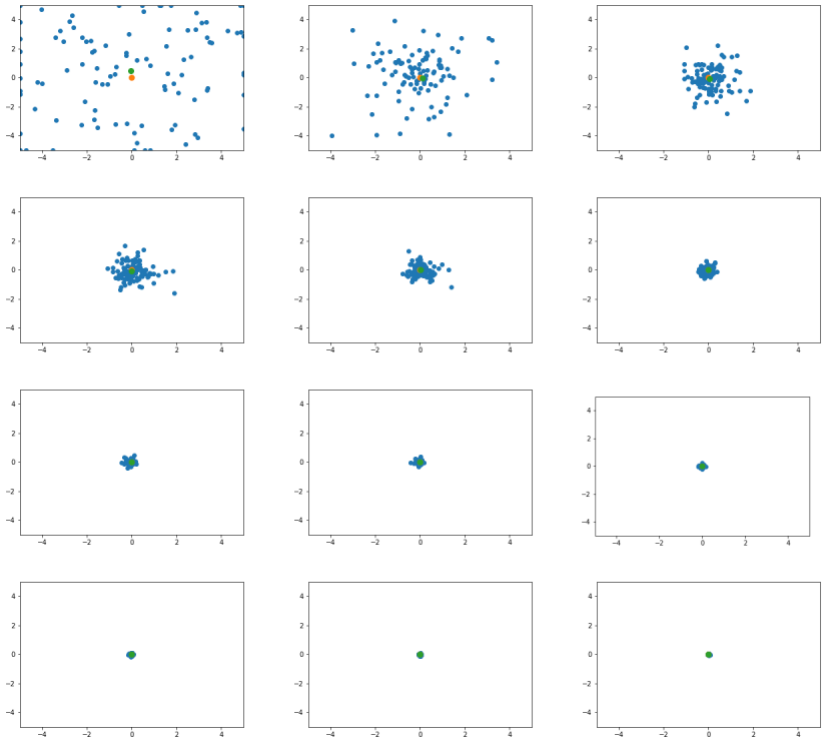


Figure 3.5: Fish Movements

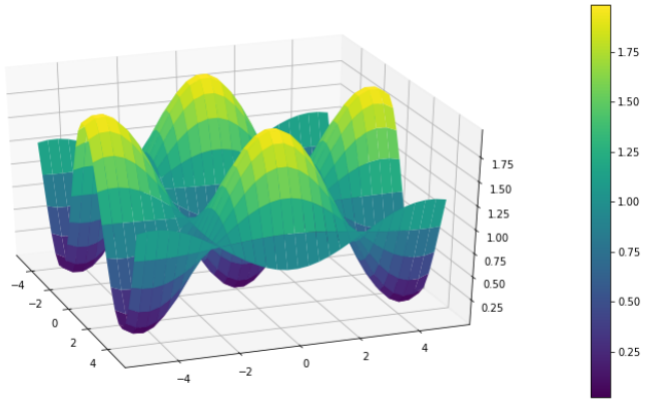


Figure 3.6: Multiple Local Optima

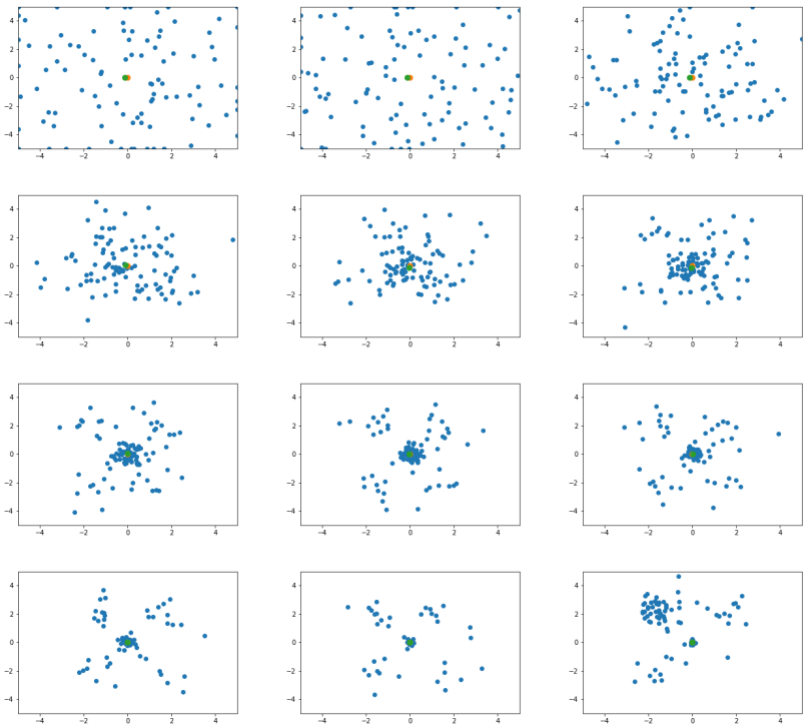


Figure 3.7: Fish Movements

3.3.2 Different Types of PSO

The literature on PSO is voluminous. Zhang, Wang, and Ji (2015) provide an excellent survey. They classify the existing PSO literature into the following strands:[8]

- modifications,[9]
- population topology,[10]
- hybridization,[11]
- extensions,[12]
- theoretical analysis,[13] and
- parallel implementation.[14]

However, Zhang, Wang, and Ji only provide applications in non-financial areas.[15][16] To date, there have been very limited number of applications in the area of finance. Within the limited literature, most noticeably is in the area of portfolio selection.[17]

[8]PSO can also vary in terms of parameterization such as center mass (see Jamous, Tharwat, Seidy, and Bayoumi (2015)).

[9]This includes quantum-behaved PSO, bare-bones PSO, chaotic PSO, and fuzzy PSO.

[10]This includes von Neumann, ring, star, random, among others.

[11]This is to combine PSO with genetic algorithm, simulated annealing, Tabu search, artificial immune system, ant colony algorithm, artificial bee colony, differential evolution, harmonic search, and biogeography-based optimization.

[12]This includes multi-objective, constrained, discrete, and binary optimization.

[13]This includes parameter selection and tuning, and convergence analysis.

[14]This involves multi-core, multiprocessor, GPU, and cloud computing forms.

[15]They are electrical and electronic engineering, automation control systems, communication theory, operations research, mechanical engineering, fuel and energy, medicine, chemistry, and biology.

[16]Kumar et. al. (2013) examines performance of various PSO algorithms: Canonical PSO, Hierarchical PSO (HPSO), Time varying acceleration coefficient (TVAC) PSO, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients (HPSO-TVAC), Stochastic inertia weight (Sto-IW) PSO and Time varying inertia weight (TVIW) PSO have been used for comparative study. These versions of PSO vary in only parameterization.

[17]See Huang (2019) for a survey.

3.3.3 Applications

§ Option Pricing

Evaluating American-style derivatives is a challenging task. In a uni-variate setting (e.g. option on one stock), lattice models – either the binomial model (e.g. Cox, Ross, and Rubinstein (1979)) or finite difference methods (e.g. see Hull (2015)) are an efficient method. However, once the derivative contract is written on multiple assets (e.g. exchange options), lattice models would become infeasible (with regard to both computation time or memory space). Furthermore, path-dependent derivatives cannot be evaluated with lattice models.

As a result, modifying the Monte Carlo method to evaluate American-style derivatives is a popular alternative. There are two approaches to achieve this goal. The first approach is proposed by Longstaff and Schwartz (2001) who approximate the continuation value of the option by a regression function (functional form can be arbitrary). They recognize that the early exercise decision is merely a comparison of the exercise value and its continuation value of the option. If the continuation value can be reasonably accurately estimated, then the early exercise problem can be easily solved and hence one can readily compute the value of an American-style derivative. The drawback of this approach is apparent – it is hard to know in advance which functional form of the regression will provide an accurate estimate for the continuation value.

The other approach is to recognize that derivatives pricing in general is a free-boundary PDE (partial differential equation) problem. If we can accurately estimate the exercise boundary, then it is just an easy integration over the boundary (as the first passage time problem). In other words, if we can accurately estimate the boundary, then the value of an American-style derivative can be calculated as it would be a barrier option. This approach is more computational efficient than the Longstaff-Schwartz model; yet if suffers the same drawback of the Longstaff-Schwartz model – the accuracy of the American-style derivative value relies upon an accurate exercise boundary. Moreover, the literature of this approach lacks the evidence on derivatives on multiple assets.

In this section, we introduce an artificial intelligence method, i.e. swarm intelligence, to locate the optimal exercise boundary. In particular, we use an optimization algorithm within the realm of swarm intelligence named PSO (particle swarm optimization) to locate the optimal exercise boundary. The intelligence by the swarm can efficiently decide piece-wise values of the boundary, one for each time step, without an approximated functional form as in the literature. As in any artificial intelligence model, PSO is efficient in high dimensional optimization problems. In the case of a truly free boundary (i.e. piece-wise), we find that PSO can ideally provide the best

solution to complex (e.g. American-style, multi-asset, path-dependent) derivatives problems.

As Carr (1998), among others, points out, if we solve an American-style derivative premium as a free-boundary problem, then we can use an explicit boundary function and the American-style derivative premium is simply an integration of payoff function (e.g. put) over the boundary.

$$\xi(t) = \mathbb{E}_t^Q \left[e^{-r\tau} \max\{[E(\tau), 0\} \right] \tag{3.16}$$

where $E(\tau)$ is the exercise value at the stopping time τ. If it is a put option without dividends which is the case in this paper, then $E(\tau) = K - S(\tau)$. On the boundary, $S(\tau) = B(\tau)$ and hence $E(\tau) = K - B(\tau)$ where $B(\tau)$ is the boundary function given exogenously. The way the boundary function works is that it serves as a stopping time. Once the stock price at time t hits the boundary $B(t)$, the process stops and the option will be exercised and paid and hence the American-style derivative can be evaluated as a barrier option.

The easiest way to perform the integration is through Monte Carlo simulations. As the derivative price is given as an expected value:

$$\xi(t) = \frac{1}{N} \sum_{j=1}^{N} e^{-r\tau_j} \max\{K - B(\tau_j), 0\} \tag{3.17}$$

We note that the recursively determined boundary function (via a lattice model) maximizes the option value, any other exogenously specified boundary function will only be "sub-optimal", that is, generating a lower value than the lattice model. This sub-optimal argument is convenient in that now we can simply try a large number of boundary functions and use the one that generates the highest option value as a good approximation.

Researchers then have tried various approximations on the exercise boundary. These approximations are explicit functions and hence can be easily integrated (and hence American-style derivative value solved for). According to a recent survey by Nunes (2008), the literature has the following functional forms:

- Constant: $B(t) = a_0$
- Linear: $B(t) = a_0 + a_1 t$
- Exponential: $B(t) = a_o e^{a_1 t}$
- Exponential-constant: $a_0 + e^{a_1 t}$
- Polynomial: $B(t) = \sum_{i=1}^{n} a_i t^{i-1}$

- Carr-Jarrow-Myneni (2008): $B(t) = \min(K, \frac{r}{q}K)e^{-a\sqrt{T-t}} + E_\infty \left[1 - e^{-a\sqrt{T-t}}\right]$

Note that the boundary is not a function of the stock price (i.e. free boundary problem). Since these boundary functions are explicit, they can be easily integrated. Certainly the accuracy of the American value depends on the accuracy of the approximated boundary function. The problem of this approach is that there is no consensus of which functional form of the boundary can consistently be the best. Often it varies with the parameters of the option (i.e. moneyness, interest rate, time to maturity, and volatility). As a result, no conclusion can be drawn on a particular functional form.

See paper.

§ Swing Contract

A swing contract allows its holder to purchase a flexible quantity of the underlying commodity at a fixed price and a fixed future date subject to local (e.g. daily) and global (e,g. all-time) storage constraints. This flexibility is an attractive choice for example for natural gas consumers and matches their risk profile better than the previously mentioned hedging alternatives. For example a swing contract could allow its holder to buy a certain amount of gas on a daily basis throughout the winter (daily decision within daily constraints) subject to a total consumption limit. Also, at each day, the buyer of the swing contract can also consume any quantity of natural gas (but up to the storage limit). As a result, it is a derivative contract that must be balanced between price and quantity in order to maximize the profit (or minimize the cost).

Swing contracts are commonly traded over-the-counter and allow their holders to purchase a flexible quantity of the underlying asset at fixed exercise dates subject to local and global constraints. For example a power producer relying on wind to generate power might have entered an agreement to deliver a certain amount to power on a monthly basis. In case wind does not produce sufficient amounts of energy they would be able to hedge their delivery risk buy buying a swing contract on power. In the same way, a gas consumer might require volume flexibility throughout the winter in order to balance their delivery risk.

Figure 3.8 depicts the inventory limitation.

See paper.

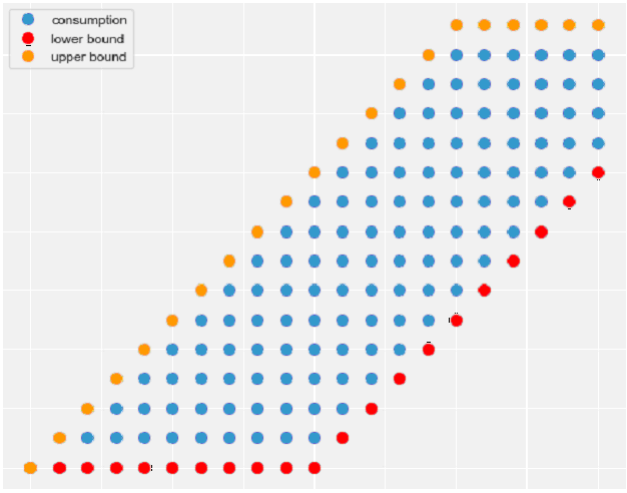


Figure 3.8: Swing Contract

§ Portfolio Optimization

The efficient frontier (EF) can be achieved by:

$$\begin{aligned} \max \quad & \mu_P = \underline{w}'\underline{\mu}_P \\ & \text{subject to} \\ & \sigma_P^2 = \frac{1}{2}\underline{w}'\Omega\underline{w} \end{aligned} \tag{3.18}$$

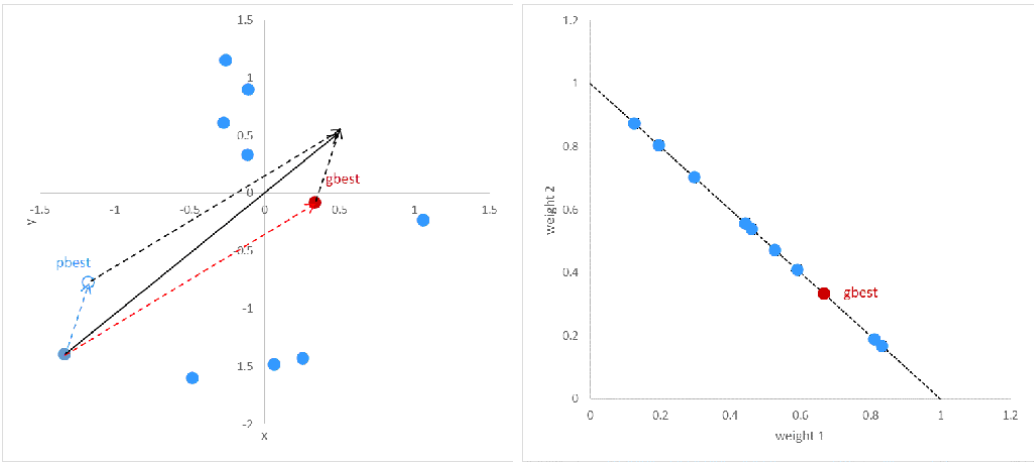
or

$$\begin{aligned} \min \quad & \sigma_P^2 = \frac{1}{2}\underline{w}'\Omega\underline{w} \\ & \text{subject to} \\ & \mu_P = \underline{w}'\underline{\mu}_P \\ & 1 = \underline{w}'\underline{1} \end{aligned} \tag{3.19}$$

We know that there exists a closed-form solution as follows:

$$\underline{\hat{w}} = \frac{\underline{1}'\Omega^{-1}\underline{1}\mu_P - \underline{1}'\Omega^{-1}\underline{\mu}}{(\underline{\mu}'\Omega^{-1}\underline{\mu})(\underline{1}'\Omega^{-1}\underline{1}) - (\underline{1}'\Omega^{-1}\underline{\mu})^2}\Omega^{-1}\underline{\mu} + \frac{-\underline{1}'\Omega^{-1}\underline{\mu}\mu_P + \underline{\mu}'\Omega^{-1}\underline{\mu}}{(\underline{\mu}'\Omega^{-1}\underline{\mu})(\underline{1}'\Omega^{-1}\underline{1}) - (\underline{1}'\Omega^{-1}\underline{\mu})^2}\Omega^{-1}\underline{1} \tag{3.20}$$

But if we seek this solution numerically, we can achieve it by using PSO. See Figure 3.9. Let $\underline{w}$ be positions of the global best. We wish this global best (whichever fish) to be the correction solution as in equation (3.20).



(a) A Generic PSO (b) PSO for Portfolio Weights

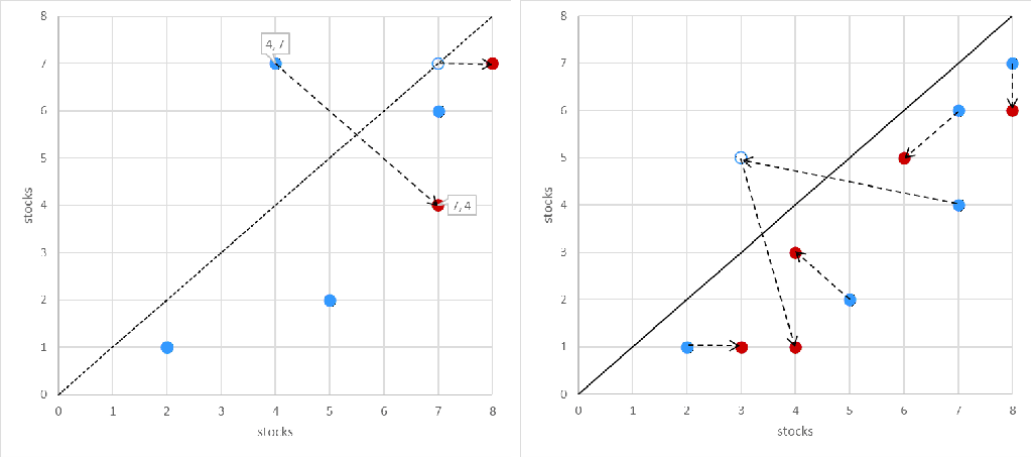
Figure 3.9: Two-Dimensional PSO

§ Stock Selection

We can also use PSO for selecting stocks. Let there be n stocks to choose from and we shall select K stocks. Now, positions in PSO will be digital (0/1) where 0 representing no-select and 1 representing select. This is known as binary PSO or digital PSO. While there are efficient methods for binary PSO, for our purpose, we can easily modify the standard PSO for the task (although may not be most efficient).

Figure 3.10 demonstrates how we can modify the standard PSO for digital choices.

In this example, we choose 2 out of 8 stocks to form a tracking portfolio. There are 5 particles in PSO labeled by blue dots (four solid dots and one hollow dot). Each particle carries two coordinates representing two stocks (by integers). For example, the lower left dot is (1,1) representing that this particle chooses the same stock (#1) twice. This is not allowed since a stock cannot be repeatedly chosen. Hence this position is not allowed and should be discarded. Another same example is (7,7) which is not allowed since stock #7 cannot be chosen repeatedly. The other three solid dots are acceptable – (5,2), (7,6), and (4,7). We note that the selection of stocks has no order and hence (4,7) and (7,4) which is marked by solid red, are identical. As a result, we can easily limit the domain space to be a half triangle (in a multi-dimensional space it would be a hyper-pyramid). Doing so is also more time efficient since there is no need to check repeats (which is m^n where m is the number of stocks chosen and n is the total number of stocks in the index). A hyper-pyramid is in the order of $n!$.



(a) A Generic PSO (b) PSO for 0/1 Selection

Figure 3.10: Two PSOs

To move from the current positions to the next positions, in a case where a particle is moved outside of the triangle, we need to let it "bounce back". The rule to reflect a newly permissible position, as marked in the above diagram is let it reflect on the 45° line. In the example, the particle (7,4) is intended to move to (4,5) which is outside of the permissible triangle. We let it reflect back to (4,1), the arrows indicate. The other particles move within the boundary and hence need not be adjusted.

3.4 Estimating Swarm

Note that PSO is an optimization tool. It is not a behavior model. Rather, we take advantage of the very nature of swarm to help us seek the global optimum. This is no different from using genetic algorithm, neural network, or reinforcement learning (to be introduced in later chapters
— 5, 8, and 10) for solving an optimization problem.

Swarm intelligence itself (or boids) is a behavioral model. By specifying various hyper parameters, we can control the movements of a swarm. This can be useful for studying the well documented behavior in investments in finance – known as herding.

When applied to data, swarm intelligence is used backwards. Since data give us positions already and we need to recover the parameters. Note that here, we can easily combine boids and PSO. For example, leader and memory are the main features of PSO and can be incorporated in a boids.

Now,

$$\min_{\underline{w}} \sum_{j=1}^{n} (v_{j,t}^{(i)} - \nu_{j,t}^{(i)})^2 \tag{3.21}$$

where $v_{j,t}^{(i)}$ is model velocity and $\nu_{j,t}^{(i)}$ is data velocity. From data, we know the positions $x_{j,t}^{(i)}$. By subtracting two consequent positions we can get velocity and then we can match the observed velocity with the formula and then solve for the hyper parameters $\underline{w}$.

3.4.1 Memory

Without alignment, cohesion, and leader,

$$\vec{v}_t^{(i)} = \vec{v}_{P,t}^{(i)} \tag{3.22}$$

where

$$\begin{aligned} \vec{v}_{P,t}^{(i)} &= (1 - \alpha_{P,t}^{(i)})\vec{u}_t^{(i)} + \alpha_{P,t}^{(i)}(\vec{p}_t^{(i)} - \vec{x}_t^{(i)}) \\ &= \vec{u}_t^{(i)} + \alpha_{P,t}^{(i)} \left(\vec{p}_t^{(i)} - \vec{x}_t^{(i)} - \vec{u}_t^{(i)} \right) \end{aligned} \tag{3.23}$$

where $\vec{p}_t^{(i)}$ is fish i's personal best.

Let's first take the partial derivative w.r.t. $\alpha_P^{(i)}$:

$$\frac{\partial v_{P,j,t}^{(i)}}{\partial \alpha_P^{(i)}} = p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}$$

The objective is to solve for α_t across all topics. Using the optimization equation (3.21), taking the derivative, and setting it to 0, we get:

$$\frac{\partial}{\partial \alpha_{P,t}^{(i)}} \sum_{j=1}^{n} (v_{j,t}^{(i)} - \nu_{j,t}^{(i)})^2 = \sum_{j=1}^{n} 2(v_{j,t}^{(i)} - \nu_{j,t}^{(i)}) \frac{\partial v_{j,t}^{(i)}}{\partial \alpha_{P,t}^{(i)}} = 0$$

Hence, the solution to $\alpha_{P,t}^{(i)}$ is:

$$0 = \sum_{j=1}^{n} (v_{j,t}^{(i)} - \nu_{j,t}^{(i)})(p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}) \tag{3.24}$$

which can be rearranged as:

$$\sum_{j=1}^{n} v_{j,t}^{(i)}(p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}) = \sum_{j=1}^{n} \nu_{j,t}^{(i)}(p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}) \tag{3.25}$$

Further rearrangements as follows:

$$\sum_{j=1}^{n}(u_{j,t}^{(i)}+\alpha_{P,t}^{(i)}(\hat{p}_{j,t}^{(i)}-p_{j,t}^{(i)}-u_{j,t}^{(i)}))(p_{j,t}^{(i)}-x_{j,t}^{(i)}-u_{j,t}^{(i)})=\sum_{j=1}^{n}\nu_{j,t}^{(i)}(p_{j,t}^{(i)}-x_{j,t}^{(i)}-u_{j,t}^{(i)})$$

and

$$\alpha_{P,t}^{(i)}\sum_{j=1}^{n}(p_{j,t}^{(i)}-x_{j,t}^{(i)}-u_{j,t}^{(i)})(p_{j,t}^{(i)}-x_{j,t}^{(i)})=\sum_{j=1}^{n}(\nu_{j,t}^{(i)}-u_{j,t}^{(i)})(p_{j,t}^{(i)}-x_{j,t}^{(i)}-u_{j,t}^{(i)}) \tag{3.26}$$

And the final solution can be shown as:

$$\alpha_{P,t}^{(i)}=\frac{\sum_{j=1}^{n}(\nu_{j,t}^{(i)}-u_{j,t}^{(i)})(p_{j,t}^{(i)}-x_{j,t}^{(i)}-u_{j,t}^{(i)})}{\sum_{j=1}^{n}(p_{j,t}^{(i)}-x_{j,t}^{(i)}-u_{j,t}^{(i)})^2} \tag{3.27}$$

Because $u_{j,t}^{(i)}$ is random, $\alpha_{P,t}^{(i)}$ is random. To calculate $\mathbb{E}[\alpha_{P,t}^{(i)}]$, we must simulate $u_{j,t}^{(i)}$ a number of times.

3.4.2 Leader

Leader alone, we have two versions.

$$\vec{v}_t^{(i)}=\begin{cases}\vec{v}_{L,t}^{(i)}\\ v_{L,t}\end{cases} \tag{3.28}$$

where

$$\vec{v}_{L,t}^{(i)}=\begin{cases}\alpha_L(\vec{g}_{t-1}-\vec{x}_{t-1}^{(i)})\\ \alpha_L^{(i)}(\vec{g}_{t-1}-\vec{x}_{t-1}^{(i)})\end{cases}$$

where the former is each firm has its own $\alpha_{L,t}^{(i)}$ and the latter is all firms share the same $\alpha_{L,t}$. In the first case, $\alpha_{L,t}^{(i)}$. Taking derivative w.r.t. $\alpha_{L,t}^{(i)}$:

$$\frac{\partial v_{j,t}^{(i)}}{\partial\alpha_{L,t}^{(i)}}=g_{j,t}^{(i)}-x_{j,t}^{(i)} \tag{3.29}$$

Differentiate the objective function (3.21) with respect to $\alpha_{L,t}^{(i)}$:

$$\frac{\partial}{\partial\alpha_{L,t}^{(i)}}\sum_{j=1}^{n}(v_{j,t}^{(i)}-\nu_{j,t}^{(i)})^2=\sum_{j=1}^{n}2(v_{j,t}^{(i)}-\nu_{j,t}^{(i)})\frac{\partial v_{j,t}^{(i)}}{\partial\alpha_{L,t}^{(i)}}=0 \tag{3.30}$$

Rearranging terms:

$$\begin{gathered}\sum_{j=1}^{n} v_{j,t}^{(i)}(g_{j,t} - x_{j,t}^{(i)}) = \sum_{j=1}^{n} \nu_{j,t}^{(i)}(g_{j,t} - x_{j,t}^{(i)}) \\ \text{and} \\ \sum_{j=1}^{n} \alpha_{L,t}^{(i)}(g_{j,t}^{(i)} - x_{j,t}^{(i)})(g_{j,t}^{(i)} - x_{j,t}^{(i)}) = \sum_{j=1}^{n} \nu_{j,t}^{(i)}(g_{j,t} - x_{j,t}^{(i)})\end{gathered} \tag{3.31}$$

Finally, the solution to $\alpha_{L,t}^{(i)}$ is:

$$\alpha_{L,t}^{(i)} = \frac{\sum_{j=1}^{n} \nu_{j,t}^{(i)}(g_{j,t}^{(i)} - x_{j,t}^{(i)})}{\sum_{j=1}^{n} (g_{j,t}^{(i)} - x_{j,t}^{(i)})^2} \tag{3.32}$$

In the second case $\alpha_{L,t}$.

$$\frac{\partial}{\partial \alpha_{L,t}} \sum_{i=1}^{m} \sum_{j=1}^{n} (v_{j,t}^{(i)} - \nu_{j,t}^{(i)})^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} 2(v_{j,t}^{(i)} - \nu_{j,t}^{(i)}) \frac{\partial v_{j,t}}{\partial \alpha_t} = 0 \tag{3.33}$$

Rearranging terms:

$$\begin{gathered}\sum_{i=1}^{m} \sum_{j=1}^{n} v_{j,t}^{(i)} \frac{\partial v_{j,t}}{\partial \alpha_{L,t}} = \sum_{i=1}^{m} \sum_{j=1}^{n} \nu_{j,t}^{(i)} \frac{\partial v_{j,t}}{\partial \alpha_{L,t}} \\ \text{and} \\ \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{L,t}(g_{j,t} - x_{j,t}^{(i)})(g_{j,t}^{(i)} - x_{j,t}^{(i)}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \nu_{j,t}^{(i)}(g_{j,t}^{(i)} - x_{j,t}^{(i)})\end{gathered} \tag{3.34}$$

The solution is:

$$\alpha_{L,t} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \nu_{j,t}^{(i)}(g_{j,t}^{(i)} - x_{j,t}^{(i)})}{\sum_{i=1}^{m} \sum_{j=1}^{n} (g_{j,t}^{(i)} - x_{j,t}^{(i)})^2} \tag{3.35}$$

Compare to the average of $\alpha_{L,t}^{(i)}$.

3.4.3 Combine Leader and Memory

Now we have,

$$\vec{v}_t^{(i)} = w\vec{v}_{P,t}^{(i)} + (1-w)\vec{v}_{L,t}^{(i)}$$

This leads to the following:

$$\begin{aligned}
&\vec{v}_{L,t}^{(i)} = \alpha_{L,t}^{(i)}(\vec{g}_{t-1} - \vec{x}_{t-1}^{(i)}) && \vec{v}_{L,t}^{(i)} = \alpha_{L,t}^{(i)}(\vec{g}_{t-1} - \vec{x}_{t-1}^{(i)}) \\
&\vec{v}_{P,t}^{(i)} = \vec{u}_t^{(i)} + \alpha_P^{(i)}(\vec{p}_t^{(i)} - \vec{x}_{t-1}^{(i)} - \vec{u}_t^{(i)}) && \vec{v}_{P,t}^{(i)} = \vec{u}_t^{(i)} + \alpha_P^{(i)}(\vec{p}_t^{(i)} - \vec{x}_{t-1}^{(i)} - \vec{u}_t^{(i)}) \\
&\frac{\partial v_{P,j,t}^{(i)}}{\partial \alpha_{P,t}^{(i)}} = p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}; \quad \frac{\partial v_{L,j,t}^{(i)}}{\partial \alpha_{P,t}^{(i)}} = 0 && \frac{\partial v_{L,j,t}^{(i)}}{\partial \alpha_{L,t}^{(i)}} = g_{j,t-1} - x_{j,t-1}^{(i)}; \quad \frac{\partial v_{P,j,t}^{(i)}}{\partial \alpha_{L,t}^{(i)}} = 0
\end{aligned} \tag{3.36, 3.37}$$

Now, differentiate the objective function (3.21) and substituting in (3.36), $\alpha_{P,t}^{(i)}$ first, to get:

$$\begin{aligned}
&\frac{\partial}{\partial \alpha_{P,t}^{(i)}} \sum_{j=1}^{n} [w v_{P,j,t}^{(i)} + (1-w) v_{L,j,t}^{(i)} - \nu_{j,t}]^2 \\
&= \sum_{j=1}^{n} 2[w v_{P,j,t}^{(i)} + (1-w) v_{L,j,t}^{(i)} - \nu_{j,t}] \frac{\partial v_{P,j,t}^{(i)}}{\partial \alpha_{P,t}^{(i)}} \\
&= \sum_{j=1}^{n} 2[w v_{P,j,t}^{(i)} + (1-w) v_{L,j,t}^{(i)} - \nu_{j,t}][p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}] = 0
\end{aligned} \tag{3.38}$$

and next $\alpha_{L,t}^{(i)}$ to get (substituting in (3.37)):

$$\begin{aligned}
&\frac{\partial}{\partial \alpha_{L,t}^{(i)}} \sum_{j=1}^{n} [w v_{P,j,t}^{(i)} + (1-w) v_{L,j,t}^{(i)} - \nu_{j,t}]^2 \\
&= \sum_{j=1}^{n} 2[w v_{P,j,t}^{(i)} + (1-w) v_{L,j,t}^{(i)} - \nu_{j,t}] \frac{\partial v_{L,j,t}^{(i)}}{\partial \alpha_{L,t}^{(i)}} \\
&= \sum_{j=1}^{n} 2[w v_{P,j,t}^{(i)} + (1-w) v_{L,j,t}^{(i)} - \nu_{j,t}][g_{j,t-1} - x_{j,t-1}^{(i)}] = 0
\end{aligned} \tag{3.39}$$

Solving $\alpha_{P,t}^{(i)}$ first to get (let $z_{j,t}^{(i)} = p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)}$):

$$\begin{aligned}
&\sum_{j=1}^{n} \left\{ w u_{j,t}^{(i)} z_{j,t}^{(i)} + w \alpha_{P,t}^{(i)} (z_{j,t}^{(i)})^2 + (1-w) \vec{v}_{L,j,t}^{(i)} z_{j,t}^{(i)} - \nu_{j,t} z_{j,t}^{(i)} \right\} = 0 \\
&\qquad \text{and} \\
&w \alpha_{P,t}^{(i)} \sum_{j=1}^{n} (z_{j,t}^{(i)})^2 = \sum_{j=1}^{n} \left\{ \nu_{j,t} z_{j,t}^{(i)} - w u_{j,t}^{(i)} z_{j,t}^{(i)} - (1-w) v_{L,j,t}^{(i)} z_{j,t}^{(i)} \right\}
\end{aligned} \tag{3.40}$$

The result is:

$$\alpha_{P,t}^{(i)} = \frac{\sum_{j=1}^{n} \left\{ \nu_{j,t} z_{j,t}^{(i)} - w u_{j,t}^{(i)} z_{j,t}^{(i)} - (1-w) v_{L,j,t}^{(i)} z_{j,t}^{(i)} \right\}}{w \sum_{j=1}^{n} (z_{j,t}^{(i)})^2} \tag{3.41}$$

Now, $\alpha_{L,t}^{(i)}$:

$$\sum_{j=1}^{n}[wv_{P,j,t}^{(i)} + (1-w)\alpha_{L,t}^{(i)}(\vec{g}_{t-1} - \vec{x}_{t-1}^{(i)}) - \nu_{j,t}](g_{j,t-1} - x_{j,t-1}^{(i)}) = 0$$

and

$$\sum_{j=1}^{n}(1-w)\alpha_{L,t}^{(i)}(\vec{g}_{t-1} - \vec{x}_{t-1}^{(i)})(g_{j,t-1} - x_{j,t-1}^{(i)})$$
$$= \sum_{j=1}^{n}[\nu_{j,t} - wv_{P,j,t}^{(i)}](g_{j,t-1} - x_{j,t-1}^{(i)}) \tag{3.42}$$

and

$$(1-w)\alpha_{L,t}^{(i)}\sum_{j=1}^{n}(\vec{g}_{t-1} - \vec{x}_{t-1}^{(i)})^2$$
$$= \sum_{j=1}^{n}(\nu_{j,t} - w(u_{j,t}^{(i)} + \alpha_P^{(i)}(p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)})))(g_{j,t-1} - x_{j,t-1}^{(i)})$$

The result is:

$$\alpha_{L,t}^{(i)} = \frac{\sum_{j=1}^{n}(\nu_{j,t} - w(u_{j,t}^{(i)} + \alpha_P^{(i)}(p_{j,t}^{(i)} - x_{j,t}^{(i)} - u_{j,t}^{(i)})))(g_{j,t-1} - x_{j,t-1}^{(i)})}{(1-w)\sum_{j=1}^{n}(\vec{g}_{t-1} - \vec{x}_{t-1}^{(i)})^2} \tag{3.43}$$

Now we combine the two results together. First $\alpha_{P,t}^{(i)}$ as follows:

$$\begin{aligned}\alpha_{P,t}^{(i)} &= \frac{\sum_{j=1}^{n}\left\{\nu_{j,t}(p_{j,t}^{(i)} - x_{j,t}^{(i)}) - (1-w)\alpha_{L,t}^{(i)}(g_{j,t-1} - x_{j,t-1}^{(i)})(p_{j,t}^{(i)} - x_{j,t}^{(i)})\right\}}{w\sum_{j=1}^{n}(z_{j,t}^{(i)})^2} \\ &= \frac{\sum_{j=1}^{n}\nu_{j,t}(p_{j,t}^{(i)} - x_{j,t}^{(i)})}{w\sum_{j=1}^{n}(p_{j,t}^{(i)} - x_{j,t}^{(i)})^2} - \alpha_{L,t}^{(i)}\frac{(1-w)\sum_{j=1}^{n}(g_{j,t-1} - x_{j,t-1}^{(i)})(p_{j,t}^{(i)} - x_{j,t}^{(i)})}{w\sum_{j=1}^{n}(p_{j,t}^{(i)} - x_{j,t}^{(i)})^2} \\ &= \pi_1 - \pi_2\alpha_{L,t}^{(i)}\end{aligned} \tag{3.44}$$

then $\alpha_{L,t}^{(i)}$ as follows:

$$\begin{aligned}\alpha_{L,t}^{(i)} &= \frac{\sum_{j=1}^{n}(\nu_{j,t} - w\alpha_P^{(i)}(p_{j,t}^{(i)} - x_{j,t}^{(i)}))(g_{j,t-1} - x_{j,t-1}^{(i)})}{(1-w)\sum_{j=1}^{n}(g_{j,t-1} - x_{j,t-1}^{(i)})^2} \\ &= \frac{\sum_{j=1}^{n}\nu_{j,t}(g_{j,t-1} - x_{j,t-1}^{(i)})}{(1-w)\sum_{j=1}^{n}(g_{j,t-1} - x_{j,t-1}^{(i)})^2} - \alpha_P^{(i)}\frac{w\sum_{j=1}^{n}(p_{j,t}^{(i)} - x_{j,t}^{(i)})(g_{j,t-1} - x_{j,t-1}^{(i)})}{(1-w)\sum_{j=1}^{n}(g_{j,t-1} - x_{j,t-1}^{(i)})^2} \\ &= \rho_1 - \rho_2\alpha_P^{(i)}\end{aligned} \tag{3.45}$$

Hence, we have a simultaneous equation system. Solving it, we get:

$$\alpha_{L,t}^{(i)} = \frac{\rho_1 - \rho_2\pi_1}{1 - \rho_2\pi_2} \qquad (3.46) \qquad\qquad \alpha_{P,t}^{(i)} = \frac{\pi_1 - \pi_2\rho_1}{1 - \pi_2\rho_2} \qquad (3.47)$$

Lastly,

$$\vec{v}_t^{(i)} = w_t^{(i)}\vec{v}_{P,t}^{(i)} + (1 - w_t^{(i)})\vec{v}_{L,t}^{(i)}$$

and

$$\frac{\partial \vec{v}_t^{(i)}}{\partial w_t^{(i)}} = \vec{v}_{P,t}^{(i)} - \vec{v}_{L,t}^{(i)}$$

Minimizing SSE, equation (3.21):

$$\begin{aligned}
&\frac{\partial}{\partial w_t^{(i)}} \sum_{j=1}^{n} (v_{j,t} - \nu_{j,t})^2 \\
&= \sum_{j=1}^{n} 2(v_{j,t} - \nu_{j,t}) \frac{\partial v_{j,t}}{\partial w_t^{(i)}} \\
&= \sum_{j=1}^{n} 2(v_{j,t} - \nu_{j,t})(v_{P,j,t} - v_{L,j,t}) \\
&= \sum_{j=1}^{n} (v_{j,t} - \nu_{j,t})\{(u_{j,t} + \alpha_{P,t}^{(i)}(p_{j,t} - x_{j,t} - u_{j,t})) - \alpha_{L,t}^{(i)}(g_{j,t} - x_{j,t})\} = 0
\end{aligned}$$

Solve (1), (2), and (3) simultaneously for $\alpha_{P,t}^{(i)}$, $\alpha_{L,t}^{(i)}$ and $w_t^{(i)}$. If $\alpha_{P,t}^{(i)} = 1$, then

$$\sum_{j=1}^{n} (v_{j,t} - \nu_{j,t})\{(p_{j,t} - x_{j,t}) - \alpha_{L,t}^{(i)}(g_{j,t} - x_{j,t})\} = 0$$

which gives the solution:

$$\alpha_{L,t}^{(i)} = \frac{\sum_{j=1}^{n} (v_{j,t} - \nu_{j,t})\{(p_{j,t} - x_{j,t})}{\sum_{j=1}^{n} (g_{j,t} - x_{j,t})}$$

which does not rely on w. We know from above the solution to $\alpha_{L,t}^{(i)}$:

$$\alpha_{L,t}^{(i)} = \frac{\sum_{j=1}^{n} (\nu_{j,t} - w\alpha_P^{(i)}(p_{j,t}^{(i)} - x_{j,t}^{(i)}))(g_{j,t-1} - x_{j,t-1}^{(i)})}{(1 - w)\sum_{j=1}^{n} (g_{j,t-1} - x_{j,t-1}^{(i)})^2}$$

Hence we can solve for w:

$$\begin{aligned}
\alpha_{L,t}^{(i)} &= \frac{\sum_{j=1}^{n}(\nu_{j,t}-w\alpha_P^{(i)}(p_{j,t}^{(i)}-x_{j,t}^{(i)}))(g_{j,t-1}-x_{j,t-1}^{(i)})}{(1-w)\sum_{j=1}^{n}(g_{j,t-1}-x_{j,t-1}^{(i)})^2} \\
1-w &= \frac{\sum_{j=1}^{n}(\nu_{j,t}-w\alpha_P^{(i)}(p_{j,t}^{(i)}-x_{j,t}^{(i)}))(g_{j,t-1}-x_{j,t-1}^{(i)})}{\alpha_{L,t}^{(i)}\sum_{j=1}^{n}(g_{j,t-1}-x_{j,t-1}^{(i)})^2} \\
1-w &= \frac{\sum_{j=1}^{n}\nu_{j,t}-w\sum_{j=1}^{n}\alpha_P^{(i)}(p_{j,t}^{(i)}-x_{j,t}^{(i)}))(g_{j,t-1}-x_{j,t-1}^{(i)})}{\alpha_{L,t}^{(i)}\sum_{j=1}^{n}(g_{j,t-1}-x_{j,t-1}^{(i)})^2} \\
1-w &= \frac{a-wb}{c} \\
c(1-w) &= a-bw \\
(b-c)w &= a-c \\
w &= \frac{b-c}{a-c}
\end{aligned}$$

3.4.4 Including Cohesion

Take partial (set $\alpha_{L,t}^{(i)}=1$ and $\alpha_{L,t}^{(i)}=1$):

$$\sum_{j=1}^{n} 2(v_{j,t}^{(i)}-\nu_{j,t}^{(i)})\frac{\partial v_{j,t}^{(i)}}{\partial w_C}=0$$

and

$$\sum_{j=1}^{n}(w_C v_{C,j,t}^{(i)}+w_L v_{L,j,t}^{(i)}+w_P v_{P,j,t}^{(i)}-\nu_{j,t}^{(i)})v_{C,t}^{(i)}=0$$

and

$$w_C\sum_{j=1}^{n}\left(v_{C,j,t}^{(i)}\right)^2+w_L\sum_{j=1}^{n}v_{L,j,t}^{(i)}v_{C,j,t}^{(i)}+w_P\sum_{j=1}^{n}v_{P,j,t}^{(i)}v_{C,j,t}^{(i)}=\sum_{j=1}^{n}\nu_{j,t}^{(i)}v_{C,j,t}^{(i)} \tag{3.48}$$

Two more other equations:

$$w_C\sum_{j=1}^{n}v_{C,j,t}^{(i)}v_{L,j,t}^{(i)}+w_L\sum_{j=1}^{n}\left(v_{L,j,t}^{(i)}\right)^2+w_P\sum_{j=1}^{n}v_{P,j,t}^{(i)}v_{L,j,t}^{(i)}=\sum_{j=1}^{n}\nu_{j,t}^{(i)}v_{L,j,t}^{(i)}$$

and

$$w_C\sum_{j=1}^{n}v_{C,j,t}^{(i)}v_{P,j,t}^{(i)}+w_L\sum_{j=1}^{n}v_{L,j,t}^{(i)}v_{P,j,t}^{(i)}+w_P\sum_{j=1}^{n}\left(v_{P,j,t}^{(i)}\right)^2=\sum_{j=1}^{n}\nu_{j,t}^{(i)}v_{P,j,t}^{(i)} \tag{3.49}$$

In a matrix form:

$$\begin{bmatrix} \left(v_{C,j,t}^{(i)}\right)^2 & \sum_{j=1}^{n}v_{C,j,t}^{(i)}v_{L,j,t}^{(i)} & \sum_{j=1}^{n}v_{C,j,t}^{(i)}v_{P,j,t}^{(i)} \\ \sum_{j=1}^{n}v_{C,j,t}^{(i)}v_{L,j,t}^{(i)} & \left(v_{L,j,t}^{(i)}\right)^2 & \sum_{j=1}^{n}v_{L,j,t}^{(i)}v_{P,j,t}^{(i)} \\ \sum_{j=1}^{n}v_{C,j,t}^{(i)}v_{P,j,t}^{(i)} & \sum_{j=1}^{n}v_{L,j,t}^{(i)}v_{P,j,t}^{(i)} & \left(v_{P,j,t}^{(i)}\right)^2 \end{bmatrix}\begin{bmatrix} w_C \\ w_L \\ w_P \end{bmatrix}=\begin{bmatrix} \sum_{j=1}^{n}\nu_{j,t}^{(i)}v_{C,j,t}^{(i)} \\ \sum_{j=1}^{n}\nu_{j,t}^{(i)}v_{L,j,t}^{(i)} \\ \sum_{j=1}^{n}\nu_{j,t}^{(i)}v_{P,j,t}^{(i)} \end{bmatrix}$$

or

$$X\underline{w}=\underline{y} \tag{3.50}$$

Hence:

$$\underline{w} = X^{-1}\underline{y}$$

3.4.5 Applications

§ Risk Factors (with Yi Tang)

In the investments literature, following winners or the doing the opposite (so called winner's curse) is a fight that never ends. My current research with professor Yi Tang is to use swarm to the judge. A new angle (hypothesis) we have is that we have a swarm measure. A stronger swarm should indicate a more significant empirical result.

The dimensions are lags. Let x_1 be the 1-day lag and x_5 be a 5-day lag. We use $x_1 \sim x_5$ (lagged returns) as features/dimensions. In other words, returns of past 5 days are plotted in the 5-dimensional hyper-cube. Hence, at any given time, there is a matrix of $x_{j,t}^{(i)}$ where $i = 1, \cdots, 10$ and $j = 1, \cdots, 5$. And then there are a time series of such matrix: $t = 1, \cdots, 250$. Differences in positions $x_{j,t+1}^{(i)} - x_{j,t}^{(i)}$ is velocity $v_{j,t+1}^{(i)}$.

Step 1 Collect daily prices of at least 10 ETFs (i.e. portfolios) for at least one year and compute their returns. [Note: You can choose stocks, but you should have an idea if there exists a leader. Otherwise you will be getting GIGO.]

Step 2: Find the leader (highest return) and loser (lowest return) of yesterday. Then compute and . Compare the two results.

Step 3: If the leader's return is positive, then buy leader and sell loser. If negative, do the opposite.

Step 3a: Do Step 3 only if is strong (set an arbitrary threshold, say, 0.5 and you can experiment this threshold)

Step 4: Do this daily till end of data. Report the cumulative return (of this long-short portfolio) – better draw an entire time series of cum returns.

Step A1: In Step 3, adjust for trading cost (you need to liquidate the previous day position)

Step B1: Simulate 1000 and compute .

§ Firm Choice (with Cameron Miller)

It is widely known in literature of firm choice that a genetic model named N-K search is used for making decisions. See the following diagrams:

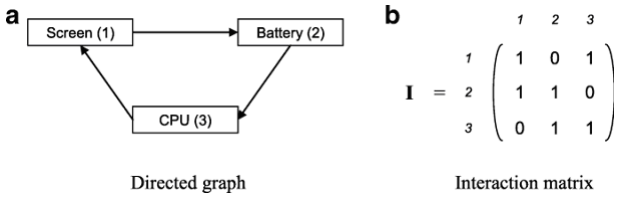


Figure 3.11: Screen and Battery and CPU

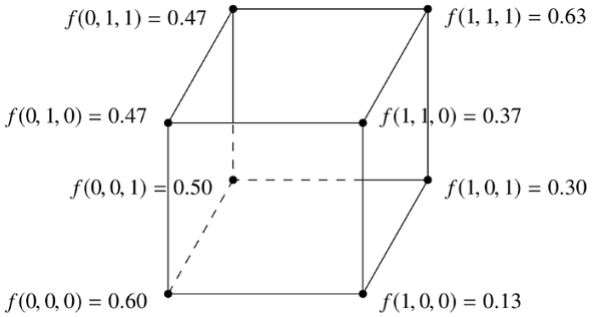


Figure 3.12: Cube

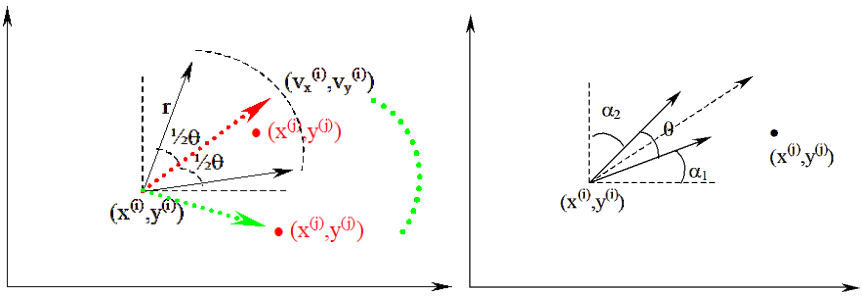
Contribution of screen	Contribution of battery	Contribution of CPU
screen CPU	battery screen	CPU battery
$c_1(0,0) = 0.9$	$c_2(0,0) = 0.8$	$c_3(0,0) = 0.1$
$c_1(0,1) = 0.5$	$c_2(0,1) = 0.0$	$c_3(0,1) = 0.5$
$c_1(1,0) = 0.3$	$c_2(1,0) = 0.0$	$c_3(1,0) = 0.2$
$c_1(1,1) = 0.7$	$c_2(1,1) = 0.3$	$c_3(1,1) = 0.9$

Figure 3.13: Payoff

$$f(s) = \frac{1}{N}\sum_{i=1}^{N} c_i(s_i; K \text{otherelementsof} s)$$

Thus, the fitness of a portable computer with small screen ($s_1 = 0$), weak battery ($s_2 = 0$), and fast CPU ($s_3 = 1$) would be:

$$\begin{aligned} f(0,0,1) &= \frac{1}{3}(c_1(0,1) + c_2(0,0) + c_3(1,0)) \\ &= \frac{1}{3}(0.5 + 0.8 + 0.2) \\ &= 0.5 \end{aligned}$$



(a) Swarm Demo (b) Swarm Demo

Figure 3.14: Two Swarm Examples

§ Institution Herding (Ren-Raw Chen)

We can study how brokers follow each other. This would require detailed trading information (price and volume) of every broker in the market. Using a proprietary dataset from Taiwan, I study the herding behavior of all brokers/dealers (affiliation and location).

3.5 Appendix

In this appendix, we draw some easy geometric for calculating movements in a swarm (also in PSO). These are easy steps to set up for any type of swarm. Note that in the examples in the main text, we do not set up certain angle and radius for a reference group. In this appendix, we show how to do so should we need to. These modifications are important for certain applications (e.g. firm choice).

Descriptions here help set up angle and radius when we simulate swarm intelligence. See Figure 3.14.

Then to detect if a fish $\vec{x}_t^{(j)}$ is within the cone of $\vec{x}_t^{(i)}$, it must be:

$$\left\| x_t^{(i)}, x_t^{(j \neq i)} \right\| < d$$

and

$$\angle\{\vec{v}_t^{*(i)}, \vec{x}_t^{(j)} - \vec{x}_t^{(i)}\} < \frac{\theta}{2}$$

shown in Figure 3.14b.

If $\alpha_1 \neq \alpha_2$, then the center vector must have the following coordinates

$$(r\cos[\alpha_1 + \tfrac{\theta}{2}] + x^{(i)}, r\cos[\alpha_2 + \tfrac{\theta}{2}] + y^{(i)}) \tag{3.51}$$

and for a fish to be inside the cone, the angle toward this center vector must be less than $\frac{\theta}{2}$ and distance must be less than r. Note that when $\alpha_1 = \alpha_2$, $\alpha + \frac{\theta}{2} = 45°$.

This can be easily extended to n dimensions

$$(r\cos[\alpha_1 + \tfrac{\theta}{2}] + x_1^{(i)}, r\cos[\alpha_2 + \tfrac{\theta}{2}] + x_2^{(i)}, \cdots, r\cos[\alpha_2 + \tfrac{\theta}{2}] + x_n^{(i)}) \tag{3.52}$$

If $\alpha_1 = \alpha_2$, then the center vector is a 45° degree line (and the coordinates are always (x, x)) and:

$$\begin{aligned} &\alpha_1 + \theta + \alpha_2 = 90° \\ &2\alpha + \theta = 90° \\ &\alpha = 45° - \tfrac{\theta}{2} \end{aligned} \tag{3.53}$$

Chapter 4

Textual Analsis

There is no use disputing with the translating machine. It will prevail.

Peter Petrovich Troyanskii, 1894-1950

4.1 Introduction

Textual analysis is by far the most impressive advance in machine learning. Benefiting from probability theory, natural language processing (NLP) has completely revolutionized how we think possible for machine to communicate with humans. ChatGPT today can easily pass the Turing test (the imitation game by Alan Turing in 1949) which is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.

In this chapter, we open certain black boxes in NLP and see how probability theory has helped the advance of NLP. While a complex system is required to generate human-like conversations such as ChatGPT, certain components can be implemented as easily as in Excel.

4.2 Basics

4.2.1 Bag-of-Words Model (Hull 9.3)

† Tokenization

Contrast to a word to humans, a token is a "word" to computers. Apparently computers can only recognize 0's and 1's. Hence a token is a set of 0's and 1's that computers can understand as a word. Tokenization is such a process. When applied to data security, tokenization is the process of substituting a sensitive data element with a non-sensitive equivalent, referred to as a token, that has no intrinsic or exploitable meaning or value.

† Hull

The most basic concept of NLP is to understand a sentence in human conversations. This begins with the concept of bag-of-words. The bag-of-words model is like a dictionary. For example, a bag (dictionary) can contain 10 words – like:

bad	good	great	much	never	product	recommend	someone	terrible	much
1	2	3	4	5	6	7	8	9	10

Table 4.1: Dictionary

Now a positive opinion is given:

The products works well. I would recommend the product to someone.

To map this to the list of words in Table 4.1,

bad	good	great	much	never	product	recommend	someone	terrible	much
1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	2	1	1	0	1

Table 4.2: Bag of Words

where the number in each position corresponds to the frequency of the word appearing. Hence, we can guess that "product" that appears twice is a very positive word, followed by "recommend", "someone", and "much".

A bag of words would translate this into an array of numbers $\{0, 0, 0, 0, 0, 2, 1, 1, 0, 1\}$ according to the dictionary. This is something computers understand. Fed by lots of

sentences like this, computer can then learn how to classify words into, for example, positive, negative, and neutral.

It is apparent that in this opinion, "product" is viewed as an important positive word and yet it is not. It requires the machine be fed with a lot of examples until the word "product" can be recognized as neutral. Certainly, we humans can do a better job because we understand these words and can define each word as positive, negative, or neutral. Yet, this is human learning, not machine learning. It is hard to say which method is better (LOL).

Note that bag-of-words does not understand order. It is merely a frequency count. To improve that, the concept of n-gram is introduced. This can remove confusion as the following sentence:

The products does not work well. I would not recommend the product to someone.

where "does not" and "would not" will be recognized together and so this sentence would not be mis-classified as a positive opinion.

4.2.2 Sentiment Analysis via Naive Bayes (Hull)

Bag-of-words can be used in naive Bayes classification, which is introduced in Chapter 2.[1]

Suppose there are m words in the dictionary. An opinion arrives and we need to decide if it is positive or negative. The Bayesian calculation of each probability is as follows

$$p[\text{pos}|\text{words}] = \frac{p_1 \cdots p_m}{p[\text{words}]} p[\text{pos}]$$
$$p[\text{neg}|\text{words}] = \frac{q_1 \cdots q_m}{p[\text{words}]} p[\text{neg}]$$

where (assuming each word is independent of others, conditional on a given opinion)
p_j is the probability of word j in a positive opinion and
q_j is the probability of word j in a negative opinion.

If a dictionary word is in an opinion, then it is counted toward p_j (j-th word in the dictionary) if it is a positive opinion (and not counted vice versa) and toward q_j if it is a negative opinion (and not counted vice versa).

Hull's example has 10 opinions with 2 words in each opinion given as follows:

Let's compile a joint distribution for x_1 and for x_2 separately.

[1] Classification tools are introduced in Chapter 9.

opinion	1	2	3	4	5	6	7	8	9	10
word 1	1	1	1	0	0	0	0	0	0	1
word 2	0	0	1	1	1	1	0	1	0	0
label	pos	pos	pos	pos	neg	neg	neg	neut	neut	neut

Table 4.3: Bag of Words

x_1	pos	neg	neu	
0	0.1	0.3	0.2	0.6
1	0.3	0.0	0.1	0.4
	0.4	0.3	0.2	1

Table 4.4: Word 1

x_2	pos	neg	neu	
0	0.2	0.1	0.2	0.5
1	0.2	0.2	0.1	0.5
	0.4	0.3	0.3	1

Table 4.5: Word 2

Table 4.6: Joint Distributions

and the conditional distributions are:

x_1	pos	neg	neu
0	0.25	1	$\frac{2}{3}$
1	0.75	0	$\frac{1}{3}$
	1	1	1

Table 4.7: Word 1

x_2	pos	neg	neu
0	0.5	$\frac{1}{3}$	$\frac{2}{3}$
1	0.5	$\frac{2}{3}$	$\frac{1}{3}$
	1	1	1

Table 4.8: Word 2

Table 4.9: Conditional Distributions

We need to compute the following probabilities (by reverse engineering Hull) if an opinion where word 1 shows up but not word 2.

$$\begin{aligned} p[\text{pos}|x_1 = 1; x_2 = 0] &= \frac{p[x_1 = 1|\text{pos}] \times p[x_2 = 0|\text{pos}]}{Q} \times p[\text{pos}] \\ &= \frac{p_1 \times p_2}{Q} \times p[\text{pos}] \\ &= \frac{0.75 \times 0.5}{Q} \times 0.4 \\ &= \frac{0.075}{Q} \end{aligned}$$

and

$$\begin{aligned} p[\text{neg}|x_1 = 1; x_2 = 0] &= \frac{p[x_1 = 1|\text{pos}] \times p[x_2 = 0|\text{pos}]}{Q} \times p[\text{neg}] \\ &= \frac{q_1 \times q_2}{Q} \times p[\text{neg}] \\ &= \frac{0 \times \frac{1}{3}}{Q} \times 0.3 \\ &= \frac{0}{Q} \end{aligned}$$

and

$$\begin{aligned} p[\text{neu}|x_1 = 1; x_2 = 0] &= \frac{p[x_1 = 1|\text{neu}] \times p[x_2 = 0|\text{neu}]}{Q} \times p[\text{neu}] \\ &= \frac{r_1 \times r_2}{Q} \times p[\text{neu}] \\ &= \frac{\frac{1}{3} \times \frac{2}{3}}{Q} \times 0.3 \\ &= 0.06667 \end{aligned}$$

Hence, $Q = 0.15 + 0 + 0.06667 = 0.2167$ and as a result,

$$\begin{aligned} p[\text{pos}|x_1 = 1; x_2 = 0] &= 0.6923 \\ p[\text{neg}|x_1 = 1; x_2 = 0] &= 0 \\ p[\text{neu}|x_1 = 1; x_2 = 0] &= 0.3077 \end{aligned}$$

Now we know word 1 only opinions are likely (probability of 69.23%) to be positive opinions. It is equally interesting to see the other three situations. The word 2 only opinions can be calculated similarly as follows:

$$\begin{aligned} p[\text{pos}|x_1 = 0; x_2 = 1] &= 0.5 \times 0.25 \div Q \times 0.4 = 0.05 \div Q \\ p[\text{neg}|x_1 = 0; x_2 = 1] &= \tfrac{2}{3} \times 1 \div Q \times 0.3 = 0.2 \\ p[\text{neu}|x_1 = 0; x_2 = 1] &= \tfrac{2}{3} \times \tfrac{2}{3} \div Q \times 0.3 = 0.1333 \div Q \end{aligned}$$

and hence $Q = 0.05 + 0.2 + 0.1333 = 0.3833$ and

$$\begin{aligned} p[\text{pos}|x_1 = 0; x_2 = 1] &= 0.13 \\ p[\text{neg}|x_1 = 0; x_2 = 1] &= 0.52 \\ p[\text{neu}|x_1 = 0; x_2 = 1] &= 0.35 \end{aligned}$$

which indicates that those word 2 only opinions are likely to be neutral opinions (highest probability of 52%). Readers can work out the three probabilities in a situation where both words show up in an opinion and a situation where both words fail to show up in an opinion.[2]

This is generally understood as sentiment analysis.

(https://www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis) Sentiment analysis, also known as opinion mining, is a subfield of NLP that includes deciding and concentrating on the emotional data in an info text. This can be an assessment, an evaluation, or an inclination about a specific point or item. Here are the fundamental sorts of feeling examination:

- Fine-grained Sentiment Analysis: This goes beyond just positive, negative, or neutral. It involves very specific ratings, like a 5-star rating, for example.

- Emotion detection: This aims to detect emotions like happiness, frustration, anger, sadness, etc. The biggest challenge here is being able to accurately identify these emotions in text.

- Aspect-based Sentiment Analysis: This is generally used to understand specific aspects of a certain product or service. For example, in a review like "The battery life of this phone is great, but the screen is not very clear", the sentiment towards the battery life is positive, but it's negative towards the screen.

- Multilingual sentiment analysis: This can be particularly challenging because the same word can convey different sentiments in different languages.

- Intent Analysis: This goes a step further to understand the user's intention behind a certain statement. For example, a statement like "I would need a car" might indicate a purchasing intent.

Sentiment analysis using NLP is a mind boggling task because of the innate vagueness of human language. Mockery, for example, is especially difficult to identify. Subsequently, the precision of opinion investigation generally relies upon the intricacy of the errand and the framework's capacity to gain from a lot of information.

[2]Opinions with none of the two words are 18% to be positive, 35% to be negative, and 47% to be neutral. Opinions with both words are 82% to be positive, 0% to be negative and 18% to be neutral.

4.3 Embedding

The general meaning of embedding is to assume that the observational data (either textual or numeral) are originated from a much simpler universe. In other words, the real and noisy world "embeds" a true world which is clean with no noise. Putting it differently, the clean world has a much smaller number of dimensions than the dirty noisy real world. As a result, embedding can be regarded as a dimension reduction process. A typical example is the Swiss roll example.

Another use of embedding is to translate natural language into machine language. Stretching your imagination, you can probably think of such language translation as one (not necessarily simpler, but more standard) structure being embedded into another (messier) structure. This is not dimension reduction for sure. This is a necessary step in NLP. Typical examples are `word2vec` and `doc2vec`.

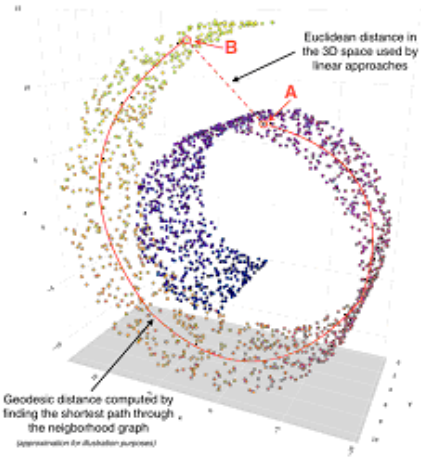
4.3.1 Dimension Reduction

§ Swiss Roll

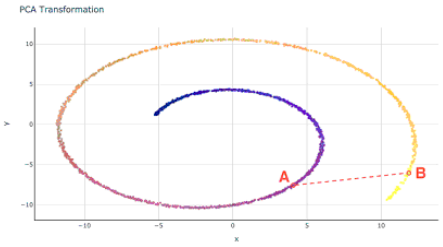
See Figure 4.1 for an animated Swiss roll, its side view, and a flattened view. This famous example demonstrates that a complex structure can be modeled with a straightforward 2-dimensional plane. A and B in the Swiss roll can be easily mistakenly regarded as two close points, and yet in the real structure (sub-figure 4.1c), they are actually far apart.

In a sense, this is similar to PCA (principle component analysis) and yet embedding does not need to be linear, as we can see in the Swiss roll example. In this example, we can successfully unfold a Swiss roll because we already knew it is a Swiss roll. What if we do not visualize a high-dimensional structure? How to "unfold" it becomes a challenge. Hence, various embedding methods are proposed.

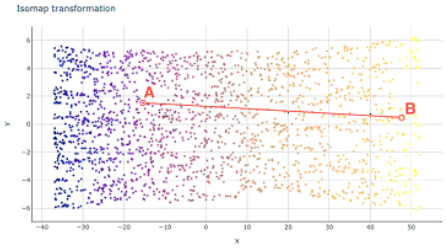
- isomap embedding – https://towardsdatascience.com/isomap-embedding-an-awesome-approach-to-non-linear-dimensionality-reduction-fc7efbca47a0 by Saul Dobilas
  - Use a KNN (k-nearest neighbors) approach to find the k nearest neighbors
  - Once the neighbors are found, construct the neighborhood graph where points are connected to each other if they are each other's neighbors
  - Compute the shortest path between each pair of data points (nodes).
  - Use multidimensional scaling (MDS) to compute lower-dimensional embedding



(a) Swiss Roll



(b) Side View of Swiss Roll



(c) Flattened Swiss Roll

Figure 4.1: Embedding of Swiss Roll

https://towardsdatascience.com/mds-multidimensional-scaling-smart-way-to-reduce-dimensionality-in-python-7c126984e60b
https://www.sjsu.edu/faculty/guangliang.chen/Math250/lec11ISOmap.pdf
by Dr. Guangliang Chen

- spectral embedding – https://medium.com/mlearning-ai/demystifying-spectral-embedding-b2368bba580
  Spectral embedding is a technique used for non-linear dimensionality reduction. Under it's hood, the algorithm in action is Laplacian Eigenmaps. Laplacian Eigenmaps is considerably similar to Isometric Feature Mapping (also referred to as Isomap). You can find an amazing reference to Isomap towards the end of this section, if you are unfamiliar with it. The primary difference between Isomap and Laplacian Eigenmaps is that the goal of Isomap is to directly preserve the global (non-linear) geometry, but the goal of Laplacian Eigenmaps is to preserve the local geometry (i.e., nearby points in the original space remain nearby in the reduced space).

  - Constructing the Adjacency Graph
  - Choosing the Weights
  - Obtaining the Eigenmaps

  A nice way of representing a set of data points $x_1, \cdots, x_N$ is in form of the similarity graph $G = (V, E)$. ε-neighborhood graph k-nearest neighbor graph

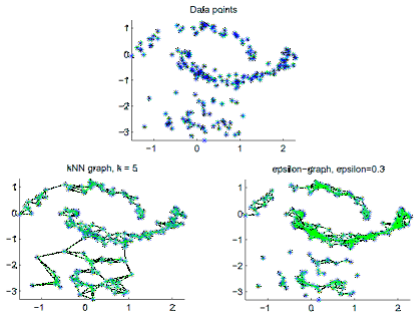


Figure 4.2: Spectral Embedding

Now that we have our graph, we need to form its associated Laplacian matrix

Run k-means

- linearly local embedding – https://scikit-learn.org/stable/modules/manifold.html

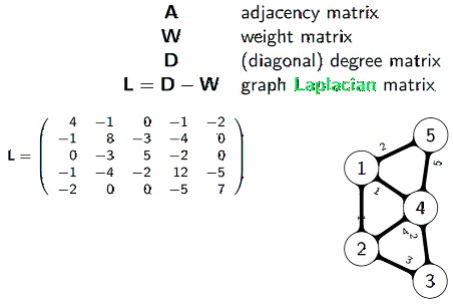


Figure 4.3: Laplacian Matrix

- t-SNE (t-distributed stochastic neighbor embedding – https://www.datacamp.com/tutorial/introduction-t-sne
- UMAP (uniform manifold approximation and projection) embedding
  - use spectral embedding
  - use topology
  - use fuzzy sets (fuzzy union operation)
- forced-directed graph
  - Force-directed graph drawing algorithms are a class of algorithms for drawing graphs in an aesthetically-pleasing way. Their purpose is to position the nodes of a graph in two-dimensional or three-dimensional space so that all the edges are of more or less equal length and there are as few crossing edges as possible, by assigning forces among the set of edges and the set of nodes, based on their relative positions, and then using these forces either to simulate the motion of the edges and nodes or to minimize their energy.
  - Force-directed graph drawing algorithms assign forces among the set of edges and the set of nodes of a graph drawing. Typically, spring-like attractive forces based on Hooke's law are used to attract pairs of endpoints of the graph's edges towards each other, while simultaneously repulsive forces like those of electrically charged particles based on Coulomb's law are used to separate all pairs of nodes.

The details of these methods are discussed in the Appendix at the end of the chapter.

§ Knowledge Graph Embedding

unfinished... Knowledge graph embedding is a type of representation learning between entities and relations in a knowledge base. The entities and relations are mapped into a low-dimensional space representing the semantic information between entities and relationships. We classify knowledge embedding into two broad areas. The first is unfolding. The most famous case is the Swiss roll example where a roll is unfolded into a plane. This includes isomap, locally linear embedding, spectral embedding, Hessian eigenmapping, local tangent space alignment, multi-dimensional scaling (MDS), t-distributed stochastic neighbor embedding (t-SNE), among others.

The second is to investigate the relation of any two nodes using textual data, known as translation distance models. Both entities and relations can be represented as vectors in the same space. This includes DistMult, TransE, TransH, TransR, TransM, ComplEx, ConvE, KG2E, among others. Note that these methods use textual data. These knowledge graph embedding methods aim to capture the semantic and structural information of entities and relations in knowledge graphs. These embeddings can then be used as features for various downstream tasks, such as knowledge graph completion, entity recommendation, and question answering.

A triplet is defined as triplet (h, r, t) where h is for "head", t is for "tail", and r is for the "relation" between head and tail. The triplet hence is a "semantic" expression of two entities (words, people, etc.) See Figure 4.4. Semantic graphs in graph database are a crucial first step, which we shall discuss later in the chapter.

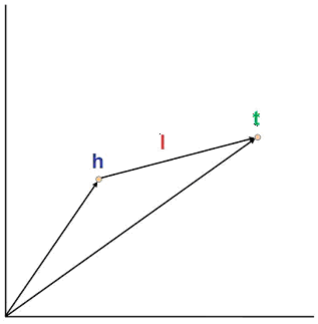


Figure 4.4: Triplet

Define the loss function as follow:

$$L = \sum_{\{h,r,t\}\in S} \sum_{\{h',r,t'\}\in S'} [\gamma + d(h + r, t) - d(h' + r, t')]^+$$

where γ represents the margin and is set to 1. We can calculate distances using L2 norm. Here, (h', r, t') represents a corrupted triple by either replacing the head or tail with a random entity.

Various triplet models are listed in Figure 4.5.

Models	score function $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$
TransE [2]	$-\|\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|\|_{1/2}$
TransR [10]	$-\|\|M_r\mathbf{h} + \mathbf{r} - \mathbf{M_r t}\|\|_2^2$
DistMult [20]	$\mathbf{h}^\top \text{diag}(\mathbf{r})\mathbf{t}$
ComplEx [16]	$\text{Real}(\mathbf{h}^\top \text{diag}(\mathbf{r})\bar{\mathbf{t}})$
RESCAL [12]	$\mathbf{h}^\top \mathbf{M_r t}$
RotatE [15]	$-\|\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|\|^2$

Figure 4.5: Various Triplet Models

Finally we should note that graphs are often used, like other machine learning tools, to perform classification, clustering, regression, anomaly detection, feature learning, among others. All of these tasks have their counterparts in network analysis. Researchers in network science have traditionally relied on user-defined heuristics to extract features from complex networks (e.g., degree statistics or kernel functions). However, recent years have seen a surge in approaches that automatically learn to encode network structure into low-dimensional embeddings, using techniques based on deep learning and nonlinear dimensionality reduction. These network representation learning (NRL) approaches remove the need for painstaking feature engineering and have led to state-of-the-art results in network-based tasks, such as node classification, node clustering, and link prediction.

4.3.2 Text Embedding

Text embedding is a technique for converting words, phrases, or entire paragraphs of text into numerical vectors. These vectors capture the semantic properties of words, such as meaning, contextual relevance, etc. Through text embedding, the model can mathematically process and analyze natural language. These embeddings are usually vectors in a high-dimensional space that can reflect the similarities and differences between different words. Text embedding is the basis for LLM to process natural language. LLMs typically contain one or more layers within their internal architecture specifically responsible for converting text into vector representations. When training an LLM, the model learns how to map words and sentences into an embedding space that can effectively express their semantic meaning. This

enables the model to capture the nuances of language, allowing for more accurate understanding and generation of natural language.

Let's begin with `word2vec` first invented by Google.

§ word2vec

In the following (oversimplified) example, `word2vec` converts the word "house" (that humans understand as a place to live in) into a vector of numbers:

$$\text{house} \rightarrow \begin{pmatrix} 2566 \\ 3 \end{pmatrix}$$

where the first number is the square footage and the second number represents the number of bedrooms in the house. Apparently two numbers aren't very helpful. There are a huge number of houses that are not at all similar but have similar sizes and 3 bedrooms. Hence we need a large vector.

`word2vec` takes a word (e.g. king) and convert it to hundreds of numbers in a vector:[3]

$$\begin{pmatrix} 31.5 \\ -2.5 \\ 9.2 \\ 7.8 \\ \vdots \\ 6.8 \\ 9.4 \\ -15.3 \\ -5.1 \end{pmatrix}$$

so we have a word (above vector) that computer can understand. With `word2vec`, we can assume that there is a one-to-one translation between natural language and machine language. Now, we are ready to start some dialogues. For example, if you ask a person what is the female person of a king, then anyone would answer queen. We would hope computers do the same: "king - man + woman = queen".

In the following simple example in Figure 4.6, it highlights how embedding can do the job.

[3]This example is a result using R. It is unclear the exact dimension of `word2vec`, and it is believed to be a few hundreds.

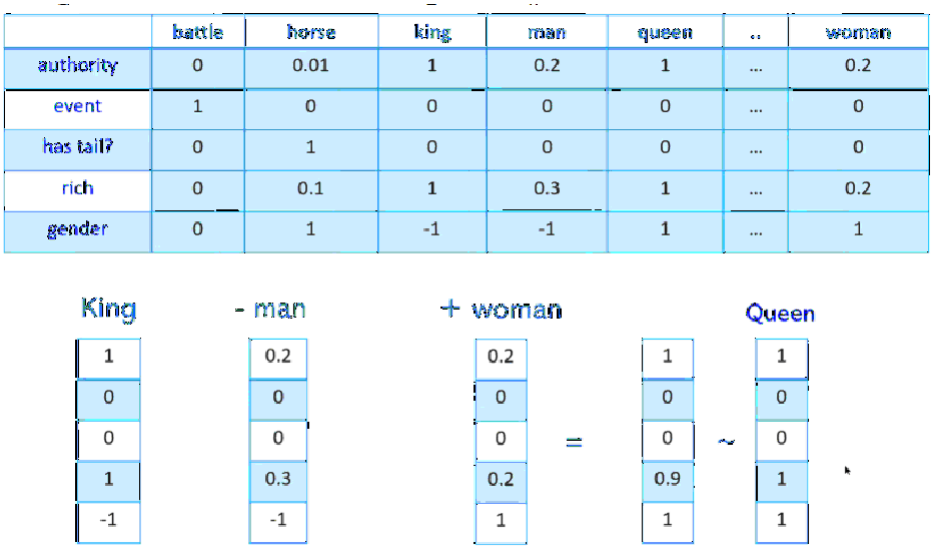


Figure 4.6: Embedding

and hence

$$\text{king} - \text{man} + \text{woman} = \text{queen}$$

We can also process the texts like and hence

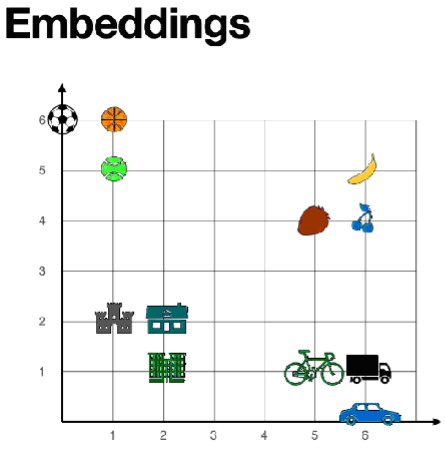
$$\text{apple} + \text{dried} = \text{plum}$$

It is pretty clear now that the key to all this textual processing is to find similar words. The ability to identify those words that are similar is highly essential to NLP. And `word2vec` allows us to achieve that. Given that each word now is a vector of numbers, we can easily tell if two vectors are "similar".

Back on the house example and assume that each word is converted to a 2-dimensional vector, as in Figure 4.7 (right panel). As in the house case, each word is converted to a 2-dimensional vector. Now these two numbers do not mean anything, but will be used to figure out similarity (see next sub-section skip-gram). Also in this example, we do not know where "apple" should belong. The right panel of the figure plots where they are in an x-y plane. It is quite clear that visually we can tell which words are close and which are not.

Similarity can be defined as:

- dot product similarity
- cosine similarity



Word	Numbers	
Apple	?	?
Banana	6	5
Strawberry	5	4
Cherry	6	4
Soccer	0	6
Basketball	1	6
Tennis	1	5
Castle	1	2
House	2	2
Building	2	1
Bicycle	5	1
Truck	6	1
Car	6	0

Figure 4.7: Embedding and Similarity

source: Serrano

Take strawberry as an example, we can calculate its similarity with banana and house. The dot product is 56 between strawberry and banana (i.e. $56 = 6 \times 6 + 6 \times 5$) and 20 between strawberry and house (i.e. $20 = 6 \times 4 + 2 \times 2$). So we know strawberry and banana are more similar than strawberry and house. The cosine similarity scores are 0.9943 and 0.9806 respectively (simply by dividing the dot product by the corresponding lengths which are 7.8 for banana, 7.2 for strawberry, and 2.8 for house.) This result also indicates that strawberry is more similar to banana than to house.

§ Training `word2vec`

CBOW (continuous bag of words) and skip-gram are two similar (and yet opposite) methods to train `word2vec`.

(ritvikmath) Skip-gram is one technique how we can create a useful `word2vec`. For example, if we want "apple" and "orange" to be similar, we first need to know the "context" of each word. To do that, we need the bag-of-words, in particular n-gram, mentioned earlier in this chapter.

Say, we use 5-gram, i.e. two words before and after the word we are interested in. This is called a "context". We want to label neighbor words 1 and non-neighbor words 0. Take the following sentence as an example:

I like data science because data science is really fun.

We first remove those stop words like "1" and "is" because they are irrelevant. We can see that for the list of remaining 8 words: "like", "data", $\cdots$, "fun" their neighbor words ("1") and non-neighbor words ("0"). For example, "like" and "data" have the label of "1" and "like" and "fun" have the label of "0". See Table 4.10. The word "like" has no words on the left but only two words "data" and "science" on the right. The word "data" has 1 word on the left and two on the right. The word "science" has two words on both sides. These words will be labeled "1". This is known as skip-gram.

main	context
like	data
like	science
data	like
data	science
data	because
science	like
science	data
science	because
science	data

Table 4.10: Main and Context Words

To introduce randomness, we need to include those label-0 words. This is known as negative sampling. See Table 4.11. Now, we can see that "fun" is not close to "like" so it is labeled "0". So is "really" which is not close to "data". In Table 4.11, we only pick one random label-0 word for each main word (i.e. the size of negative sampling = 1). For example, for "like", we choose only one label-0 word and randomly we choose "fun". Similarly, we randomly choose "really" for the main word "science".

main	context	label
like	data	1
like	science	1
like	fun	0
data	like	1
data	science	1
data	because	1
data	really	0

Table 4.11: with Labels

For the sake or easy exposition and visualization, we use two dimensions x_1 and x_2. Set up an embedding as in Table 4.12 for the main words. Initially, these embeddings are randomly assigned. As we iterate, they will improve. Design another table of of list of values as in Table 4.13.

	main	
	x_1	x_2
like	0.2	-1
data	-0.7	0.8
science	0.4	0.3
⋮		

Table 4.12: Embeddings of Main Words

So for every word, we have two embeddings, main embedding and context embedding. For example, the word "like" as an embedding of $(0.2, 1)$ as a main word, and $(0.2, -1)$ as a context embedding.

	context	
	x_1	x_2
like	-0.2	-1
data	0.75	-0.25
science	-1	2.1
fun	0.1	2
⋮		

Table 4.13: Embeddings of Context Words

Now we dot-product each vector in the main embedding with a different vector in the context embedding. This result is then fed to the sigmoid function (to get a number between 0 and 1).

If the dot product is very positive, then the main vector and context vector are close together, then the sigmoid value is close to 1. Reversely, if the dot product is very negative, then the two vectors are far apart, then the sigmoid value is close to 0. (Note that the sigmoid value is 0.5 when the dot product is 0.)

Then, the sigmoid value is compared to the label. If the sigmoid value is close to 1 (meaning the main word and the context word is similar) and the label is 1 (meaning the context word is within the n-gram of the main word), then the error is small. Similarly, if the sigmoid value is close to 0 (meaning the main word and the context word is not similar) and the label is 0 (meaning the context word in not

within the n-gram of the main word), then the error is also small. Otherwise the error would be big, and we need to adjust the vectors.

Take the example of "like" (main word) and "data" (context word). The dot product of the two is:

$$\sigma[(0.2, -1) \cdot (0.75, 0.25)] = \sigma[0.4] = 0.6$$

where as the label of the two words is 1 and hence the error is 0.4. Also, "like" (main word) and "fun" (context word) has the following result:

$$\sigma[(0.2, -1) \cdot (0.1, -2)] = \sigma[2.02] = 0.88$$

whereas the label is 0 and hence the error is -0.88.

Hence, we know "like" (main word) and "data" (context word) should be moved closer together (due to small error) and "like" (main word) and "fun" (context word) should be move farther apart (due to large error). See Figure 4.8. To exactly calculate the next set of vectors, commonly people use neural networks for it.

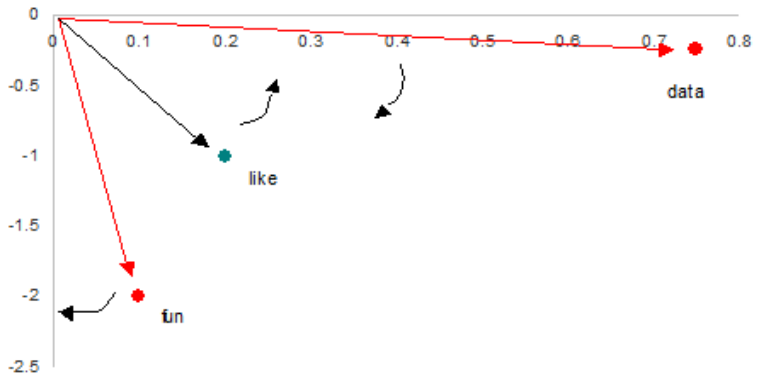


Figure 4.8: Main and Context Vectors

This is really it. As you can see, `word2vec` is really simple to do and yet it is exceedingly powerful. It is the foundation of the Transformer technology today. Once we have achieved good embedding, we can then try to predict missing words in a sentence, or the next word to appear in a partial sentence (sounds familiar?)

§ `doc2vec`

`doc2vec` converts a document into a vector embedding.

† gensim

software: https://radimrehurek.com/gensim/models/doc2vec.html

https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html

† medium:

https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

https://pub.towardsai.net/an-intuitive-introduction-of-document-vector-doc2vec-42c6205ca5a2

† github

https://shuzhanfan.github.io/2018/08/understanding-word2vec-and-doc2vec/

§ `node2vec`

From its name, it is clear that `node2vec` is to convert a set of nodes (vertices) of a graph into a vector embedding, similar to `doc2vec`. But how to create a document (or a corpus) from a graph? `node2vec` uses random walks. The architecture of `node2vec` can be seen in Figure 4.9.

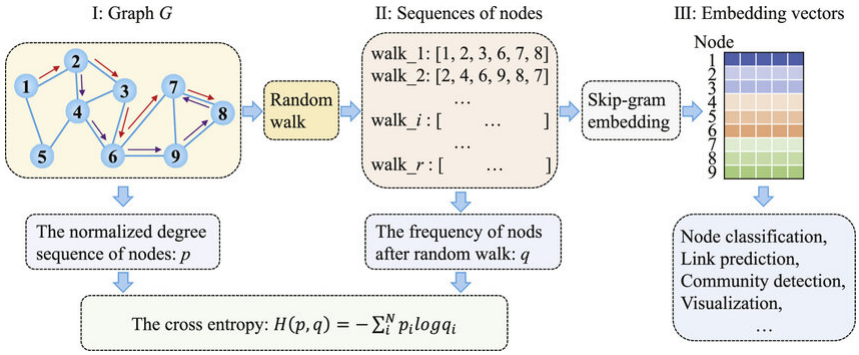


Figure 4.9: A Demonstration

source: The Architecture of node2vec: fig3

An actual training example can be found in https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html#:~:text=Doc2Vec 20is 20a 20Model 20that,and 20test 20corpora 20(see 20Corpus).

§ `fund2vec`

To study a bipartite graph of mutual funds using `node2vec`. See Chen, Luo, and Zhang (2025) for the similarity study of funds.

4.3.3 TF-IDF

See Hull.

Term frequency-inverse document frequency (TF-IDF) is a simple yet useful index to quickly convert documents into vectors. First, term frequency (TF) is defined for each words in vocabulary (v_i) and documents (w_d). This is sometimes referred to as raw term frequency. Inverse document frequency (IDF) is defined as an adjusted inverse of document frequency (DF). Finally, TF-IDF is a product of TF and IDF.

Computing TF-IDF for all documents and words in a corpus yields a document-term matrix. Row vectors of the matrix can be viewed as document embeddings.

4.4 Attention Mechanisms

Consider we are attempting machine translation on the following sentence: "The dog is a Labrador." If you were to ask someone to pick out the key words of the sentence, i.e. which ones encode the most meaning, they would likely say "dog" and "Labrador." Articles like "the" and "a" are not as relevant in translation as the previous words (though they aren't completely insignificant). Therefore, we focus our attention on the important words.

Attention seeks to mimic this by adding attention weights to a model as trainable parameters to augment important parts of our input. Consider an encoder-decoder architecture such as the one Google Translate uses. Our encoder recurrent neural network (RNN) encodes our input sentence as a context vector in some vector space, which is then passed along to the decoder RNN which translates it into the target language. The attention mechanism scores each word in the input (via dot product with attention weights), then passes these scores through the softmax function to create a distribution. This distribution is then multiplied with the context vector to produce an attention vector, which is then passed to the decoder. In the example in the first paragraph, our attention weights for "dog" and "Labrador" would hopefully become larger in comparison to those for the other words during training. Note that all parts of the input are still considered since a distribution must sum to 1, just some elements have more effect on the output than others.

The advantages of attention is its ability to identify the information in an input most pertinent to accomplishing a task, increasing performance especially in natural language processing - Google Translate is a bidirectional encoder-decoder RNN with attention mechanisms. The disadvantage is the increased computation. In humans, attention serves to reduce our workload by allowing us to ignore unim-

portant features, however in a neural network, attention entails overhead as we are now generating attention distributions and training our attention weights (we are not actually ignoring the unimportant features, just diminishing their importance).

4.4.1 Attention

The materials here are based upon Luis Serrano (Serrano Academy).

Continue from Figure 4.7 and but now we have a word "apple" which could mean either a fruit or a brand of a cellphone. So where to put it, as shown in Figure 4.10? Embedding (which is unique to each word) cannot help us. This leads to attention. Later, we will define attention more formally as a function of query, key and value. First we learn self-attention (then later multi-head attention).

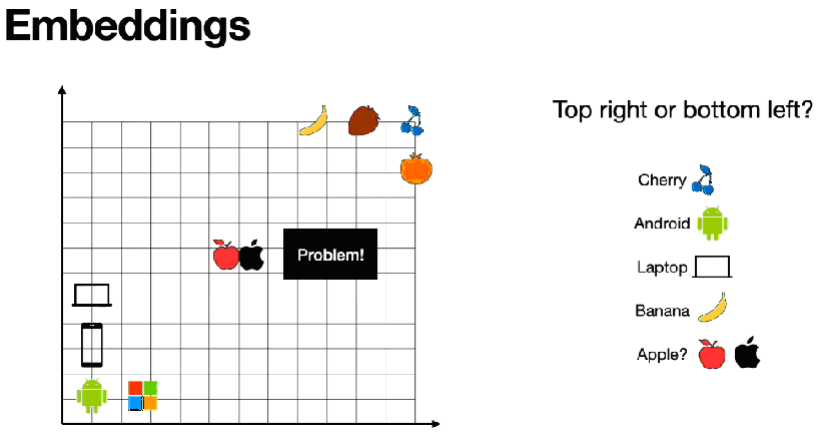


Figure 4.10: Embedding and Similarity

Attention, simply speaking, is that it can read the whole sentence and measure how each word is related to "apple" and clearly here only "orange" is related to "apple" (since all other words (e.g. "please", "buy", "an", and "and") have nothing to do with "apple". Here, we have two sentences, like

- please buy an apple and an orange
- apple unveiled an iPhone

In Figure 4.11, we can see that phone is positioned (via its embedding) at the coordinate (1,1) and orange is at (11,11). Apple so far is positioned at (6,7). We would like the apple in the first sentence close to orange and the second apple to phone.

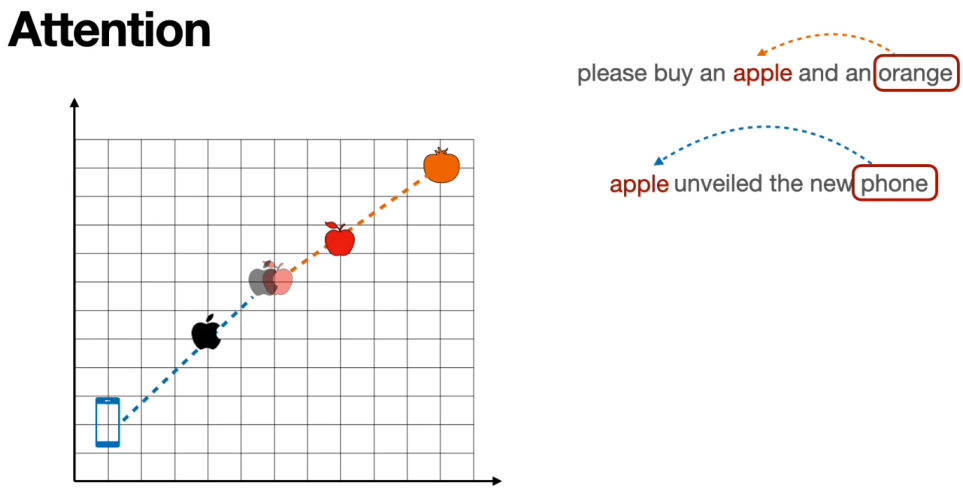


Figure 4.11: Attention

To do that, Attention Mechanisms take in the whole sentence and measure the distance of apple with all the other words in the sentence, as in Figure 4.12. It is clear now that apple should be placed very close to orange as opposed to other (quite meaningless) words.

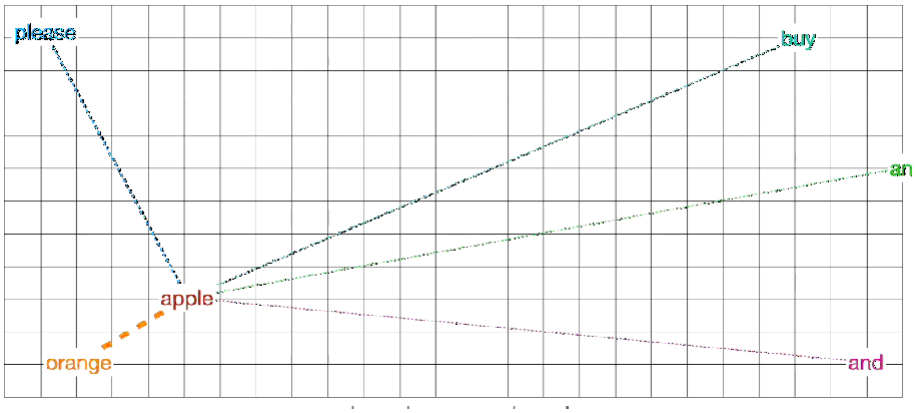


Figure 4.12: Where to Place "Apple"

4.4.2 Linear Transformation

In reality, there are many embeddings (Google, OpenAI, Lamma, etc.) and perhaps one embedding could place apple well but not other words. Similarly, other embeddings can place other words better. How to choose the best embedding is the key

to be able to successfully place a target word, such as apple. The way to do that is to use linear transformation of a chosen embedding. A particular embedding may not be the best for the chosen job, and yet via linear transformation, it will.

Take a look at the three linear transformations in Figures 4.13 and 4.14. We can rotate and stretch the embedding matrix to a desired set of coordinates.

Figure 4.13: Rotate, Stretch Tall, Stretch Wide

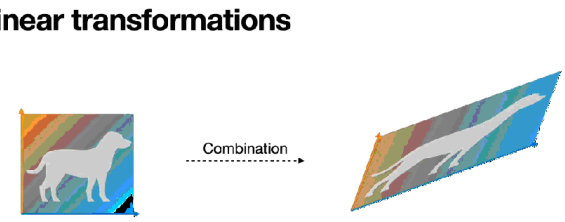


Figure 4.14: Combination

For example, consider three different embeddings as in Figure 4.15. The left is the best and the middle is the worst (because we would like phone and orange to be as far apart as possible). Should we know our task is to place apple, then certainly we would choose the left embedding. Unfortunately, we do not know that the best embedding is the left one. So we might be just unlucky and choose the worse (middle) one.

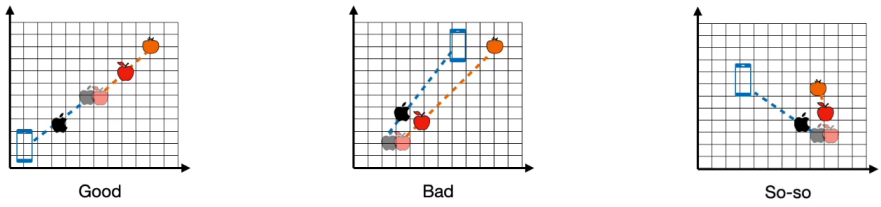


Figure 4.15: Different Embeddings

Linear transformations allow us to make a bad embedding a good embedding. See Figure 4.16. In the figure, we take a bad embedding and turn it into a good embedding (right panel).

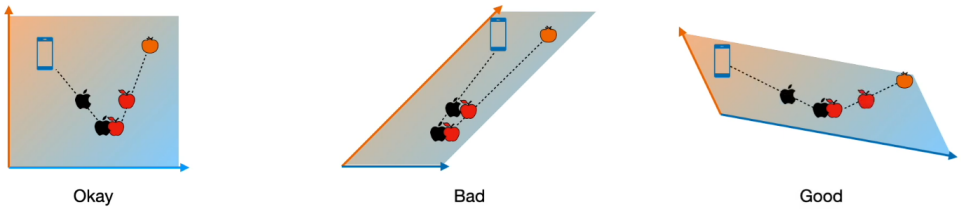
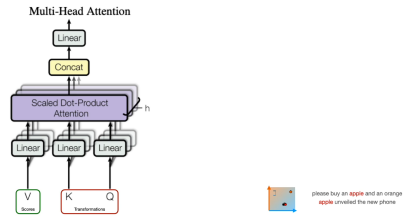
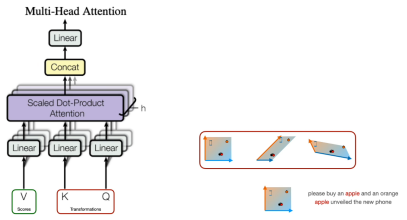


Figure 4.16: Linear Transformation

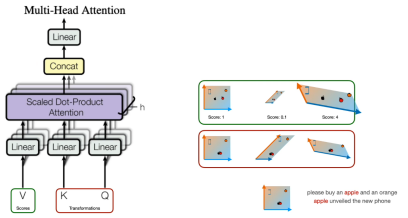
To achieve the result (i.e. the best linear transformation), we must train these linear transformations in a neural network. Let an NN be represented in Figure 4.17a. On the right of the diagram, an embedding is fed into NN. The keys and queries are applied (to be introduced later). These are the linear transformations needed and should result in Figure 4.17b. Then an evaluation is given to each result, as in Figure 4.17c. Finally, the best linear transformation is achieved via concatenating various linear transformation results, as in Figure 4.17d.



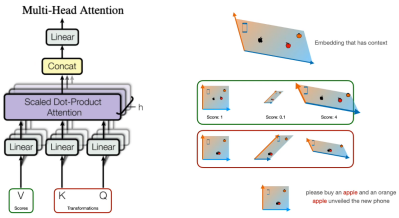
(a) Feed Embedding to NN



(b) Perform Linear Transformations



(c) Evaluate

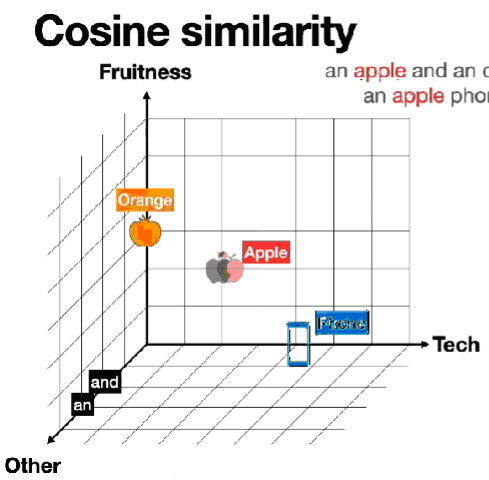


(d) Concatenate

Figure 4.17: Multi-head Attention

Now we go back and analyze the sentences: *buy an apple and an orange* and *buy an Apple phone*. We first compute similarity scores as in Figure 4.18 using the

embedding (that has 3 dimensions: tech, fruit, and other). The chosen words are: "orange", "phone", "apple", "and", and "an". The embedding vectors are shown on the top right of the figure and their similarity scores are shown on the bottom right of the figure.



	Tech	Fruitness	Other
Orange	0	3	0
Phone	4	0	0
Apple	2	2	0
And	0	0	2
An	0	0	3

	Orange	Phone	Apple	And	An
Orange	1	0	0.71	0	0
Phone	0	1	0.71	0	0
Apple	0.71	0.71	1	0	0
And	0	0	0	1	1
An	0	0	0	1	1

Figure 4.18: Embedding and Similarity

Now, look at the words of the first sentence: *buy an apple and an orange*. There are four words: "orange", "apple", "and" and "an". Next is that we need to change their coordinates by computing a weighted sum across other words:

$$\text{orange} = 1 \times \text{orange} + 0.71 \times \text{apple} + 0 \times \text{and} + 0 \times \text{an}$$
$$\text{apple} = 0.71 \times \text{orange} + 1 \times \text{apple} + 0 \times \text{and} + 0 \times \text{an}$$

Then, we need to normalize the coordinates in case they get too large:

$$\text{orange} = \frac{1 \times \text{orange} + 0.71 \times \text{apple}}{1 + 0.71} = 0.58 \times \text{orange} + 0.42 \times \text{apple}$$
$$\text{apple} = \frac{0.71 \times \text{orange} + 1 \times \text{apple}}{1 + 0.71} = 0.42 \times \text{orange} + 0.58 \times \text{apple}$$

so the weights now sum to 1.

However the above normalization, although intuitive, could explode should the denominator be 0. To avoid that, we use the softmax function instead:

$$\begin{aligned}\text{orange} &= \frac{e^{1\times\text{orange}} + e^{0.71\times\text{apple}} + e^{0\times\text{and}} + e^{0\times\text{an}}}{e^1 + e^{0.71} + e^0 + e^0} \\ &= 0.4 \times \text{orange} + 0.3 \times \text{apple} + 0.15 \times \text{and} + 0.15 \times \text{an}\end{aligned}$$

This means that the new position of "apple" is somewhere among the old position of "apple" (30%), "orange" (40%), "and" (15%), and "an" (15%). But for simplicity, we continue to use the linear weights (which is 42% orange and 58% apple):

$$\text{apple} = 0.42 \times \text{orange} + 0.58 \times \text{apple}$$

This means that "apple" needs to move to a new position toward "orange" by 42% (and keep 58% from its old position) and do not move toward "and" and "an". See Figure 4.19. The new position for "apple" now is $(1.14, 2.43)$.

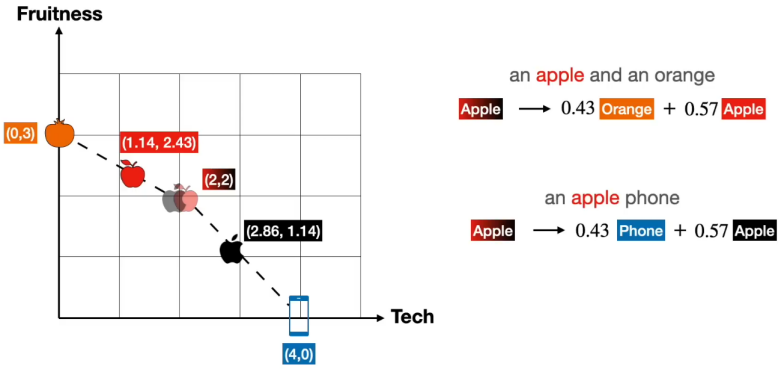


Figure 4.19: Similarity with New Position

In the second sentence: *buy an apple phone*, we have "phone", "apple", and "an". We need to change their coordinates (linear weights) like before. The results are:

$$\text{phone} = 0.57 \times \text{phone} + 0.43 \times \text{apple}$$
$$\text{apple} = 0.43 \times \text{phone} + 0.57 \times \text{apple}$$
$$\text{an} = 1 \times \text{an}$$

and the new position for "apple" is $(2.86, 1.14)$. See Figure 4.19.

4.4.3 Keys and Queries

It goes from this...

to this...

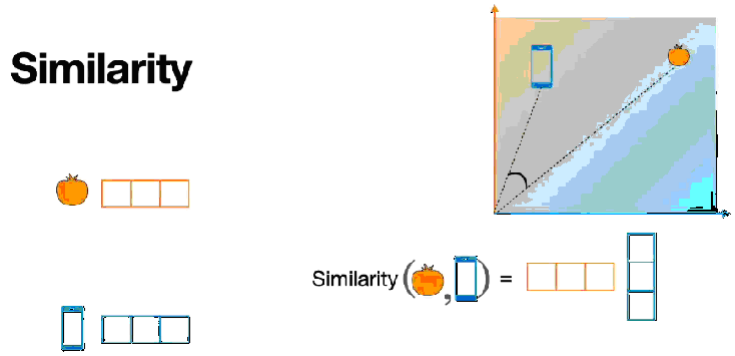


Figure 4.20: Keys and Queries

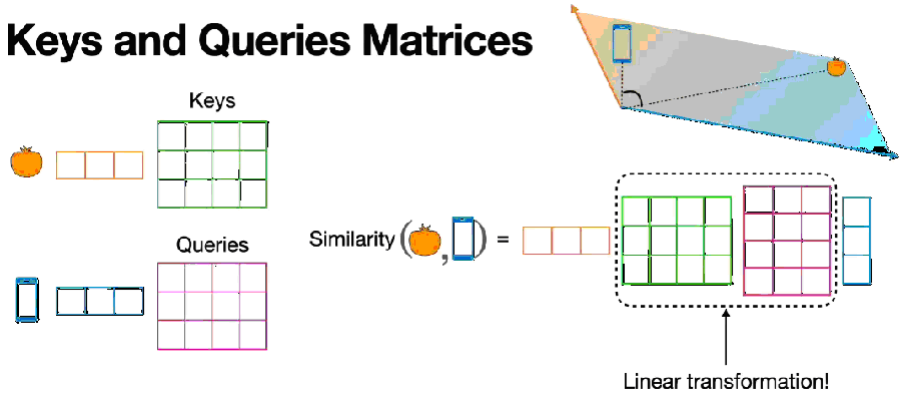


Figure 4.21: Keys and Queries

Now the values matrix. The left of Figure 4.22 is the matrix that we use to calculate similarity scores. But when we move the word "apple", we do it on the right diagram of Figure 4.22. This is because the diagram on the left is optimized for finding similarity scores and the diagram on the right is best optimized for embedding. The reason is that in Transformer, we need to continue to predict the next word. Hence, we need an embedding that is the best to predict the correct next word.

The left diagram is operated using keys and queries and after being multiplied by the values matrix we obtain the embedding on the right. The reason is that the embedding on the left is best for understanding the features of words (e.g. color, size, fruitness, technology, etc.) The embedding on the right knows the best how two words are close to each other and hence should appear in the same context.

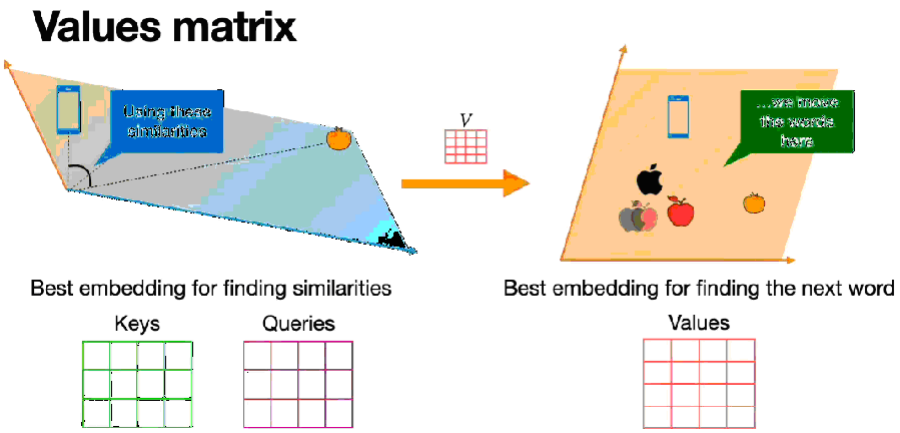


Figure 4.22: Value Matrix

4.4.4 Transformer

A Transformer is an architecture that is trained to predict the next object (which can be code writing, chats, poem writing, etc.) The architecture of Transformer is as follows:

- tokenization
- embedding
  - word2vec
- attention mechanisms
- neural network

4.5 Topic Modeling

To explore the patterns of online shoppers or readers/viewers and then make suggestions of related items and articles/videos is an essential task of many online platforms (such as Amazon). Topic modeling is a perfect tool for it. In finance, we can also use it for risk factors of various financial assets (e.g. stocks). In the past, linear and parametric models are used (e.g. PCA), but now we can use topic modeling to explore nonlinear and non-parametric relations between assets and their fundamental risk factors.

Topic modeling is a type of statistical modeling that uses unsupervised machine learning to identify clusters or groups of similar words within a body of text. This text mining method uses semantic structures in text to understand unstructured data without predefined tags or training data. Like PCA for numerals, topic modeling is for texts.

It is a bag-of-words model.

The contents in this section is heavily based upon lectures by Luis Serrano (Serrano Academy).

4.5.1 Latent Dirichlet (dee.ruhsh.lay) Allocation

(wiki) In natural language processing, latent Dirichlet allocation (LDA) is a Bayesian network (and, therefore, a generative statistical model) for modeling automatically extracted topics in textual corpora. The LDA is an example of a Bayesian topic model. In this, observations (e.g., words) are collected into documents, and each word's presence is attributable to one of the document's topics. Each document will contain a small number of topics.

(wiki) In probability and statistics, the Dirichlet distribution (after Peter Gustav Lejeune Dirichlet) is a family of continuous multivariate probability distributions parameterized by a vector α of positive reals. It is a multivariate generalization of the beta distribution. Dirichlet distributions are commonly used as prior distributions in Bayesian statistics, and in fact, the Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution. The infinite-dimensional generalization of the Dirichlet distribution is the Dirichlet process. An example (of 3 dimensions) is given as follows:

Note that with 4 variables, the distribution can be depicted as a tetrahedron which can be still visualizable. However, in higher dimensions than 4, it is a high dimensional simplex. As we can see, the basic idea of a Dirichlet distribution is to allocate probability mass either near the corners or near the center. As a result, it is perfectly suitable for assigning topics.

A given corpus has 4 documents, as in Figure 4.24 in which the words are "ball", "planet", "referendum", and "galaxy".

There are three topics "science", "politics", and "sports". We do not now what the topic of each document is and to name these documents is our job, as in Figure 4.25.

We now put the documents and topics in a triangle like Figure 4.26. On each tip is a topic and documents are put inside the triangle. If a document is close

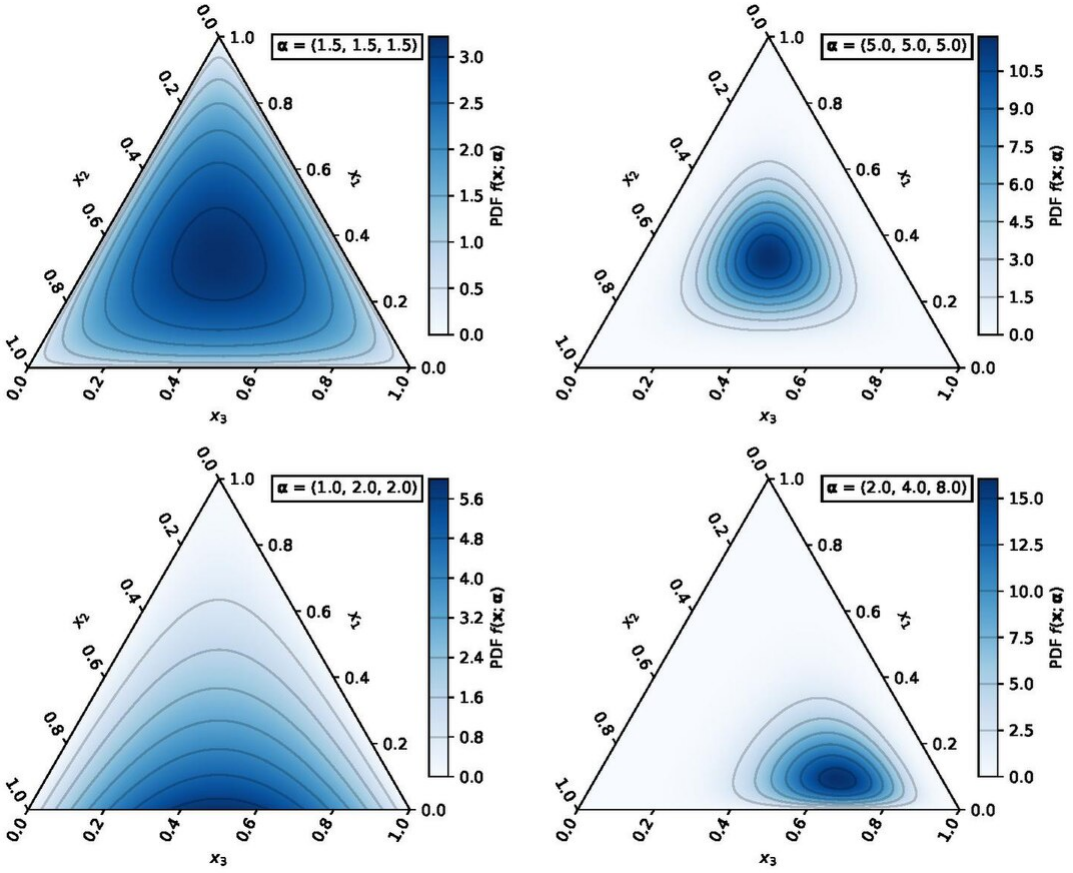


Figure 4.23: Dirichlet Distribution (3d)

Figure 4.24: 4 Documents

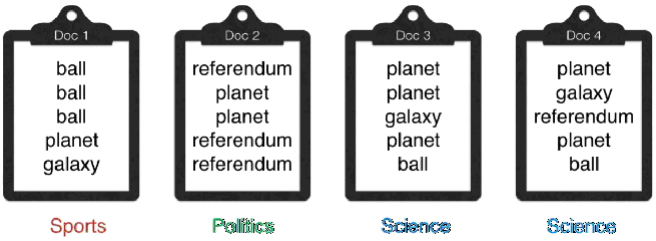


Figure 4.25: 4 Documents (named)

to a topic then it will be placed near that topic. For example, the first document is classified as a sport document, and hence it will be place near to top tip of the triangle. If a document is half sports and half science, then it will be place in the middle on the left edge of the triangle.

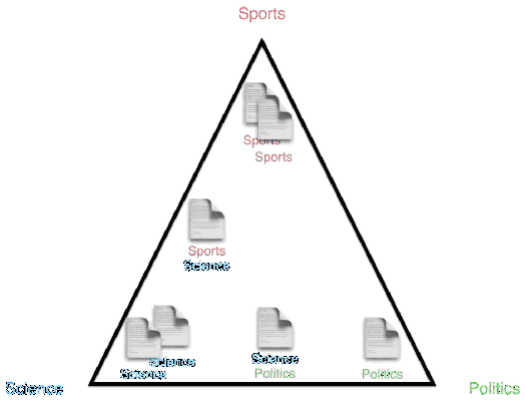


Figure 4.26: Documents Placed in a Triangle

The triangle reflects a Dirichlet distribution. If documents are drawn to the corners, then each document has a clear topic. If documents are concentrated in the center, then no document has a clear topic. This is the major property of the Dirichlet distribution. The Dirichlet distribution is given as follows:

$$f(x_1, \cdots, x_K) = \frac{1}{B(\alpha_1, \cdots, \alpha_K)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{4.1}$$

where

$$B(\alpha_1, \cdots, \alpha_K) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}$$

We can visualize the distribution in Figure 4.27. Three possible Dirichlet distributions are presented. The left is when documents are randomly situated. The middle is the case where topics are clear (density is high at the corners). The right is when documents have no clear topic.

We can plot the $K = 3$ case in Figure 4.28.

Now we take the middle example of Figure 4.27 and assign probabilities. See Figure 4.29 and Table 4.14. Each document (A $\sim$ E) has a distribution associated with the three topics.

Now, lets look at the topic distribution. In Figure 4.30, there are four words (hence a pyramid) and we try to see how each topic is related to these words where the red dot is sports, the blue dot is science, and the green dot is politics.

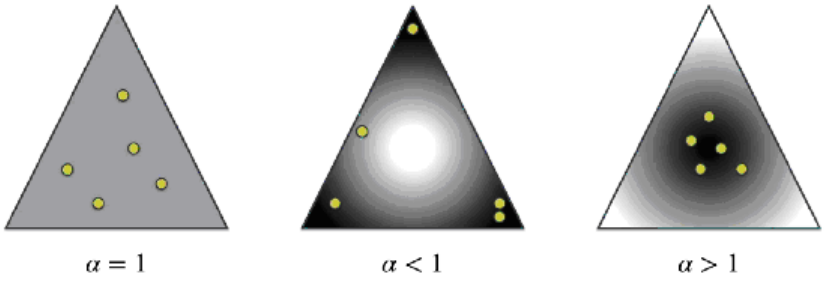


Figure 4.27: A Demo of Dirichlet Distribution

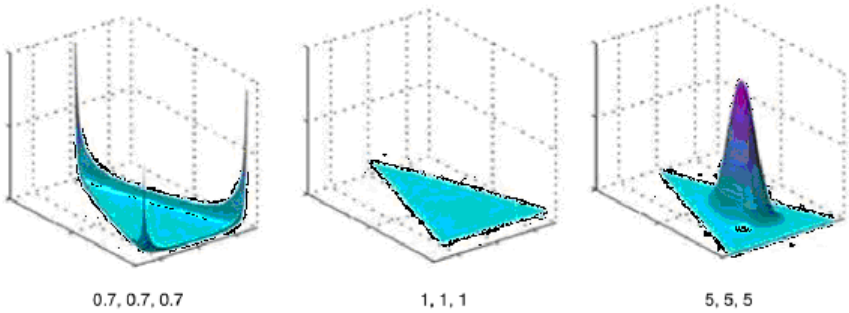


Figure 4.28: Dirichlet Distribution ($K = 3$)

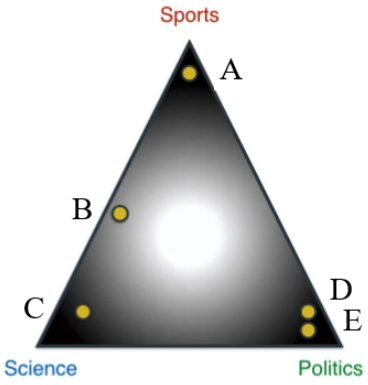


Figure 4.29: Dirichlet Distribution for Documents

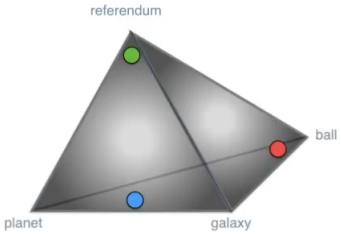


Figure 4.30: Dirichlet Distribution for Topics

	science	politics	sports
A	0.07	0.03	0.9
B	0.45	0.05	0.5
C	0.8	0.1	0.1
D	0.05	0.8	0.15
E	0.05	0.9	0.05

Table 4.14: Distribution of 5 Documents

The probabilities of each topic is associated with words are given as in Table 4.15.

	galaxy	planet	ball	ref'm
science	0.4	0.4	0.1	0.1
sports	0.3	0.1	0.5	0.1
politics	0.1	0.1	0.1	0.7

Table 4.15: Distribution of 3 Topics

So now we have two Dirichlet distributions, one associated with the documents (with topics) and one associated with the topics (with words).

Think about LDA as a video machine, as in Figure 4.31, that produces documents. There are settings (dials on the upper right) and gears (lower left). Gears are turning to produce documents and dials are used to fine tune the documents.

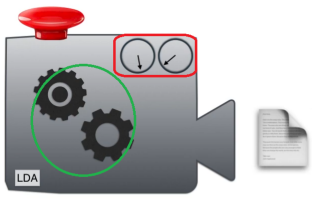


Figure 4.31: Video Camera

Ideally, we would like the machine to produce documents as close to the ones in the dataset by adjusting the settings (dials). By reading the settings, we decide what a topic should be. The settings are two Dirichlet distributions, see Figure 4.32. The two Dirichlet distributions are then used to generate multi-nomial distributions.

A flow chart of LDA can be seen in Figure 4.33. In this chart, there are

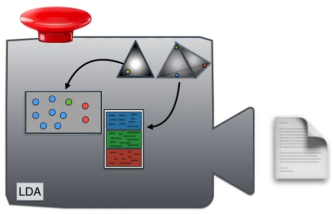


Figure 4.32: Video Camera with Dirichlet Distribution

two Dirichlet distributions, α and β, which are inputs to the machine. Inside the machine, topics and words are processed.

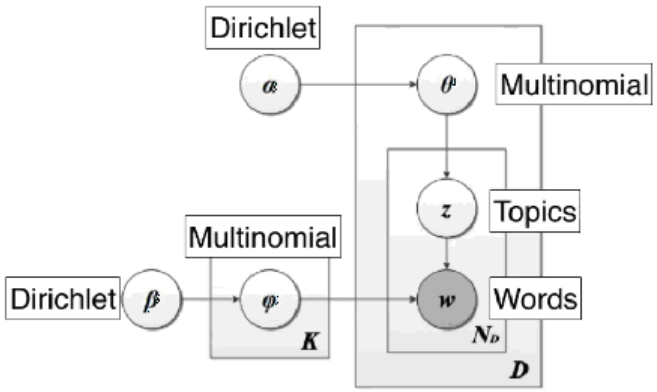


Figure 4.33: Flowchart of LDA

The probability distribution of LDA is:

$$p(W, Z, \theta, \varphi|\alpha, \beta) = \underbrace{\prod_{j=1}^{M} p(\theta_j|\alpha)}_{\text{document (topics)}} \underbrace{\prod_{i=1}^{K} p(\varphi_i|\beta)}_{\text{topic (words)}} \prod_{n=1}^{N} p(Z_{j,n}|\theta_j) p(W_{j,n}|\varphi_i, Z_{j,n}) \tag{4.2}$$

where
M is the number of documents,
K is the number of topics, and
N is the total number of words in the entire corpus.

Also, W represent document and Z represents topic. If we let N_j be the number of words in the j-the document, then $N = \sum_j^M N_j$.

There are two Dirichlet distributions for θ and φ. Using θ, we set a distribution to generate the distribution for topics Z. Then using both Z and φ jointly, we generate the distribution for words W. Finally, stringing all the distributions together, we have the LDA model, as in Figure 4.34.

The probabilistic model behind LDA and the analogy of a video machine can be seen in Figures 4.34 and 4.35.

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$

Figure 4.34: LDA: dials and gears

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$

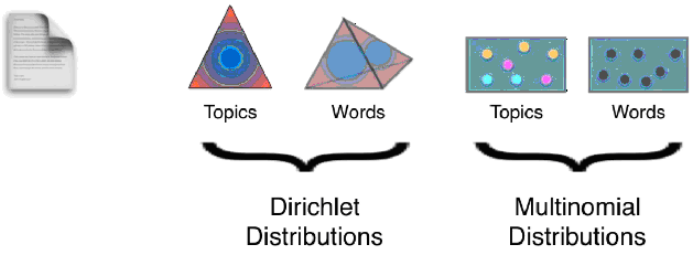


Figure 4.35: LDA: Dirichlet and multinomial distributions

The way the LDA equation in Figures 4.34 and 4.35 works is as follows. We pick a document and see its distribution (probability to each topic). For example, a random document is drawn from the triangle (i.e. Dirichlet distribution) which is 70% related to science, 10% related to politics, and 20% related to sports. Then we sample from this distribution a series of topics, as shown in Figure 4.36.

Similarly, we now simulate a topic distribution (w.r.t. words) and the use it to draw words. For example, as shown in Figure 4.37, we pick a blue topic (science) which has a distribution of 0.4 related to the word galaxy, 0.4 related to the word planet, 0.1 related to the word ball, and 0.1 related to the word referendum. Then we randomly draw these words according to this distribution.

In Figure 4.37, the first sample is topic science. Hence, we sample the 10 words with the science topic distribution. As we can see, most of the ten words are science (4 galaxy's and 4 planet's) and only one sports (ball) and one politics (referendum).

Then, the second sample of topic is (green) politics, and we sample another 10 words using the politics distribution. We have 6 referendum's (politics) but only 1 ball (sports) and 2 science words (1 galaxy and 1 planet).

Finally, the third sample is sports and we sample 10 words using the sports

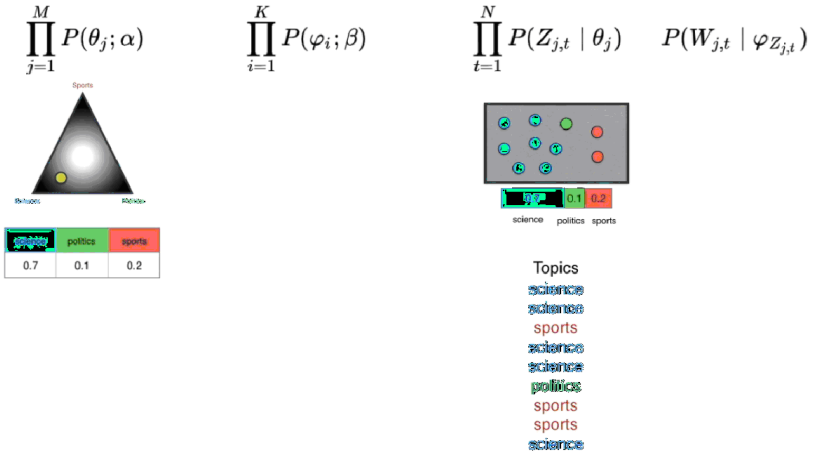


Figure 4.36: LDA (Using Document Distribution to Sample Topics)

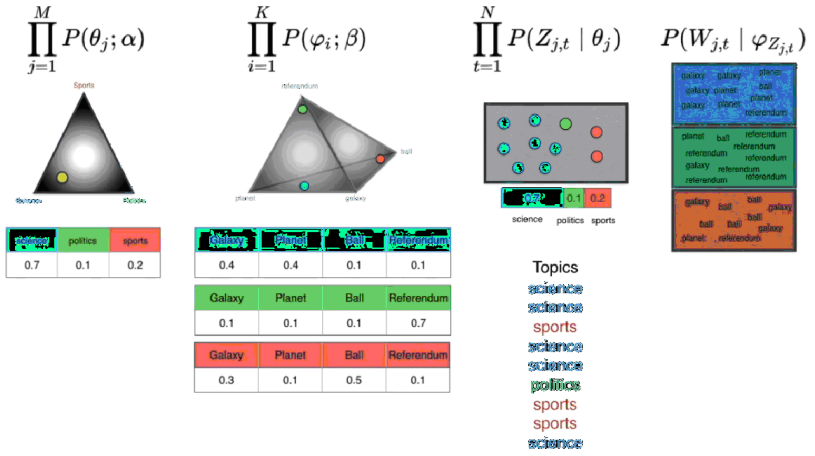


Figure 4.37: LDA (Using Topic Distribution to Sample Words)

distribution. We have 5 balls (sports), 1 referendum (politics), and 4 science words (3 galaxy's and 1 planet).

Once we have separately simulate topics and words, we now combine them, as in Figure 4.38. In Figure 4.38, we first look at the first topic that gets simulated: "science" (as the fat right arrow shows). Then we go to the first box (scienceblue, as the fat left arrow shows) and draw a word randomly. Say, we draw the word "planet". Hence, from the first topic science, we draw the word planet.

Now, we repeat the process. From the second topic which is also science, we draw again from the blue box and get the word "galaxy". Keeping the random draws until we finish the entire list of topics. See Figure 4.39 for the whole collection of

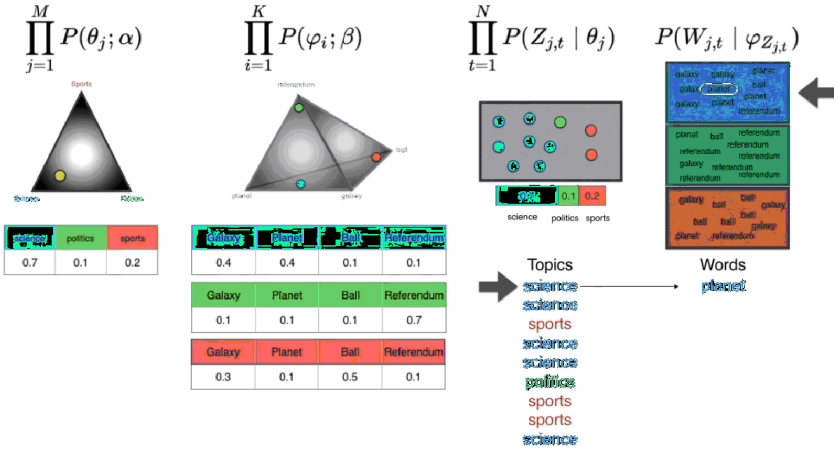


Figure 4.38: LDA (Topic to word)

words.

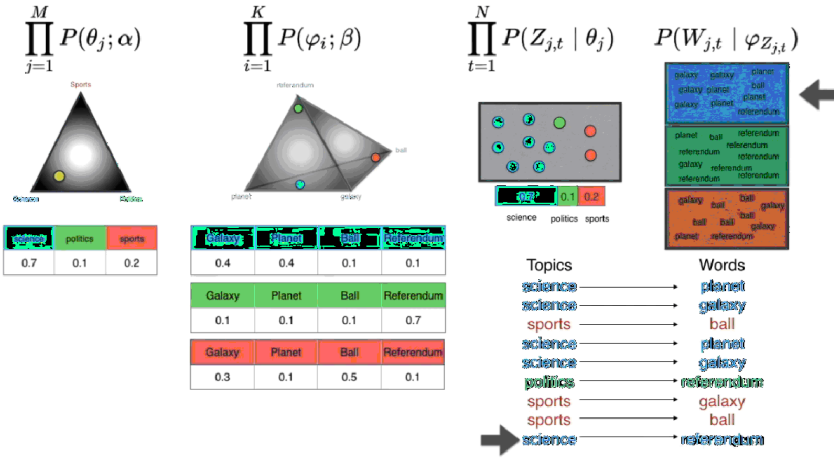


Figure 4.39: LDA (Topics to words)

The collection of words form a document as in Figure 4.40. Certainly, in this demonstration, the document is gibberish. But this is just an initial attempt.

The training is more like Figure 4.41. We generate a large number of documents and compare with the input documents. Certainly the initial match is very bad. But we keep iterating by adjusting the dials (settings) which are the two Dirichlet distributions and hopefully after enough iterations, the matching errors are minimal.

Actually, it is empirically seen that the actual matches are extremely bad.

Figure 4.40: LDA (Document generated)

Figure 4.41: LDA (Whole Corpus)

Yet, this is not the point. The main point is that even the actual match is very bad, a worse topic distribution is relatively much worse than a not-so-worse topic distribution. In other words, we are talking about relative errors, not absolute errors. As long as we could choose the best topic distribution, even thought it does not match the original document well, we have achieved our goal.

As a conclusion, we draw the two flowcharts side-by-side in Figure 4.42 so it is easy to compare and match distributions with the flowchart. With this visualization, it is quite easy to see that from each of the Dirichlet distributions, we simulate θ and φ. Then a multi-nomial simulation is done to generate topics z. And finally z and φ together are used to generate words which then form a document. Lots of such documents form a corpus.

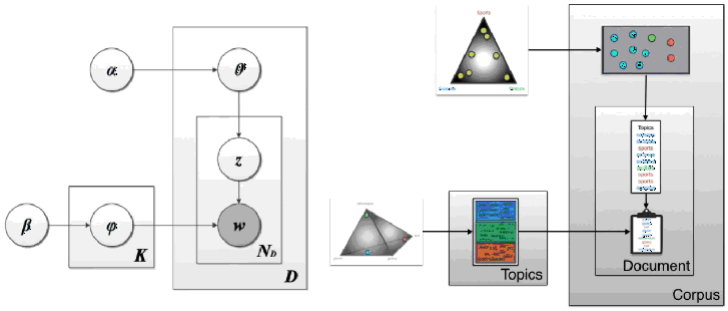


Figure 4.42: LDA (Two Flowcharts Side by Side)

One thing is left out of the example is when the documents are not equally long (i.e. different number of words). This is a more complex discussion and not covered here. In such a case, a Poisson distribution is used to handle documents of different lengths. Another discussion left out is Gibbs sampling. Gibbs sampling is used to train an LDA model.[4]

What we have discussed so far is that each document is associated with a known topic and so is each word. This requires a dictionary. In reality, there are numerous situations where such dictionary doesn't exist. In these situations, LDA must first randomly assign a topic to each word (but note that a word may have multiple topics (recall apple which can be a fruit or a phone). The goal is let both documents and words as monochromatic as possible. This is what Gibbs sampling does.

[4]For a detailed discussion of implementation, see https://medium.com/analytics-vidhya/latent-dirichelt-allocation-1ec8729589d4

4.5.2 Use of LDA

Once the LDA is trained, we can then compute the following. For every word in every document,W_n (where $n = 1, \cdots, N$), and for each topic Z_j (where $j = 1, \cdots, K$), we calculate:

- the proportion of words in document W_n that are currently assign to topic Z_j – $p(Z_j|W_n)$
- the proportion of assignments to topic Z_j over all documents that come from this word W_n – $p(W_n|Z_j)$
- reassign each word $W_{n^\dagger}$ to a new topic where we choose topic Z_{j^*} with probability $p(Z_{j^*}|W_{n^\dagger}) \times p(W_{n^\dagger}|Z_{j^*})$. This is essentially that topic Z_{j^*} generating word $W_{n^\dagger}$.

After repeating the previous step large number of times, we eventually reach a roughly steady state where the assignments are acceptable. At the end we have each document assigned to a topic. We can search for the words that have highest probability of being assigned to a topic.

4.5.3 Implementing LDA from Scratch, by Sihyung Park

1. https://naturale0.github.io/2021/02/14/LDA-1-background-topic-modelling
2. https://naturale0.github.io/2021/02/14/LDA-2-The-Model
3. https://naturale0.github.io/2021/02/15/LDA-3-Variational-EM
4. https://naturale0.github.io/2021/02/16/LDA-4-Gibbs-Sampling
5. https://naturale0.github.io/2021/02/17/LDA-5-Smooth-LDA

4.5.4 How to Determine the Optimal Number of Topics

"Selecting the Number and Labels of Topics in Topic Modeling: A Tutorial" by Sara J. Weston, Ian Shryock, Ryan Light, and Phillip A. Fisher, Advances in Methods and Practices in Psychological Science, April-June 2023, Vol. 6, No. 2, pp. 1–13.

A challenging step of topic modeling is determining the number of topics to extract.[5] There is no single correct number of topics for any corpus. Thus, topic modeling requires subjective decision making informed by the data and research questions. In the case of open-ended survey questions, one might consider the specificity of the prompt, the total number of responses, and the relative length of those

[5]See https://journals.sagepub.com/doi/pdf/10.1177/25152459231160105

responses. A final consideration is practical: Solutions with more topics take more time and computer memory to evaluate and are difficult to label.

The general procedure for identifying the ideal number of topics is to (1) examine the fit statistics of numerous possible solutions, (2) narrow down these solutions to a tractable number of candidate models, and (3) select one (or a small number) of models for further evaluation. Corpora can contain as few as three and as many as hundreds of topics. This is related both to the researcher's goals and the number, length, and complexity of responses in the corpora.

4.6 Appendix

https://en.wikipedia.org/wiki/Simplex

In geometry, a simplex (plural: simplexes or simplices) is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. The simplex is so-named because it represents the simplest possible polytope in any given dimension. For example,

a 0-dimensional simplex is a point,

a 1-dimensional simplex is a line segment,

a 2-dimensional simplex is a triangle,

a 3-dimensional simplex is a tetrahedron, and

a 4-dimensional simplex is a 5-cell.

Specifically, a k-simplex is a k-dimensional polytope that is the convex hull of its $k+1$ vertices. More formally, suppose the $k+1$ points are affinely independent, which means that the k vectors $u_1 - u_0, \cdots, u_k - u_0$ are linearly independent. Then the simplex determined by them is the set of points:

$$C = \left\{ \theta_0 u_0 + \cdots + \theta_k u_k | \sum_{i=0}^{k} \theta_i = 1, \theta_i \geqslant 0 \right\}$$

A regular simplex is a simplex that is also a regular polytope. A regular k-simplex may be constructed from a regular $(k-1)$-simplex by connecting a new vertex to all original vertices by the common edge length.

The standard simplex or probability simplex is the $(k-1)$-dimensional simplex whose vertices are the k standard unit vectors in

$$\{x \in \mathbb{R}^k : x_0 + \cdots + x_{k-1} = 1, x_i \geqslant 0\}$$

In topology and combinatorics, it is common to "glue together" simplices to form a simplicial complex. The associated combinatorial structure is called an abstract simplicial complex, in which context the word "simplex" simply means any finite set of vertices.

4.7 Homework: Hull's exercise

§ Homework

1. Hull's exercise
2. Run an LDA example (patent? Wiki?)

§ Project

Systemic Risk and Bank Networks: A Use of Knowledge Graph with ChatGPT.

Chapter 5

Reinforcement Learning

Don't Panic! Reinforcement learning is full of magical things patiently waiting for our wits to grow sharper.

Marlos C. Machado, University of Alberta

5.1 Introduction

Reinforcement learning (RL) is one of the four pure artificial intelligence models (in my opinion).[1] The typical example is mouse walking in a maze. Mice learn how to walk out of a maze by trial and error – the basic algorithm behind RL. As a result, it is then apparent that the RL algorithm must contain reward and penalty. A correct move will be rewarded and similarly a wrong move will be penalized. Mice learn how to walk out of a maze by following rewards.

Note that mice also need to explore in order to get out of a stuck. Hence in the RL algorithm, there is embedded a random action. It is this random action that provides the ultimate result. At the beginning, mice must explore a lot, and then as they become more experienced, they explore less (and exploit more).

RL has been used widely. For example, self-parking algorithm is a RL model. One can see https://www.youtube.com/watch?v=VMp6pq6_QjI. In this chapter, we explore how RL is used in pricing financial assets. For example, we will evaluate American options. At the end, an exotic derivative contract – swing contract is used as a demonstration. We begin this chapter by two simple examples – multi-armed

[1]The other three are swarm intelligence, genetic algorithm, and neural network. Note that my definition of AI is a model that must be behavioral.

bandit and game of nim. We use Hull's book for this.

5.2 Q Learning

5.2.1 Basics

Follow Hull (page 174) and define "Q" value as sum of rewards over time:

$$\begin{aligned} Q^{\text{new}} &= \frac{1}{T}\sum_{t=1}^{T} R(t) \\ &= \frac{1}{T}R(T) + \frac{T-1}{T}\frac{1}{T-1}\sum_{t=1}^{T-1} R(t) \\ &= \frac{1}{T}R(T) + \frac{T-1}{T}Q^{\text{old}} \end{aligned} \tag{5.1}$$

where there are a total of T iterations. This implies that the new value of Q is a weighted average of old value of Q and the new reward. This implies that in general, we can write updated Q as:

$$\begin{aligned} Q &\leftarrow \alpha R + (1-\alpha)Q \\ &= Q + \alpha(R - Q) \end{aligned} \tag{5.2}$$

where α is known as the learning rate. When it is big ($\rightarrow 1$), clearly we take full advantage of learning. When it is small ($\rightarrow 0$), then we don't learn at all (and Q will never change).

Equation (5.2) can be extended if there is more than one reward channels. In this case, each reward channel will learn on its own and we should also learn from all channels. For each reward channel, we use equation (5.2) as follows:

$$\begin{aligned} Q_k &\leftarrow \alpha_k R_k + (1-\alpha_k)Q_k \\ &= Q_k + \alpha_k(R_k - Q_k) \end{aligned} \tag{5.3}$$

and the overall Q is updated as follows:

$$\begin{aligned} Q &\leftarrow \alpha R_{k^*} + (1-\alpha)\max_k\{Q_k\} \\ &= Q + \alpha[R_{k^*} - \max_k\{Q_k\}] \end{aligned} \tag{5.4}$$

where k^* is the k which provides the maximum Q_k.

We can further augment Q to $Q(s, a)$ and R to $R(s, a)$ where s represents state and a represents action. In doing so, we are able to use Q-learning for a wide variety of problems. In this chapter, we provide several examples to demonstrate the use of Q-learning. In the multi-arm bandit game, action is $a = \{_0^1$ where 0 is exploitation and 1 is exploration. In the maze problem, state is the room the mouse is at and action is which room the mouse goes next.

5.2.2 Multi-armed Bandit – from Hull

The term multi-armed bandit comes from slot machines that are traditionally known as one-armed "bandits" because of their notorious reputation for taking money from players. Let there be 4 levers of a mult-armed bandit, as shown in Figure 5.1.

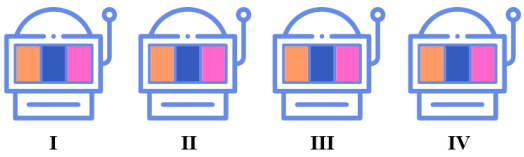


Figure 5.1: 4 Slot Machines

Each lever has a random payoff that is normally distributed with means of $\mu_1 = 1.2$, $\mu_2 = 1.0$, $\mu_3 = 0.8$, $\mu_4 = 1.4$ and the same standard deviations of 1. In this case, we have 4 duplicates of equation (5.2). That is, Q_k where $k = 1, \cdots, 4$. Since we need to choose the best lever, we also have a grand Q value.

Now we start a random lever to begin, say #2 and a random payoff is given as 1.812. In this case, $R_2 = 1.812$ and so is Q_2. Given that we only just pulled one lever, the grand Q value is also 1.812.

Since we don't know any better, let the second iteration be also random, say lever #3, and then a random payoff is drawn as -0.325. Now, $R_3 = -0.325$ and so is Q_3. Now we need to update our grand Q which is the average of the previous two values: Q_2 and Q_3:

$$\begin{aligned} Q &= \frac{Q_2 + Q_3}{2} \\ &= \frac{1.812 - 0.325}{2} = 0.743 \end{aligned}$$

For the sake of argument (i.e. every lever is drawn once), the third chice of lever is also randomly chosen to be #4 and the payoff (random) is 2.328. So

$R_4 = 2.328$ and so is Q_4. Again the grand Q is the average of all previous 3 values:

$$\begin{aligned} Q &= \frac{1.812 - 0.325 + 2.328}{3} = 1.271 \\ &= \frac{Q_2 + Q_3}{3} + \frac{Q_4}{3} \\ &= \frac{2}{3}\frac{Q_2 + Q_3}{2} + \frac{1}{3}Q_4 \end{aligned}$$

We can view the above calculation alternatively as:

$$Q^{\text{new}} = \frac{2}{3}Q^{\text{old}} + \frac{1}{3}Q_4$$

This is similar to equation (5.1) where the newly arrived reward is $R = Q_4$.

Again, for the sake of argument (i.e. every lever is drawn once), the fourth iteration is random and it is lever #1. The payoff is 0.431; and $R_1 = Q_1 = 0.431$. Now we need to update Q using equation (5.1):

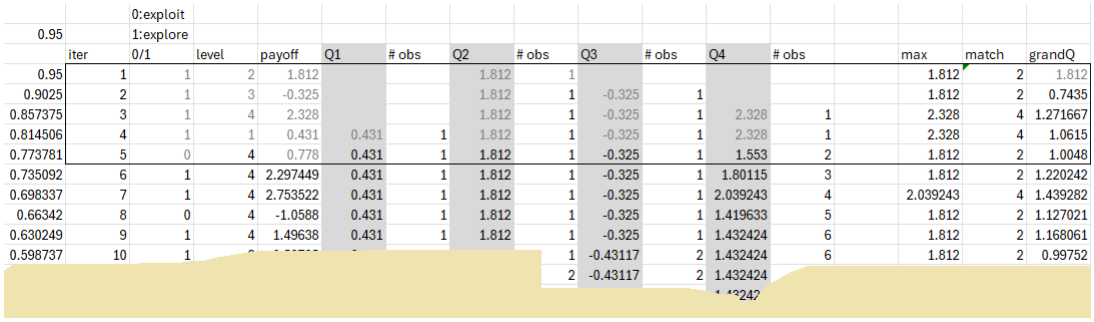
$$\begin{aligned} Q^{\text{new}} &= \frac{3}{4}Q^{\text{old}} + \frac{1}{4}Q_1 \\ &= \frac{3}{4}1.271 + \frac{1}{4}0.431 = 1.0615 \end{aligned}$$

Now, we start to toggle between exploitation (i.e. choose the best lever so far) and exploration (i.e. a random lever). The choice depends on a parameter β. The probability of exploration (i.e. random lever) at each iteration can be set as β^{t-1} (let's set β to be 0.9 for now). Say in the 5th iteration $\beta^{t-1} = 0.9^4 = 0.6561$ and we happen to draw a random number that is greater than 0.6561, which means we will exploit as opposed to explore. In this case, we will look at the past best performer which is lever #4. And a random draw of lever #4 gives a payoff of 0.778. Then, we can reculate Q_4 using equation (5.2).

There are two updates: one is Q_k itself (which is Q_4 right now) and the other is the general Q. The former is just (5.3) and the latter follows equation (5.4). Now, $k^* = 4$. $\alpha_4 = ½$ and $\alpha = ¼$. Then

$$\begin{aligned} Q_4 &\leftarrow 2.328 + (0.778 - 2.328)/2 = 1.553 \\ Q &\leftarrow 1.0615 + (0.778 - 1.061)/5 = 1.0048 \end{aligned}$$

The same process is repeated as many times as possible until the best lever is chosen. See bandit.xlsx (Figure 5.2). For this example, lever 4 is the best.



		0:exploit													
0.95		1:explore													
	iter	0/1	level	payoff	Q1	# obs	Q2	# obs	Q3	# obs	Q4	# obs		max	match	grandQ
0.95	1	1	2	1.812			1.812	1						1.812	2	1.812
0.9025	2	1	3	-0.325			1.812	1	-0.325	1				1.812	2	0.7435
0.857375	3	1	4	2.328			1.812	1	-0.325	1	2.328	1		2.328	4	1.271667
0.814506	4	1	1	0.431	0.431	1	1.812	1	-0.325	1	2.328	1		2.328	4	1.0615
0.773781	5	0	4	0.778	0.431	1	1.812	1	-0.325	1	1.553	2		1.812	2	1.0048
0.735092	6	1	4	2.297449	0.431	1	1.812	1	-0.325	1	1.80115	3		1.812	2	1.220242
0.698337	7	1	4	2.753522	0.431	1	1.812	1	-0.325	1	2.039243	4		2.039243	4	1.439282
0.66342	8	0	4	-1.0588	0.431	1	1.812	1	-0.325	1	1.419633	5		1.812	2	1.127021
0.630249	9	1	4	1.49638	0.431	1	1.812	1	-0.325	1	1.432424	6		1.812	2	1.168061
0.598737	10	1						1	-0.43117	2	1.432424	6		1.812	2	0.99752
								2	-0.43117	2	1.432424					
											1.43242					

Figure 5.2: Example

5.2.3 Game of Nim – from Hull

Nim is a strategy game in which two players take turns removing (or "nimming") objects from a pile. On each turn, a player must remove at least any number of objects (from 1 to as many as the rule defines). Whoever takes the last object loses. Nim is fundamental to the Sprague-Grundy theorem, which essentially says that every impartial game is equivalent to a nim game with a single pile.

In this game, we face a state (number of marbles left) and must decide an action to take (number of marbles to remove). In each simulation, a series of such state/action result in a win (gain \$1) or loss (lose \$1). We then run a number of such simulations. In each simulation, we need to update our $Q(s, a)$ value via the following formula:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha R$$

which is equation (5.2).

In Hull, an example is given as a pile of 8 marbles and each player (computer and I) can nim 1∼3 marbles each time. The learning rate is set at $\alpha = 0.05$. Assuming I go first and randomly choose 1 (7 left) marble to remove. Then the machine randomly chooses 3 (4 left). Then I randomly choose 1 (3 left). Then the machine chooses 3 (0 left) and I win. The reward is \$1.

In this simulation, when the state is 8, I choose 1 and when it is 4, I choose 3. Hence:

$$Q(8, 1) = 0.95 \times 0 + 0.05 \times \$1 = \$0.05$$
$$Q(4, 1) = 0.95 \times 0 + 0.05 \times \$1 = \$0.05$$

In the second simulation, I randomly choose 2 marbles to remove (6 left). Then the machine chooses 2 (4 left). Then I choose 1 (3 left). Then the machine chooses

2 which leaves only 1 for me to choose and I lose. The reward is $\$-1$. Hence:

$$Q(8,2) = 0.95 \times 0 + 0.05 \times (-\$1) = -\$0.05$$
$$Q(4,1) = 0.95 \times 0.05 + 0.05 \times (-\$1) = -\$0.0025$$
$$Q(1,1) = 0.95 \times 0 + 0.05 \times (-\$1) = -\$0.05$$

In the above $Q(s,a)$ values, two $Q(8,2)$ and $Q(1,1)$ are new; but $Q(4,1)$ has an existing value that needs to be updated. As we can see, the way to update its value is the same equation as in equation (5.2). So the game of nim is an augment of the multi-armed gambit game to include multiple states. In the multi-armed gambit game, there is always one lever to pull but here there are multiple states.

The process is repeated as many times as needed to reach the final result, as shown in Hull.

However, such a process takes a long time to converge. It is clear that when there are 2∼4 marbles left, I win (because I can then nim 1∼3 respectively and leave just one marble for computer). It is also clear that when there is 1 marble left, then I lose. Hence, we can take advantage of this known fact and stop simulation when the number of marbles is ≤ 4.

	1	2	3	4	5	6	7	8
1	-1	1	0	0	-1	1	0	-0.76
2	0	0	1	0	-1	-1	0	0.50
3	0	0	0	1	-1	-1	0	1

Table 5.1: Hull Example

5.2.4 Traveling Salesman's Problem

Manuel Amunategui demonstrates honey bee problem which is the same. He provides useful Python code. See https://amunategui.github.io/reinforcement-learning/index.html and https://www.youtube.com/watch?v=nSxaG_Kjw_w.

(wiki) In the theory of computational complexity, the travelling salesman problem asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" It is an **NP-hard** problem in combinatorial optimization, important in theoretical computer science and operations research. Go visit the wiki website for an animation of the problem.

A simple demonstration of the problem can be see in Figure 5.3 where we need to find the shortest path to connect all points.

Figure 5.3: Room

source: mathworld

Recall equation (5.4): $Q \leftarrow Q+\alpha[R_{k^*}-\max_k\{Q_k\}]$. We can rename the learning rate as follows (this is the key equation used in bee seeking honey and maze in the next section):

$$Q(s,a) \leftarrow R(s,a) + \gamma \max_k\{Q(s^+,a_k)\} \tag{5.5}$$

where s^+ is the state in the next iteration and $k = 1, \cdots, n$. (should $R(s,a)$ be $R(s,a_{k^*})$?)

What this equation says is that at the current state (at current time), we look at all next possible states and what the action would be taken in each every state, and we choose the best action a_{k^*} which maximizes Q (i.e. $Q(s^+,a_{k^*}) = \max_k\{Q(s^+,a_k)\}$).

As it will be clear later, how to structure learning is irrelevant. It is a balance matter between fast convergence (but possibly a local optimum) and global optimum.

The traveling salesman's problem, like many other problems, can be solved by swarm intelligence and genetic algorithms. Please see Chapters 3 and 10.

5.2.5 Maze – from XXX

Another popular example[2] is to walk out of a room (similar to walking out of a maze). In this example, rewards do not come sequentially; but rather already given in advance.

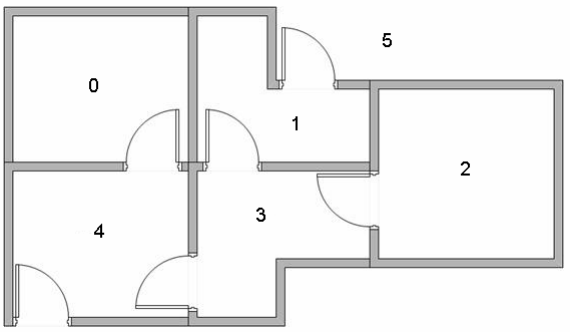


Figure 5.4: Room

In this room, there are 5 compartments labeled #0 $\sim$ #4. We also label the outside as #5. As we can see, not every room is connected to every other rooms. Hence, we draw a graph like the following:

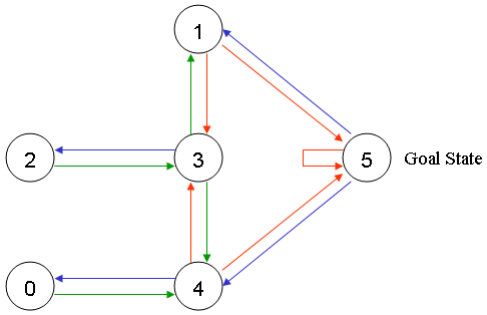


Figure 5.5: Maze

Now every compartment is a vertex and edges show how vertices are connected. It is now quite clear what we do if we want to get to the outside (i.e. vertex #5). For example, if we begin at #0, then the fastest way to get out is:

$$\#0 \to \#4 \to \#5$$

[2]The example shown in this sub-section is taken from https://www.sefidian.com/2022/09/06/reinforcement-learning-q-learning-numerical-example: A Painless Q-Learning Tutorial.pdf

Certainly, one can get out via:

$$\#0 \to \#4 \to \#3 \to \#1 \to \#5$$

but that would be inefficient. In order the machine can learn the fastest route, we must put in reward values. Note that, different from the multi-armed bandit example, here the rewards are pre-given (and not randomly drawn). Quite arbitrarily, we can set the rewards as follows:

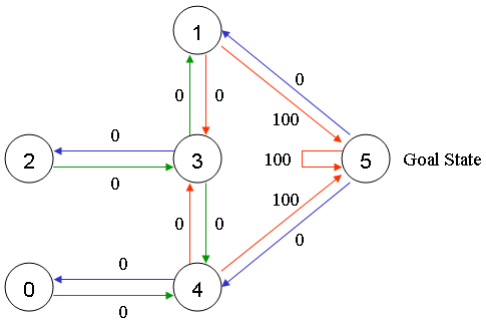


Figure 5.6: Rewards

where only direct routes to outside (i.e. #5) is given a very high value (e.g. 100) whereas everywhere else is assigned 0 (because there is no difference between any two vertices unless they can lead to outside). We write these values in a (reward) matrix, as follows:

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Figure 5.7: Reward R Matrix

and the inital Q matrix as follows:
which are all 0's.

In the reward matrix, routes (i.e. edges) that do not connect any vertex are given a value of -1 (can be any negative value), just a little penalty so that machine knows this is not a desirable route. Note that again, different from mult-armed bandit, now rewards/penalties are based upon state and action whereas in multi-armed bandit it is based upon action only since there are no states in mult-armed bandit.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 5.8: Initial Q Matrix

Now, we can start our iterations (as in multi-armed bandit). Suppose the agent is in state 2. From state 2, he can only go to state 3. From state 3 he can go either state 1 or state 4. Then it is quite straightforward to go to state 5 which outside of the house.

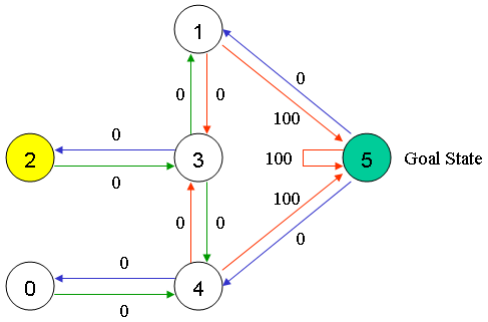


Figure 5.9: Graph

By equation (5.5) as follows:

$$Q(s,a) \leftarrow R(s,a) + \gamma \max_k \{Q(s^+, a_k)\} \tag{5.6}$$

where s^+ is the state in the next iteration and $k = 1, \cdots, n$. (should $R(s,a)$ be $R(s,a_{k^*})$?)

If the initial state is room #1, then from the reward matrix in Figure 5.7, the agent should obviously choose action 5 which will lead to the outside. This is a convergent state in that in Figure 5.7, state 5 will also stay in state 5 by taking action 5. As a result, the agent will stop moving. But let's compute the Q matrix anyway (just to see how it works).

$$\begin{aligned} Q(1,5) &\leftarrow R(1,5) + \gamma \max_k \{Q(s^+, a_k)\} \\ &= R(1,5) + \gamma \max\{Q(5,1), Q(5,4), Q(5,5)\} \\ &= 100 + 0.8 \cdot 0 \\ &= 100 \end{aligned}$$

given that $Q(5,1) = Q(5,4) = Q(5,5) = 0$. Then, $Q(1,5) = 0$ is replaced by $Q(1,5) = 100$.

Now, we are in state 5. Apply the same equation:

$$\begin{aligned} Q(5,5) &\leftarrow R(5,5) + \gamma \max_k \{Q(s^+, a_k)\} \\ &= R(5,5) + \gamma \max\{Q(5,1), Q(5,4), Q(5,5)\} \\ &= 100 + 0.8 \cdot Q(5,5) \\ &= 100 + 0.8 \cdot 100 \\ &= 180 \end{aligned}$$

which replaces $Q(5,5) = 0$ is replaced by $Q(5,5) = 180$. If the agent keeps going, he will always in state 5 and $Q(5,5)$ will become 244 (which is $100 + 180 \times 0.8$). This is a simple sum of a geometric series with 0.8 ratio and equal to $\frac{1}{1-0.8} = 4$. Hence, we know the ultimate Q value for $Q(5,5)$ is $100 \times (1+4) = 500$.

Let's begin with another room, say room #3. From the reward matrix, we know that it can go to rooms 1, 2, and 4. Given that the payoffs are all 0 (indifferent), it is randomly decided to be room 1. So

$$\begin{aligned} Q(3,1) &\leftarrow R(3,1) + \gamma \max_k \{Q(s^+, a_k)\} \\ &= R(3,1) + \gamma \max\{Q(1,3), Q(1,5)\} \\ &= 0 + 0.8 \cdot 0 \\ &= 0 \end{aligned}$$

So there is no update of $Q(3,1)$ which remains 0. Now we are at room 1. Clearly, this repeats what is discussed previously and it will lead to outside very quickly. We are lucky!

It is clearly that if we move to room 4, we will get outside immediately as well.

Let's say, unluckily, we draw a random number that leads us to room 2. Then,

$$\begin{aligned} Q(3,2) &\leftarrow R(3,2) + \gamma \max_k \{Q(s^+, a_k)\} \\ &= R(3,2) + \gamma \max\{Q(2,3)\} \\ &= 0 + 0.8 \cdot 0 \\ &= 0 \end{aligned}$$

which will lead us right back to room 3 in the next iteration, given there is nowhere else to go from room 2.

If we keep iterating, sooner or later, we will get lucky and go to either room 1 or room 4. We can do this for every room and find the optimal path.

§ Alternative Calculation

Or alternatively, since computing is cheap, we just start randomly from any room k and update $Q(k, k^+)$ where k^+ is the next allowable room (but do not continue). And we keep repeating this until the Q matrix converges (as follows).

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

Figure 5.10: Q Matrix after Many Iterations

Then, from this final Q matrix, we can find the optimal path of any room. If we start from room #3, we will go to room #1 or #4 since their payoffs are same (i.e. 400). We will not go to room 2 since the payoff is less (i.e. 256). Then from room #1 or #4 we will go right outside since the payoff is 500.

This alternative calculation (which leads to the same result) is very important, in that we can now expand the algorithm to include more complex structures. One immediate expansion is to add time (so that we can incorporate stochastic processes). When we add time and stochasticity, it is very hard for us to trace one path at a time. It is better that we finish computing the Q matrix at once using all possible initial states and then trace the elements of the Q matrix to find the optimal path per initial state. We explain this in the next sub-section using the swing contract.

It is useful to switch $Q(k, k^+)$ to $Q(s, a)$ where s represents the current state (e.g. current room) and a represents action (i.e. which room to move into next). We can write the above iterative steps as:

$$Q(s, a) \leftarrow R(s, a) + \gamma \max_k \{Q(s^+, a_k)\}$$

where s^+ is the next state.

§ Add α

Now we can add α into the algorithm. First of all, we recognize equation (5.5) as a whole as a "gain":

$$G = R(s, a) + \gamma \max_k \{Q(s^+, a_k)\}$$

Then, we simply use this gain to replace the reward in equation (5.2), which

is $Q \leftarrow \alpha R + (1-\alpha)Q$, to update Q with G instead of R:

$$\begin{aligned} Q(s,a) &\leftarrow \alpha G + (1-\alpha)Q(s,a) \\ &= Q(s,a) + \alpha(G - Q(s,a)) \\ &= Q(s,a) + \alpha\left[R(s,a) + \gamma \max_k\{Q(s^+,a_k)\} - Q(s,a)\right] \\ &= (1-\alpha)Q(s,a) + \alpha\left[R(s,a) + \gamma \max_k\{Q(s^+,a_k)\}\right] \end{aligned} \tag{5.7}$$

which is the version used by Wikipedia.

We can see that this is a weighted average of the current Q value and an improvement. It is often expressed as follows:

$$Q(s,a) + \alpha\left[R(s,a) + \gamma \max_k\{Q(s^+,a_k)\} - Q(s,a)\right] \tag{5.8}$$

where the bracketed term is the entire reward (which includes the current reward $R(s,a)$ and an improvement discounted by γ.)

There is an alternative version to the above equation (5.7).

$$\begin{aligned} Q(s,a) &\leftarrow R(s,a) + (1-\alpha)Q(s,a) + \alpha\gamma \max_k\{Q(s^+,a_k)\} \\ &= R(s,a) + Q(s,a) + \alpha\left[\gamma \max_k\{Q(s^+,a_k)\} - Q(s,a)\right] \end{aligned} \tag{5.9}$$

As we can see, how learning takes place is not crucial. It is merely a balance of higher likelihoods to achieve global optimum (which needs to learn slowly) and efficiency (but might lands on a local optimum).

5.3 Markov Decision Process

In the above section, the Q learning algorithms are static. Q learning can be combined with the Bellman equation and be used in a dynamic setting.

$$V(s) = \max_a\left(R(s,a) + \gamma \sum\nolimits_{s^+} p(s,a,s^+)V(s^+)\right)$$

where $p(\cdot)$ is transition probability from state s to state s^+.

5.3.1 Basics

§ Augmented (5.1)

Reinforcement Learning: Markov-Decision Process (Part 1) by Ayush Singh

Let policy π be a probability (p.12):

$$\pi(a|s) = p(A_t = a|S_t = s)$$

Then (p.19),

$$\begin{aligned} Q_t^\pi(s,a) &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi [R_{t+1} | S_t = s] + \mathbb{E}_\pi \left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi [R_{t+1} | S_t = s, A_t = a] + \mathbb{E}_\pi [Q_{t+1}^\pi(s,a)] \end{aligned}$$

This is the Bellman equation.

§ Augmented (5.5)

In general, equation (5.5) needs to be augmented to include time, as follows.

Wikipedia:

$$Q^{new}(S_t, A_t) \leftarrow (1 - \underbrace{\alpha}_{\text{learning rate}}) \cdot \underbrace{Q(S_t, A_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \Big(\underbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(S_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} \Big)$$

Figure 5.11: Q Learning Wikipedia

5.3.2 The Longstaff-Schwartz Model

The LS model is an American option pricing model using Monte Carlo simulations. As well-known, lattice models are best for American options since exercises require backward induction.[3]

The last period (maturity) yields the put exercise payoff:

[3]"Valuing American Options by Simulation: A Simple Least-Squares Approach," Review of Financial Studies, Spring 2001 Vol. 14. No. 1, pp. 113-147.

t=0	t=1	t=2	t=3
1	1.09	1.08	1.34
1	1.16	1.26	1.54
1	1.22	1.07	1.03
1	0.93	0.97	0.92
1	1.11	1.56	1.52
1	0.76	0.77	0.9
1	0.92	0.84	1.01
1	0.88	1.22	1.34

Table 5.2: Longstaff-Schwartz Example

stock (t=2)	stock (t=3)	put (t=3)	put (disc)
1.08	1.34	0	0
1.26	1.54	0	0
1.07	1.03	0.07	0.065924
0.97	0.92	0.18	0.169518
1.56	1.52	0	0
0.77	0.90	0.20	0.188353
0.84	1.01	0.09	0.084759
1.22	1.34	0	0

Table 5.3: Put Payoff, $K = 1.1$

Now we use this for regression. Note that continuation value of the put option is:

$$\mathbb{E}_t \left[e^{-r\Delta t} C_{t+1} | S_t \right] = f(S_t)$$

and we know that and conditional expectation is a function of the current stock price. This also implies (because $\mathbb{E}[e_{t+1}|S_t] = 0$):

$$e^{-r\Delta t} C_{t+1} = \mathbb{E}_t \left[e^{-r\Delta t} C_{t+1} | S_t \right] + e_{t+1}$$

is a regression:

$$\begin{aligned} e^{-r\Delta t} C_{t+1} &= f(S_t) + e_{t+1} \\ &= \beta_0 + \beta_1 S_t + \beta_2 S_t^2 + e_{t+1} \end{aligned}$$

The quadratic equation is merely an example (used by Longstaff and Schwartz). Later authors find better functions to use. This is known as the basis function in Q learning.

At the last period (maturity, t=3), we know C_3 which is the payoff. Hence, we can run the following regression:

$$e^{-r\Delta t} C_3 = \beta_0 + \beta_1 S_2 + \beta_2 S_2^2 + e_3 \tag{5.10}$$

For some unspecified reason, Longstaff and Schwartz recommend to use only in-the-money paths for the regression. These are the red numbers in Table 5.3. Note that in t=2, in-the-money paths are those stock prices less than the strike (1.1). The regression result is in Table 5.4.

β_0	-1.06999
β_1	2.98341
β_2	-1.81358

Table 5.4: Betas (t=2)

We can use the regression coefficients in equation (5.10) to calculate the continuation values of various paths. Then, we can compare the continuation values ($\hat{y}$) and exercise values in Table 5.5.

where the red numbers are those paths where the put option is exercised (i.e. the exercise value is greater than the continuation value). They are paths 4, 6, and 7.

Now, we move back to t=2. The payout (and its discount) now, from above, is:

t=2	ex value	cont value
1.08	0.02	0.036741
1.26	0	0
1.07	0.03	0.045898
0.97	0.13	0.117527
1.56	0	0
0.77	0.33	0.151969
0.84	0.26	0.156418
1.22	0	0

Table 5.5: Continuation vs. Exercise Values

t=1	payout	pay (disc)
1.09	0	0
1.16	0	0
1.22	0	0
0.93	0.13	0.122429
1.11	0	0
0.76	0.33	0.310782
0.92	0.26	0.244859
0.88	0	0

Table 5.6: Put Payoff

β_0	2.03751
β_1	-3.33544
β_2	1.35646

Table 5.7: Regression

Out of these paths, the in-the-money ones are those in red. Hence, in the regression, we can only use these paths. The regression result is in

Then, we can compute the continuation values for t=2. From Table 5.8, we know that they are paths 4, 6, 7, and 8. Note that paths 4, 6, and 7 are the same paths that are exercised at t=2, clearly these paths will be exercised here at t=1 instead of t=2. In other words, no path will be exercised at t=2. In conclusion, the put option will be exercised at t=1 along the paths of 4, 6, 7, and 8. No exercise at t=2. Then the put will be exercise along path 3 at t=3 (maturity).

t=1	ex value	cont value
1.09	0.01	0.013485
1.16	0	0
1.22	0	0
0.93	0.17	0.108749
1.11	0	0
0.76	0.34	0.286065
0.92	0.18	0.117009
0.88	0.22	0.152762

Table 5.8: Exercise vs. Continuation Values

Now we can calculate the put value using the exercise values at those paths and take an average and we have 0.114434.

5.3.3 LSPI

Q learning discussed in the previous section can be extended to a dynamic environment (a.k.a. the Markov decision process, or MDP). To use MDP for American options, we adopt a similar trick as the Longstaff-Schwartz model – least squares policy iteration (LSPI).

The material here is based upon Li, Szepesvari and Schuurmans (2009, p.353)[4] and the tutorial notes by Ashwin Rao (2020).[5]

Let there be m paths and n periods. We need to loop over all policies to find the maximum $Q(s, a)$.

[4]Li, Yuxi, Csaba Szepesvari and Dale Schuurmans, 2009, "Learning Exercise Policies for American Options," Proceedings of the 12th International Conference on Artificial Intelligence and Statistics. `li09d.pdf`

[5]"RL for Optimal Exercise of American Options," 2020.
See https://web.stanford.edu/class/cme241/lecture_slides.

Let

$$\begin{aligned} Q(s,a) &= \sum_{k=0}^{\kappa} w_k \phi_k(s,a) \\ &= w_0 + w_1 S + w_2 S^2 \end{aligned}$$

So, given a decision matrix, choose $\{w_0, w_1, w_2\}$ to minimize the following SSE:

$$\begin{aligned} &\sum_{s,a,r,s^+} [r + \gamma Q(s^+, \pi(s^+)) - Q(s,a)]^2 \\ =&\sum_{s,a,r,s^+} \left[0 + \gamma Q(s^+, \pi(s^+)) - (w_0 + w_1 S + w_2 S^2)\right]^2 \end{aligned} \tag{5.11}$$

where "+" represents the value of the next time period.

Now we consider $Q(s^+, \pi(s^+))$:

- when $K - S^+ \geqslant w_0 + w_1 S^+ + w_2 (S^+)^2$ then $Q(s^+, \pi(s^+)) = K - S^+$
- when $K - S^+ < w_0 + w_1 S^+ + w_2 (S^+)^2$ then $Q(s^+, \pi(s^+)) = w_0 + w_1 S^+ + w_2 (S^+)^2$

Then (5.11) becomes:

$$\min_{\underline{w}} \sum_{s,a,r,s^+} \left[\gamma \times \{^{K - S^+}_{w_0 + w_1 S^+ + w_2 (S^+)^2} - (w_0' + w_1' S + w_2' S^2)\right]^2 \tag{5.12}$$

Li, Szepesvari and Schuurmans seem to suggest that $\sum_{s,a,r,s'}$ is to sum over all paths over all periods. If so, then there is no iteration over policies.

Equation (5.12) can be solved analytically, given that it is a linear system, and the solution is analogous to linear regression (hence named least squares). Yet, this is not necessary as numerical optimizers are fast and quick.

Ashwin Rao recommends a sample of basis (feature) functions p.14.[6] We find that this is very important when applying GPU.[7]

$$\begin{aligned} \phi_0 &= 1 \\ \phi_1 &= e^{-1/2 S'} \\ \phi_2 &= e^{-1/2 S'} (1 - S') \\ \phi_3 &= e^{-1/2 S'} (1 - 2S' + 1/2 S'^2) \end{aligned}$$

where $S' = S/K$.

[6] AmericanOptionRL.pdf – slides

[7] This is because GPU does not allow optimizers and must adopt analytical solutions, and hence matrix inversions are a must.

5.3.4 Swing Contract

A swing contract is a commodity derivative that tries to maximize the profit of trading the commodity. For the sake of easy argument, let's use (liquid) natural gas (in mega tons) as the commodity. For simplicity, we assume that the purchase and sale prices are the same. So the only way to make money is to buy when the price is low and wait to sell when the price rises. In doing so, we must obey the storage limits, which is the major challenge that we need a model for. For further simplicity, we also assume there is no storage cost (i.e. we can store natural gas for free indefinitely).

The swing contract is more complex than the maze example in that there is an additional time dimension. This is a step toward building a RL model for MDP (Markov decision process). Hence, the rewards are not just a fixed matrix and the Q matrix is not one matrix but a collection of matrices over time. As a first step, we assume all prices over time are known so we can see the connection between the maze example and the swing contract. Later, we will allow prices to be stochastic (i.e. MDP).

In the previous maze example, action is defined as one of the reachable rooms from the existing room which is shown on each row. Here in the swing contract, action is "buy", "hold", or "sell" (represented numerically as "1", "0", or "-1").

In the previous maze example, state is given as the existing room (shown on columns). Here, state is how much quantity we currently hold.

In the previous maze example, rewards are fixed and given. Here, rewards need to be computed at each time and are defined as negative buy price, 0, and positive sell price, since we pay to buy (i.e. negative cash) and receive cash by selling. Quantity of buy or sell is 1 for simplification.

In this swing contract example, we follow a paper by Behrndt and Chen (2022).[8] where a hypothetical example is given. The contract has 22 days till maturity and all 22 futures prices are known and given. The maximum storage limit is 5 units.

On each day, a decision must be made to buy, hold or sell. The price is given in Table 5.9 (which is either \$5 or \$10).

On day 1, the state is 0 since there is no inventory. On this day, we must decide to hold or buy (cannot sell since there is nothing to sell). If we buy, the reward is $-\$5$; and if we hold, then the reward is \$0. The decision is random, since

[8]Behrndt, Tapio, and Ren-Raw Chen, 2022, "A New Look at the Swing Contract: From Linear Programming to Particle Swarm Optimization," Journal of Risk and Financial Management, 15(6), 1-20.

Day	1	2	3	4	5	6	7	8	9	10	11
Price	\$5	5	10	10	5	5	10	10	5	5	10
Day	12	13	14	15	16	17	18	19	20	21	22
Price	\$10	5	5	10	10	5	5	10	10	5	5

Table 5.9: Natural Gas Futures Prices

we don't know any better, one way or the other.

This action and its resulting reward $R(s_t, a) = \{^0_{-5}$ where $t = 1$.

In the equation

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[R(s_t, a) + \gamma \max_k \{Q(s_{t+1}, a_k)\} - Q(s_t, a) \right]$$

we know that $Q(s_{t+1}, a_k) = 0$ for all k since there is no value at this initial stage; and hence $\max_k \{Q(s_{t+1}, a_k)\} = 0$. As a result,

$$\begin{cases} Q(s_1, 0) \leftarrow 0 + 0.9 \times [0 + 0.8 \times 0 - 0] = 0 \\ Q(s_1, 1) \leftarrow 0 + 0.9 \times [-5 + 0.8 \times 0 - 0] = -4.5 \end{cases}$$

Now we move to $t = 2$. If the action at $t = 1$ is hold, then nothing will happen ever. If the action at $t = 1$ is buy, then there are two possible decisions in $t = 2$: either hold or sell. Since sell gives a positive payoff of \$5. In this situation, $Q(s_2, -1) = 4.5$, as shown below (this is because $Q(s_2, -1)$ is computed by the same equation which must take into account of $Q(s_3, a)$ values, i.e. $\max_k \{Q(s_3, a_k)\}$):

$$\begin{cases} Q(s_2, 0) \leftarrow 0 + 0.9 \times [0 + 0.8 \times 0 - 0] = 0 \quad (*) \\ Q(s_2, -1) \leftarrow 0 + 0.9 \times [5 + 0.8 \times 0 - 0] = 4.5 \end{cases}$$

where all $Q(s_3, a)$ values are 0.

The process will stop here. This is because the sell action will leave no inventory and the process naturally stops. (Although the hold action (*) will leave 1 unit of inventory for $t = 3$ to sell (or any future t to sell) but that action will never be allowed since at $t = 1$, the best action for the next time period is to sell (because \$5 > \$0).)

In the next iteration, the above results will then used to revise $Q(s_1, 1)$ from -4.5 to -0.9405. Updates from iterations are given as follows (Note that Q values are not necessarily updated in each iteration since some iterations the decision is hold. Only when a buy action is taken there will be an update of Q values)

- $Q(s_1, 1) = -4.5$ because $Q(s_2, -1) = 0$
- $Q(s_1, 1) = -0.9405$ because $Q(s_2, -1) = 4.5$
- $Q(s_1, 1) = -0.1836$ because $Q(s_2, -1) = 4.95$
- $Q(s_1, 1) = -0.067815$ because $Q(s_2, -1) = 4.995$
- $\cdots$
- $Q(s_1, 1) = -0.05$ because $Q(s_2, -1) = 5$

Apparently this is not an ideal solution, let alone the optimal solution. The reason is that except for the initial state, there is no exploration afterwards. There is no possibility of buy or hold in $t = 2$ since sell dominates (i.e. generating positive payoff which then always get selected via $\max_k\{Q(s_{t+1}, a_k)\}$).

As a natural remedy, we simply let each time the agent can choose buy, hold, or sell randomly. We allow a small chance (say, ε) that a decision is made randomly (with equal probability). In this case, all sorts of possibilities can happen, and through this exploration, we will reach the optimal solution.

Note that each run/iteration will almost surely not an optimal path, and yet, as in the maze example, by keeping iterating the calculation from $t = 1$ to $t = 22$ many times, since in each iteration, values from the previous iterations are utilized, we will reach the final Q values that not only converge but are optimal. Then we can, just like the maze example, trace the Q matrices and find the optimal path.

This is quite important in Q-learning in that exploration is the only way there is any chance of reaching the optimal path.

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[R(s_t, a) + \gamma \max_k \{Q(s_{t+1}, a_k)\} - Q(s, a) \right]$$

$$Q(s_1, a) \leftarrow 0 + 0.9 \times \left[\{^0_{-5} + 0.8 \times 5 - 0 \right] = \{^0_{-4.5}$$

- there are 22 days, so there are 22 Q matrices – row 8, 15, 22, ... , 155 and (col I)
- in each Q matrix, actions are vertical and states are horizontal
- Q matrix grows as number of states grows
- "max" is the maximum payout over various actions under each state (row 12, 19, ..., 159)

- "action" is to choose (via vlookup) the action of the highest payout, which is chosen when $u > \varepsilon$
- "random action" is chosen if $u < \varepsilon$

5.4 Homework

§ Homework

1. Solve the following traveling saleman's problem:

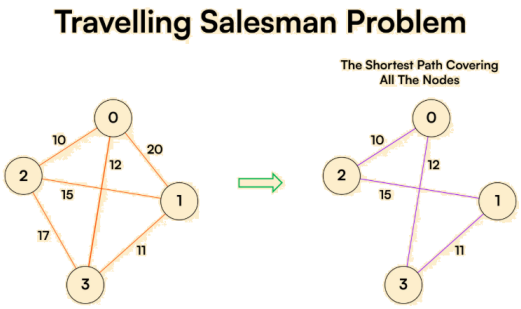


Figure 5.12: Lystloc: TSP

2. Replicate the LS example (2001).

§ Project

1. Redo LS with 10,000 paths and 100 periods.
2. Do LSPI with 10,000 paths and 100 periods.

Chapter 6

Undirected Graph

We pass by imperceptible gradations from the brute to the savage and from the savage to Euler and Newton.

Marquis Marie Jean Antoine Nicolas Caritat Condorcet, 1743 - 1794.

6.1 Introduction

Ever be puzzled why coffee mug and donut are the same? Ever heard of the 4-color theorem that you can color all the states of the United States with no more than 4 colors and no adjancent states have the same color? Ever heard of the 7-bridge tale that you can never cross each bridge only once? It is time to know Leonard Euler who provides the answers to all the above questions.

Graph theory is another important mathematical branch, equally important as Newton's calculus. It is originated from Leonard Euler's 7 bridge theorem (Königsberg Bridge Problem) on August 26, 1735. Königsberg is known as Kaliningrad today. It is an exclave of Russia inside of the Baltics (particularly Lithuania). See https://kostelisevich.com/2023/04/01/the-kaliningrad-oblast-a-time-to-detach of a map in Figure 6.1.

According to Wikipedia, "In mathematics, graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices (also called nodes or points) which are connected by edges (also called arcs, links or lines). A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically. Graphs are one of the principal objects of study in discrete mathematics."

Figure 6.1: Königsberg(Kaliningrad)

source: https://kostelisevich.com/2023/04/01/the-kaliningrad-oblast-a-time-to-detach

The first part of this chapter leverages heavily on Sarada Kerke's lectures.

6.2 Basic Definitions

Let G be a graph, $V(G)$ be vertices (nodes) of the graph, and $E(G)$ be edges of the graph. $N_G[v] = \{u | u, v \in E(G)\}$ is the neighborhood of v. $|E|$ is the number of edges and $|V|$ is the number of vertices. $|R|$ is the number of regions.

§ Clique

A clique of a graph G is a set X of vertices of G with the property that every pair of distinct vertices in X are adjacent (complete sub-graph). A maximal clique of a graph G is a clique X such that there is no clique Y that contains all of X and at least one other vertex. In Figure 6.2, the red circle is a clique but not maximal clique, but blue circle is the maximal clique.

The size of the biggest clique is symbolized as $\omega(G)$. In the above example, $\omega(G) = 4$.

§ Degree

The degree of a vertex is the number of edges connecting that vertex. The highest degree is symbolized as $\delta(G)$.

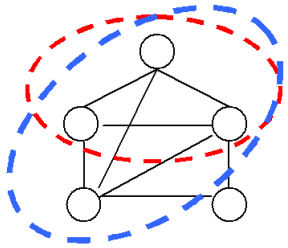


Figure 6.2: Clique

A graph G is r-regular if $\deg_G(v) = r$ for all $v \in V(G)$.

§ Chromatic Number

$\chi(G)$, called the chromatic number, is the smallest number of colors used to color (the vertices of) the graph.

§ Colorable

A graph is called k-colorable if its vertices can be colored by k colors without adjacent vertices sharing the same color.

§ Minor and Subgraph

A subgraph G' of a graph G is another graph formed from a subset of the vertices and edges of G. In other words, it is a subset of the larger original graph. Or alternatively, we can understand G' that is formed by deleting edges and/or evetices of G.

An undirected graph H is called a minor of the graph G if H can be formed from G by deleting edges, vertices and by contracting edges. A minor of a graph G describes a substructure of G that is more general than a subgraph.

If we take a subgraph of G and then contract some connected pieces in this subgraph to single points, the resulting graph is called a minor of G.

§ Adjacency Matrix

The adjacency matrix, sometimes also called the connection matrix, of a simple labeled graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are adjacent or not.

For example, see Figure 6.3.

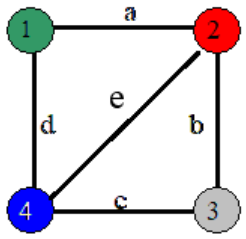


Figure 6.3: Adjacency Matrix

We can write the adjacency matrix as:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

The above adjacency matrix is expressed in terms of vertices. There is an alternative way to write the adjacency matrix via the graph's edges. We label Figure 6.3's edges as follows:

Then, we re-write the adjacency matrix as follows:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

§ Independent Set

An independent set is a set where no two vertices in the set are adjacent.

A k-coloring of G partitions the vertex set V into k sets $V_1, V_2, \cdots, V_k$.

This means $V = V_1 \cup \cdots V_k$ and $V_i \cap V_j = \{\}$.

$V_1, V_2, \cdots, V_k$ are then called color classes.

Peterson graph, as shown in Figure 6.4 can be separated into independent sets. The red set $V_{\text{red}} = \{v_1, u_3, u_4\}$ forms an independent set, and so do the blue set $V_{\text{blue}} = \{u_2, v_3, v_5\}$ and the green set $V_{\text{green}} = \{u_1, u_5, v_2, v_4\}$. Note that the union of all three indepedent sets is the whole graph, $V = V_{\text{red}} \cup V_{\text{blue}} \cup V_{\text{green}}$. This is an example of 3-coloring of the Petersen graph.

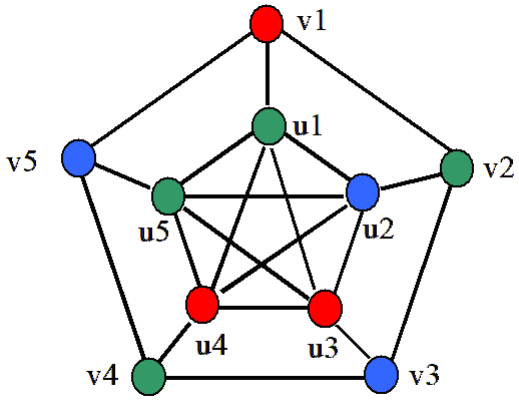


Figure 6.4: Independent Set

§ Girth

wiki: In graph theory, the girth of an undirected graph is the length of a shortest cycle contained in the graph. If the graph does not contain any cycles (that is, it is a forest), its girth is defined to be infinity. For example, a 4-cycle (square) has girth 4. A grid has girth 4 as well, and a triangular mesh has girth 3.

A graph with girth four or more is triangle-free.

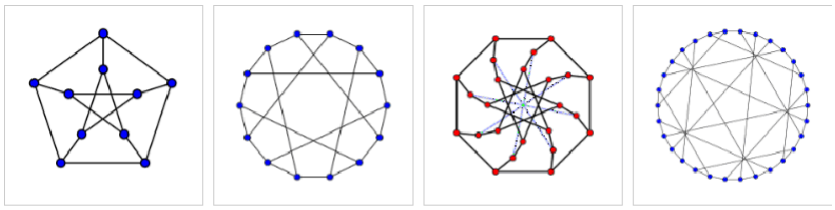


Figure 6.5: Girth

6.3 Basic Graphs

As we can see, a graph in graph theory is a network graph. It describes how vertices are connected. In other words, through a graph (network), we can study and understand how relationships among vertices are. In this section, we first review some basic graphs.

6.3.1 Chain Graph

An simplest chain graph can be seen in Figure 6.6. This is an undirected graph.

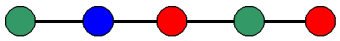


Figure 6.6: Chain Graph

A tree graph can be easily seen as a chain graph, as in Figure 6.7.

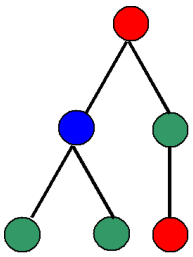


Figure 6.7: Tree Graph

Note that Figure 6.8 is also a chain graph. In this graph there are certain edges that are directional. Note that the graph would not be a chain graph if those directions are removed (then 1-2-4 become a cycle). Directed graphs (directional edges) will be discussed in the next chapter.

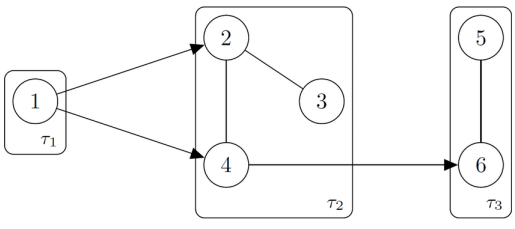


Figure 6.8: Chain Graph

6.3.2 Cycle Graph

Usually labeled as C_n where n is number of vertices. In graph theory, a cycle graph or circular graph is a graph that consists of a single cycle, or in other words, some number of vertices (at least 3, if the graph is simple) connected in a closed chain.

(a) Chain Graph (b) Cycle Graph

Figure 6.9: Chain and Cycle Graphs

Note that although Figure 6.9a doesn't look like Figure 6.6, they are same graphs.

6.3.3 Complete Graph

A complete graph is a graph in which each vertex is connected to every other vertex. Such graphs are denoted as K_n.

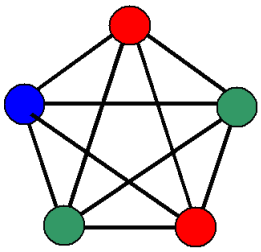


Figure 6.10: Complete Graph: K_5

A complete bipartite graph or biclique is a special kind of bipartite graph where every vertex of the first set is connected to every vertex of the second set. Such graphs are denoted as $K_{m,n}$, seen in Figure 6.11.

6.3.4 Bipartite Graph

A bipartite graph is a graph where the vertices can be divided into two disjoint sets such that all edges connect a vertex in one set to a vertex in another set. There are

no edges between vertices in the disjoint sets.

Figure 6.11 is a complete bipartite graph (every red vertex is connected to every green vertex).

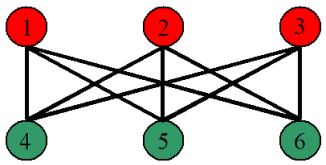


Figure 6.11: Bi-partite Graph

6.3.5 Regular Graph

The number of edges of every vertex in the graph is same. A regular graph with a degree k is called k-regular. Figure 6.12 demonstrates various 3-regular graphs.

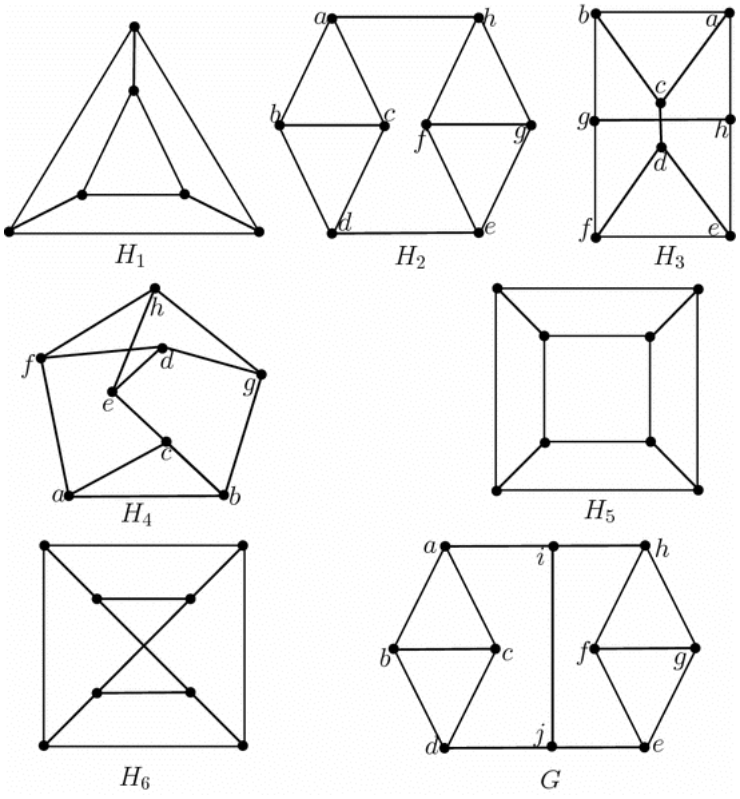


Figure 6.12: 3-regular graphs

Regular graphs have certain properties.

- Complete graphs are regular, i.e. K_n is $(n-1)$-regular.
- All cycles, denoted as C_n, are 2-regular.
- 2-regular graphs consist of disjoint union of cycles and infinite chains.
- 3-regular, also called cubic.
- Chain graph is not regular (middle have 2 and ends have 1), except for "infinite chains".

https://www.geeksforgeeks.org/regular-graph-in-graph-theory/

Theorem

The total number of degrees of a graph is twice the number of edges.

$$\sum_{v \in V(G)} \deg_G(v) = 2\,|E(G)|$$

□

This is trivial but the following corollary is useful.

Corollary

In any graph, there is an even number of odd degree vertices (i.e. cannot have odd number of odd degree vertices – say 3 vertices with 3 degrees)
□

6.3.6 Isomorphic Graphs

These two graphs are isomorphic $\Theta : V(G) \to V(H)$ where Θ is a bijection which preserves adjacency and non-adjacency.

$$u, v \in E(G) \Leftrightarrow \Theta(u)\Theta(v) \in E(H)$$

In Figure 6.13, the two graphs are isomorphic via the following mapping:

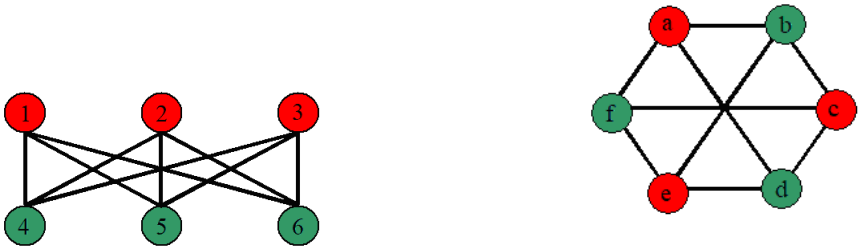
$$\Theta \quad \begin{pmatrix} a & b & d & e & c \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

(a) Cycle (b) Star

Figure 6.13: Isomorphic Graphs

Figure 6.14 is another example of two isomorphic graphs. They are bipartite graphs via the following mapping:

$$\Theta \begin{pmatrix} a & c & e & b & d & f \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$



(a) (b)

Figure 6.14: Isomorphic Bi-Partite Graphs

To determine if two graphs are isomorphic, they must have:

1. same number of vertices
2. same number of edges
3. structural similarities or differences

Theorem

A graph is bi-partite if and only if it is 2-colorable.
□

Corollary

A tree graph (T) is bi-partite.
□

6.3.7 Planar Graph

A planar graph is a graph that can be drawn on a plane such that its edges intersect only at their end nodes. A plane graph is a graph whose edges don not intersect. In other words, a planar graph is a graph that has an isomorphic plane graph (planar = "plane-able"?). For example, see Figure 6.15. The left is a graph that can be easily redrawn as the right one (by moving the top left vertex to the right) which is a plane graph (i.e. no edge crossing)

(a) Planar Graph (b) Plane Graph

Figure 6.15: Planar Graphs

The complete graph K_4, as in Figure 6.16, is a planar graph. Both figures in Figure 6.17a are planar graphs. It can be made into a plane graph as in Figure 6.17.

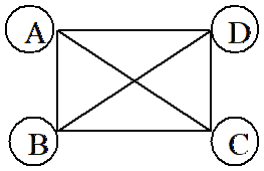
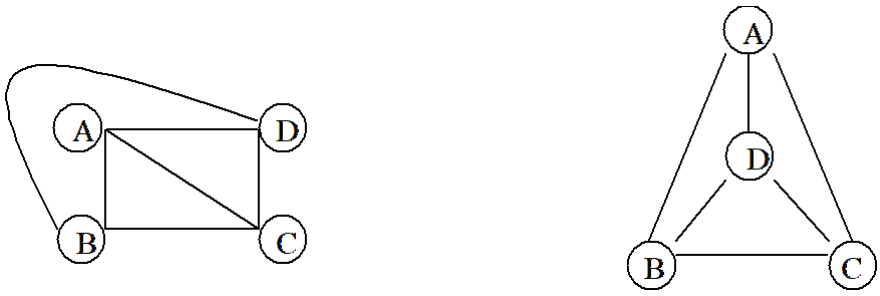


Figure 6.16: Complete Graph K_4

Figure 6.17a is just to pull B-D edge outside of the square (kind of a cheat). Figure 6.17a(b) is more legit and clearly it is isomorphic to Figure 6.17b. (edges don't need to be straight lines, known as the Fary's theorem which says that if there is an arc edge then there must exist a straight edge).

K_5 on the other hand, is not planar (while the proof is complex).

$K_{3,3}$ cannot be made planar either. See Figure 6.19.



(a) Planar Graph 1 (b) Planar Graph 2

Figure 6.17: K_4 Planar Graphs

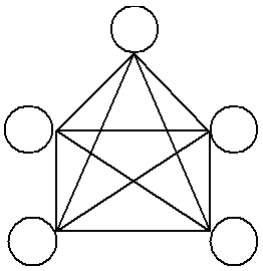


Figure 6.18: Complete Graph K_5

Since K_5 does not have planar embedding, K_6 is not a planar graph in that (to be shown later) it contains K_5 as a sub-graph (i.e. by removing one vertex and its connecting edges from K_6).

There are several properties of planar graphs:

- If a connected planar graph G has $|E|$ edges and $|R|$ regions or faces then $|R| \leq \frac{2}{3}|E|$.
- If a connected planar graph G has $|E|$ edges, $|V|$ vertices, and $|R|$ regions, then $|V| - |E| + |R| = 2$. This is known as the Euler's formula. See later.
- If a connected planar graph G has $|E|$ edges and $|V|$ vertices, then $3|V| - |E| \geq 6$.
- A complete graph K_n is planar iff $n < 5$.
- A complete bipartite graph $K_{m,n}$ is planar iff $m < 3$ or $n \geq 3$.
- If and only if a sub-graph of G is homomorphic to K_5 or $K_{3,3}$, then G is non-planar (K_5 and $K_{3,3}$ are two very special graphs).

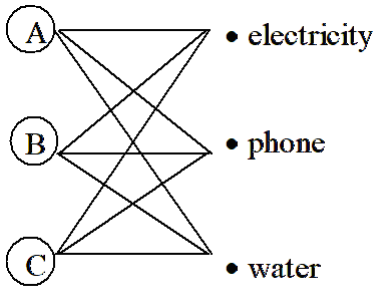


Figure 6.19: $K_{3,3}$ Graph

Any violation of one or more of the above properties is a certain proof that the graph is not planar.

§ Kuratowski's theorem

G is planar if and only if G has no K_5 or $K_{3,3}$ subdivisions. $K_{3,3}$ and K_5 (above) are the only ones that cannot be made planar. Extensions and subdivisions of these two graphs will not be planar embeddable. All other graphs are planar embeddable.
□

§ Wagner's theorem

A finite graph is planar if and only if its minors include neither K_5, nor $K_{3,3}$.
□

§ 4-color theorem

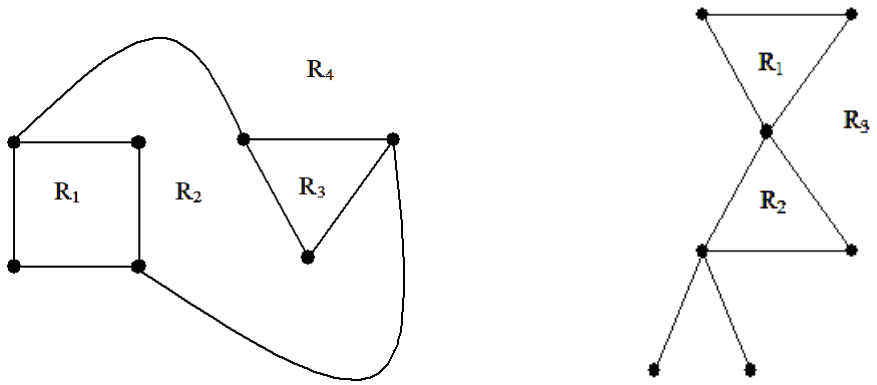
One very interesting (and USEFUL) result of planar graphs is that you can color them with no more than 4 colors!
□

While the proof of the 4-color theorem is complex, the intuition is quite easy to understand. Given Kuratowski's theorem, we know that K_5 is not planar embeddable and a planar embeddable must not contain K_5. Hence it is quite easy to conjecture that any planar graph has less than 5 chromatic colors.

§ Euler's formula

For any connected plane graph, $|V| - |E| + |R| = 2$
□

Take Figure 6.20 as an example. The number of vertices in two sub-graphs is both 7 ($|V| = 7$). Example 1 has 9 edges ($|E| = 9$) and 4 regions ($|R| = 4$) while example 2 has 8 edges and 3 regions. Both graphs satisfy the Euler's formula.



(a) Example 1 (b) Example 2

$|V| = 7, |E| = 9, |R| = 4, |V| - |E| + |R| = 2$ $|V| = 7, |E| = 8, |R| = 3, |V| - |E| + |R| = 2$

Figure 6.20: Euler Theorem

A corollary of Euler's theorem can be derived as $|V| - |E| + |R| = 1 + c(G)$ where $c(G)$ is the number of connected components and G is a planar graph. See Figure 6.21 as an example.

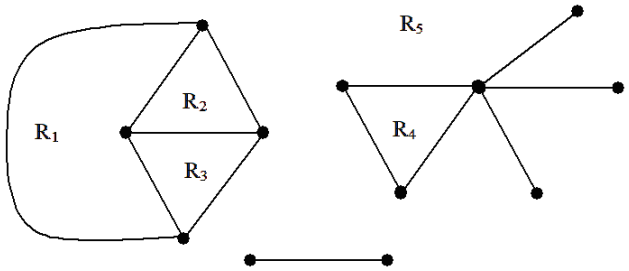


Figure 6.21: Corollary

§ Euler's Polyhedron Formula

In Figure 6.22, the left is a polyhedron with 5 vertices ($|V| = 5$), 8 edges ($|E| = 8$), and 5 faces ($|F| = 5$), and the right is an associated graph with 5 vertices, 8 edges, and 5 regions. Then, $|V| - |E| + |F| = |V| - |E| + |R| = 2$.

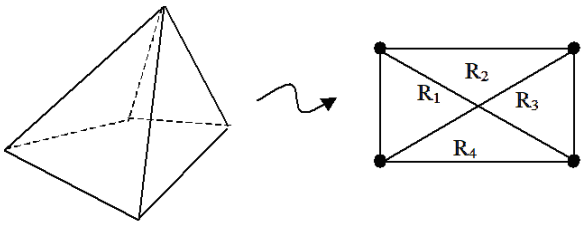


Figure 6.22: Euler's polyhedron formula

Without a formal proof, the intuition behind the theorem is quite clear. We can ask the following question: "Does there exist a graph embedding that fulfills Euler's formula and is not planar?" If the embedding has crossings, then start putting new vertices at each crossing, so that the new vertex subdivides the two crossing edges. With each new vertex, the number of vertices increases by one, the number of edges increases by two, and the number of regions does not change. Keep doing this, and we will get the planar embedding of a graph with $|V| + c$ vertices and $|E| + 2c$ edges, where c is the number of crossings in the initial embedding. Euler's formula now tells you that the number of regions is $|E| - |V| + 2 + c$. Since the number of regions does not change, this is also true for the original embedding.

Since for a non-planar graph every embedding will have crossings, this shows that none of the embeddings satisfies Euler's formula.

§ Maximal Planar Graph

A simple graph is called maximal planar if it is planar but adding any edge (on the given vertex set) would destroy that property.

If G is a maximal planar graph iff $|E| = 3|V| - 6$.

§ Theorem 1

A planar graph implies $|E| \leq 3\,|V| - 6$.
□

Note that the reverse is not true (i.e. $|E| \leq 3\,|V| - 6$ does NOT imply planar.)

But if the condition is violated (i.e. $|E| > 3|V| - 6$), then we know the graph is not planar.

Take K_5, $|V| = 5$ and $|E| = 10$, so 10 is not less than $3 \times 5 - 6 = 9$. The condition is violated, so, K_5 is not planar embeddable.

Take $K_{3,3}$ which we know not a planar graph. $|V| = 6$ and $|E| = 9$, so 9 is less than 12. The inequality is satisfied but the graph is not planar.

§ Theorem 2

G is planar and G has no triangle, then $|E| \leq 2|V| - 4$
□

It is quite amazing that Theorem 1 can be improved with an triangle (cycle of 3).

$K_{3,3}$ now is 9 which is greater than $2 \times 6 - 4 = 8$. The condition is violated. This means that either $K_{3,3}$ is not planar or it has a triangle. Since it has no triangle, it must be not planar.

§ Corollary to Theorem 2

G is planar with $|E| \geq g$ and G has no triangle, then $|E| \leq \frac{g}{g-2}(|V| - 2)$
□ (planar.pdf)

6.3.8 Complement of a Graph

The complement of a graph G is a graph G' on the same set of vertices as of G such that there will be an edge between two vertices (v, e) in G', if and only if there is no edge in between (v, e) in G. Simply speaking, any adjacent vertices become non-adjacent and vice versa (non-adjacent vertices become adjacent).

Take Figure 6.23 as an example (which is a cycle K_5). Its complement is clearly Figure 6.24a (a star). Yet note that Figure 6.24a and Figure 6.24b are isomorphic. Hence interestingly Figure 6.23 and Figure 6.24b (both cycles) are complement to each other, only moving the positions of the vertices – A-B-C-D-E versus A-D-B-E-C.

6.3.9 Perfect Graph

https://www.youtube.com/watch?app=desktop&v=C4Zr4cOVm9g&t=526s

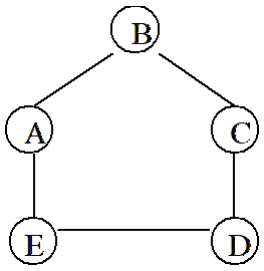
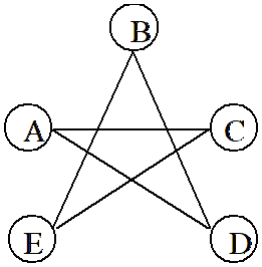
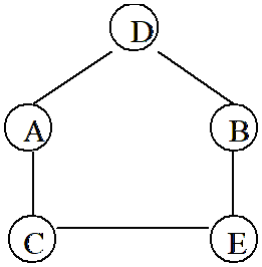


Figure 6.23: Cycle, K_5



(a) 1a



(b) 1b

Figure 6.24: Complement Graphs

A perfect graph is a graph in which the chromatic number equals the size of the maximum clique, both in the graph itself and in every induced subgraph. In all graphs, the chromatic number is greater than or equal to the size of the maximum clique, but they can be far apart. A graph is perfect when these numbers are equal, and remain equal after the deletion of arbitrary subsets of vertices.[1] In other words, G is perfect if $\chi(H) = \omega(H)$ for every H obtained from G by deleting vertices (and of course for G itself, $\chi(G) = \omega(G)$).

This was first conjectured by Claude Berge (1961) and known as the weak perfect graph theorem. The strong perfect graph theorem states that the only graphs that are not perfect are those that can reduce to odd cycles or their complements. This was proven by Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas was announced in 2002.

Figure 6.25 is an odd cycle (of 5 vertices) C_5. It is obvious that all cycles have $\omega(C_n) = 2$, but $\chi(G) = 3$. Hence it is not perfect.

Take any cycle C_n,

- if n is odd then the minimum number of colors must be $\chi(G) = \chi(C_{2n+1}) = 3$
- if n is even then $\chi(G) = \chi(C_{2n}) = 2$

[1] https://en.wikipedia.org/wiki/Perfect_graph

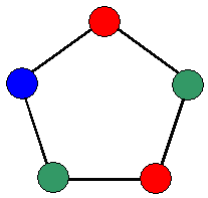


Figure 6.25: Odd Cycle

$$\omega(C_5) = 2, \chi(C_5) = 3$$

Hence, even cycles are perfect, and odd cycles are not perfect.

Bipartite graphs are all 2-colorable, $\chi(G) = 2$. Also their clique $\omega(G) = 2$. So all bipartite graphs are perfect.

Now, look at the complement of C_5 (Figure 6.26) which is a star. It is not perfect ($\omega = 2, \chi = 3$).

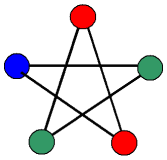


Figure 6.26: Complement of C_5

$$\omega(G) = 2, \chi(G) = 3$$

C_5, is not perfect (any odd cycle is not perfect) because $\chi(C_5) = 3$ and $\omega(C_5) = 2$. Now add a triangle to C_5 and call it G. G is not perfect because removing the triangle leads to H which is C_5 which is not perfect. See Figure 6.27. And hence any graph G that can reduce to it will not be perfect.

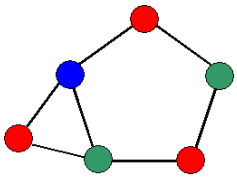


Figure 6.27: A Triangle added to C_5

$$\omega(G) = 2, \chi(G) = 3$$

Look at Figure 6.28 (which is a planar graph – by the 4-color theorem, $\chi(G) \leq 4$), $\chi(G) = 3$. Note that 1-2-3 is a clique (set of pair-wise adjacent vertices). $\omega(G)$ is

the biggest size of a clique and in this graph, $\omega(G) = 3$ which is 1-2-3, since 2-3-4-5 has only clique of 2. Hence in this case, the graph is perfect.

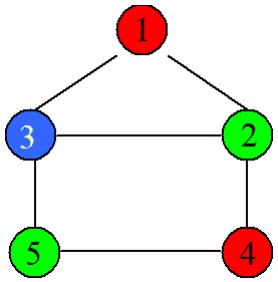


Figure 6.28: This Planar Graph is Perfect

$$\omega(G) = 3, \chi(G) = 3$$

Figure 6.29 is a complete graph K_5, also called a clique (set of pair-wise adjacent vertices), and hence $\chi(K_n) = n = 5$. It is not a planar graph (edges crossing). Since it is complete, $\omega(K_5) = 5$ as well. Hence, all complete graphs are perfect (Claude Berge (1961)).

$$\underset{\text{chromatic number}}{\chi(H)} = \underset{\text{largest clique}}{\omega(H)}$$

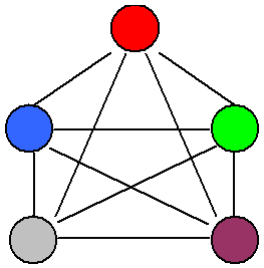


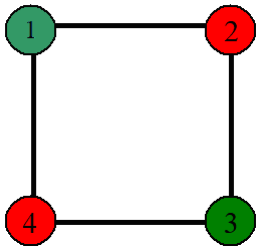
Figure 6.29: Complete Graph is a Perfect Graph

$$\omega(K_5) = \chi(K_5) = 4$$

Theorem

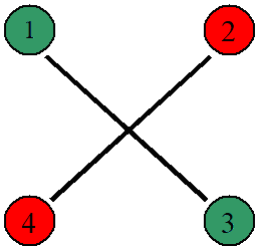
An undirected graph is perfect if and only if its complement graph is also perfect. □

This result had been conjectured by Claude Berge (1961, 1963), and it is sometimes called the weak perfect graph theorem to distinguish it from the strong perfect



(a) Even Cycle, C_4

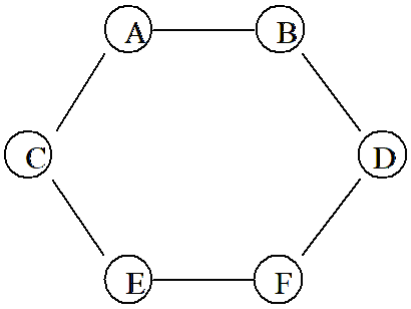
$\omega(G) = 2, \chi(G) = 2$



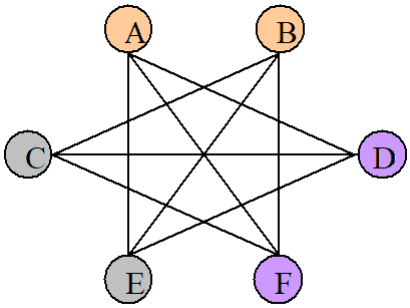
(b) Complement to C_4

$\omega(G) = 2, \chi(G) = 2$

Figure 6.30: Perfect Graphs



(a) $\chi(G) = 2; \omega(G) = 2$



(b) $\chi(G) = 3; \omega(G) = 3$

Figure 6.31: Complement Graphs

graph theorem characterizing perfect graphs by their forbidden induced subgraphs.[2] Figure 6.30b is complement to Figure 6.30a. Even cycles are perfect, their complements are also perfect. Note that the converse is absolutely not true: small ω implies small χ.

§ Corollary

The complement of an even cycle is a perfect graph. (See Figure 6.31.)
□

Note that only the complement of C_5 is a C_5. Not all odd cycles whose complements are odd cycles.

[2]https://en.wikipedia.org/wiki/Perfect_graph_theorem#:~:text=A 20perfect 20graph 20is 20an,a 20coloring 20of 20the 20subgraph.

§ Strong Perfect Graph Theorem

G is perfect if no induced subgraph of G is an odd cycle of length at least five or the complement of one.
□ (https://annals.math.princeton.edu/wp-content/uploads/annals-v164-n1-p02.pdf)

In plain words, a perfect graph is one such that it cannot be extensions of odd cycles (C_n where $n \geq 5$ or their complements).

6.3.10 Brook's Theorem

Before introducing Brook's theorem, here is a more general theorem.

§ Theorem

If a graph contains a complete subgraph (also known as a clique), then the chromatic number (number of colors used to color the graph) of the graph must be at least the number of vertices in the graph (size of the largest clique).

$$\underset{\substack{\text{chromatic}\\\text{number}}}{\chi(G)} \geq \underset{\substack{\text{largest}\\\text{clique}}}{\omega(G)}$$

□

This is quite intuitive. Given that the largest clique $C(G)$ (i.e. a complete subgraph) has a chromatic color $\chi(C(G))$ same as $\omega(G)$ because it is a complete graph, it is obvious that $\chi(G)$ cannot be less than $\omega(G)$.

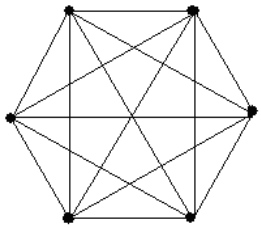
§ Theorem (R. Leonard Brooks (1941))

In a connected graph in which every vertex has at most δ neighbors, the vertices can be colored with only δ colors, except for two cases, complete graphs and cycle graphs of odd length, which require $\delta + 1$ colors.
□

In other words, the previous inequality can be extended as (bounds for chromatic number):

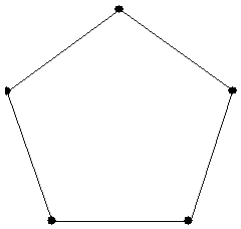
$$\underset{\substack{\text{largest}\\\text{clique}}}{\omega(G)} \leq \underset{\substack{\text{chromatic}\\\text{number}}}{\chi(G)} \leq \underset{\substack{\text{highest}\\\text{degree}}}{\delta(G)} \tag{6.1}$$

except for complete graphs and cycles. Note that a complete graph $\chi(G) = \delta(G) + 1$ because $\chi(K_n) = n$ and $\delta(K_n = n - 1)$.



(a) Complete Graph, K_6

$\delta(G) = 5, \chi(G) = 6$



(b) Cycle, C_5

$\delta(G) = 2, \chi(G) = 3$

Figure 6.32: Brook Graphs

Any graph G satisfies $\chi(G) \leq \delta(G)$ unless G is a complete graph or an odd cycle, in which case $\chi(G) = \delta(G) + 1$. (note that G is not planar).

To see that, take an example of Figure 6.32 where the left is a complete graph ($\delta(G) = 5, \chi(G) = 6$) and the right is an odd cycle ($\delta(G) = 2, \chi(G) = 3$). Both cases are exceptions of the Brook's theorem. We can see that in both cases the theorem is violated.

If cycles are not perfect (note that even cycles are perfect $\chi(G) = \omega(G) = 2$ and odd cycles are not prefect), then anything that can be reduced to cycles will not be perfect.

Now, let's take an example (Petersen graph) like Figure 6.33 (not a complete graph nor a cycle). The maximal clique is just any two vertices, $\omega(G) = 2$ (if the center star had a circumference, then it would be the maximal clique and it is 5). The highest degree is $\delta(G) = 3$ (actually it is a 3-regular graph). So the chromatic number is between 2 and 3, $2 \leq \chi(G) \leq 3$.

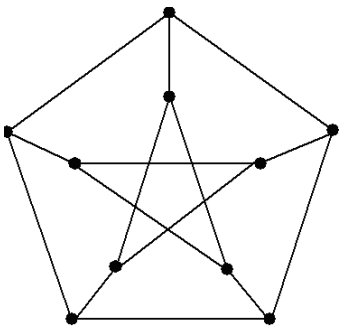


Figure 6.33: Brook (Petersen) Graph

$\delta(G) = 3, \chi(G) = 3$

6.3.11 Odd Graph

(mathworld) The odd graph O_n of order n is a graph having vertices given by the $(n-1)$-subsets of $\{1, \cdots, 2n+1\}$ such that two vertices are connected by an edge iff the associated subsets are disjoint (Biggs 1993, Ex. 8f, p. 58). Hence the total number of vertices is

$$\binom{2n-1}{n-1}$$

For example, O_1 has just one vertex; O_2 has 3 vertices; and O_3 has 10 vertices. In fact, O_1 is a dot; O_2 is a triangle; and O_3 is the famous Petersen graph (see a later section of special graphs). See Figure 6.34.

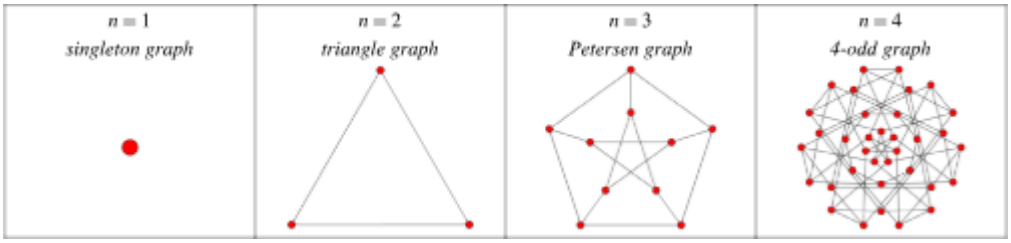


Figure 6.34: Odd Graphs

6.4 Special Graphs

6.4.1 Petersen Graph

(Wiki) The Petersen graph is an undirected graph with 10 vertices and 15 edges. It is a small graph that serves as a useful example and counterexample for many problems in graph theory. The Petersen graph is named after Julius Petersen, who in 1898 constructed it to be the smallest bridgeless cubic graph with no three-edge-coloring. Figure 6.35 gives a few examples of the Petersen graph.

The Peterson graph does not have planar embedding. This is because the Petersen graph contains a sub-graph that is a subdivision of $K_{3,3}$. To see that, remove two edges from Figure 6.36a: 7-10 and 3-4 to get a sub-graph, as in Figure 6.36b. We know that by Kuratowski's theorem, it is not planar.

There is another proof that the Petersen graph is not planar. See Figure 6.37. The Petersen graph has a K_5 minor.[3]. Perform edge extraction on edges

[3] H is called a minor of the graph G if H can be formed from G by deleting edges, vertices and by contracting edges. See section 6.2

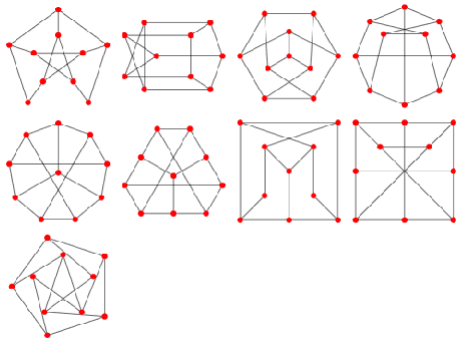
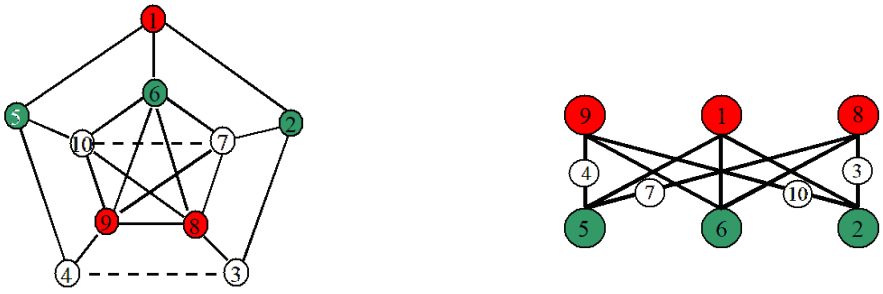


Figure 6.35: Peterson Graphs

source: https://mathworld.wolfram.com/PetersenGraph.html



(a) Petersen Graph (b) Bipartite Graph

Figure 6.36: Petersen Graph is not Planar, due to $K_{3,3}$

e_1, e_2, e_3, e_4, e_5. The resulting graph is isomorphic to K_5. Then by Wagner's theorem, the Petersen graph is not planar embeddable.

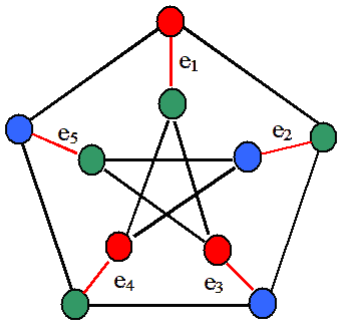


Figure 6.37: Petersen Graph is not planar due to K_5

The Petersen graph is nonhamiltonian.

6.4.2 Hamiltonian Graph

A Hamiltonian graph is a graph which has a closed path (cycle) that visits each vertex exactly once, ending on the same vertex as it began. This closed path is also called a Hamiltonian cycle. Examples are given in Figure 6.38.

Examples (wiki)

- A complete graph with more than two vertices is Hamiltonian
- Every cycle graph is Hamiltonian
- Every tournament has an odd number of Hamiltonian paths (Rédei 1934)
- Every platonic solid, considered as a graph, is Hamiltonian
- The Cayley graph of a finite Coxeter group is Hamiltonian (For more information on Hamiltonian paths in Cayley graphs, see the Lovász conjecture.)
- Cayley graphs on nilpotent groups with cyclic commutator subgroup are Hamiltonian.
- The flip graph of a convex polygon or equivalently, the rotation graph of binary trees, is Hamiltonian.

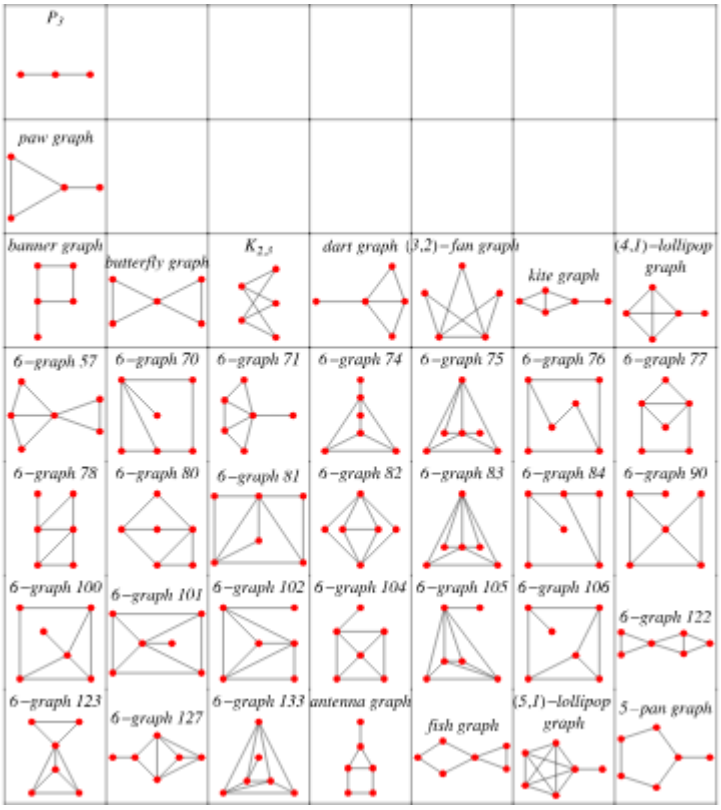


Figure 6.38: Hamiltonian Graphs

6.4.3 Eulerian Path/Trail

Wiki: Eulerian trail (or Eulerian path) is a trail in a finite graph that visits every edge exactly once. See Figure 6.39.

- A graph is Eulerian if all vertices have even degree.
- Semi-Eulerian. (traversable) Contains a semi-Eulerian trail - an open trail that includes all edges one time. A graph is semi-Eulerian if exactly two vertices have odd degree.

6.4.4 Sudoku Graph

A Sudoku puzzle is a game (as shown in Figure 6.40) in which some cells are filled in with symbols and the rest must be filled in by the person solving the puzzle. It corresponds to the pre-coloring extension problem on this graph.

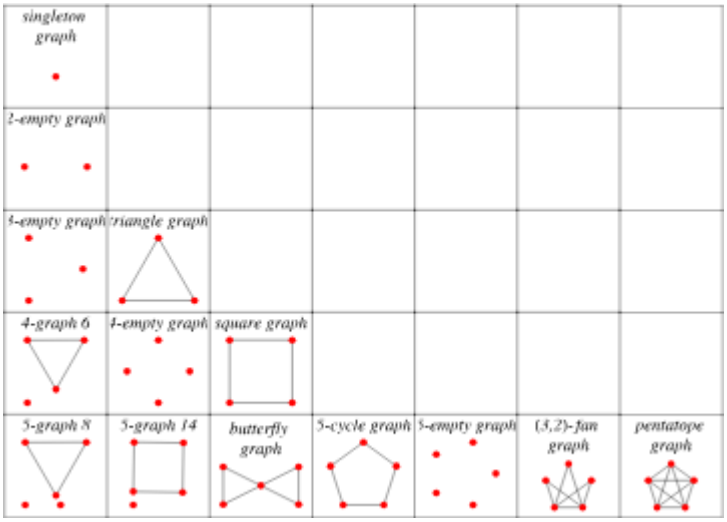


Figure 6.39: Eulerian Graph

source: https://mathworld.wolfram.com/EulerianGraph.html

From wikipedia: "On a Sudoku board of size $n^2 \times n^2$, hence the Sudoku graph has n^4 vertices, each with exactly $3n^2 - 2n - 1$ neighbors. Therefore, it is a regular graph. The total number of edges is $\frac{n^4(3n^2-2n-1)}{2}$.

For instance, for a 4×4 board, has 16 vertices and 56 edges, and is 7-regular. For the most common form of Sudoku, on a 9×9 board, the Sudoku graph is a 20-regular graph with 81 vertices and 810 edges.

Each row, column, or block of the Sudoku puzzle forms a clique in the Sudoku graph, whose size equals the number of symbols used to solve the puzzle. A graph coloring of the Sudoku graph using this number of colors (the minimum possible number of colors for this graph) can be interpreted as a solution to the puzzle.

6.5 Topology

https://en.wikipedia.org/wiki/Topological_space
Around 1735, Leonhard Euler discovered the formula

$$V - E + F = 2$$

relating the number of vertices (V), edges (E) and faces (F) of a convex polyhedron, and hence of a planar graph. The study and generalization of this formula, specifically by Cauchy (1789–1857) and L'Huilier (1750–1840), boosted the study of topology.

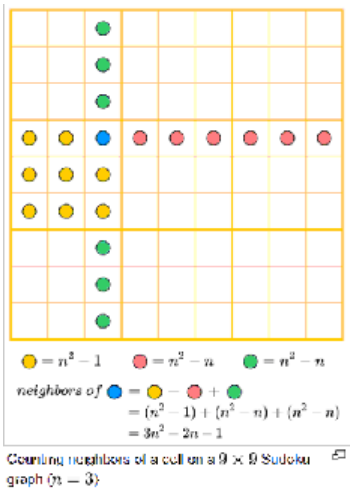


Figure 6.40: Sudoku

In 1827, Carl Friedrich Gauss published *General Investigations of Curved Surfaces* in which he defines the curved surface in a similar manner to the modern topological understanding: "A curved surface is said to possess continuous curvature at one of its points A, if the direction of all the straight lines drawn from A to points of the surface at an infinitesimal distance from A are deflected infinitesimally from one and the same plane passing through A."

Möbius and Jordan seem to be the first to realize that the main problem about the topology of (compact) surfaces is to find invariants (preferably numerical) to decide the equivalence of surfaces, that is, to decide whether two surfaces are homeomorphic or not.

The term "topology" was introduced by Johann Benedict Listing in 1847 by combining two Greek words $\tau\acute{o}\pi o\varsigma$ and $\lambda\acute{o}\gamma o\varsigma$ where $\tau\acute{o}\pi o\varsigma$ means location and $\lambda\acute{o}\gamma o\varsigma$ means logos.

6.6 Modeling Undirected Graph

Usually a joint (multivariate) Gaussian distribution is assumed. However, unlike a generic multivariate Gaussian, the distribution of a graph can be largely simplified. This is known as probabilistic graphical models (PGMs).

6.6.1 Probability Factorization

Assume n vertices of a graph, and each vertex is represented as x_i where $i = 1, \cdots, n$. Then $p(x_1, \cdots, x_n)$ follows some joint probability distribution as follows:

$$p(x_1, \cdots, x_n) \propto \frac{\prod_{C \in C(G)} \psi_C(x_C)}{\prod_{S \in S(G)} \psi_S(x_S)} \tag{6.2}$$

where $\psi_C(x_C)$ is the potential for a clique C, and $\psi_S(x_S)$ is the potential for a separator S.

Probability distributions can be used as potential functions. However, in this case, we cannot let all potentials be either marginal probabilities or conditional probabilities. So the potential function for this graph often are not probability distributions.

§ Example 1

See a chain graph as in Figure 6.41. We can see that Y is a separator (separating X from Z). Also, $X - Y$ is a clique and $Y - Z$ is a clique. Using equation (6.2), we can write $p(x, y, z) \propto p(x, y)p(y, z)/p(y)$.[4]

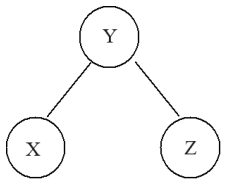


Figure 6.41: Chain Graph

There are other ways to express the joint distribution. One is to recognize that $X \perp\!\!\!\perp Z|Y$. Secondly, it can be a combination of two cliques: $X - Y$ and $Y - Z$. Lastly, it can be seen as just a chain.

$$p(x, y, z) = \begin{cases} p(x|y)p(z|y)p(y) & X \text{ and } Z \text{ are independent conditional on } Y \\ p(x, y)p(z|y) & \text{clique } (X, Y) \text{ and } Z \text{ by itself (conditional on } Y) \\ p(x|y)p(y|z)p(z) & \text{chain} \end{cases} \tag{6.3}$$

[4]In this simple example, it is actually equal.

Note that the three factorizations are all equal. To see that the first and the last are equal:

$$\begin{aligned} p(x|y)p(y|z)p(z) &= p(x|y)p(y,z) \\ &= p(x|y)p(z|y)p(y) \end{aligned} \tag{6.4}$$

and to see the first and second are equal:

$$\begin{aligned} p(x,y)p(z|y) &= p(x,y)\frac{p(y,z)}{p(y)} \\ &= p(x|y)p(y,z) \\ &= p(x|y)p(z|y)p(y) \end{aligned} \tag{6.5}$$

§ Example 2

See a tree graph as in Figure 6.42. https://en.wikipedia.org/wiki/Graphical_model

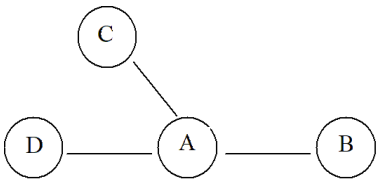


Figure 6.42: Tree Graph

We can write the factorization as a combination of all cliques:

$$p(A,B,C,D) = p(A,B)p(A,C)p(A,D) \tag{6.6}$$

Alternatively, since $B \perp\!\!\!\perp C \perp\!\!\!\perp D|A$, we can write it as follows:

$$\begin{aligned} p(A,B,C,D) &= p(B,C,D|A)p(A) \\ &= P(B|A)p(C|A)p(D|A)p(A) \\ &= \frac{p(A,B)}{p(A)}\frac{p(A,C)}{p(A)}\frac{p(A,D)}{p(A)}p(A) \\ &= p(A,B)p(A,C)p(A,D)/p(A)/p(A) \end{aligned} \tag{6.7}$$

Now they are not equal (differ by $p(A)$). Note that equality is not necessary (as A is a separator).

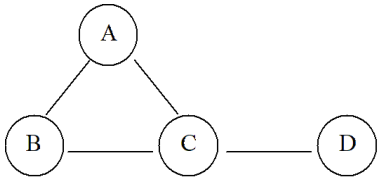


Figure 6.43: Circle Graph

§ Example 3

Figure 6.43 is similar to example 2 but $A - B - C$ form a circle

The circle is the maximal clique of the graph. Hence, we can write the factorization as:

$$p(A, B, C, D) = p(A, B, C)p(C, D) \tag{6.8}$$

or alternatively, we can write it as a combination of all two-vertex cliques:

$$p(A, B, C, D) = p(A, B)p(B, C)p(C, A)p(C, D) \tag{6.9}$$

Hence, factorization is not unique.

§ Example 4

See Figure 6.44.

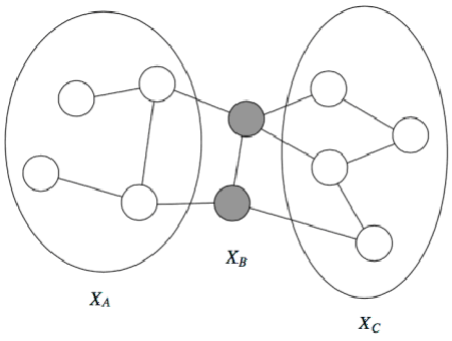


Figure 6.44: Partitioned Graph

There are several ways to express the joint distribution:

1. given x_B, x_A and x_C are independent
2. within x_A and x_C, write down the product of cliques

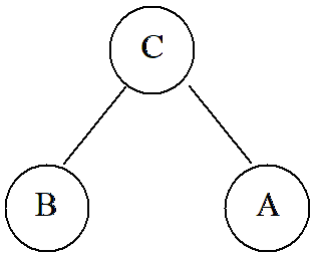


Figure 6.45: Undirected Graph of Three Vertices

3. note that within x_C, there are many ways to factorize (different ways to see cliques)

6.6.2 Conditional Independence

§ Precision Matrix

Precision matrix is just the inverse of the covariance matrix. It is also called partial correlation matrix. It provides information on conditional independence. If any element in the precision matrix is 0, it indicates that the two variables are conditionally independent given (condition on) the remaining variables.

Take Figure 6.45 chain graph as an example. We know that $A \perp\!\!\!\perp B|C$.

Without loss of generality, let A, B, and C be related as follows:

$$\begin{aligned} x_B &= x_C + u \\ x_A &= x_C + v \end{aligned} \tag{6.10}$$

where u and v are independent (because A and B are independent given C).

Then,

$$\begin{aligned} \sigma_{AB} &= \sigma_C^2 \\ \sigma_{AC} &= \sigma_C^2 \\ \sigma_{BC} &= \sigma_C^2 \end{aligned} \quad \text{and} \quad \begin{aligned} \sigma_A^2 &= \sigma_C^2 + \sigma_v^2 \\ \sigma_B^2 &= \sigma_C^2 + \sigma_u^2 \end{aligned} \tag{6.11}$$

Then,

$$\Sigma = \begin{bmatrix} \sigma_C^2 + \sigma_u^2 & \sigma_C^2 & \sigma_C^2 \\ \sigma_C^2 & \sigma_A^2 + \sigma_v^2 & \sigma_A^2 \\ \sigma_A^2 & \sigma_A^2 & \sigma_A^2 \end{bmatrix} \tag{6.12}$$

Assume that all variances are 1. Then, Σ equals:

$$\begin{array}{c|ccc} & A & B & C \\ A & 2 & 1 & 1 \\ B & 1 & 2 & 1 \\ C & 1 & 1 & 1 \end{array}$$

Then inverse (precision matrix) is:

$$\begin{array}{c|ccc} & A & B & C \\ A & 1 & 0 & -1 \\ B & 0 & 2 & -1 \\ C & -1 & -1 & 3 \end{array}$$

It is clear that A and B are independent given C. This validates the model specified in equation (6.10).

Now given a variance-covariance matrix in general (n dimensions) as follows:

$$\Sigma = \begin{bmatrix} V_1 & R \\ R' & V_2 \end{bmatrix} \tag{6.13}$$

where the variance-covariance matrix Σ is partitioned into 4 segments: V_1, R, R', and V_2. Let its inverse be:

$$\Sigma^{-1} = \begin{bmatrix} K_1 & H \\ H' & K_2 \end{bmatrix} \tag{6.14}$$

Then, we have the following (derivation is straightforward and left as an exercise):

$$\begin{aligned} K_1^{-1} &= V_1 - RV_2^{-1}R' \\ H &= -K_1RV_2^{-1} \\ K_2^{-1} &= V_2 - R'K_1^{-1}R \end{aligned} \tag{6.15}$$

Now partition Σ into three segments: A, B, and C:

$$\Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} & \Sigma_{AC} \\ \Sigma_{BA} & \Sigma_{BB} & \Sigma_{BC} \\ \Sigma_{CA} & \Sigma_{CB} & \Sigma_{CC} \end{bmatrix} = \begin{bmatrix} \Sigma_{AB\times AB} & \Sigma_{AB\times C} \\ \Sigma_{C\times AB} & \Sigma_{CC} \end{bmatrix} \tag{6.16}$$

where

$$\begin{aligned} \Sigma_{AB\times AB} &= \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} \\ \Sigma_{AB\times C} &= \begin{bmatrix} \Sigma_{AC} \\ \Sigma_{BC} \end{bmatrix} \\ \Sigma_{C\times AB} &= \begin{bmatrix} \Sigma_{CA} & \Sigma_{CB} \end{bmatrix} \end{aligned} \tag{6.17}$$

Then by the result of equation (6.20), the conditional distribution of A and B given C has covariance matrix as:

$$K_1^{-1} = \Sigma_{AB\times AB} - \Sigma_{AB\times C}\Sigma_{CC}^{-1}\Sigma_{C\times AB} \tag{6.18}$$

which is a direct substitution of the inverse formula above. Expand K^{-1} into the following:

$$\begin{aligned} K^{-1} &= \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} - \begin{bmatrix} \Sigma_{AC} \\ \Sigma_{BC} \end{bmatrix} \Sigma_{CC}^{-1} \begin{bmatrix} \Sigma_{CA} & \Sigma_{CB} \end{bmatrix} \\ &= \begin{bmatrix} \Sigma_{AA} - \Sigma_{AC}\Sigma_{CC}^{-1}\Sigma_{CA} & \Sigma_{AB} - \Sigma_{AC}\Sigma_{CC}^{-1}\Sigma_{CB} \\ \Sigma_{BA} - \Sigma_{BC}\Sigma_{CC}^{-1}\Sigma_{CA} & \Sigma_{BB} - \Sigma_{BC}\Sigma_{CC}^{-1}\Sigma_{CB} \end{bmatrix} \end{aligned} \tag{6.19}$$

$A \perp\!\!\!\perp B|C$ implies that the off-diagonal term is 0, i.e. $\Sigma_{AB} - \Sigma_{AC}\Sigma_{CC}^{-1}\Sigma_{CB} = 0$.

Conditional independence (i.e. $A \perp\!\!\!\perp B|C$) is defined as the rank of $\Sigma_{AC\times BC}$ is equal to the rank of Σ_{CC}. To see that,

$$\text{rank } \Sigma_{AC\times BC} = \text{rank} \begin{bmatrix} \Sigma_{AB} & \Sigma_{AC} \\ \Sigma_{CB} & \Sigma_{CC} \end{bmatrix} = \text{rank} \begin{bmatrix} S & \Sigma_{AC} \\ 0 & \Sigma_{CC} \end{bmatrix} \tag{6.20}$$

where

$$S = \Sigma_{AB} - \Sigma_{AC}\Sigma_{CC}^{-1}\Sigma_{CB} \tag{6.21}$$

Note that equation (6.21) can be achieved by subtracting the right column multiplied by $\Sigma_{CC}^{-1}\Sigma_{CB}$ from the left column (note that column (or row) operation is an easy method to compute a rank).

Note that S is precisely the upper-right element of matrix K^{-1}. It is 0 if $A \perp\!\!\!\perp B|C$. Then we have proved that the rank of $\Sigma_{AC\times BC}$ is equal to the rank of Σ_{CC}.

§ Conditional Independence Test

(https://online.stat.psu.edu/stat504/lesson/5/5.3/5.3.4) Using Figure 6.45, we can write the joint probability distribution as:

$$p(A, B, C) = p(C)p(A|C)p(B|C)$$

If we sample the data, let the total size be n. Then, the frequency of C is:

$$\begin{aligned} p(C) &= n_c/n \\ p(A|C) &= n_{a,c}/n_c \\ p(B|C) &= n_{b,c}/n_c \end{aligned}$$

where n_c/n is the marginal probability. Similarly for others.

The null is $p(A, B|C) = p(A|C)p(B|C)$ which is $\frac{n_{a,b} \times n_{a,c}}{n_c}$. The test for conditional independence of A and B given C is equivalent to separating the table by levels of $i = 1, \cdots, n_c$ and testing for independence within each level.

There are two ways we can test for conditional independence:

- The overall C^2 statistics can be found by summing the individual test statistics for A, B independence across the levels of C. The total degrees of freedom for this test must be $n_c(n_a - 1)(n_b - 1)$.
- Cochran-Mantel-Haenszel Test (using option CMH in PROC FREQ/ TABLES/ in SAS and mantelhaen.test in R). This test produces the Mantel-Haenszel statistic also known as the "average partial association" statistic.

6.6.3 Linear Regression

Given joint normality, it is important for us to understand linear regression. It is well-known that linear regression can be studied under OLS (ordinary least square) and MLE (maximum likelihood). Here, joint normality is the latter.

An n-dimensional multivariate Gaussian distribution is:

$$p(\underline{x}) = (2\pi)^{-n/2}|\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(\underline{x} - \underline{\mu})'\Sigma^{-1}(\underline{x} - \underline{\mu})\right\} \tag{6.22}$$

Use the result above (equation (6.18)), we can partition the joint distribution as:

$$p\begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \end{pmatrix} \sim N\begin{pmatrix} \underline{\mu}_1 \\ \underline{\mu}_2 \end{pmatrix}, \quad \begin{matrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{matrix} \tag{6.23}$$

where

$$p(\underline{x}_1|\underline{x}_2) \sim N(\underline{\mu}_{1|2}, \Sigma_{1|2}) \tag{6.24}$$

$$\begin{aligned} \underline{\mu}_{1|2} &= \underline{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\underline{x}_2 - \underline{\mu}_2) \\ \Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned} \tag{6.25}$$

It is clear that equation (6.18) is verified (by substituting the following into $\Sigma_{1|2}$).

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} = \begin{bmatrix} \Sigma_{AB\times AB} & \Sigma_{AB\times C} \\ \Sigma_{C\times AB} & \Sigma_{C\times C} \end{bmatrix} \tag{6.26}$$

§ The Example of (Famous) Simple Regression

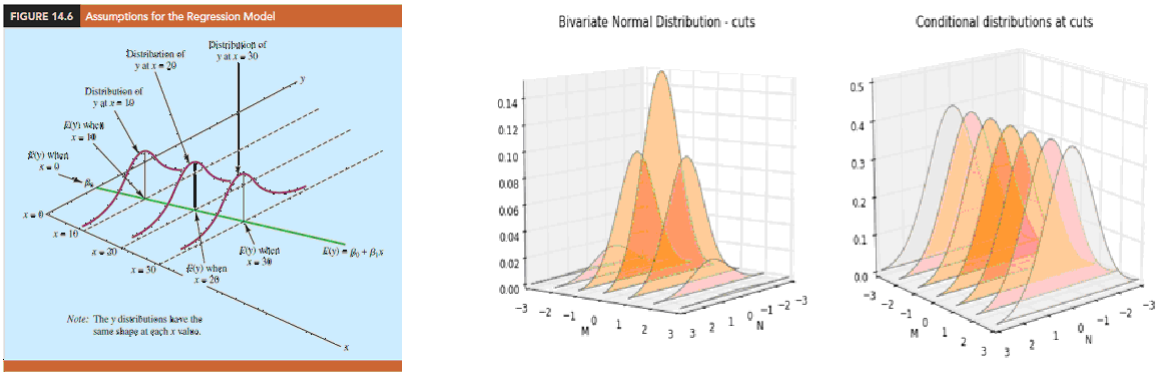
Take a simple (i.e. univariate) regression,

$$y = \alpha + \beta x + \varepsilon$$

where x and y follow a bi-variate normal distribution. Then

$$\begin{aligned} \mathbb{E}[y|x] &= \mu_y + \sigma_{xy}\frac{1}{\sigma_x^2}(x - \mu_x) \\ &= (\mu_y - \beta\mu_x) + \beta x \\ &= \alpha + \beta x \end{aligned} \tag{6.27}$$

It is quite fascinating to visualize how this conditional normality leads to a simple regression. In Figure 6.46, there are two plots to visualize the how such a simple regression can be represented in conditional normality. The green line on the left graph is the regression line. As the graph shows, it passes through all the mean values of a series of normal (conditional) distributions. Each of these normal distributions is a $y|x$ distribution (normalized).



(a) (b)

Figure 6.46: Conditional Normality

This is consistent with OLS (ordinary least square) with data ($t = 1, \cdots, T$):

$$y_t = \alpha + \beta x_t + \varepsilon_t$$

and

$$\min \sum_{t=1}^{T} \varepsilon_t^2$$

See Chapter 2 (appendix) for details.

§ LASSO (and Ridge) Regression

Hull Chapter 3 (3.4 ∼ 3.7).

LASSO regression is adding an appendix to the regular regression as follows:

$$\text{SSE} = \sum_{t=1}^{T} (y_t - \beta_0 - \beta_1 x_{t,1} - \cdots - \beta_K x_{t,K})^2 + \lambda \sum_{k=1}^{K} |\beta_k| \tag{6.28}$$

and ridge regression is adding an appendix to the regular regression as follows:

$$\text{SSE} = \sum_{t=1}^{T} (y_t - \beta_0 - \beta_1 x_{t,1} - \cdots - \beta_K x_{t,K})^2 + \lambda \sum_{k=1}^{K} {\beta_k}^2 \tag{6.29}$$

When λ is large, OLS will force β values to be small. Those β's that are insignificant will first go to 0. In doing so, we can eliminate the number of (unimportant) explanatory variables (features). Certainly, the fit will be worse (i.e. the errors are large). This is a useful technique to reduce dimensions.

What it does is graphically can be seen in Figure 6.47. The left is LASSO and the right is ridge.

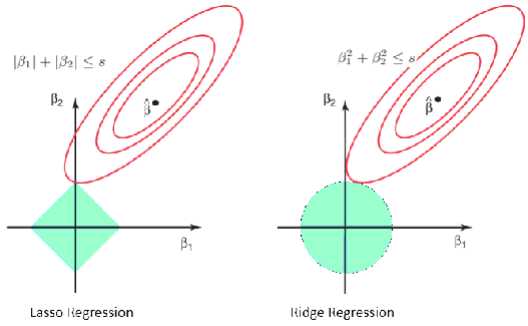


Figure 6.47: LASSO and Ridge

To see this more clearly, see Figure 6.48 where each ellipse represents a particular error level ($\sum e^2$) and each circle represents a λ value.

When $\lambda = \lambda_1$, the solution $\{\beta_1, \beta_2\}$ satisfies $\beta_1^2 + \beta_2^2 = c_1^2$. When $\lambda = \lambda_2$, the solution satisfies $\beta_1^2 + \beta_2^2 = c_2^2$. But at this point, we also have $\beta_1 = 0$ and $\beta_2 = \lambda_2$ (but the optimal solution is on a smaller circle (not shown) which is $\beta_1^2 + \beta_2^2 = c_3^2$ corresponding to an even smaller $\lambda = \lambda_3$). Note that there are many λ's (for $c > c_2$ ($\lambda < \lambda_2$)) at which $\beta_1 = 0$ but we are looking for the smallest c (this is the tangent).

There is closed-form solution to ridge regression. First, re-write equation (6.29) in a matrix form:

$$\text{SSE} = \underline{e'e} + \lambda \text{tr} \underline{\beta'\beta} = (\underline{y} - X\underline{\beta})'(\underline{y} - X\underline{\beta}) = \underline{y'y} - 2\underline{\beta'}X'\underline{y} + \underline{\beta'}X'X\underline{\beta} + \text{tr}\underline{\beta'\beta}$$

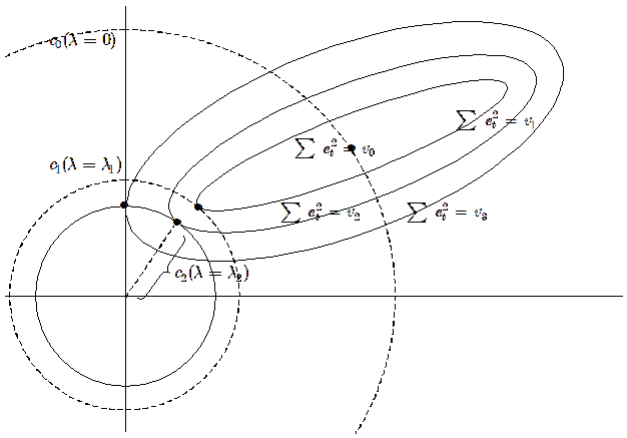


Figure 6.48: Ridge Regression

Then take partial derivatives w.r.t. regression coefficients (and set it to 0):

$$\frac{\partial}{\partial \underline{\beta}} SSE = -2X'\underline{y} + 2X'X\underline{\beta} + 2\lambda I\underline{\beta} = 0$$

Solving the equation, we get the estimates:

$$X'\underline{y} = (X'X + \lambda I)\underline{\beta}$$
$$\hat{\underline{\beta}} = (X'X + \lambda I)^{-1}X'\underline{y}$$

§ Graphic LASSO

(Wiki) the graphical LASSO (least absolute shrinkage and selection operator) is a sparse penalized maximum likelihood estimator for the concentration or precision matrix (inverse of covariance matrix) of a multivariate elliptical distribution. The original variant was formulated to solve Dempster's covariance selection problem for the multivariate Gaussian distribution when observations were limited. Subsequently, the optimization algorithms to solve this problem were improved and extended to other types of estimators and distributions.

Hence, it is slightly different from (although conceptually same as) the LASSO regression introduced in the previous sub-section. We would like to reduce the dimensions in the precision matrix – Σ^{-1}. Recall the multi-variate Gaussian distribution of equation (6.22).[5] The likelihood function (i.e. with data of n observations)

[5]Chapter 13 The Multi-variate Gaussian and Chapter 17 Undirected Graphic Models

can be written as (given that data are sampled independently):

$$\mathscr{L}(X) = (2\pi)^{-nT/2} \left|\Sigma\right|^{-T/2} \prod_{t=1}^{T} \exp\left\{-\frac{1}{2}(\underline{x}_t - \underline{\mu})'\Sigma^{-1}(\underline{x}_t - \underline{\mu})\right\} \tag{6.30}$$

Take the logarithmic operation of the function (log likelihood function):

$$\begin{aligned} \ln \mathscr{L} &= -\frac{nT}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma\right|^{-1} - \frac{1}{2}\sum_{t=1}^{T}(\underline{x}_i - \underline{\mu})'\Sigma^{-1}(\underline{x}_t - \underline{\mu}) \\ &= -\frac{nT}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma\right|^{-1} - \frac{1}{2}\sum_{t=1}^{T}\mathrm{tr}\left[(\underline{x}_t - \underline{\mu})'\Sigma^{-1}(\underline{x}_t - \underline{\mu})\right] \end{aligned} \tag{6.31}$$

Trace is applied because it is a scalar. Now note that components within a trace can be arbitrarily rearranged,

$$\begin{aligned} \ln \mathscr{L} &= -\frac{nT}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma\right|^{-1} - \frac{1}{2}\sum_{t=1}^{T}\mathrm{tr}\left[(\underline{x}_i - \underline{\mu})'\Sigma^{-1}(\underline{x}_t - \underline{\mu})\right] \\ &= -\frac{nT}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma\right|^{-1} - \frac{1}{2}\sum_{t=1}^{T}\mathrm{tr}\left[(\underline{x}_t - \underline{\mu})(\underline{x}_t - \underline{\mu})'\Sigma^{-1}\right] \\ &= -\frac{nT}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma\right|^{-1} - \frac{1}{2}\sum_{t=1}^{T}\mathrm{tr}\left[\mathrm{S}\Sigma^{-1}\right] \end{aligned} \tag{6.32}$$

Taking partial w.r.t. Σ^{-1} and setting it to 0 leads to (note that the inverse of a determinant is equal to the determinant of an inverse):

$$\frac{\partial}{\partial\Sigma^{-1}}\ln\mathscr{L} = \frac{T}{2}\left|\Sigma\right| - \frac{1}{2}\sum_{i=t}^{T}\mathrm{tr}\left[\mathrm{S}\right] = 0 \tag{6.33}$$

Now adding LASSO to equation (6.32),

$$\ln\mathscr{L}^* = -\frac{T}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma^{-1}\right| - \frac{1}{2}\sum_{t=1}^{T}\mathrm{tr}\left[(\mathrm{S}\Sigma^{-1}\right] - \lambda|\Sigma^{-1}| \tag{6.34}$$

The resulting partial is (and set it to 0):

$$\frac{\partial}{\partial\Sigma^{-1}}\ln\mathscr{L}^* = \frac{T}{2}\left|\Sigma\right| - \frac{1}{2}\sum_{i=t}^{T}\mathrm{tr}\left[\mathrm{S}\right] - \lambda|\Sigma^{-1}|\mathrm{tr}[\Sigma^{-1}] = 0 \tag{6.35}$$

6.6.4 Knowledge Graph

It is not easy to explain what a knowledge graph (KG) is. Superficially speaking, a KG is a network graph (undirected or directed) in which entities of interest (vertices) are connected via edges. There are various ways to model edges (i.e. knowledge).

One easiest way is to use precision matrix. That is, we find data of the entities. For example, if the entities are stocks, then we can collect their returns (single feature). The most intuitive way these stocks are connected is their correlation coefficients (or variance-covariance matrix). Since these stocks form a network (i.e. graph), we can study their conditional dependence – precision matrix. It is then quite straightforward to use the values in the precision matrix as edges. One can easily set a cutoff so that small conditional correlations are ignored.

A more sophisticated method is graphic lasso. As seen earlier, graphic lasso uses lasso regression to remove unimportant coefficients. This is equivalent to removing small conditional correlations.

The beauty of KG is that we do not necessarily use numeral data. We can also use textual data. Certainly textual data are not possible to compute variances and covariances. Yet there are methods to convert textual data to numeral numbers. See the NLP chapter for more details. The textual data can be processed in various ways to suit a chosen model (i.e. knowledge). For example, textual data can be used to compute similarity scores which can then be combined with neural network models to generate KGs.

A simple demonstration by neo4j is Figure 6.49 where a KG of Harry Potter is drawn. It is based upon the appearances of the characters. A sample program by neo4j can be found in https://neo4j.com/developer-blog/turn-a-harry-potter-book-into-a-knowledge-graph/. Interested readers are highly encouraged to explore it. Note that J.K. Rowling's novels are textual data, so it involves some NLP which is introduced in Chapter 4.

6.6.5 Graphic Database

(Wiki) A graphic database is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. A key concept of the system is the graph (or edge or relationship). The graph relates the data items in the store to a collection of nodes and edges, the edges representing the relationships between the nodes. The relationships allow data in the store to be linked together directly and, in many cases, retrieved with one operation. Graphic databases hold the relationships between data as a priority. Querying relationships is fast because they are perpetually stored in the database. Relationships can be intuitively visualized using graph databases, making them useful for heavily interconnected data.

(Amazon) Both graph databases and relational databases store data items with predefined relationships between them. However, they represent the data relationships very differently. Relational databases store data in a tabular format with rows

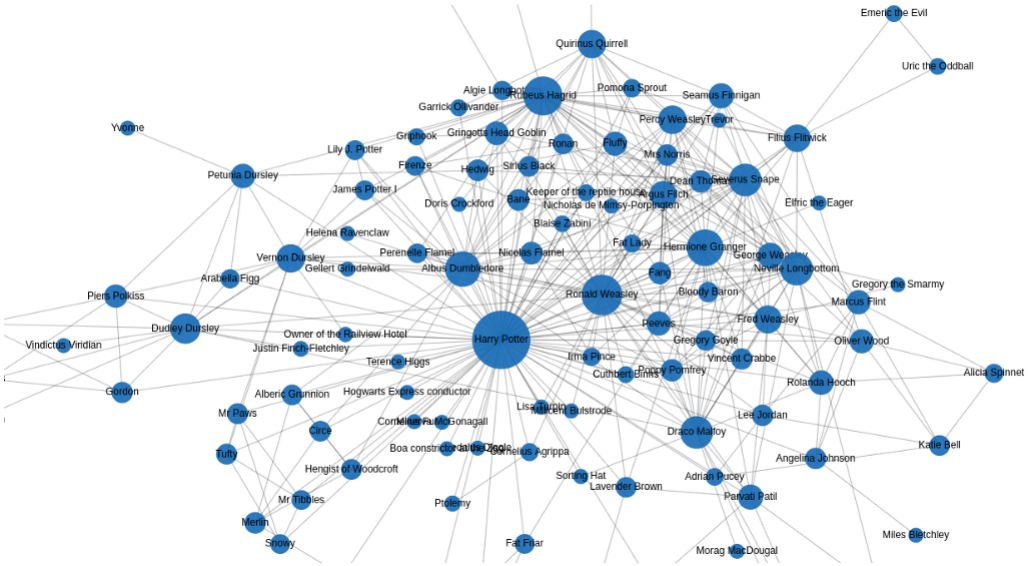


Figure 6.49: Knowledge Graph of Harry Potter

source: neo4j.com/developer-blog/turn-a-harry-potter-book-into-a-knowledge-graph

and columns. Related data is also stored in tables, and the data points link back to the original table. Operations related to data relationships become inefficient as they require multiple data table lookups. In contrast, a graph database stores data as a network of entities and relationships. It uses mathematical graph theory to store and perform operations on data relationships. Graph databases are much more efficient in relationship modeling. They improve application performance significantly for use cases with complex data interconnections.

Alina Luo.

6.7 Homework

Homework

Project

Chen and Zhang "From liquidity risk to systemic risk: A use of knowledge graph," Journal of Financial Stability, 2024.

Chapter 7

Directed Graph

Most studies in the health, social and behavioral sciences aim to answer causal rather than associative questions... many statistical researchers have not yet benefited from causal inference results in (i) counterfactual analysis, (ii) non-parametric structural equations, (iii) graphical models, and (iv) the symbiosis between counterfactual and graphical methods...

Judea Pearl, UCLA

7.1 Introduction

In this chapter, we particularly focus on "directed acyclic graph" (or DAG). DAG is a way to model causality. We will briefly discuss directed cyclical graph at the end of the chapter. In other words, this is a chapter of causation.

This chapter is heavily based upon Brady Neal's Lectures.[1]

7.2 Simpson's Paradox

Consider a Covid example where two treatments, A and B, are available. There are a total of 2,050 patients. 1,500 received treatment A and 550 received treatment B. 240 treated with A died and 105 treated with B died. The result is summarized in Table 7.1.

It seems like treatment A is more effective (causing lower death rate). Table

[1]Introduction to Causal Inference: from a machine learning perspective, December 17, 2020.

	total
A	16% (240/1500)
B	19% (105/550)
	$\mathbb{E}[Y\|T]$

Table 7.1: Covid Example

7.1 can be viewed as conditional expectations (conditional treatment): $\mathbb{E}[Y|T = A] = 16\%$ and $\mathbb{E}[Y|T = B] = 19\%$. The result (A is a better treatment) can be only true if the conditions of patients before treatment are identical.

Examine the following DAG, Figure 7.1. In this diagram, condition affects treatment. As a result, the above result is not valid.

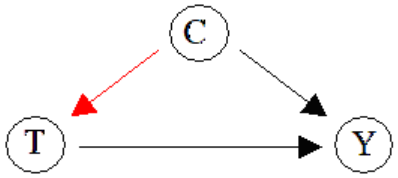


Figure 7.1: Simple Causality DAG

It is likely that some patients are sicker than others. Those who are sicker get a stronger medicine (treatment B) and those who are less sick receive treatment A (weaker drug). In such a case, then we need to look at the Table 7.2.

	mild	severe	total
A	15% (210/1400)	30% (30/100)	16% (240/1500)
B	10% (5/50)	20% (100/500)	19% (105/550)
	$\mathbb{E}[Y\|T, C = 0]$	$\mathbb{E}[Y\|T, C = 1]$	$\mathbb{E}[Y\|T]$

Table 7.2: Covid Example

But now we look at the result inside each condition, treatment B is always better (10% versus 15% in the mild condition and 20% versus 30% in the severe condition). In this situation, condition (mild or severe) matters. This is known as the Simpson's paradox.

The original result can be calculated as follows. Note that in the first calculation (treatment A), 15% is given a much larger weight ($\frac{1400}{1500}$) and 30% is given a much smaller weight ($\frac{100}{1500}$) – causing the weighted average death rate to bias toward 15%. Similarly, in the second calculation (treatment B), the bias is toward a higher death rate 20% ($\frac{500}{550}$).

$$\mathbb{E}[Y|T=A] = \mathbb{E}[Y|T=A;C=0]p(C=0|T=A) + \mathbb{E}[Y|T=A;C=1]p(C=1|T=A)$$
$$16\% = 0.15 \times \frac{1400}{1500} + 0.30 \times \frac{100}{1500}$$

and

$$\mathbb{E}[Y|T=B] = \mathbb{E}[Y|T=B;C=0]p(C=0|T=B) + \mathbb{E}[Y|T=B;C=1]p(C=1|T=B)$$
$$19\% = 0.10 \times \frac{50}{550} + 0.20 \times \frac{500}{550}$$

The last column of Table 7.2 is known as total association. This result does not consider the impact from C to T (clearly, condition can determine what treatment the patient is received), as indicated in Figure 7.1.

If we condition on C (either $C = 0$ for mild or $C = 1$ for severe), then we remove the impact from C to T, as shown in Figure 7.2, the we can directly measure the impact of treatment T. Note that from now on, conditioning on a variable is represented as a gray node (while intervention is represented as a blue node).

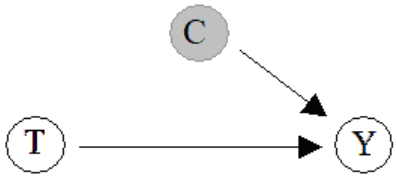


Figure 7.2: Simple DAG for Causality

It could be possible that treatment can affect condition. Imagine that treatment B is a rarer drug (stronger) and the wait time is longer. Hence in the treatment patients must wait a longer time, which could cause the condition of the patient to worsen, from mild to severe (this is the reason why in treatment B there are 500 out of 550 patients). On the other hand, treatment A is abundant and there is no wait. Hence whoever walks in gets treated right away, and no patients migrate to severe condition (1,400 out of 1,500 stay in mild condition).

This can be demonstrated in Figure 7.3. In this DAG, the effect of treatment on outcome has two paths: one directly from T to Y and the other indirectly through C. We will study such a DAG (frontdoor adjustment) later in the chapter.

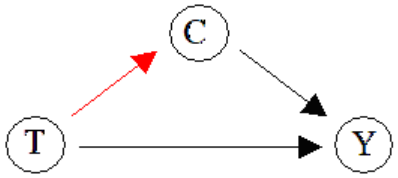


Figure 7.3: Simple Causality DAG

7.3 Chain, Fork, and Immorality

Chain, fork, and immorality are the building blocks of DAG. This is also often known as the V-structure. A complex DAG can be broken down into chains, forks, and immoralities and causal relationships can be examined.

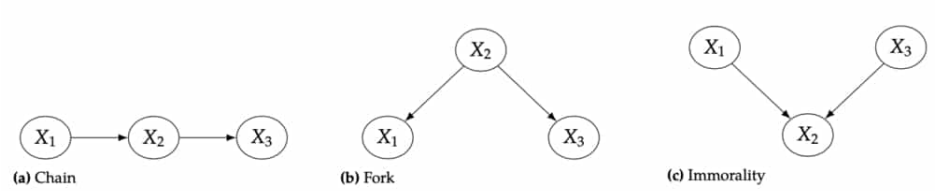


Figure 7.4: Chain, Fork, and Immorality

7.3.1 Chains and Forks

For chains and forks we can block association between variables by conditioning on the 'middle' variable. Notice that by conditioning on the 'middle' variable in the immorality we actually open up association between the variables by forcing the value of the parent variables.

As in undirected graphs, we can write the mathematical equation (Bayes rule) for chains and forks. In particular, we can write the joint probability distribution of a DAG in a stream of conditional probabilities. This has a indication of maximum likelihood (details later) estimation.

For example, the equation for the chain in Figure 7.4 can be written as:

$$p(x_1, x_2, x_3) = p(x_3|x_2)p(x_2|x_1)p(x_1) \tag{7.1}$$

Then, we can show that x_1 and x_3 are independent conditional on x_2:

$$\begin{aligned} p(x_1, x_3|x_2) &= \frac{p(x_1, x_2, x_3)}{p(x_2)} \\ &= \frac{p(x_3|x_2)p(x_2|x_1)p(x_1)}{p(x_2)} \\ &= p(x_3|x_2) \times \frac{p(x_1, x_2)}{p(x_2)} \\ &= p(x_1|x_2)p(x_3|x_2) \end{aligned} \tag{7.2}$$

which says x_1 and x_3 are independent given x_2.

The equation for the fork in Figure 7.4 is:

$$p(x_1, x_2, x_3) = p(x_1|x_2)p(x_3|x_2)p(x_2) \tag{7.3}$$

The derivation that x_1 and x_3 are independent conditional on x_2 is straightforward and is left for exercise.

7.3.2 Immoralities

Immorality is so named because two unassociated (unwed) parents share a child (which is immoral). Immorality is also known as collider. As clear in Figure 7.4, x_1 and x_3 are independent (and x_2 the child depends on its two parents.) Hence, the Bayesian equation for it is:

$$p(x_1, x_2, x_3) = p(x_1)p(x_3)p(x_2|x_1, x_3) \tag{7.4}$$

While it is apparent that x_1 and x_3 are independent, it is not so apparent that x_1 and x_3 are NOT independent once x_2 is conditioned. To demonstrate that conditional on x_2, x_1 and x_3 are not independent, consider the following simple example.

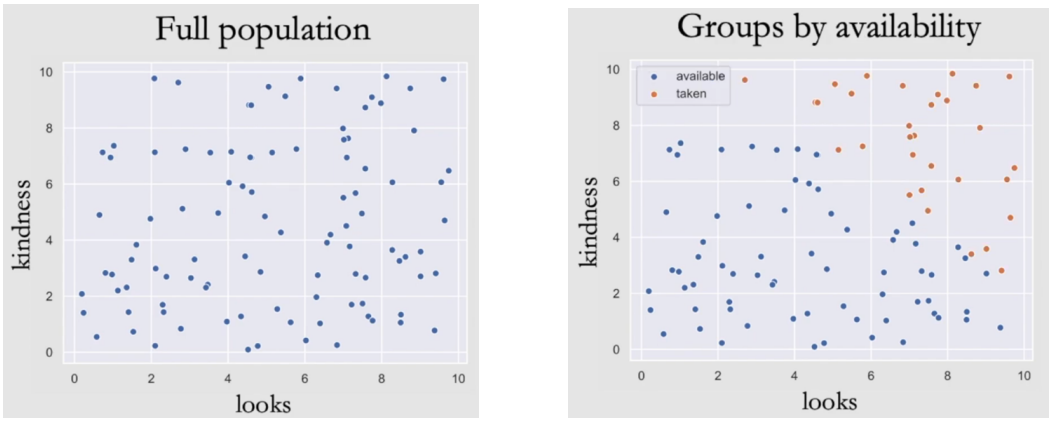
$$x_1 = \begin{cases} 1 & \text{good-looking} \\ 0 & \text{bad-looking} \end{cases} \tag{7.5}$$

and

$$x_3 = \begin{cases} 1 & \text{kind} \\ 0 & \text{jerk} \end{cases} \tag{7.6}$$

and

$$x_2 = x_1 \text{ and } x_3 = \begin{cases} 1 & \text{in relationship} \\ 0 & \text{else} \end{cases} \tag{7.7}$$



(a) (b)

Figure 7.5: Example of Collider

Hence, $x_2 = 1$ can only be 'good-looking' AND 'kind'. When conditioning on $x_2 = 0$, then the group can be either 'bad-looking' OR 'jerk'.

This will result in Figure 7.5. The general distribution of x_1 and x_3 are on the left. On the right, the red dots are those in relationships who are both kind and good-looking. Those blue dots are either bad-looking or jerks. Clearly, conditioning on $x_2 = 1$ (i.e. in relationship), x_1 and x_3 are negatively correlated.

7.4 Causation

As mentioned, the main purpose of DAG is for studying causation. It is because of the amazing theory of DAG, causation can be well-studied. The major beneficiary is the medical research. As we know, we cannot rely on statistical significance to advance a drug. We must rely on causal significance. Human trials are all small samples and hence hardly able reach statistical significance anyway.

7.4.1 Correlation is not Causation

"The number of suicides in hanging" and "expenditure on science, space and technology in the United States", "the number of drowning in swimming pool" and "the number of film appearances by Nicolas Cage", "cheese consumption per American" and "the number of tangled with bed sheets and deaths". Unrelated events? Their

correlations may surprise you.[2]

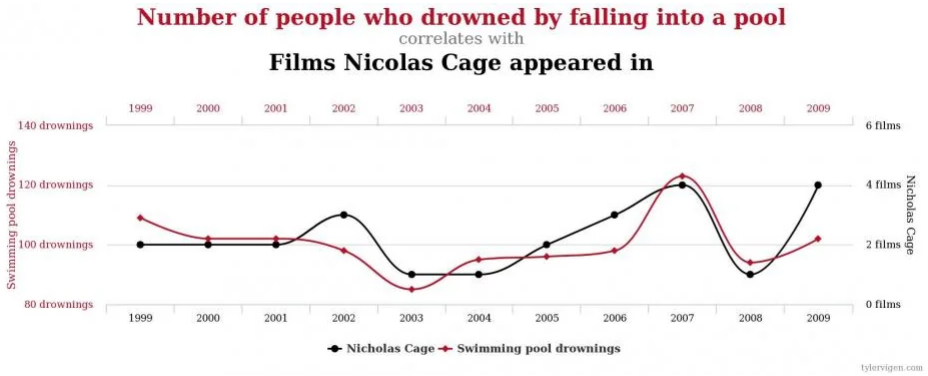


Figure 7.6: Cage Movies and Pool Drownings

7.4.2 Counterfactual

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1239917/#: :text=In 20the 20counterfactual 20model 2C 20a,multiple 20causal 20factors 20are 20allowed.

As already mentioned, one can evaluate a fixed individual i at a fixed time only under one condition. Usually, no objective criteria exist to assess with a single observation whether an outcome, such as treatment success, has been caused by the received treatment or by other factors. In the absence of such criteria, one can only estimate average causal effects. This requires several observations, involving different individuals or different time points or both. Many observations are also required for statistically stable conclusions.

The treatment that individual i actually does not receive is called "counterfactual" treatment. Likewise, the outcome under this treatment is referred to as counterfactual or "potential outcome". The term potential outcome reflects the perspective before the treatment assignment and is more widespread in statistics. In contrast, the term counterfactual outcome denotes the perspective after the allocation; it originated in philosophy and has caught on in epidemiology.

Details are in the Potential Outcome Section below.

[2]Ice Cream Sales vs. Shark Attacks, Master's Degrees vs. Box Office Revenue, Pool Drownings vs. Nuclear Energy Production, Measles Cases vs. Marriage Rate, High School Graduates vs. Pizza Consumption.

7.4.3 Confounding

Recall earlier, we have a DAG with condition affecting treatment in a Covid example. Now consider a similar example where we observe a majority of people who go to bed with their shoes on and wake up in the morning with a headache. Such high correlation can be caused by one of the two reasons:

- shoes-on are drinkers and shoes-off are non-drinkers – this explains the correlation
- confounding – total association is a combination of confounding and causal association

So now, we introduce a formal terminology of Figure 7.1. The fact that condition has a causal effect on treatment is called "confounding". We draw the shoes-on DAG in Figure 7.7.

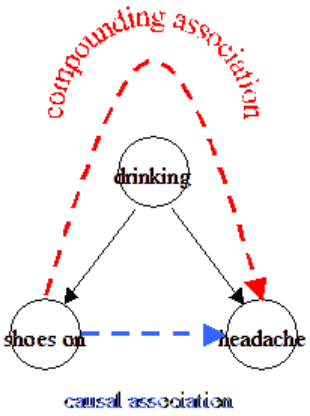


Figure 7.7: Shoes-on and Headache

The people who sleep with their shoes on also drank heavily the night before. So drinking the night before is the common cause.

One way to think of this problem is that shoe-sleepers are different from non-shoe-sleepers. The former drank the night before and the latter did not. So the groups are not comparable. You would want the two groups to be comparable (homogeneous). You would want them to be the same in every way except for the treatment (sleep with shoes on).

The other way to think about this is confounding. The common cause (drinking) confounds shoe-sleeping and headache. In Figure 7.7, there is a confounding association from shoes-on and headache (red dotted line). This is different from the causal association (blue dotted line) we are interested in.

Total association (i.e. correlation) is the combination of the two.

7.4.4 Potential Outcome and do-Operation

Suppose you have a headache. If you take a pill and the headache goes away, then you know the pill makes the headache go away. Suppose you also know that if you don't take the pill and the headache doesn't go away, then you can reach the conclusion that the pills makes the headache go away. But if you don't take the pill and the headache still goes away, then you could not be sure if the pill makes the headache go away (the pill is just a sugar pill).

To study this, we need to first introduce a new concept in causality – potential outcome. Formally (in a binary case),

$$Y_i|\text{do}(T=1) \triangleq Y_i(1)$$
$$Y_i|\text{do}(T=0) \triangleq Y_i(0)$$

where T is treatment, Y is outcome, and i is individual. The operation $\text{do}(T=1)$ is new. It describes a treatment (of 1) is performed. Note that

$$Y_i|\text{do}(T=1) \neq Y_i|T=1$$

More details will be given later. $\text{do}(T=1)$ here represents taking the pill. The causal effect is then

$$Y_i(1) - Y_i(0) \tag{7.8}$$

This is known as the ITE (individual treatment effect). We want to measure the direct (causal) treatment effect.

7.4.5 Fundamental Problem of Causal Effect

There is a fundamental problem in calculating equation (7.8). Note that $Y_i(1)$ is take the outcome of taking the pill and $Y_i(0)$ is the outcome of not taking the pill. However, for individual i, either he does take the pill (and wait for the outcome) or does not take the pill (and also wait for the outcome). He cannot take and not take the pill at the same time. In other words, we will not be able to compute equation (7.8) for individual i. This is the fundamental problem of causal effect.

Take a look at Table 7.3. Three individuals (2, 3, 6) received treatment and three did not (1, 4, 5). Since these two groups do not overlap, we cannot calculate $Y_i(1) - Y_i(0)$, the ITE (individual treatment effect).

i	T	Y	$Y(1)$	$Y(0)$	$Y(1) - Y(0)$
1	0	0	?	0	?
2	1	1	1	?	?
3	1	0	0	?	?
4	0	0	?	0	?
5	0	1	?	1	?
6	1	1	1	?	?

Table 7.3: The Fundamental Problem of Causal Effect

Hence, the best we can do is to average across individuals – known as ATE, or average treatment effect. It is incorrect to think of ATE as the difference of average $Y(1)$ which is $\frac{2}{3}$ and the average $Y(0)$ which is $\frac{1}{3}$. This is because the data for $Y_i(1)$ and the data for $Y_i(0)$ are not comparable (actually in this case, they are non-overlapping). In fact, they are (due to non-overlapping) conditional on treatment. To see this formally, ATE is calculated as:

$$\begin{aligned} \frac{1}{N}\sum_{i=1}^{N} Y_i(1) - Y_i(0) &= \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] \\ &\neq \mathbb{E}[Y|T=1] - \mathbb{E}[Y|T=0] \\ &= \tfrac{2}{3} - \tfrac{1}{3} = \tfrac{1}{3} \end{aligned} \tag{7.9}$$

The second line is a usual statistical equation (conditional expectation). It would be nice if the causal effect can be calculated via a statistical equation. But it cannot. This is of course because we have confounding association, as indicated in Figure 7.7.

Hence, if we can remove confounding, then Figure 7.7 becomes Figure 7.8.

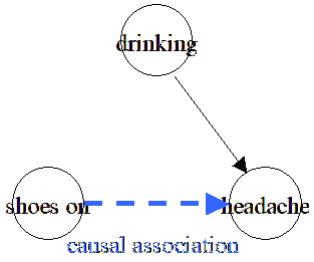


Figure 7.8: Shoes-on and Headache: Confounding Removed

Then, direct causal effect can be measured. It turns out that this is exactly

the Randomized Control Trials (RCT) do. Let's turn to the next sub-section.

7.4.6 Randomized Control Trials

It means treatment can only be determined by a coin flip. It cannot have any causal parent. Recall Figure 7.2 where C is conditioned (gray node). Here we randomize T which effectively cuts off its dependence on C. We use blue node to represent RCT. See Figure 7.9.

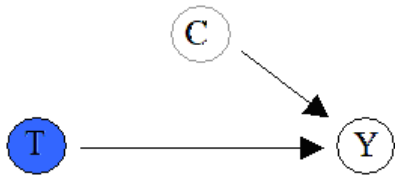


Figure 7.9: Randomized Control Trials

Another way of thinking of this is that the treatment groups must be comparable. Let's go back to the shoe-on example. The problem there is that shoe-sleepers are different from non-shoe-sleepers. Shoe-sleepers drank the night before and non-shoe-sleepers didn't.

Hence, when there is no confounding,

$$\mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] = \mathbb{E}[Y|T=1] - \mathbb{E}[Y|T=0] \tag{7.10}$$

We now turn the inequality of equation (**??**) into equality of equation (7.10). Simply speaking, T cannot have any causal parents. Randomization easily takes care of that.

Think of the following example where one sample has all the red dots (people sleep with their shoes on) and the other all the blue dots (people sleep without their shoes on), as in Figure 7.10. The two samples are not comparable. Hence their outcomes are not comparable.

RCT randomly assigns observations to each sample, as in Figure 7.11, and as a result, the two samples are now comparable, then equation (7.10) holds. Understandably, RCT can only be implemented in an experimental environment (e.g. drug trials). For an economic subject, RCT cannot be implemented. We simply cannot assign different inflationary scenarios on our economic data. As we can see, RCT is powerful and yet it is not easily available for the problems we are interested in.

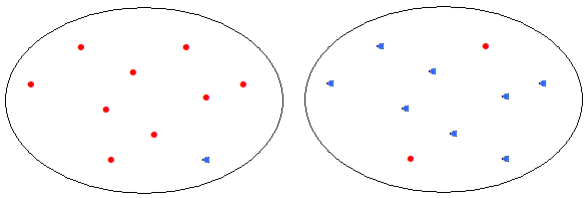


Figure 7.10: Randomized Control Trials

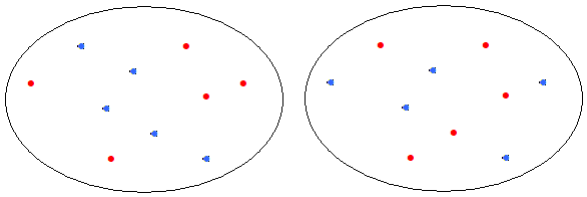


Figure 7.11: Randomized Control Trials

Alternatively, we can think of extrapolating the samples of the two groups, as in Figure 7.12. Here, both samples contain observations with all outcomes. However, we cannot do this via RCT. This can be only achieved via structural equations which will be introduced later.

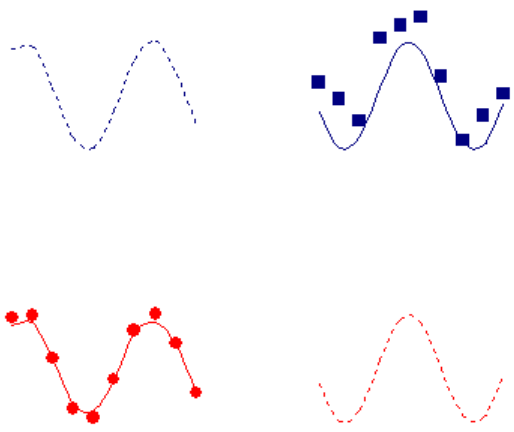


Figure 7.12: Randomized Control Trials

7.5 Backdoor Adjustment (Use of Observational Data)

While RCT takes care of everything, it is not easily implemetable. There are many reasons why – ethical, political, cost, limitation of economic data, etc. In reality, we only have observational data that carry confounding association. So the question is how to remove confounding from observational data, so the equality can hold in equation (**??**), that is equation (7.10).

To do that, we need to condition on the confounder. Let W be the confounder which is conditioned as in Figure 7.13 (similar to treatment in Figure 7.2). By conditioning W, we cutoff the causal parent of T. Hence, the impact of T on Y is a direct measure of causality.

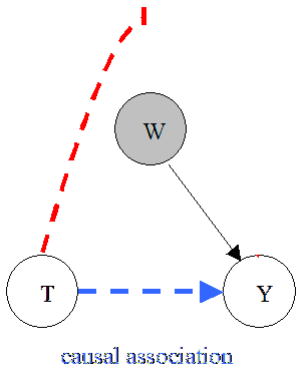


Figure 7.13: Condition on Confounder

Note that the red dotted line is the confounding association, which is now being cutoff. This is known as "backdoor" adjustment. Mathematically, we can write it as:

$$\begin{aligned}\mathbb{E}[Y(t)|W=w] &\triangleq \mathbb{E}[Y|\text{do}(T=t), W=w] \\ &= \mathbb{E}[Y|t,w]\end{aligned}$$

provided that W is the only confounder. If there are other confounders, then they all must be conditioned.

Then we can simply integrate (sum over) the confounder W to obtain ATE

(average treatment effect):

$$\begin{aligned}\mathbb{E}[Y(t)] &\triangleq \mathbb{E}[Y|\text{do}(T=t)] \\ &= \mathbb{E}[\mathbb{E}[Y|\text{do}(T=t), W]] \quad \text{law of iterative expectations} \\ &= \mathbb{E}[\mathbb{E}[Y|t, W]] \qquad \text{only if } W \text{ is the only confounder} \\ &= \textstyle\sum_w \mathbb{E}[Y|t, w]\end{aligned}$$

Now, let's get back to the Covid example.

$$\begin{aligned}\mathbb{E}[Y(t)] &\triangleq \mathbb{E}[Y|\text{do}(T=t)] \\ &= \textstyle\sum_c \mathbb{E}[Y|t, c]\end{aligned}$$

From Table 7.2, we know the two probabilities associated with two conditions (mild and severe) are:

$$p(C=0) = \frac{1450}{2050}$$
$$p(C=1) = \frac{600}{2050}$$

where 0 is mild and 1 is severe. Plugging these probabilities into equation (7.5), we have:

$$\begin{aligned}\mathbb{E}[Y(T=A)] &= p(C=0)\mathbb{E}[Y|A,0] + p(C=1)\mathbb{E}[Y|A,1] \\ &= \tfrac{1450}{2050} \times 0.15 + \tfrac{600}{2050} \times 0.30 = 0.194\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[Y(T=B)] &= p(C=0)\mathbb{E}[Y|B,0] + p(C=1)\mathbb{E}[Y|B,1] \\ &= \tfrac{1450}{2050} \times 0.10 + \tfrac{600}{2050} \times 0.20 = 0.129\end{aligned}$$

Now, we can revise Table 7.2 as follows in Table 7.4. The causal effects are opposite of the unconditional (total association or general correlation) effects. We can find the difference of ATE as $0.194 - 0.129 = 6.5\%$.

Simpson's paradox is hence resolved!

7.6 Structural Equation

Although DAG described in the previous section is useful (and powerful) in measuring true causality, it is often hard to implement. This is because we do not know if we have blocked all the confounders. The reason is that many confounders are unobservable. Since they are unobservable, they cannot be blocked. Furthermore,

	mild	severe	total	causal
A	15% (210/1400)	30% (30/100)	16% (240/1500)	19.4%
B	10% (5/50)	20% (100/500)	19% (105/550)	12.9%
	$\mathbb{E}[Y\|T, C=0]$	$\mathbb{E}[Y\|T, C=1]$	$\mathbb{E}[Y\|T]$	$\mathbb{E}[Y\|\text{do}(t)]$

Table 7.4: Covid Example: Causal Effect

we don't even know if they exist. As a result, often we write explicit (structural) equations, frequently linear, of the graph. This allows us to estimate the model more easily. See Figure 7.14.

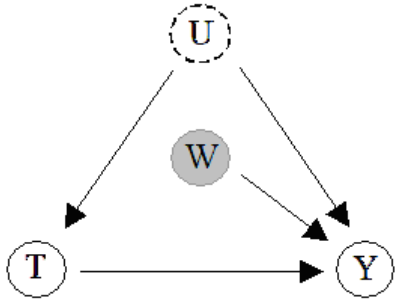


Figure 7.14: Hidden Confounder(s)

7.6.1 A Simple Example

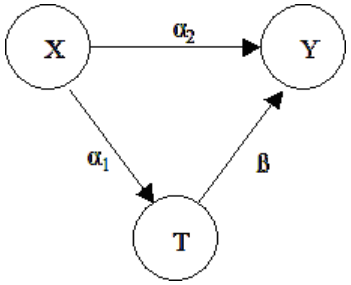


Figure 7.15: Structural Equation: $Y = \beta T + \alpha_2 X$

The structural equations are:

$$T = \alpha_1 X$$
$$Y = \beta T + \alpha_2 X$$

We would like to get $\mathbb{E}[Y|\text{do}(t)]$:

$$\begin{aligned}\mathbb{E}[Y|\text{do}(t)] &= \sum\nolimits_X \mathbb{E}[Y|T=t,X]p(X) \\ &= \sum\nolimits_X \mathbb{E}[\beta t + \alpha_2 X|T=t,X]p(X) \\ &= \sum\nolimits_X (\beta t + \alpha_2 X)p(X) \\ &= \beta t + \alpha_2 \mathbb{E}[X]\end{aligned}$$

and then

$$\frac{\partial}{\partial t}\mathbb{E}[Y|\text{do}(t)] = \beta$$

But (note that $X = T/\alpha_1$)

$$\begin{aligned}\mathbb{E}[Y|T=t] &= \mathbb{E}[\beta T + \alpha_2 X|T=t] \\ &= \mathbb{E}[\beta T + \alpha_2 \tfrac{T}{\alpha_1}|T=t] = \beta t + \tfrac{\alpha_2}{\alpha_1} t\end{aligned}$$

and then

$$\frac{\partial}{\partial t}\mathbb{E}[Y|T=t] = \beta + \tfrac{\alpha_2}{\alpha_1}$$

known as the confounding bias.

7.6.2 Another Example

Given the following structural equations:[3]

$$\begin{aligned}X_1 &= 0.8X_2 + e_1 \\ X_2 &= e_2 \\ X_3 &= 0.8X_2 + e_3 \\ Y &= -X_1 + 2X_2 - X_3 + \varepsilon\end{aligned}$$

we can draw Figure 7.16,

First, we apply multiple regression $y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$. Then we get $\beta_1 = -1$, $\beta_2 = 2$, and $\beta_3 = -1$. Hence X_2 is the most influential variable.

[3]See p.3138 of "Estimating High-Dimensional Intervention Effects from Observational Data," Marloes H. Maathuis, Markus Kalisch, and Peter Buhlmann, 2009, The Annals of Statistics, p.3133-3164.

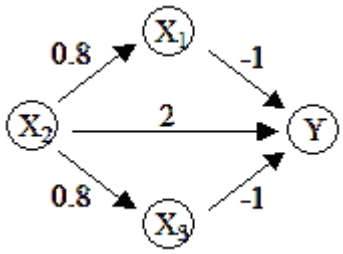


Figure 7.16: Structural Equation: $Y = -X_1 + 2X_2 - X_3$

Now, we apply intervention calculus. Let $\Theta = \{\theta_1, \theta_2, \theta_3\}$ where θ_i represents the causal effect of X_i on Y. Since $\text{pa}_1 = \{X_2\}$, $\text{pa}_2 = \{\}$, and $\text{pa}_3 = \{X_2\}$, we have

$$\theta_1 = \beta_{1|X_2} = -1$$
$$\theta_2 = \beta_{2|\{\}} = 0.4 \ (= 2 - 1 \times 0.8 - 1 \times 0.8)$$
$$\theta_3 = \beta_{3|X_2} = -1$$

We see that $\theta_1 = \beta_1$, $\theta_3 = \beta_3$, but $\theta_2 \neq \beta_2$. Hence X_2 is the least influential variable.

Now, see a different graph, as in Figure 7.17. The structural equation is $Y = X_1 + X_3 + \varepsilon$. Applying regression leads to $\beta_1 = 1$, $\beta_2 = 0$, and $\beta_3 = 1$ and X_2 is the least important.

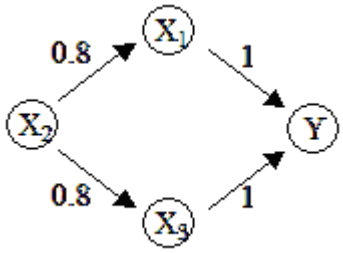


Figure 7.17: Structural Equation: $Y = X_1 + X_3$

Consider intervention. We get

$$\theta_1 = \beta_{1|X_2} = 1$$
$$\theta_2 = \beta_{2|\{\}} = 1.6 \ (= 1 \times 0.8 + 1 \times 0.8)$$
$$\theta_3 = \beta_{3|X_2} = 1$$

We again see that $\theta_1 = \beta_1$, $\theta_3 = \beta_3$, but $\theta_2 \neq \beta_2$. Hence X_2 is the most influential variable.

7.6.3 Sodium Intake Example

Sodium intake simulations by Luque-Fernandez et. al. (2019)[4] The equations are (both u and v are standard normal variates):

$$\text{blood pressure} = \beta_1 \times \text{sodium} + \beta_2 \times \text{age} + u$$

where $\beta_1 = 1.05$ and $\beta_2 = 2.00$; and

$$\text{proteinuria} = \gamma_1 \times \text{SBP} + \gamma_2 \times \text{sodium} + v$$

where $\gamma_1 = 2.0$ and $\gamma_2 = 2.8$.

7.7 Markov Equivalence Class

They share the same skeleton. Chains and forks.

Immoralities are themselves a Markov equivalence class. Hence, immoralities are not in the same Markov eqv. class as chains and forks. they are on their own Markov equivalence class.

Markov equivalence class is same as CPDAG (Completed Partially Directed Acyclic Graph).

7.8 d-Separation and do-Calculus

7.8.1 d-Separation

The concept of d-separation is a criterion for deciding, from a given a causal graph, whether a set X of variables is independent of another set Y, given a third set Z.

This section is taken from (see d-separation.doc) https://bayes.cs.ucla.edu/BOOK-2K/d-sep.html

Unconditional Separation

Rule 1: X and Y are d-connected if there is an *unblocked path* between them.

[4]"Educational Note: Paradoxical Collider Effect in the Analysis of Non-communicable Disease Epidemiological Data: A Reproducible Illustration and Web Application," International Journal of Epidemiology, 2019 (April), 48(2):640-653.

By a "path" we mean any consecutive sequence of edges, disregarding their directionalities. By "unblocked path" we mean a path that can be traced without traversing a pair of arrows that collide "head-to-head". In other words, arrows that meet head-to-head do not constitute a connection for the purpose of passing information, such a meeting will be called a "collider".

Example 1: See Figure 7.18. This graph contains one collider, at T. The path $X - R - S - T$ is unblocked, hence X and T are d-connected. So is also the path $T - U - V - Y$, hence T and Y are d-connected, as well as the pairs U and Y, T and V, T and U, X and S etc.... However, X and Y are not d-connected; there is no way of tracing a path from X to Y without traversing the collider at T. Therefore, we conclude that X and Y are d-separated, as well as X and V, S and U, R and U, etc. (The ramification is that the covariance terms corresponding to these pairs of variables will be zero, for every choice of model parameters).

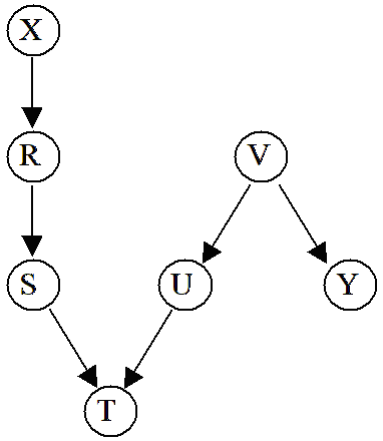


Figure 7.18: d-Separation Rule 1

Blocking by Conditioning

Rule 2: X and Y are d-connected, conditioned on a set $\mathbb{Z}$ of nodes, if there is a collider-free path between X and Y that traverses no member of $\mathbb{Z}$. If no such path exists, we say that X and Y are d-separated by $\mathbb{Z}$, We also say then that every path between X and Y is "blocked" by $\mathbb{Z}$.

Example 2: See Figure 7.19. Let $\mathbb{Z}$ be the set $\{R, V\}$ (boxed letters in the figure). Rule 2 tells us that X and Y are d-separated by $\mathbb{Z}$, and so are also X and S, U and Y, S and U etc. The path $X - R - S$ is blocked by $\mathbb{Z}$, and so are also the paths $U - V - Y$ and $S - T - U$. The only pairs of unmeasured nodes that remain d-connected in this example, conditioned on $\mathbb{Z}$, are S and T and U and T. Note

that, although T is not in $\mathbb{Z}$, the path $S-T-U$ is nevertheless blocked by $\mathbb{Z}$, since T is a collider, and is blocked by Rule 1.

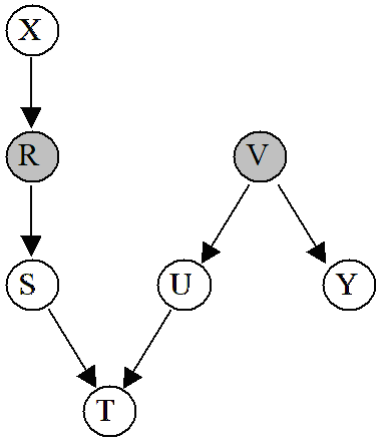


Figure 7.19: d-Separation Rule 2

Conditioning on Colliders

Rule 3: If a collider is a member of the conditioning set $\mathbb{Z}$, or has a descendant in $\mathbb{Z}$, then it no longer blocks any path that traces this collider.

Example 3: See Figure 7.20. Let $\mathbb{Z} = \{R, P\}$ (shadowed letters in the figure). Rule 3 tells us that S and Y are d-connected by $\mathbb{Z}$, because the collider at T has a descendant (P) in $\mathbb{Z}$, which unblocks the path $S-T-U-V-Y$. However, X and U are still d-separated by $\mathbb{Z}$, because although the linkage at T is unblocked, the one at R is blocked by Rule 2 (since R is in $\mathbb{Z}$).

This completes the definition of d-separation, and the reader is invited to try it on some more intricate graphs.

Typical application: Suppose we consider the regression of y on p, r and x,

$$y = c_1 p + c_2 r + c_3 x$$

and suppose we wish to predict which coefficient in this regression is zero. From the discussion above we can conclude immediately that c_3 is zero, because Y and X are d-separated given P and R, hence the partial correlation between Y and X, conditioned on P and R, must vanish. c_1 and c_2, on the other hand, will in general not be zero, as can be seen from the graph: $\mathbb{Z} = \{R, X\}$ does not d-separate Y from P, and $\mathbb{Z} = \{P, X\}$ does not d-separate Y from R.

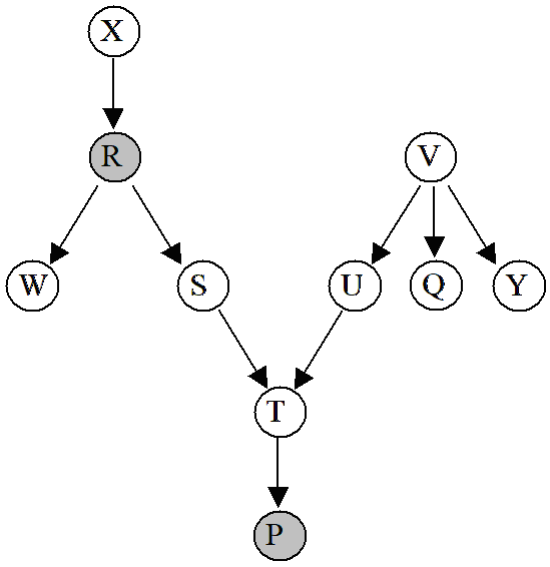


Figure 7.20: d-Separation Rule 3

7.8.2 do-Calculus

https://www.youtube.com/watch?v=rdQ-VwTYPyQ (25'50")

Pearl: "In applications involving identifiability, the role of the do-calculus is to remove the do=operator from the query expression." where "query expression" is the one involves do$Y$. To be specific, a query expression is identifiable if $p(y|\text{do}(x)) = \sum_z p(y|x,z)p(z)$

https://pyimagesearch.com/2023/11/27/a-brief-introduction-to-do-calculus/

In simple words, this means to identify the effect or effects for a particular cause from data that is continuous rather than having discrete values.

Consider the following directed acyclic graph in Figure 7.21 (G) where X, Y, Z, and W are arbitrary disjoint nodes. $G_{\bar{X}}$ is the manipulated graph where all incoming edges to X have been removed.

Rule 1: Insertion/Deletion of Observation

$$P(y|\text{do}(x), z, w) = P(y|\text{do}(x), w) \text{ if } (Y \perp\!\!\!\perp Z|X, W) \text{ for } G_{\overline{X}}$$

This means that if Y is d-separated from Z given X and W, then the expression of probability $P(y|\text{do}(x), z, w)$ resolves to $P(y|\text{do}(x), w)$. An easier way to understand this is by getting rid of the do-operators on both sides of the equality

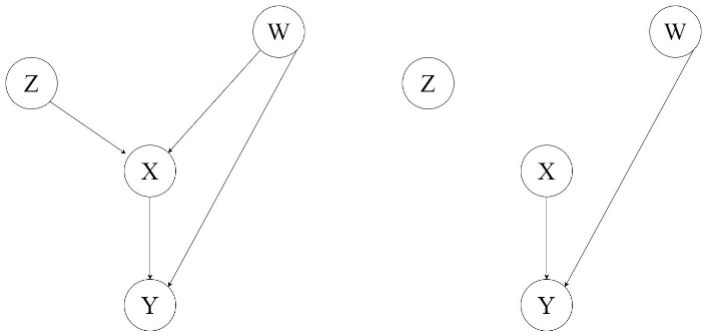


Figure 7.21: Parents of X Removed

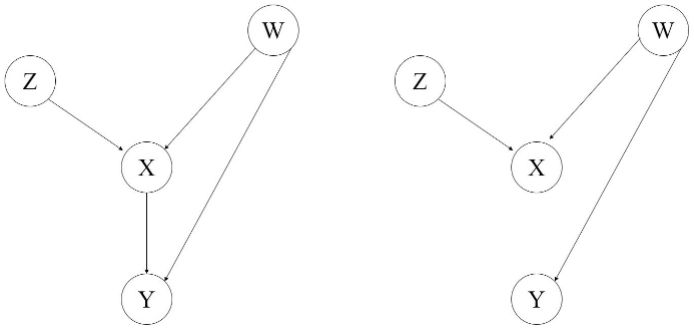


Figure 7.22: Children of X Removed

sign.

$$P(y|z,w) = P(y|w) \text{ if } (Y \perp\!\!\!\perp Z|W) \text{ for } G$$

The above expression simply implies conditional independence within the variables in the distribution given regular d-separation.

Rule 2: Action/Observation Exchange

$$P(y \mid \text{do}(x), \text{do}(z), w) = P(y|\text{do}(x), z, w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W) \text{ for } G_{\overline{X}\underline{Z}}.$$

To simplify the expression above, let us again remove do(x) or consider X to be an empty set.

$$P(y|\text{do}(z), w) = P(y|z, w) \text{ if } (Y \perp\!\!\!\perp Z|W) \text{ for } G_{\underline{Z}}$$

This expression refers to the backdoor-adjustment criteria that we saw in Part 3. Therefore, this rule gives us the interventional distribution for the backdoor adjustment criteria.

Rule 3: Insertion/Deletion of Action

$$P(y|\text{do}(x), \text{do}(z), w) = P(y|\text{do}(x), w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W) for G_{\overline{X}\,\overline{Z(W)}}$$

where $Z(W)$ is the set of Z nodes that are not ancestors of any W node in $G_{\overline{X}}$.

Again, for the sake of simplification, let us remove the do(x) operator from the above expression.

$$P(y|\text{do}(z), w) = P(y|w) \text{ if } (Y \perp\!\!\!\perp Z|W) \text{ for } G_{\overline{Z(W)}}$$

Let's pause here and really understand what this means. On the paper, it means that we can remove the intervention term do(z) provided there is no causal association flowing from $Z \to Y (Y \perp\!\!\!\perp Z \mid W)$ in the graph $G_{\overline{Z(W)}}$.

But that's not all. We have a strange term called $Z(W)$, which does not quite fit in.

The simplified expression should have been:

$$P(y|\text{do}(z), w) = P(y|w) \text{ if } (Y \perp\!\!\!\perp Z \mid W) \text{ for } G_{\overline{Z}}$$

where removal of incoming edges to Z should result in the d-separation of Y and Z, and no causal association should flow from Z to Y. However, instead of this simple term, we end up with an expression containing $Z(W)$. To understand this better, let us consider Figure 7.23.

Now, the intuitive idea is to remove incoming edges to $Z(G_{\overline{Z}})$. But if we do that, then we risk changing the distribution of Y altogether through the backdoor path consisting of U and V.

Instead, what we can do is take a sub-node of Z, say Z_2, which is not an ancestor of any node in W, and then remove all the incoming edges to it $(G_{\overline{Z_2}})$. This is shown in Figure 7.24.

$\perp$

$X \perp\!\!\!\perp Y$

$X \not\perp\!\!\!\perp Y$

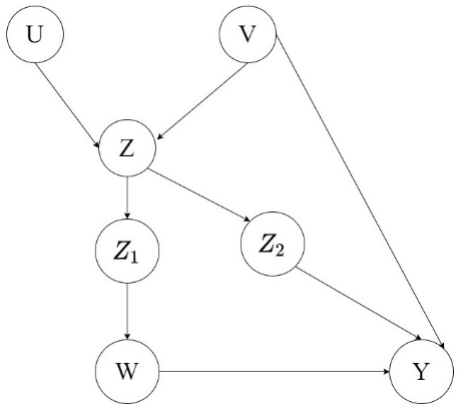


Figure 7.23: d-Separation

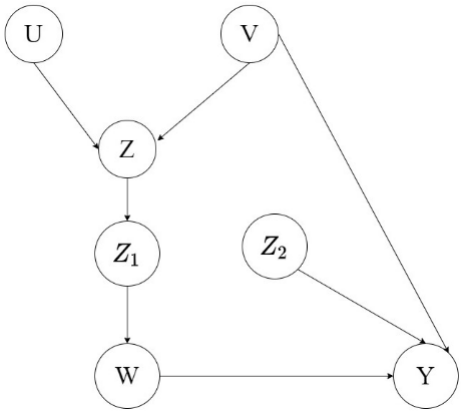


Figure 7.24: Removing incoming edges

7.9 Causal Discovery from Interventions

7.9.1 Two-node

7.9.2 Multi-node

7.10 Counterfatuals and Mediation

7.11 Markov Blanket and Faithfulness

7.11.1 Markov Blanket

(Wiki) In statistics and machine learning, when one wants to infer a random variable with a set of variables, usually a subset is enough, and other variables are useless. Such a subset that contains all the useful information is called a Markov blanket. If a Markov blanket is minimal, meaning that it cannot drop any variable without losing information, it is called a Markov boundary. Identifying a Markov blanket or a Markov boundary helps to extract useful features. The terms of Markov blanket and Markov boundary were coined by Judea Pearl in 1988. A Markov blanket can be constituted by a set of Markov chains.

In a Bayesian network, the Markov boundary of node A includes its parents, children and the other parents of all of its children.

7.11.2 Faithfulness

(https://library.fiveable.me/causal-inference/unit-9/directed-acyclic-graphs-dags/study-guide/Qpnmif3bOLg50HDp) Causal faithfulness assumption:

The causal faithfulness assumption states that the conditional independence relationships implied by the DAG are exactly those that hold in the probability distribution

This assumption rules out the possibility of accidental or fine-tuned cancellations of causal effects.

Faithfulness ensures that the causal structure can be inferred from the observed data and that the DAG provides a complete representation of the causal relationships.

7.12 Algorithms to Estimate DAG

Ideally, we can easily estimate a DAG using experimental data. Unfortunately, such data are often unavailable. What are more commonly available are observational data.

There are broadly three approaches in estimating a DAG using observational data:

1. constraint based (non parametric)
    - PC algorithm
    - IDA
    - DoWhy
2. score based (fitness function) (parametric)
    - likelihood
    - Bayesian
    - greedy search, simulated annealing, MCMC
3. hybrid

DAG is really hard to estimate given observational data. Many algorithms have been developed and yet they have their pros and cons. While no one algorithm dominates, PC-algorithm is the most popular one.

7.12.1 Basics

Recall the basic graphs of DAG – chain, fork, and immorality. Such a v-structure has four possible DAGs, as in Figure 7.25.

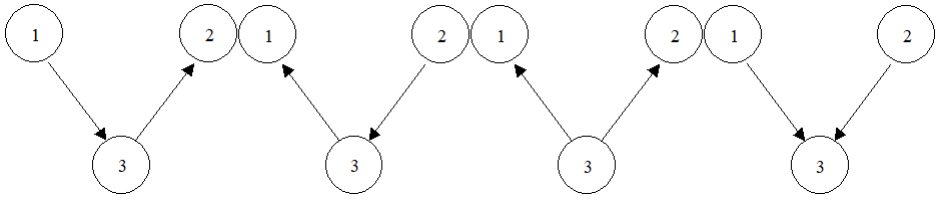


Figure 7.25: Four Possible v-structures

As introduced earlier, the first two are chains. The third is a fork. The last is a collider. Recall that in chains and forks, the two ends are conditionally independent (on the middle node). This is the first step in the PC-algorithm.

7.12.2 PC Algorithm

Peter-Clark algorithm is developed by Peter Spirtes and Clark Glymour in 1990. It is the most popular algorithm in estimating DAGs. PC algorithm is an FCI (fast causal inference). It is not perfect, yet it is most intuitive and easy to understand.

There are three basic steps in estimating a DAG:

1. identify the skeleton
2. identify immoralities and orient them
3. orient qualifying edges that are incident on colliders

Step 1: Conditional Independence Test

Between any two vertices, do the following:

- check pairs conditional on null set (i.e. unconditional independence):
  - Is $A \perp\!\!\!\perp B$? If yes, then remove the edge
  - Is $A \perp\!\!\!\perp C$? If yes, then remove the edge
  - $\cdots$ until all pairs are tested
- check pairs conditional on one other vertex:
  - $\cdots$ until all pairs are tested
- condition on two vertices
- condition on three vertices
- $\cdots$

Note that such conditional independent tests do not allow colliders (recall if a collider is conditioned then the two immoral parents will not be independent).

Step 2: Collider Identification

For every triplet, $X - Z - Y$,

- there is no edge between X and Y

- from step 1, we know that X and Y are not independent when conditioned on Z
- hence we know Z is a collider

Step 3: Identifying Directions

For the remaining edges, we investigate one by one (of every pair of vertices) which direction the arrow goes.

1. Rule 1: if k→i—j; then k→i→j
   if k→i–j, then i is a collider which is infeasible. See Figure 7.26
2. Rule 2: if i→k→j with i–j; then i→j
   if i←j, then i, j, and k form a cycle which is infeasible. See Figure 7.27
3. Rule 3: i—j, i—k, i—l, k→j, l→j but k and l not connected; then i→j if i←j (and i–l), then the other two undirected edges i—k and i—l will cause conflicts:
   (a) either i→k or i→l will create a cycle (i.e. i→k→j or i→l→j) but the other direction
   (b) i←k and i←l will make i a collider (i.e. k→i←l) which is a conflict. See Figure 7.28

(a) if k→i—j (b) then k→i→j

Figure 7.26: Rule 1

An Example

Assume a true graph as in Figure 7.29a. Given there are 5 vertices, we begin our edge elimination from a K_5 as shown in Figure 7.29b (see Chapter 6).

(a) if i→k→j with i—j (b) then i→j

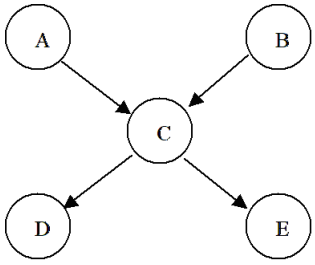
Figure 7.27: Rule 2

(a) if i—j, i—k, i—l, k→j, l→j but k and l not connected (b) then i→j

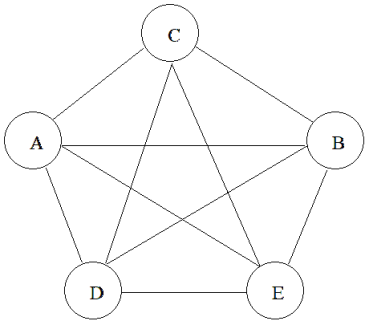
Figure 7.28: Rule 3

Is $A \perp\!\!\!\perp B \mid \phi$?	yes	remove edge
Is $A \perp\!\!\!\perp C \mid \phi$?	no	
Is $A \perp\!\!\!\perp D \mid \phi$?	no	
Is $A \perp\!\!\!\perp E \mid \phi$?	no	
Is $B \perp\!\!\!\perp C \mid \phi$?	no	
Is $B \perp\!\!\!\perp D \mid \phi$?	no	
Is $B \perp\!\!\!\perp E \mid \phi$?	no	
Is $C \perp\!\!\!\perp D \mid \phi$?	no	
Is $C \perp\!\!\!\perp E \mid \phi$?	no	
Is $D \perp\!\!\!\perp E \mid \phi$?	no	

Table 7.5: Condition on Empty Set



(a) True Graph



(b) Complete Graph

Figure 7.29: An Example

Step 1 is to test for conditional independence. We first condition on null set:

Hence, so far we can remove only one edge between A and B, as shown in Figure 7.30.

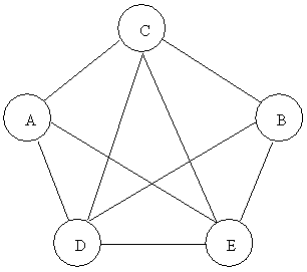


Figure 7.30: Remove A-B

Now we condition on one vertex at a time, as in Table 7.6.

Is $B \perp\!\!\!\perp C \mid A$?	no	
Is $B \perp\!\!\!\perp D \mid A$?	no	
Is $B \perp\!\!\!\perp E \mid A$?	no	
Is $C \perp\!\!\!\perp D \mid A$?	no	
Is $C \perp\!\!\!\perp E \mid A$?	no	
Is $D \perp\!\!\!\perp E \mid A$?	no	
Is $A \perp\!\!\!\perp C \mid B$?	no	
Is $A \perp\!\!\!\perp D \mid B$?	no	
Is $A \perp\!\!\!\perp E \mid B$?	no	
Is $C \perp\!\!\!\perp D \mid B$?	no	
Is $C \perp\!\!\!\perp E \mid B$?	no	
Is $D \perp\!\!\!\perp E \mid B$?	no	
Is $A \perp\!\!\!\perp D \mid C$?	yes	remove edge
Is $A \perp\!\!\!\perp E \mid C$?	yes	remove edge
Is $B \perp\!\!\!\perp D \mid C$?	yes	remove edge
Is $B \perp\!\!\!\perp E \mid C$?	yes	remove edge
Is $D \perp\!\!\!\perp E \mid C$?	yes	remove edge
Is $A \perp\!\!\!\perp C \mid D$?	no	
Is $B \perp\!\!\!\perp C \mid D$?	no	
Is $C \perp\!\!\!\perp E \mid D$?	no	
Is $A \perp\!\!\!\perp C \mid E$?	no	
Is $B \perp\!\!\!\perp C \mid E$?	no	
Is $C \perp\!\!\!\perp D \mid E$?	no	

Table 7.6: Condition on One Vertex

Now we remove five edges: $A - D$, $A - E$, $B - D$, $B - E$, and $D - E$. This is actually already the final skeleton. But since we do not observe the real graph, we will have to continue to condition on two vertices.

Is $A \perp\!\!\!\perp C \mid B, D$?	no
Is $A \perp\!\!\!\perp C \mid B, E$?	no
$\cdots$	no

Table 7.7: Condition on Two Vertices

and three vertices.

Is $A \perp\!\!\!\perp C \mid B, D, E$?	no
$\cdots$	no

Table 7.8: Condition on Three Vertices

The skeleton is now complete. See 7.31.

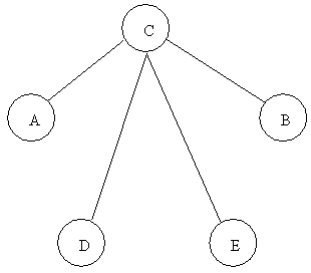


Figure 7.31: Remove B-D-E

The next step (step 2) is to examine colliders. For this, we need to examine $A - C - B$, $A - C - D$, $A - C - E$, $B - C - D$, $B - C - E$, and $D - C - E$. And quickly, we find that C is a collider for A and B and nowhere else. Hence, we now add arrows to $A \to C$ and $B \to C$ and update the graph accordingly. See Figure 7.32.

Now we only have two remaining edges to decide (using step 3) their directions $D - C - E$. Note that $D \not\to C$ because then C is collider (with A). Similarly, $E \not\to C$ because then C is collider (with A). Hence, the only case is that it $D \leftarrow C \to E$, which is Figure 7.33.

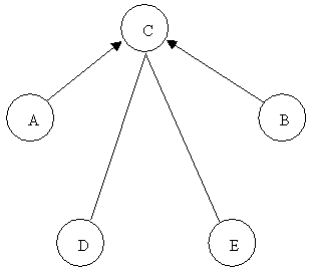
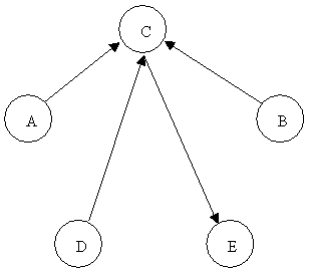
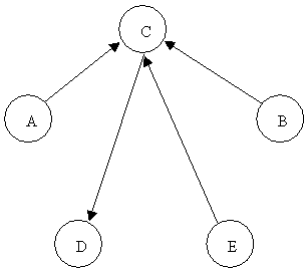


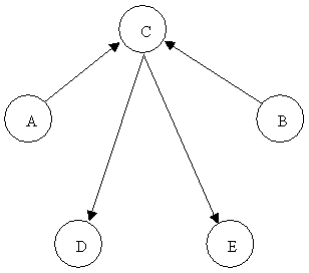
Figure 7.32: C is a Collider between A and B



(a) chain: $D \to C \to E$



(b) chain: $E \to C \to D$



(c) $D \leftarrow C \to E$

Figure 7.33: CPDAG

7.12.3 IDA

IDA (Intervention when the DAG is Absent) assumes that the data are multivariate Gaussian and faithful to the true (but unknown) underlying causal DAG (without hidden variables). Under these assumptions, IDA estimates the multiset of possible total causal effect of x on y, where the total causal effect is defined via Pearl's do-calculus as $\frac{\partial}{\partial z}\mathbb{E}[Y|\text{do}(X = z)]$ (this value does not depend on z, since Gaussianity implies that conditional expectations are linear). We estimate a multi-set of possible total causal effect instead of the unique total causal effect, since it is typically impossible to identify the latter when the true underlying causal DAG is unknown (even with an infinite amount of data).[5]

Apparently, IDA is part of the family of structural equation models (see the previous section 7.6). Since IDA assumes normality, from Chapter 6, we know that condtional dependence can be expressed via linear regressions. As a result of that, we can obtain regression coefficients that represent the strengths of edges – something PC algorithm cannot achieve.

[5]See "Causal Inference using Graphical Models with the R Package pcalg" in Journal of Statistical Software.

7.12.4 DoWhy

(https://www.pywhy.org/dowhy/v0.12/) Much like machine learning libraries have done for prediction, DoWhy is a Python library that aims to spark causal thinking and analysis. DoWhy provides a wide variety of algorithms for effect estimation, causal structure learning, diagnosis of causal structures, root cause analysis, interventions and counterfactuals.

(https://www.pywhy.org/dowhy/v0.8/) Much like machine learning libraries have done for prediction, "DoWhy" is a Python library that aims to spark causal thinking and analysis. DoWhy provides a principled four-step interface for causal inference that focuses on explicitly modeling causal assumptions and validating them as much as possible. The key feature of DoWhy is its state-of-the-art refutation API that can automatically test causal assumptions for any estimation method, thus making inference more robust and accessible to non-experts. DoWhy supports estimation of the average causal effect for backdoor, frontdoor, instrumental variable and other identification methods, and estimation of the conditional effect (CATE) through an integration with the EconML library.

(https://www.microsoft.com/en-us/research/blog/dowhy-a-library-for-causal-inference/)

7.13 Other Directed Graphs

When there is a cycle in a directed graph, there cannot be clear causality. Such graphs do exist and need to be analyzed. The most common metrics for studying such directed graphs (and yet not DAGs) are centrality measures. We can try to find which vertices are more important than others.

7.13.1 Shortest Path

(w3schools.com) To solve the shortest path problem means to check the edges inside the Graph until we find a path where we can move from one vertex to another using the lowest possible combined weight along the edges.

This sum of weights along the edges that make up a path is called a path cost or a path weight.

Algorithms that find the shortest paths, like Dijkstra's algorithm or the Bellman-Ford algorithm, find the shortest paths from one start vertex to all other vertices.

To begin with, the algorithms set the distance from the start vertex to all vertices to be infinitely long. And as the algorithms run, edges between the vertices

are checked over and over, and shorter paths might be found many times until the shortest paths are found at the end.

Every time an edge is checked and it leads to a shorter distance to a vertex being found and updated, it is called a relaxation, or relaxing an edge.

7.13.2 Centrality

From Wiki: "In graph theory and network analysis, indicators of centrality assign numbers or rankings to nodes within a graph corresponding to their network position. Applications include identifying the most influential person(s) in a social network, key infrastructure nodes in the Internet or urban networks, super-spreaders of disease, and brain networks. Centrality concepts were first developed in social network analysis, and many of the terms used to measure centrality reflect their sociological origin. Centrality indices are answers to the question "What characterizes an important vertex?" The answer is given in terms of a real-valued function on the vertices of a graph, where the values produced are expected to provide a ranking which identifies the most important nodes."

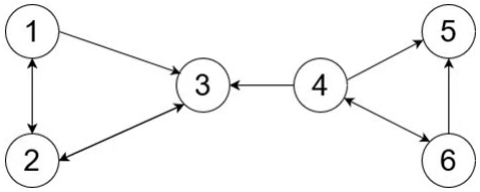


Figure 7.34: an Example

Closeness Centrality

For undirected graphs, the closeness centrality is calculated as:

$$C_C(i) = \frac{n-1}{\sum_{j=1}^{n} d(i,j)}$$

where n is the total number of vertices and $d(i,j)$ is the smallest number of edges i can travel to j. In the example,

$$C_C(1) = \frac{5}{1+1+2+3+3} = \frac{5}{10} = 0.5$$

For directed graphs, the formula is slightly modified to those directions apply. In the example,

$$C_C^*(1) = \frac{5}{1+1+N+N+N} = \frac{5}{3N+2}$$

where N is a very large number. This is because vertex 3 does not go to vertex 4.

Degree Centrality

For undirected graphs, the degree centrality is calculated as:

$$C_D(i) = \frac{d(i)}{n-1}$$

where $d(i)$ is the total number of edges directly connected to vertex i. Again, For directed graphs, the formula is slightly modified to those directions apply. In the example, the degree centrality of vertex 1 only considers vertices 2 and 3 (i.e. directly connected) and other vertices (4, 5, and 6) are not considered.

Betweenness Centrality

This sub-section is taken from https://www.sci.unich.it/ francesc/teaching/network/betweeness.html#:20centrality 20measures 20the 20extent,over 20information 20passing 20between 20others.

Betweenness centrality measures the extent to which a vertex lies on paths between other vertices. Vertices with high betweenness may have considerable influence within a network by virtue of their control over information passing between others. They are also the ones whose removal from the network will most disrupt communications between other vertices because they lie on the largest number of paths taken by messages.

Mathematically, let $n_{s,t}^i$ be the number of geodesic paths from s to t that pass through i and let $n_{s,t}$ be the total number of geodesic paths from s to t. Recall that a geodesic path is not necessarily unique and the geodesic paths between a pair of vertices need not be node-independent, meaning they may pass through some of the same vertices. Then the betweenness centrality of vertex i is:

$$b_i = \sum_{s,t} w_{s,t}^i = \sum_{s,t} \frac{n_{s,t}^i}{n_{s,t}}$$

where by convention the ratio $w_{s,t}^i = 0$ if $n_{s,t} = 0$. Notice that each pair of vertex s, t contribute to the sum for i with a weight $w_{s,t}^i$ between 0 and 1 expressing the betweenness of i with respect to the pair s, t. Observe that:

1. the given definition counts separately the geodesic paths in either direction between each vertex pair. Since these paths are the same on an undirected graph this effectively counts each path twice;

2. the definition includes paths starting or ending with i (s can be equal to i or t can be equal to i), as well as paths from a vertex to itself (s can be equal to t). It seems reasonable to define a vertex to be on a path between itself and someone else, or between some vertex and itself, since normally a vertex has control over information flowing from or to itself.

It makes little difference in practice to consider the alternative definitions, since one is usually concerned only with the relative magnitudes of the centralities and not with their absolute values. The sum can be normalized by dividing by the total number of ordered pairs of nodes, which is n^2, so that betweenness lies strictly between 0 and 1.

Consider the following graph (Figure 7.35):

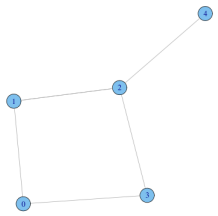


Figure 7.35: between centrality example 1

From node 0 to node 4 there are two geodesics, both going through node 2. Hence $w^2_{0,4} = 2/2 = 1$, $w^1_{0,4} = w^3_{0,4} = 1/2$.

Consider once again the following simple network Figure 7.36:

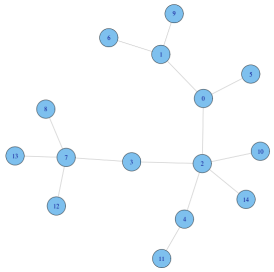


Figure 7.36: between centrality example 2

Function betweenness (R, C) computes betweenness centrality (the function counts undirected paths in only one direction and computes the sum in the be-

tweenness formula for $s \neq t$, $s \neq i$ and $t \neq i$). This is the same network with nodes labelled with their betweenness centrality Figure 7.37:

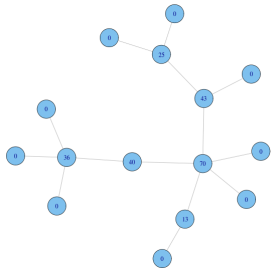


Figure 7.37: between centrality example 3

Notice that our sample graph is in fact a tree, that is a connected acyclic undirected graph, hence the number $n_{s,t}$ of geodesics from s to t is always 1, and the number $n^i_{s,t}$ of geodesics from s to t that pass through i is either 0 or 1. Hence the computed betweenness score for a vertex is the actual number of distinct paths that strictly contain the vertex in between.

Betweenness centrality differs from the other centrality measures. A vertex can have quite low degree, be connected to others that have low degree, even be a long way from others on average, and still have high betweenness. Consider a vertex A that lies on a bridge between two groups of vertices within a network. Since any path between nodes in different groups must go through this bridge, node A acquires high betweenness even though it is not well connected (it lies at the periphery of both groups) and hence it might not have particularly high values for degree, eigenvector, and closeness centrality. Vertices in roles like this are sometimes referred to as brokers.

The maximum possible value for betweenness occurs for the central node of a star graph, a network composed of a vertex attached to $n-1$ other vertices, whose only connection is with the central node. All paths, except the $n-1$ paths from the peripheral vertices to themselves, go through the central vertex, hence its betweenness is $n^2-(n-1) = n^2-n+1$. At the other end of the scale, the smallest possible value for betweenness occurs for a leaf node in a graph with a single component, that is a node that is connected to the rest of the network with only one edge. The leaf vertex lies on every path that starts or ends with itself. These are $2n-1$ in total: $n-1$ paths from a vertex to others, $n-1$ from others to the vertex, and 1 from the vertex to itself.

Betweenness centrality values are typically distributed over a wide range. Taking again the example of the film actor network, the individual with highest betweenness in the largest component of the network is Fernando Rey (The French

Connection). It is no coincidence that he appeared in both European and American films, played roles in several languages, and worked in both film and television, hence he is the archetypal broker. Rey has a betweenness score of $7.47 \cdot 10^8$, while the lowest score of any actor in the large component is $8.91 \cdot 10^5$. Thus there is a ratio of almost a thousand between the two limits, much larger than the ratio of 3.6 we saw in the case of closeness. One consequence is that there are clear winners and losers in the betweenness centrality competition. The second highest betweenness score is attributed to Christopher Lee again, with $6.46 \cdot 10^8$, a 14% difference from the winner.

Betweenness centrality, as defined above, is a measure of information control assuming two important hypothesis: (i) every pair of vertices exchange information with equal probability, and (ii) information flows along the geodesic (shortest) path between two vertices, or one of such path, chosen at random, if there are several. However, information not always takes the shortest route. In social network, for instance, a news about a friend of us might not come directly from the friend but from another mutual friend.

Calculating closeness and betweenness centrality for all nodes in a graph involves computing the (unweighted) shortest paths between all pairs of nodes in the graph. A breath-first visit from a source node can be used to compute all shortest paths from the source in time $O(n+m)$, where n and m are the number of nodes and edges of the graph. If this visit is repeated for all the source nodes of the graph the cost amounts to $O(n \cdot (n+m))$. Typically, real networks are sparse graphs, meaning the number m of edges is of the order of the number n of nodes. For instance, it is very rare in a social network that a significant number of actors have contacts with all the other actors of the network. Hence the overall cost in practical cases is quadratic.

Eigen Centrality

In graph theory, eigenvector centrality (also called eigencentrality or prestige score) is a measure of the influence of a node in a connected network. Relative scores are assigned to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. A high eigenvector score means that a node is connected to many nodes who themselves have high scores.

Google's PageRank and the Katz centrality are variants of the eigenvector centrality.

Page Rank

PageRank (PR) is a Google algorithm that ranks web pages in search results by evaluating the number and quality of links to a page. It operates on the principle that pages receiving more high-quality links are deemed more important and are thus ranked higher.

PageRank was created by Google co-founders Sergey Brin and Larry Page in 1997 when they were at Stanford University, and the name is a reference to both Larry Page and the term "webpage."

In many ways, it's similar to a metric called "impact factor" for journals, where more cited = more important. It differs a bit in that PageRank considers some votes more important than others.

By using links along with content to rank pages, Google's results were better than competitors. Links became the currency of the web.

7.14 Exercise

Exercise

Using Figure 7.38, answer the following questions: (https://web.mit.edu/jmn/www/6.034/d-separation.pdf)

1. Are A and B conditionally independent, given D and F? (Same as "P(A|BDF) =? P(A|DF)" or "P(B|ADF) =? P(B|DF)")

2. Are A and B marginally independent? (Same as "P(A|B) =? P(A)" or "P(B|A) =? P(B)")

3. Are A and B conditionally independent, given C?

4. Are D and E conditionally independent, given C?

5. Are D and E marginally independent?

6. Are D and E conditionally independent, given A and B?

7. P(D|BCE) =? P(D|C)

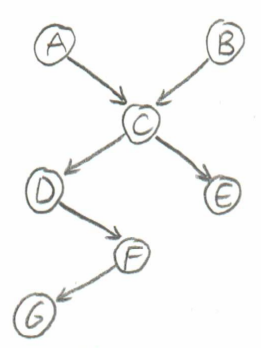


Figure 7.38: d-separation example

Project

1. Use stocks to estimate a CPDAG.

2. Use indices or sectors to estimate a CPDAG (different sectors could have a clear lead-lag relationship).

3. Use interest rates to estimate a CPDAG. (see my work)

Chapter 8

Neural Networks

Turing Test Passed, Now What?

Stefan Carter, Ho Chi Minh City

8.1 Introduction

Neural Networks (NN), or sometimes called Artificial Neural Networks (ANN, the name I don't like because what is not artificial in AI?! Not to mention that a prefix now is used to represent a specific NN), are one of the four artificial intelligence (AI) branches as they imitate networks of neurons.[1] See Figure 8.1.

Yet, just like any other AI or ML (machine learning) tool, NN have been "packaged" as tools to solve various classification problems. For example, we can train a Neural Network to recognize digits ($0 \sim 9$) or cat/dog.[2] In finance, various NN tools have been used to provide credit classifications (i.e. ratings) and predict default. Over the years, as computation powers have increased, very complex NN have been developed and they can be "trained" (i.e. can "learn") to perform various complex tasks and so earn the name "deep learning".

[1]The other two artificial intelligences are swarm intelligence and genetic algorithm. The former imitates low-intelligent animals such as bees, birds, ants, or fish and the latter imitates genes. To be short, an artificial intelligence is a computerized algorithm of a natural intelligence which is based upon biology.

[2]The other two artificial intelligences are swarm intelligence and genetic algorithm. The former imitates low-intelligent animals such as bees, birds, ants, or fish and the latter imitates genes. To be short, an artificial intelligence is a computerized algorithm of a natural intelligence which is based upon biology.

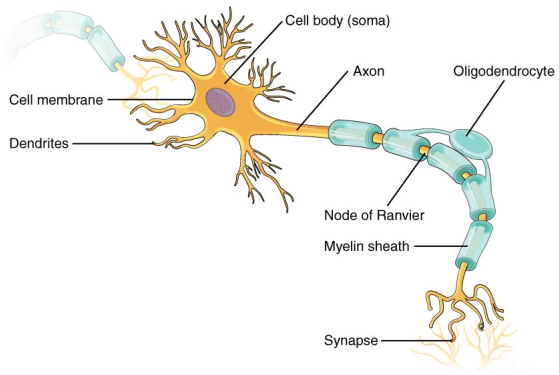


Figure 8.1: Neuron

source: https://www.healthline.com/health/neurons

Let's begin with a simple linear discriminant function (see Chapter 9 for detailed discussions of discriminant analysis). In Figure 8.2, we try to classify blue dots from red dots and we fit a linear line through the plane.

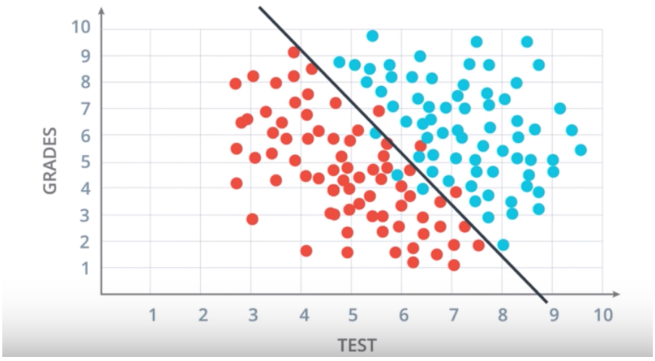


Figure 8.2: Linear Discriminant Function

source: Andy Li's slides

The linear equation clearly is $x_2 = b_0 + b_1 x_1$ where x_2 is grade and x_1 is test. In the discriminant analysis, however, it is more convenient (and equivalent) to express it as:

$$w_1 x_1 + w_2 x_2 + b = 0$$

where w_1 and w_2 are weights. We fit (train) all data through this equation and solve for w_1, w_2 and b. Then, we can predict a new observation by plugging its

coordinates (w_1, w_2) into the above equation and predict:

$$y = \begin{cases} 1 & w_1x_1 + w_2x_2 + b \geqslant 0 \\ 0 & w_1x_1 + w_2x_2 + b < 0 \end{cases}$$

This is shown graphically in Figure 8.3 where w_1, w_2 and b are inputs to a linear discriminant function. Then, the result is passed to a step function to obtain 0 or 1. As we can see, the discriminant analysis is amazingly analogous to an NN where the activation function is a step function.

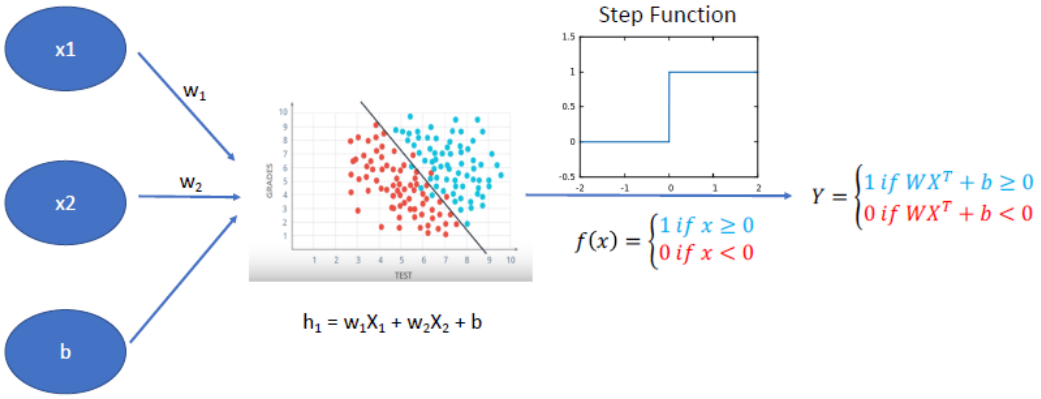


Figure 8.3: The Process of Linear Discriminant Analysis

source: Andy Li's slides

Combine Figure 8.3 with Figure 8.1, we can easily see that the linear discriminant analysis is a special case of NN, as shown side-by-side in Figure 8.4. Now, the linear discriminant analysis is expressed in a format of NN.

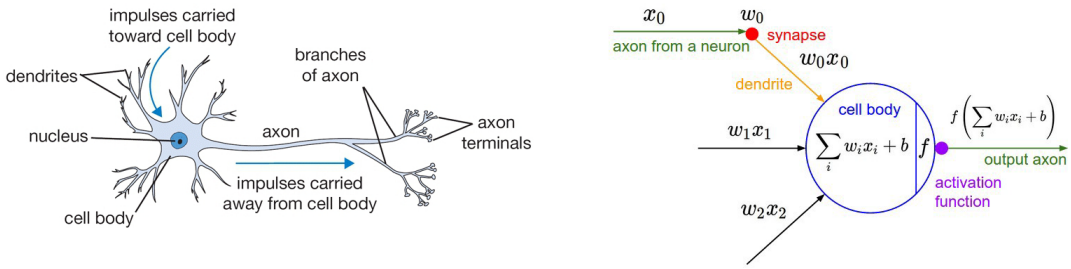


Figure 8.4: Linear Discriminant Function as NN

source: Andy Li's slides

8.2 Basics

The following diagram explains a one-layer, one-neuron, NN.

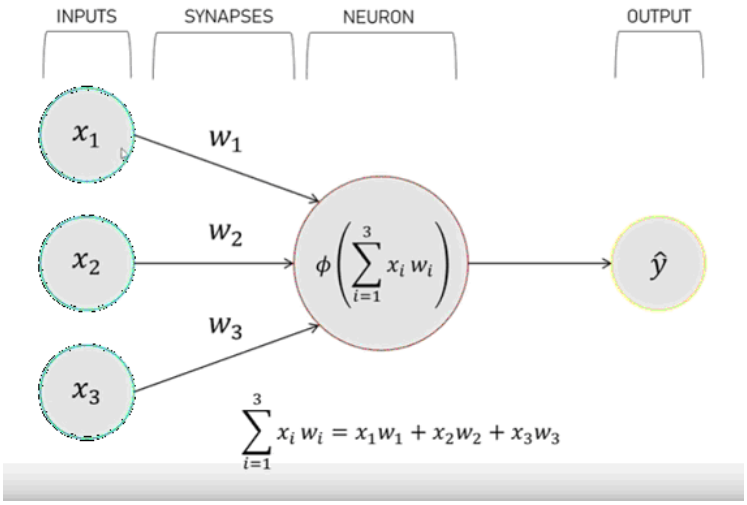


Figure 8.5: Simple NN

where synapses are weights. $\phi(\cdot)$ is known as the activation function. The common choices of the function in NN are (where $z = \sum w_i x_i$)

- $\phi(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$ (logit, sigmoid)
- $\phi(z) = \max\{z, 0\}$ (ReLu)
- $\phi(z) = \ln[1 + e^z]$ (softplus)
- $\phi(z) = \tanh[z]$ (tanh)
- $\phi(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$ (softmax)

The objective of this network is to find the best fit:

$$\min \left\{ \sum_{j=1}^{N} (y_j - \hat{y}_j)^2 \right\}$$

where j is the number of observations. In the easiest example, y can be default (i.e. 1) or no-default (i.e. 0). More complexly, y can be rating categories: AAA (9), AA (8), ..., D (0).

A two-layer NN can be constructed as:

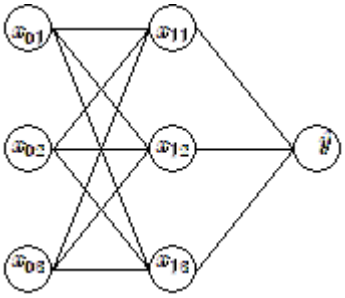


Figure 8.6: 2-layer NN

In this network, the value is stored as $x_{i,k(i)}$ where $i = 1, \cdots, m$ is the number of network (hidden) layers (note that $i = 0$ represents the "input layer" (not a network layer) that contains input variables); $k(i) = 1, \cdots, n(i)$ is the number of neurons of a given layer i; and finally the data contain $j = 1, \cdots, N$ observations.

Here, $x_{0,k(0)}$ are inputs and $k(0) = 1, \cdots, n(0)$ represents the number of input variables. In our example, $n(0) = 3$. There is one hidden layer, symbolized by $x_{1,k(1)}$ where $k(1) = 1, \cdots, n(1)$ and $n(1) = 3$ representing that there are three neurons in the layer. $\hat{y}$ is output. The number of synapses are 3×3 which is 9 (if a constant is given to every neuron, then there are 12) for the first layer, and 3×1 which 3 (if a constant is given then there is 4) for the second layer. There are two activation functions, one of each layer. The first layer has 3 such functions and the second layer has 1. These activation functions need not be same.

If we have no activation function, then

$$\begin{aligned} x_{11} &= w_{11}^{(1)} x_{01} + w_{12}^{(1)} x_{02} + w_{13}^{(1)} x_{03} + w_{14}^{(1)} \\ x_{12} &= w_{21}^{(1)} x_{01} + w_{22}^{(1)} x_{02} + w_{23}^{(1)} x_{03} + w_{24}^{(1)} \\ x_{13} &= w_{31}^{(1)} x_{01} + w_{32}^{(1)} x_{02} + w_{33}^{(1)} x_{03} + w_{34}^{(1)} \end{aligned}$$

or

$$\begin{aligned} x_{1i} &= \sum_{j=1}^{3} w_{ij}^{(1)} x_{0j} + w_{14}^{(1)} \\ \underline{x}_1 &= W_{3\times 4}^{(1)} \underline{x}_0 \end{aligned}$$

Define an activation function as arbitrary $\phi^{(i)}$. Then,

$$\begin{aligned} x_{11} &= \phi^{(1)} \left(w_{11}^{(1)} x_{01} + w_{12}^{(1)} x_{02} + w_{13}^{(1)} x_{03} \right) \\ x_{12} &= \phi^{(1)} \left(w_{21}^{(1)} x_{01} + w_{22}^{(1)} x_{02} + w_{23}^{(1)} x_{03} \right) \\ x_{13} &= \phi^{(1)} \left(w_{31}^{(1)} x_{01} + w_{32}^{(1)} x_{02} + w_{33}^{(1)} x_{03} \right) \end{aligned}$$

Then

$$x_{21} = \hat{y} = \phi^{(2)} (w_{11}^{(2)} x_{11} + w_{12}^{(2)} x_{12} + w_{13}^{(2)} x_{13})$$

where $\phi^{(i)}$ may or may not be same across i layers. For simplicity, we can set:

$$\phi^{(i)} = \phi(w'x) = \frac{1}{1+e^{-w'x}}$$

for all layers $i = 1, \cdots, m$. Another popular choice of the activation function is:

$$\phi(x) = \max\{x, a\}$$

If a is negative enough, then $\phi(x) = x$, which is what we want.

Then

$$\begin{aligned}\hat{y} &= \phi\left(\sum_{i=1}^{n} x_{1i} w_i^{(2)}\right) \\ &= \phi\left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{ij}^{(1)} x_{0j}\right) w_i^{(2)}\right) \\ &= \phi\left(\sum_{j=1}^{n} \left(\sum_{i=1}^{n} w_{ij}^{(1)} w_i^{(2)}\right) x_{0j}\right)\end{aligned}$$

With another activation function, then it is different!

$$\begin{aligned}\hat{y} &= \phi\left(\sum_{i=1}^{n} x_{1i} w_i^{(2)}\right) \\ &= \phi\left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{ij}^{(1)} x_{0j}\right) w_i^{(2)}\right) \\ &= \phi\left(\sum_{j=1}^{n} \left(\sum_{i=1}^{n} w_{ij}^{(1)} w_i^{(2)}\right) x_{0j}\right)\end{aligned}$$

Then

$$x_{21} = \hat{y} = \phi^{(2)}(w_{11}^{(2)} x_{11} + w_{12}^{(2)} x_{12} + w_{13}^{(2)} x_{13})$$

where $\phi^{(i)}$ may or may not be same across i layers.

$$\phi(x) = \begin{cases} \max\{x, a\} \\ \dfrac{1}{1+e^{-\beta x}} \end{cases}$$

If a is negative enough, then $\phi(x) = x$, which is what we want.

For multiple layers,

$$\begin{aligned}&\phi^{(1)}(W^{(1)}\underline{x}_0) = \underline{x}_1 \\ &\phi^{(1)}(W^{(2)}\underline{x}_0) = \underline{x}_2 \\ &\vdots \\ &\phi^{(1)}(W^{(n)}\underline{x}_0) = \underline{x}_n = \underline{\hat{y}}\end{aligned}$$

where the dimension of each weight matrix W_i is dependent on the dimensions of $\underline{x}_{i-1}$ and $\underline{x}_i$. The errors are defined as $\underline{e} = \underline{y} - \underline{\hat{y}}$. We minimize either the sum of the squared errors and the sum of absolute errors.

8.3 Backpropagation

This is nothing more than the chain rule in calculus.

8.3.1 Gradient Decent

This is simply the Newton's rule. Take the sigmoid function as an example. We first recognize that:

$$\begin{aligned}\frac{\partial}{\partial z}\phi(z) &= \frac{\partial}{\partial z}\frac{e^z}{1+e^z} \\ &= \frac{\partial}{\partial z}(1+e^{-z})^{-1} \\ &= (1+e^{-z})^{-2}e^{-z} \\ &= \phi(z)(1-\phi(z))\end{aligned}$$

Then, since $z = \sum w_i x_i$, we have:

$$\begin{aligned}\frac{\partial}{\partial w_i}\phi(z) &= \frac{\partial}{\partial z}\phi(z)\frac{\partial z}{\partial w_i} \\ &= \phi(z)(1-\phi(z))\frac{\partial}{\partial w_i}\sum_{i=1}^{n} w_i x_i \\ &= \phi(z)(1-\phi(z))x_i\end{aligned}$$

The error function is:

$$\min_{\underline{w}} \sum_{j=1}^{m} (y_j - \phi(z))^2$$

Hence,

$$\begin{aligned}&\frac{\partial}{\partial w_i}\sum_{j=1}^{m}(y_j-\phi(z))^2 = 0 \\ &2\sum_{j=1}^{m}(y_j-\phi(z))\frac{\partial}{\partial w_i}\phi(z) = 0 \\ &2\sum_{j=1}^{m}(y_j-\phi(z))\frac{\partial}{\partial w_i}\phi(z) = 0 \\ &\sum_{j=1}^{m}(y_j-\phi(z))\phi(z)(1-\phi(z))x_i = 0 \\ &\sum_{j=1}^{m}\left(y_j-\phi\left(\sum_{i=1}^{n}w_i x_i\right)\right)\phi\left(\sum_{i=1}^{n}w_i x_i\right)\left(1-\phi\left(\sum_{i=1}^{n}w_i x_i\right)\right)x_i = 0\end{aligned} \tag{8.1}$$

There are n such equations. To solve for all w_i's, we need to solve a system of linear equations. Clearly, this simultaneous equation system is not easy to solve. Hence, we must use a numerical solution – gradient decent.

Recall the Newton's rule. Suppose we have an arbitrary equation $y = f(x)$. The Taylor's series expansion is:

$$\begin{aligned} dy &= \frac{\partial y}{\partial x} dx + e \\ y_1 - y_0 &= \left.\frac{\partial y}{\partial x}\right|_{x=x_0} (x_1 - x_0) + e \\ y_1 &= y_0 + \left.\frac{\partial y}{\partial x}\right|_{x=x_0} (x_1 - x_0) + e \end{aligned}$$

where the update is:

$$x_i = x_{i-1} - \left.\frac{\partial y}{\partial x}\right|_{x=x_{i-1}}$$

From Figure 8.7, we solve a function: $0 = (x-6)^2 - 5$. In a couple of iterations, we find the solution to be 3.7639.

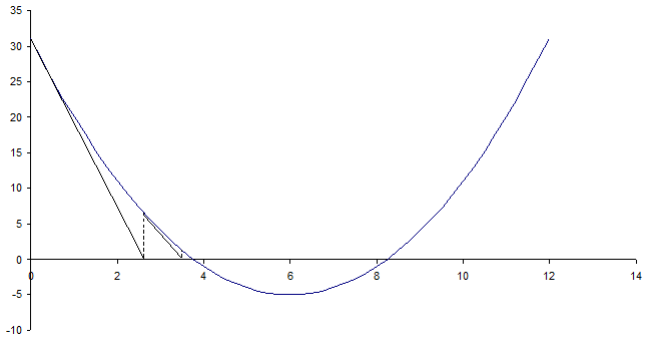


Figure 8.7: Newton's Rule

In a multi-variate case, the process is similar.

$$\begin{aligned} dy &= \textstyle\sum_{i=1}^{n} x_i \dfrac{\partial y}{\partial x_i} dx_i + e \\ y_1 - y_0 &= \textstyle\sum_{i=1}^{n} x_i \left.\dfrac{\partial y}{\partial x_i}\right|_{x_i=x_{i,0}} (x_i - x_{i,0}) + e \\ y_1 &= y_0 + \textstyle\sum_{i=1}^{n} x_i \left.\dfrac{\partial y}{\partial x}\right|_{x_i=x_{i,0}} (x_i - x_{i,0}) + e \end{aligned} \tag{8.2}$$

The update is:

$$x_{i,0} = x_{i-1,0} - \left.\frac{\partial y}{\partial x}\right|_{x=x_{i-1,0}} \tag{8.3}$$

Now we need to solve (8.1). We, following the Newton's rule, take partial derivative of (8.1) with respect to w_i and b. Then we update the estimates using (8.3). Clearly, the partial derivatives are in closed form and yet they are very long. As a result, one can just use numerical partials. Optimization libraries often allow numerical partials (when the function is fed). In the sigmoid example, the equation, although long, is simple and the numerical partials should be quite accurate.

Have an example as in Figure 8.8. The loss function is assumed to be:

$$\phi = (1 - y)^2$$

and

$$\begin{aligned} h_1 &= \phi(w_1 x_1 + w_2 x_2 + b_1) \\ h_2 &= \phi(w_3 x_1 + w_4 x_2 + b_2) \\ y = o_1 &= \phi(w_5 h_1 + w_6 h_2 + b_3) \end{aligned}$$

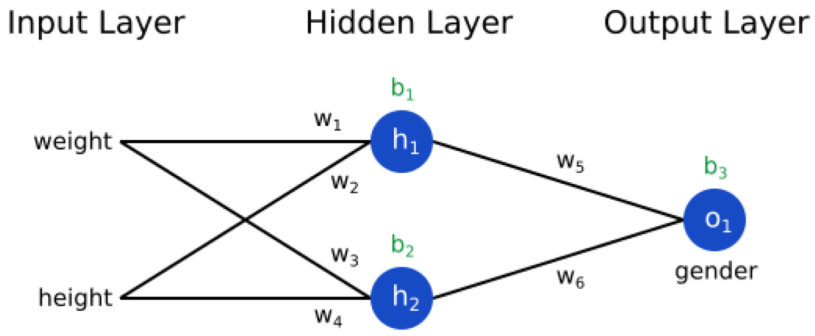


Figure 8.8: An Example

Let initial values be: $b_i = 0$, $w_i = 1$, $x_1 = -2$, $x_2 = -1$, and $y = 1$. Then

$$\begin{aligned} h_1 &= \phi(-2 - 1 + 0) = \phi(-3) = 0.0474 \\ h_2 &= \phi(-2 - 1 + 0) = \phi(-3) = 0.0474 \\ y &= \phi(0.0474 + 0.0474 + 0) = 0.524 \end{aligned}$$

Then

$$\begin{aligned} \frac{\partial \phi}{\partial w_1} &= \frac{\partial \phi}{\partial y} \frac{\partial y}{\partial h_1} \frac{\partial h_1}{\partial w_1} \\ &= -2(1 - y) \times w_5 \phi'(w_5 h_1 + w_6 h_2 + b_3) \times x_1 \phi'(w_1 x_1 + w_2 x_2 + b_1) \\ &= 0.0214 \end{aligned}$$

8.3.2 Vanishing Gradient

As more layers are added, gradient approaches 0 exponentially. See Figure **??**. This is because when n hidden layers use an activation like the sigmoid function, n small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.

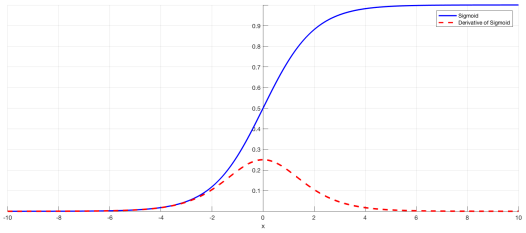


Figure 8.9: Sigmoid Function and Its Derivative

Solutions:

- Use other activation functions, such as f, which doesn't cause a small derivative.
- Residual networks
- Batch Normalization

8.3.3 An Example – Cleveland data

In this sub-section, we analyze a real dataset provided by StatQuest. The data contain patients with (and without) heart disease. There are 13 features in this data, such as `age`, `sex`, etc. In this sub-section, we only use two features as a demonstration. There is no hidden layer, as shown in Figure 8.5.

8.4 Logistic regression versus NN

The Logit model, while called logistic regression, is not a regression model but a maximum likelihood estimation. Under the Logit model, the dependent variable is binary and hence follows a binomial (or multinomial) distribution. As a result, the

likelihood function is (n data points):

$$\mathscr{L} = \prod_{j=1}^{N} p_j 1_{y_j=1}(1-p_j)1_{y_j=0}$$
$$\ln \mathscr{L} = \sum_{j=1}^{N} \ln[p_j 1_{y_j=1}] + \ln[(1-p_j)1_{y_j=0}]$$

Each data sample collected $< y_j; x_{1j}, x_{2j}, \cdots, x_{Kj} >$ with the dependent variable being $y_j = 1$ or $y_j = 0$. There are n data points.

The logistic function is:

$$p = \frac{\exp\left(\sum_{k=0}^{K} \beta_k x_k\right)}{1 + \exp\left(\sum_{k=0}^{K} \beta_k x_k\right)}$$

If we estimate the parameters by maximizing (or minimizing since is negative) then it is MLE. Or alternatively, we can minimize SSE (sum of squared errors):

$$\min\left\{\sum_{j=1}^{N} (y_j - \hat{y}_j)^2\right\}$$

which will be then consistent with NN with no hidden layers. The regression beta coefficients are weights in NN.

THIS HAS BEEN VERIFIED.

I TEST THIS WITH A YOUTUBE VIDEO DATA

https://www.youtube.com/watch?v=C4N3_XJJ-jU&t=3s

processed.cleveland.data.xls

Logistic Regression in R, Clearly Explained!!!!

StatQuest with Josh Starmer

If the output is also multi-variate, then $y_i = \phi_i(x_1, x_2, \cdots, x_n) + e_i$. An example could be a multi-variate logit model where ϕ_i is simply an indicator function such as:$y_i = \chi_i$(say rating BBB) $\chi_{i-1} < \hat{y} < \chi_i$ don't know how to describe it.

Here is the question:

- Should we do multiple layers in financial problems?
- Should we include ConvLayers to help results?
- This is not to say that the methods are not transferable. In fact, part of the purpose of this paper is to explore how these tools can benefit from each

other's advantages and hence can better provide better solutions. In other words, we will delve into the NN theory (open up the black box) to be able to create a better tool (by combining the advantages of currently separate tools.

Neural Networks (NN) used in credit ratings has a long history. Yet it is used in a very narrow sense – as a classification tool. When used as a classification tool, such an NN has a very simple structure. It has multiple (hidden) layers. There isn't any restriction or guideline of how many layers, how many neurons (i.e. nodes) in each layer, and how two consecutive layers are connected (i.e. weights and activation functions).

Figure 8.10: Simple NN

https://xkcd.com/1838/?fbclid=IwAR04p87_80uam5k6UCytH6aRxpCI7kvJnwlbhNzBDgWFCrf

† ????????

https://en.wikipedia.org/wiki/Rectifier_(neural_networks)#Softplus

$$f(x) = \ln[1 + e^x]$$
$$f(x;k) = \frac{\ln[1 + e^{kx}]}{k}$$

The derivative of softplus is the logistic function. Starting from the parametric version,

$$f'(x) = \frac{e^{kx}}{1 + e^{kx}}$$

The logistic sigmoid function is a smooth approximation of the derivative of the rectifier, the Heaviside step function.

The multivariable generalization of single-variable softplus is the LogSumExp with the first argument set to zero:

$$\ln[1 + e^{x_1} + e^{x_2} + \cdots + e^{x_n}]$$

The LogSumExp function itself is

$$\ln[e^{x_1} + e^{x_2} + \cdots + e^{x_n}]$$

and its gradient is the softmax; the softmax with the first argument set to zero is the multivariable generalization of the logistic function. Both LogSumExp and softmax are used in machine learning.

8.5 Various NNs

8.5.1 Recurrent Neural Networks, RNN

A recurrent neural network (RNN) is a network architecture for deep learning that predicts on time-series or sequential data.

RNNs are particularly effective for working with sequential data that varies in length and solving problems such as natural signal classification, language processing, and video analysis.

§ Long short-term memory (LSTM)

LSTM networks are a specialized form of the RNN architecture. RNNs use past information to improve the performance of a neural network on current and future inputs. They contain a hidden state and loops, which allow the network to store past information in the hidden state and operate on sequences. RNNs have two sets of weights: one for the hidden state vector and one for the inputs. During training, the network learns weights for both the inputs and the hidden state. When implemented, the output is based on the current input, as well as the hidden state, which is based on previous inputs.

LSTM is to avoid the exploding/vanishing gradient problem, such as

cell state for long term

hidden state for short term rolling

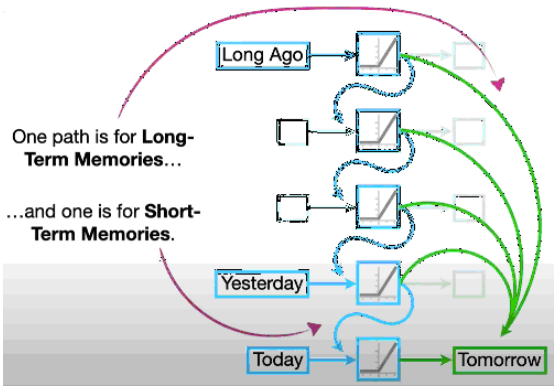


Figure 8.11: LSTM

8.5.2 Convolutional Neural Networks, CNN

(wiki) A convolutional neural network (CNN) is a regularized type of feedforward neural network that learns features by itself via filter (or kernel) optimization. This type of deep learning network has been applied to process and make predictions from many different types of data including text, images and audio. An example is given in Figure 8.12 where a handwritten 7 needs to be recognized.

Figure 8.12: Handwritten 7

We then put this image onto a pixel chart (Excel in this case) as in Figure 8.13. Each pixel has a value that reflects the darkness of the pixel. The ones that contain the number 7 will be labeled 1 and the ones that contain nothing will be labeled 0. Those pixels that are not so dark will be labeled as 0.2, 0.3, 0.5, accordingly.

Now we set up a convolutional layer as a 3×3 matrix.

0.979	0.278	0.940
0.713	0.048	0.564
0.604	0.327	0.853

Table 8.1: A Convolutional Layer

Figure 8.13: Handwritten 7

This convolutional layer is then applied on a moving basis as follows in Figure 8.14. Then, we move pixel by pixel over as in Figure 8.15.

In Figure 8.17, the left highlights the top edge, the middle highlights the left edge, and the right highlights the right edge

See https://dev.to/shri50/n-d-arrays-understanding-with-real-life-examples-5b5l

8.5.3 Recursive Neural Networks, RvNN

(wiki) A recursive neural network is a kind of deep neural network created by applying the same set of weights recursively over a structured input, to produce a structured prediction over variable-size input structures, or a scalar prediction on it, by traversing a given structure in topological order. These networks were first introduced to learn distributed representations of structure (such as logical terms), but have been successful in multiple applications, for instance in learning sequence and tree structures in natural language processing (mainly continuous representations of phrases and sentences based on word embeddings).

(https://www.geeksforgeeks.org/recursive-neural-network-in-deep-learning/)

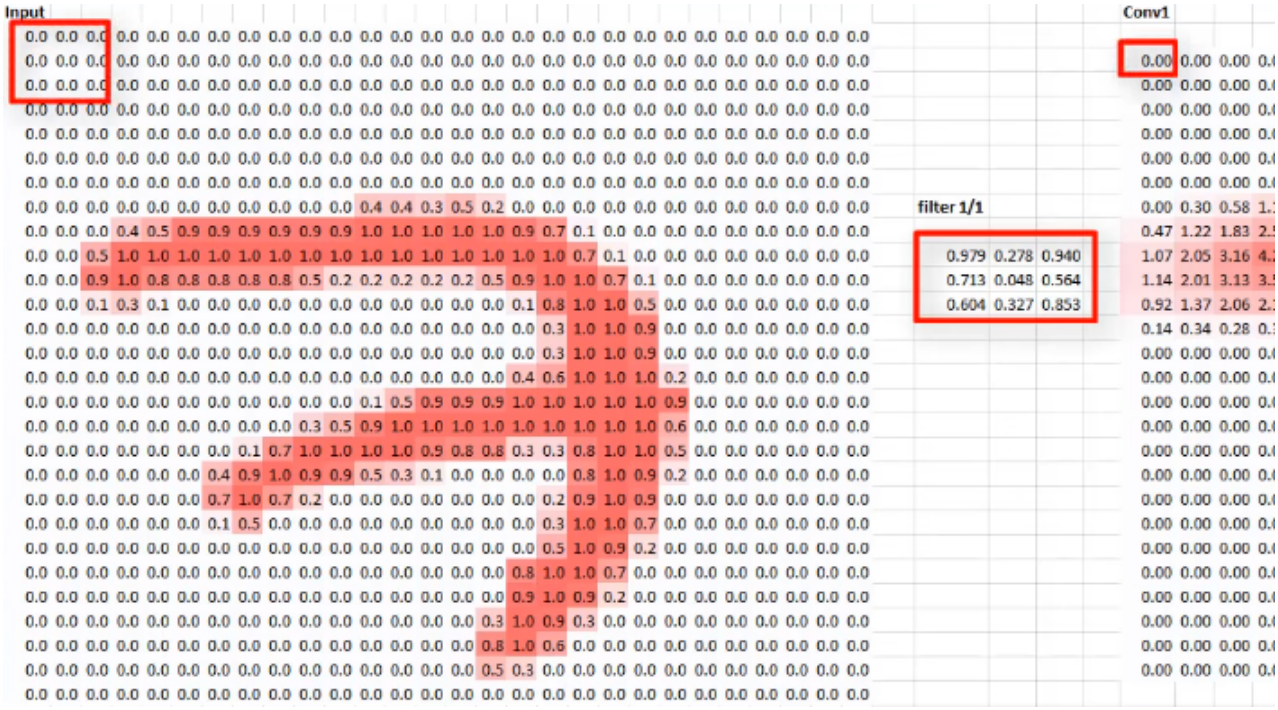


Figure 8.14: Handwritten 7

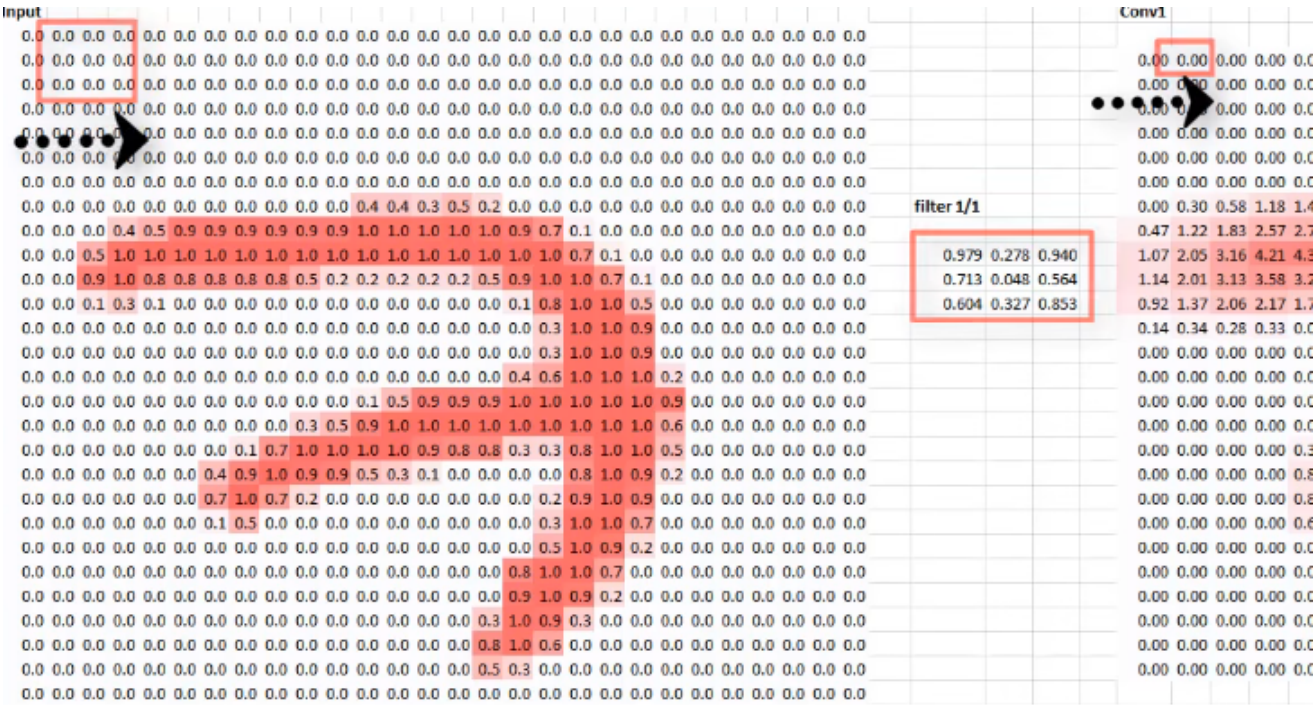


Figure 8.15: Handwritten 7

Conv1

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.32	0.45	0.61	0.72	0.54	0.36	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.30	0.58	1.18	1.41	1.64	1.64	1.64	1.64	1.91	1.95	2.21	2.31	2.10	2.04	1.69	0.87	0.48	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.47	1.22	1.83	2.57	2.73	3.00	3.00	3.00	3.00	3.39	3.50	3.83	3.97	3.71	3.57	3.16	2.30	1.46	0.54	0.05	0.00	0.00	0.00	0.00	0.00	0.00
1.07	2.05	3.16	4.21	4.39	4.74	4.74	4.54	4.20	4.01	3.85	3.90	3.92	4.10	4.48	4.65	4.06	3.11	1.53	0.54	0.05	0.00	0.00	0.00	0.00	0.00
1.14	2.01	3.13	3.58	3.28	3.22	3.22	3.09	2.91	2.72	2.51	2.51	2.51	2.65	2.99	3.91	4.38	4.12	2.85	1.37	0.33	0.00	0.00	0.00	0.00	0.00
0.92	1.37	2.06	2.17	1.79	1.71	1.71	1.49	1.14	0.82	0.53	0.53	0.53	0.77	1.27	2.43	3.72	4.53	3.98	2.38	0.96	0.00	0.00	0.00	0.00	0.00
0.14	0.34	0.28	0.33	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	1.29	2.83	4.25	4.66	2.76	1.65	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.35	1.16	2.91	3.93	5.13	3.07	2.19	0.12	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.42	0.97	1.33	1.59	1.91	2.39	3.69	4.36	5.19	3.83	2.50	0.71	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.49	1.10	1.68	2.29	2.65	2.96	3.41	3.72	4.61	4.91	5.24	4.12	2.53	1.20	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.12	0.67	1.32	1.89	2.57	3.17	3.96	4.43	4.77	4.38	4.22	4.49	4.79	5.19	4.53	2.72	1.60	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.38	0.96	1.82	2.60	3.32	3.77	3.94	3.80	3.65	3.38	2.97	2.93	3.56	4.09	5.00	3.84	2.48	0.97	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.82	1.71	2.96	3.51	3.68	3.30	3.04	2.48	2.23	1.93	1.29	1.28	2.43	3.29	4.83	3.43	2.34	0.56	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.85	1.96	2.72	3.18	2.58	1.77	1.32	0.65	0.38	0.12	0.00	0.39	2.22	3.27	4.63	2.96	2.02	0.16	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.66	1.40	1.65	1.74	0.74	0.24	0.00	0.00	0.00	0.00	0.00	0.83	2.51	3.67	4.25	2.61	1.50	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.07	0.48	0.20	0.48	0.00	0.00	0.00	0.00	0.00	0.00	0.70	1.72	3.28	3.98	3.56	2.23	0.80	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	2.27	3.98	3.90	2.82	1.58	0.22	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	2.26	3.10	4.46	3.42	2.04	0.80	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.89	2.56	3.36	3.81	2.51	1.22	0.22	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	1.18	2.01	2.84	2.42	1.46	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	1.02	1.34	1.98	1.31	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 8.16: Convolved

Recursive Neural Networks are a type of neural network architecture that is specially designed to process hierarchical structures and capture dependencies within recursively structured data. Unlike traditional feedforward neural networks (RNNs), Recursive Neural Networks or RvNN can efficiently handle tree-structured inputs which makes them suitable for tasks involving nested and hierarchical relationships. In this article, we will discuss RvNN, its work principles, and some frequently asked questions.

8.5.4 Grahpic Neural Networks, GNN

(https://www.youtube.com/watch?v=GXhBEj1ZtE8&t=218s) What if someone told you that a machine has discovered an antibiotic that can treat previously untreatable strains of bacteria? Yes, you heard that right. The antibiotic is a chemical named Halicin, and it was discovered by a pioneering machine-learning approach called graph neural networks.

Graph neural networks (GNN) are specialized artificial neural networks that are designed for tasks whose inputs are graphs.

Some use cases of graphs:
(https://www.youtube.com/watch?v=xFMhLp52qKI)

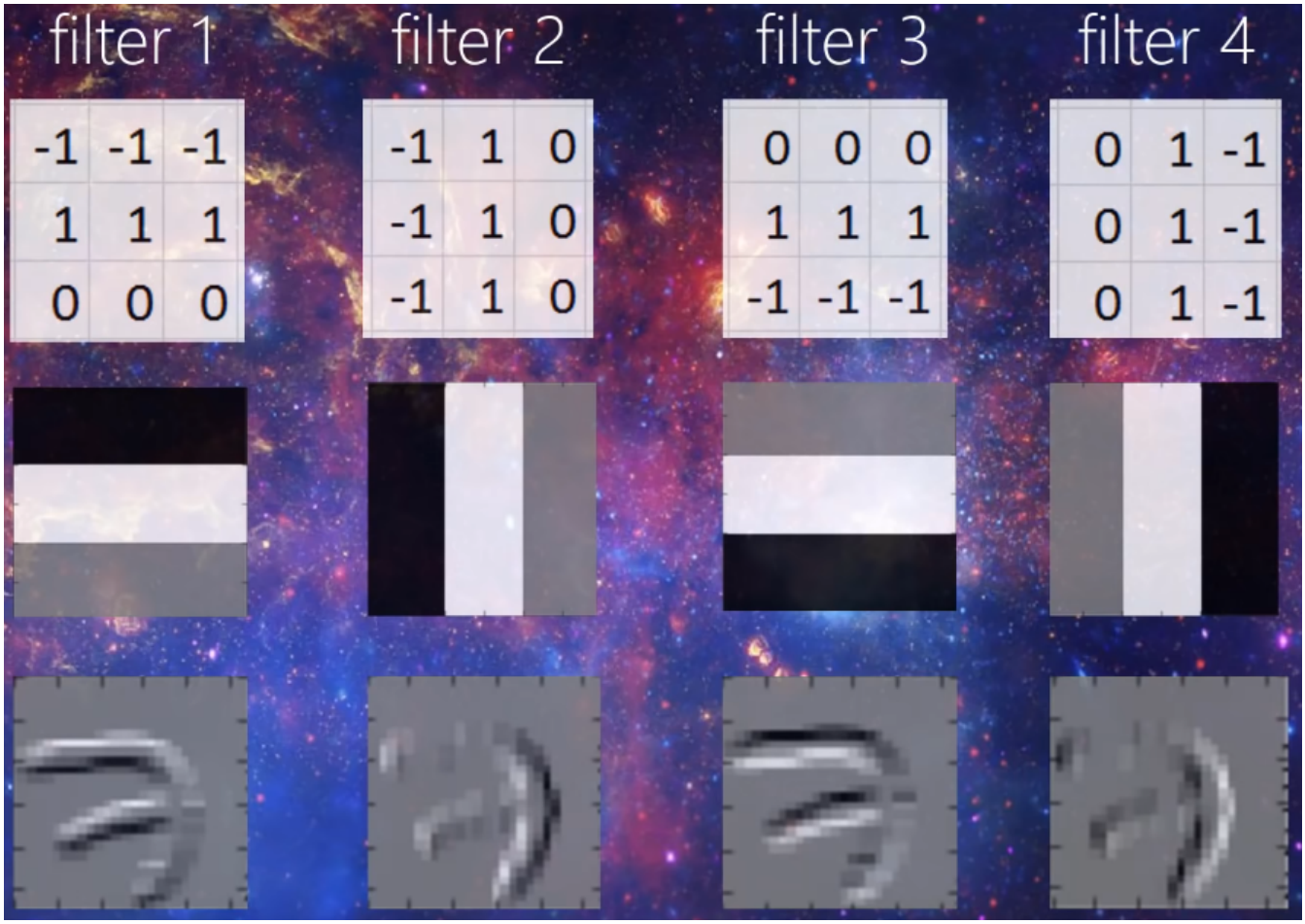


Figure 8.17: Filters

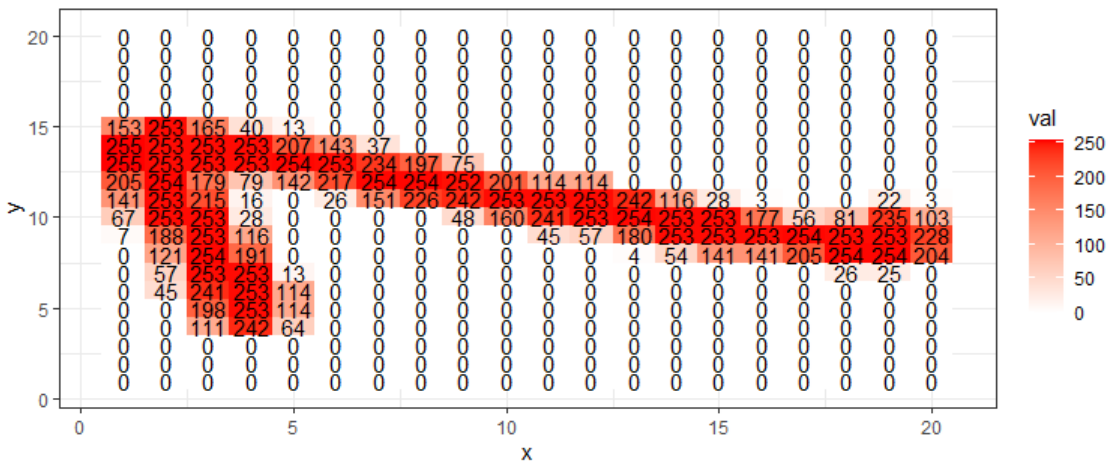


Figure 8.18: Handwritten 7

source: https://dev.to/shri50/n-d-arrays-understanding-with-real-life-examples-5b5l

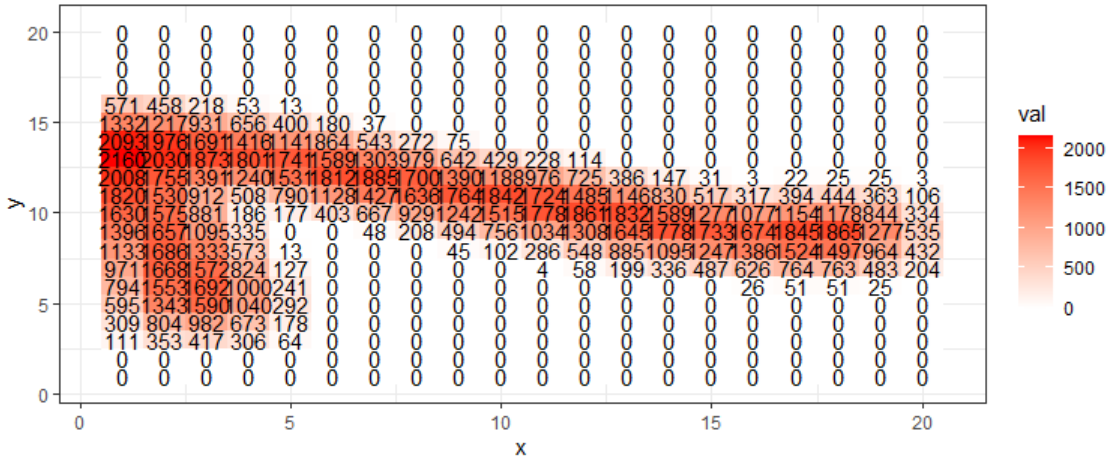


Figure 8.19: Handwritten 7

- node classification
- node clustering
- graph classification
- link prediction (in recommendation systems)
- influence maximization (node(s) that is most influential)

8.5.5 Kolmogorov-Arnold Network, KAN

Instead of training weights, KAN trains activation functions. By using the Kolmogorov-Arnold theorem, we can approximate amazingly closely any continuous function and hence achieve the classification we desire with no error (near 100%)!

§ Kolmogorov-Arnold Theorem

(wiki) In real analysis and approximation theory, the Kolmogorov–Arnold representation theorem (or superposition theorem) states that every multivariate continuous function can be represented as a superposition of continuous single-variable functions.

§ Relation to Fourier Series

https://www.youtube.com/watch?v=r6sGWTCMz2k&t=625s
https://www.youtube.com/watch?v=ds0cmAV-Yek
For those of you who are familiar with Fourier series, the Kolmogorov-Arnold theorem is similar to the concept of Fourier series that a continuous function can be approximated by a collection of Fourier series.

§ KAN

Here is a simple example of KAN.

8.6 Transformer

Transformer knows "bank of a river" and "withdraw money from a bank" by positional encoding and multi-head attention.

https://www.youtube.com/watch?v=l4is4uHvKlU

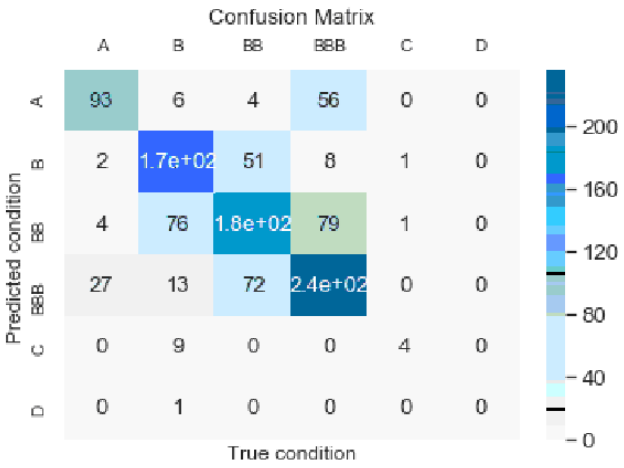


Figure 8.20: Confusion Matrix

8.7 Other Issues

8.7.1 Overfitting

(mathworks) https://www.mathworks.com/help/deeplearning/ug/improve-neural-network-generalization-and-avoid-overfitting.html

"One of the problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations."

8.7.2 Early Stopping

The default method for improving generalization is called early stopping. This technique is automatically provided for all of the supervised network creation functions, including the backpropagation network creation functions such as feedforwardnet.

In this technique the available data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error.

However, when the network begins to overfit the data, the error on the validation set typically begins to rise. When the validation error increases for a specified number of iterations (net.trainParam.max_fail), the training is stopped, and the weights and biases at the minimum of the validation error are returned.

The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error in the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set.

8.7.3 Regularization

"Another method for improving generalization is called regularization. This involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set. The next section explains how the performance function can be modified, and the following section describes a routine that automatically sets the optimal performance function to achieve the best generalization."

§ Modified Performance Function

The typical performance function used for training feedforward neural networks is the mean sum of squares of the network errors.

It is possible to improve generalization if you modify the performance function by adding a term that consists of the mean of the sum of squares of the network weights and biases msereg $= \gamma$msw $+ (1 - \gamma)$mse, where γ is the performance ratio, and

$$\text{msw} = \frac{1}{n} \sum_{j=1}^{n} w_j^2$$

Using this performance function causes the network to have smaller weights and biases, and this forces the network response to be smoother and less likely to overfit.

§ Summary and Discussion of Early Stopping and Regularization

Early stopping and regularization can ensure network generalization when you apply them properly.

For early stopping, you must be careful not to use an algorithm that converges too rapidly. If you are using a fast algorithm (like trainlm), set the training param-

eters so that the convergence is relatively slow. For example, set mu to a relatively large value, such as 1, and set mu_dec and mu_inc to values close to 1, such as 0.8 and 1.5, respectively. The training functions trainscg and trainbr usually work well with early stopping.

With early stopping, the choice of the validation set is also important. The validation set should be representative of all points in the training set.

When you use Bayesian regularization, it is important to train the network until it reaches convergence. The sum-squared error, the sum-squared weights, and the effective number of parameters should reach constant values when the network has converged.

With both early stopping and regularization, it is a good idea to train the network starting from several different initial conditions. It is possible for either method to fail in certain circumstances. By testing several different initial conditions, you can verify robust network performance.

When the data set is small and you are training function approximation networks, Bayesian regularization provides better generalization performance than early stopping. This is because Bayesian regularization does not require that a validation data set be separate from the training data set; it uses all the data.

https://datascience.stackexchange.com/questions/43500/how-does-a-bayes-regularization-works

$$F = \gamma \sum_{j=1}^{M} w_j^2 + (1-\gamma) \sum_{i=1}^{N} e_i^2$$

is equivalent to maximising likelihood P(w|D,y), which can be computed according to Bayes' theorem

$$p(w|D,\gamma) = \frac{p(D|w,\gamma)p(w|\gamma)}{p(D|\gamma)}$$

Likelihood P(w|y) is assumed to be a Gaussian one and can be computed as:

$$p(w|\gamma) = \left(\frac{\gamma}{2N}\right)^{1/2M} e^{-1/2w'w}$$

Likelihood $P(D|\gamma)$ can be computed too as:

$$p(D|\gamma) = \left(\frac{\pi}{\gamma}\right)^{-1/2N} \left(\frac{\pi}{1-\gamma}\right)^{-1/2M} \frac{(2\pi)^{1/2M}(-e^{F(w)})}{\sqrt{|H|}}$$

And now my questions are: how is $P(D|w,\gamma)$ computed? Or maybe is there any special assumptions about it? And should I minimalize or maximise $P(w|D,\gamma)$

to minimalize objective function F. I have found answer for my question here: http://hagan.okstate.edu/icnn97a.pdf If anyone needs it in future: all above likelihoods are assumed to be Gaussian distributions. Likelihood $P(\gamma)$ is assumed to be uniform. In the article it is shown how exactly are they defined and how to compute further is described in detail in article linked above.

8.8 Financial Applications

NN is perhaps one of the earliest AI models adopted in finance. This is quite understandable as classification has always been a necessary tool in financial analysis. From default, to credit ratings, to industry categorizations, classification is everywhere in finance. As introduced in Chapter 9, various parametric techniques, such as discriminant analysis, probit/logit, etc. have been widely used, till today. AI/ML certainly helps in providing non-parametric and non-linear tools in classification. NN is no exception.

The adoption of NN in default prediction can be traced back to early 90's.

Chapter 9

Classification

9.1 Introduction

Classification is ...

9.2 Naive Bayes

See Chapter 2.

Hull Chapter 4 (4.4)

9.3 Logistic and Probit Models

It is often common for the independent and/or the dependent variables to be binomial. As a result econometric methods for such situations are important. Independent binomials can be done with dummies but the dependent binomials cannot.

Hull Chapter 3 (3.9)

9.3.1 The Basic Idea

Recall a coin flip example and the probability of head is p. $y = 1$ represents head.

$$f(y) = \begin{cases} p & y = 1 \\ 1-p & y = 0 \end{cases}$$
$$= p^y(1-p)^{1-y}$$

The log of maximum likelihood function is:

$$\max \ln \mathscr{L} = \max \sum_{i=1}^{n} \ln \mathscr{L}(y_i|p) = \max \sum_{i=1}^{n} \ln \left[p^{y_i}(1-p)^{1-y_i}\right]$$
$$= \max \sum_{i=1}^{n} y_i \ln p + (1-y_i) \ln[1-p]$$

Take the derivative with respective to p:

$$\frac{\partial \ln \mathscr{L}}{\partial p} = \sum_{i=1}^{n} \frac{y_i \ln p}{\partial p} + \frac{(1-y_i)\ln[1-p]}{\partial p}$$
$$= \frac{1}{p}\sum_{i=1}^{n} y_i - \frac{1}{1-p}\sum_{i=1}^{n}(1-y_i) = 0$$

Solve for p and get:

$$p = \frac{\sum_{i=1}^{n} y_i}{n}$$

We wish to model the probability p by specifying the model:

$$p = f\left(\alpha + \sum_k \beta_k x_k\right)$$

The two popular ways to specify the functional form are:

$$p = \begin{cases} \frac{\exp(\alpha+\sum \beta_k x_k)}{1+\exp(\alpha+\sum \beta_k x_k)} & \text{logit model} \\ \int_{-\infty}^{\alpha+\sum \beta_k x_k} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right) du & \text{probit model} \end{cases}$$

which are called “link functions". Lets explore more what this means.

9.3.2 Other Link Functions

Recall linear regression:

$$\begin{cases} p = \sum \beta_k x_k + \varepsilon & \text{regression equation} \\ \mathbf{P}_{T\times 1} = \mathbf{X}_{T\times K}\beta_{K\times 1} + \mathbf{e}_{T\times 1} & \text{data equation} \end{cases}$$

where T is the length of data series (T observations) and k is the number of independent variables. A link function is a function to transform y from its raw data to create a nonlinear regression. For example, the link function can be $y^* = \ln[y]$:

$$\ln[p] = \sum \beta_k x_k + \varepsilon$$

The meaning of this is that: $\Delta \ln[p] = \sum \beta_k \Delta x_k + \varepsilon$ and hence β is a return sensitivity measure (as opposed to a $ sensitivity measure in the linear case.)

9.3.3 Probit Function

From the Probit model, it is clear that $p = N(\alpha + \sum \beta_k x_k) = N(z)$. Hence, $z = N^{-1}(p)$.

The idea is that y is binary. As we can see that a linear fit will not work. So we need to do two things. First is to let y be continuous between 0 and 1, that is $p \in (0, 1)$. Then, we transform p to $z = N^{-1}(p) \in (-\infty, \infty)$ which is our link function. An example taken from O'Halloran is in the following.

$$z = N^{-1}(p) = \sum \beta_k x_k + \varepsilon$$

This is the Probit Link function (short for Prob(ability Un)it). According to O'Halloran, this term was coined in the 1930's by biologists studying the dosage-cure rate link. It is straightforward to back out the value for y:

$$\sum \beta_k x_k \to z \to p = N(z)$$

The S curve in the fit (see graph) is the cumulative normal curve.

9.3.4 Logit Function

Let $z = \alpha + \sum \beta_k x_k$. Then the logit model is:

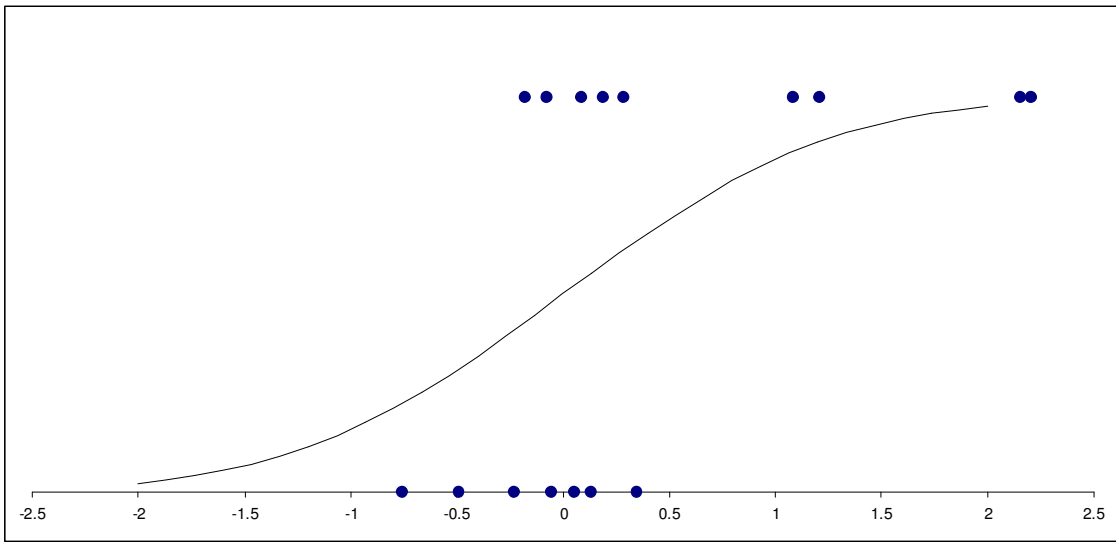


Figure 9.1: Binary Dependent Variable

$$p = \frac{\exp(\alpha + \sum \beta_k x_k)}{1 + \exp(\alpha + \sum \beta_k x_k)} = \frac{e^z}{1 + e^z}$$
$$e^z = O(z) = \frac{p}{1-p}$$
$$z = \ln\left[\frac{p}{1-p}\right]$$

where $\frac{p}{1-p}$ is known as the "odds ratio" (as opposed to $z = N^{-1}(p)$.)

Clearly that $p = 0$ corresponds to $O(z) = 0$ and $p = 1$ to $O(z) = \infty$. ($p = 1/4$ is $O(z) = 1/3$, $p = 1/2$ is $O(z) = 1/2$, and $p = 3/4$ to $O(z) = 3$.) To extend the odds ratio to $-\infty$, we simply take logs to obtain z.

This logit function and the normal probit function are very similar. See graph.

Put it back in regression:

$$z = \sum \beta_k x_k + \varepsilon$$
$$\hat{z} = \sum \beta_k x_k$$
$$\hat{p} = \frac{e^{\hat{z}}}{1 + e^{\hat{z}}}$$

which is a lot easier than looking up the normal table.

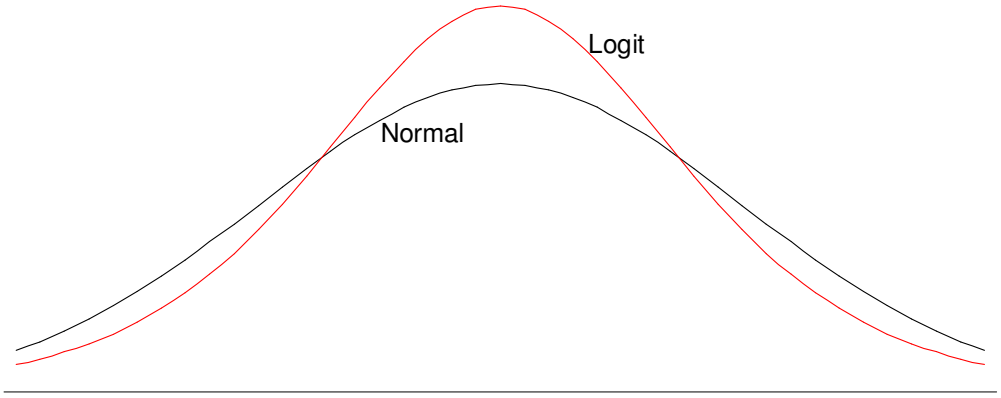


Figure 9.2: Logit vs. Probit

9.3.5 Maximum Likelihood Estimator

§ Logit

Let i be the i-th issuer. Then:

$$\ln\left(\frac{p_i}{1-p_i}\right) = \sum_{k=0}^{K} x_{ik}\beta_k$$
$$p_i = \frac{\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}{1+\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}$$

The log likelihood function is naturally:

$$\mathscr{L}(\theta) = f(\underline{y}|\underline{\beta}) = \prod_{i=1}^{N} \frac{n_i!}{y_i!(n_i-y_i)!} p_i^{y_i}(1-p_i)^{n_i-y_i}$$
$$= \prod_{i=1}^{N} \frac{n_i!}{y_i!(n_i-y_i)!} \prod_{i=1}^{N} \left(\frac{p_i}{1-p_i}\right)^{y_i} (1-p_i)^{n_i}$$
$$= \prod_{i=1}^{N} \frac{n_i!}{y_i!(n_i-y_i)!} \prod_{i=1}^{N} \left[\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)\right]^{y_i} \left(1-\frac{\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}{1+\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}\right)^{n_i}$$
$$= \prod_{i=1}^{N} \frac{n_i!}{y_i!(n_i-y_i)!} \prod_{i=1}^{N} \exp\left(y_i \sum_{k=0}^{K} x_{ik}\beta_k\right) \left(1+\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)\right)^{-n_i}$$

$$\ln \mathscr{L}(\theta) = \sum_{i=1}^{N} y_i \sum_{k=0}^{K} x_{ik}\beta_k - n_i \ln\left[1 + \exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)\right]$$

$$\begin{aligned} \frac{\partial \ln \mathscr{L}(\theta)}{\partial \beta_k} &= \sum_{i=1}^{N} y_i x_{ik} - n_i \frac{x_{ik}\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}{1 + \exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)} \\ &= \sum_{i=1}^{N} y_i x_{ik} - n_i p_i x_{ik} = 0 \end{aligned} \tag{9.1}$$

Note that the above implies that

$$\hat{p}_i = \frac{\sum_{i=1}^{N} y_i x_{ik}}{n_i x_{ik}}$$

which is logical. To solve for the parameter values (β) we need to solve for the system of $K+1$ equations and solving for each β_k.

Standard Error

Recall that:

$$\ln\left[\frac{p_i}{1-p_i}\right] = \sum_{k=0}^{K} x_{ik}\beta_k$$

for $i = 1, \cdots, n$ and that:

$$\begin{aligned} f(y|\beta) &= \prod_{i=1}^{n} \frac{n_i!}{y_i!(n_i - y_i)!} p_i^{y_i}(1-p_i)^{n_i - y_i} \\ &= \mathscr{L}(\beta) \end{aligned}$$

We continue from equation (9.1) and take one more derivative of the log likelihood function and get the Hessian matrix:

$$\begin{aligned} \frac{\partial^2 \ln \mathscr{L}(\beta)}{\partial \beta_k \partial \beta_{k'}} &= \frac{\partial}{\partial \beta_k} \sum_{i=1}^{N} y_i x_{ik} - n_i x_{ik} p_i \\ &= \frac{\partial}{\partial \beta_k} \sum_{i=1}^{N} - n_i x_{ik} p_i \\ &= -\sum_{i=1}^{N} n_i x_{ik} \frac{\partial}{\partial \beta_k}\left(\frac{\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}{1 + \exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}\right) \\ &= -\sum_{i=1}^{N} n_i x_{ik} p_i (1-p_i) x_{ik'} \end{aligned}$$

The last line is achieved by:

$$\frac{\partial}{\partial x}\frac{e^{u(x)}}{1+e^{u(x)}} = \frac{e^{u(x)}}{1+e^{u(x)}}\frac{1}{1+e^{u(x)}}\frac{\partial u(x)}{\partial x}$$

9.3.6 Probit

Take the following regression:

$$z = \sum \beta_k x_k + \varepsilon$$

and $\varepsilon \sim N(0, \sigma^2)$. If $z > 0$ then $\sum \beta_k x_k + \varepsilon > 0$. Or $\varepsilon > -\sum \beta_k x_k$. Then,

$$\begin{aligned}\Pr(z > 0|x) = \Pr(y = 1|x) &= \Pr(\varepsilon > -\sum \beta_k x_k) \\ &= \Pr\left(\frac{\varepsilon}{\sigma} > \frac{-\sum \beta_k x_k}{\sigma}\right) \\ &= N\left(\frac{\sum \beta_k x_k}{\sigma}\right)\end{aligned}$$

and similarly:

$$\begin{aligned}\Pr(y = 0|x) &= 1 - N\left(\frac{\sum \beta_k x_k}{\sigma}\right) \\ &= N\left(-\frac{\sum \beta_k x_k}{\sigma}\right)\end{aligned}$$

It is quite convenient to set $\sigma = 1$. Say for an observation y_i, we plug in independent variable values $x_1 \cdots x_k$ and coefficients $\beta_1 \cdots \beta_k$ in the standard normal probability function $\Pr(y_i = 1|x) = N(\sum \beta_i x_i) = 0.8$. We then say that the likelihood of $y_i = 1$ is 0.8 and the likelihood of $y_i = 0$ is 0.2. We can say the same thing for $\Pr(y_i = 0|x)$.

A Demonstration

We take one x as an example. The following graph illustrates the MLE.

The 0's and 1's are actual dependent variable (Y) and the solid line is the estimates of Y. Note that the solid line is computed as:

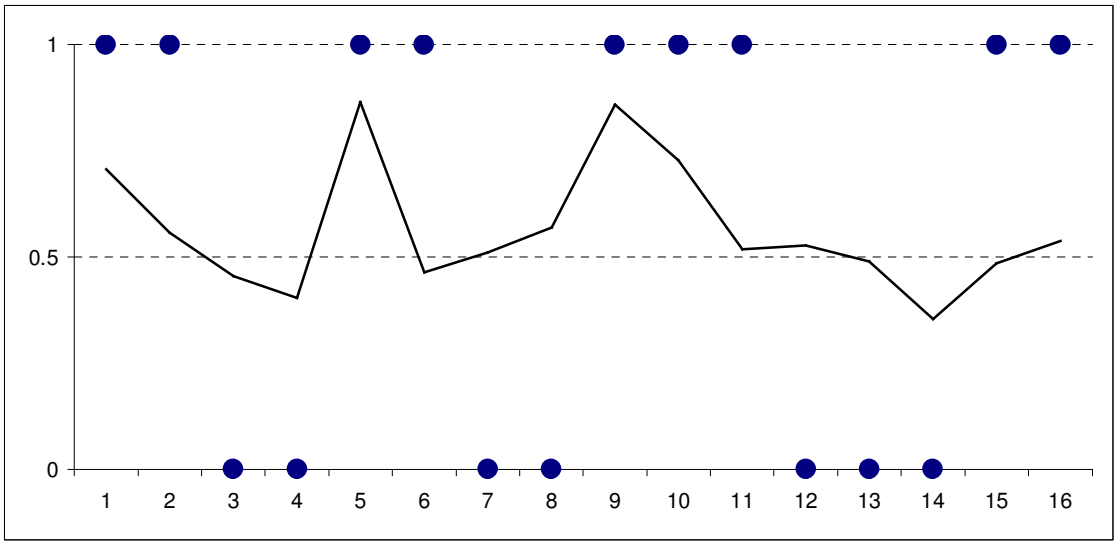


Figure 9.3: Illustration of the Likelihood function

$$N(\mathbf{Z}) = N(\mathbf{X}'\beta)$$

Given that there are only 16 observations there are 16 estimates. I use β_1 and β_2. So $\mathbf{X}$ is a 16×2 matrix.

It turns out that the distance between the data and the line is $1 - \mathscr{L}_i(\theta)$ for each data point (i.e. $i = 1, \cdots, 16$). And the likelihood function is just:

$$\mathscr{L}(\theta) = \prod_{i=1}^{n} \mathscr{L}_i(\theta)$$
$$\ln \mathscr{L}(\theta) = \sum_{i=1}^{n} \ln \mathscr{L}_i(\theta)$$

To solve for the estimates, due to the non-linearity, we must adopt an iterative search.

9.3.7 Likelihood ratio test

$$2(\ln \mathscr{L}_1 - \ln \mathscr{L}_0)$$

A similar statistic to R^2 is:

$$R^2 = 1 - \frac{\ln \mathscr{L}_1}{\ln \mathscr{L}_0}$$

Recall that in an OLS, R^2 is:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}} = 1 - \frac{\sum_{i=1}^{n} (z_i - \hat{z}_i)^2}{\sum_{i=1}^{n} (z_i - \bar{z}_i)^2}$$

9.3.8 Marginal effects

In linear regression, if the coefficient on x is β, then a 1-unit increase in x increases y by β. But what exactly does it mean in probit that the coefficient on x is 0.0923 and significant? It means that a 1% increase in x will raise the z-score of $\Pr(Y = 1)$ by 0.0923. And this coefficient is different from 0 at the 5% level. So raising x has a constant effect on Y^* and does not translate into a constant effect on the original Y.

Marginal impact of changing a variable is not constant. To see that:

$p = N(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)$

$$\frac{\partial p}{\partial x_i} = z(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)\beta_i$$

This expression depends on not just β_i, but on the value of x_i and all other variables in the equation. Typical options are to set all variables to their means or their medians. Or to fix the x_j and let x_i vary from its minimum to maximum values and Then you can plot how the marginal effect of x_i changes across its observed range of values.

9.3.9 Multinomial dependent variable

§ MLE

[Multinomial Logistic]

Let J be the number of discrete categories of the dependent variable. Z be one of the values of J. For the multinomial logistic regression we equate the linear component to the log of the odds of the j-th observation compared to the J-th observation.

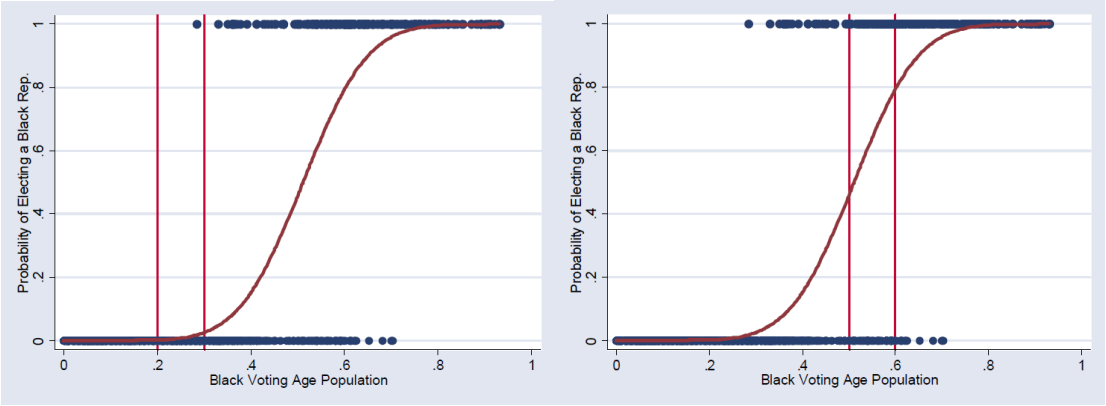


Figure 9.4: Marginal Effects

$$\ln\left[\frac{p_{ij}}{p_{iJ}}\right] = \ln\left[\frac{p_{ij}}{1-\sum_{j=1}^{J-1} p_{ij}}\right] = \sum_{k=0}^{K} x_{ik}\beta_k$$

Solving for p_{ij}, we have:

$$p_{ij} = \frac{\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}{1+\sum_{j=1}^{J-1}\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}$$
$$p_{iJ} = \frac{1}{1+\sum_{j=1}^{J-1}\exp\left(\sum_{k=0}^{K} x_{ik}\beta_k\right)}$$

Likelihood function:

$$\begin{aligned}\mathscr{L}(\beta) = f(y|\beta) &= \prod_{i=1}^{N}\frac{n_i!}{\prod_{j=1}^{J} y_{ij}!}\prod_{j=1}^{J} p_{ij}^{y_{ij}} \\ &= C\prod_{i=1}^{N}\prod_{j=1}^{J} p_{ij}^{y_{ij}}\end{aligned}$$

When $J = 2$, then this is identical to equation (9.3.5). The constant C is the fraction of factorials.

Replacing the J-th term, the above equation can be re-writtn as:

$$\begin{aligned}
\mathscr{L}(\beta) &= C\prod_{i=1}^{N}\prod_{j=1}^{J-1} p_{ij}^{y_{ij}}\left(p_{iJ}^{n_i-\sum_{j=1}^{J-1}y_{ij}}\right)\\
&= C\prod_{i=1}^{N}\prod_{j=1}^{J-1} p_{ij}^{y_{ij}}\frac{p_{iJ}^{n_i}}{p_{iJ}^{\sum_{j=1}^{J-1}y_{ij}}}\\
&= C\prod_{i=1}^{N}\prod_{j=1}^{J-1} p_{ij}^{y_{ij}}\frac{p_{iJ}^{n_i}}{\prod_{j=1}^{J-1}p_{iJ}^{y_{ij}}}\\
&= C\prod_{i=1}^{N}\frac{\prod_{j=1}^{J-1} p_{ij}^{y_{ij}}}{\prod_{j=1}^{J-1}p_{iJ}^{y_{ij}}}p_{iJ}^{n_i}\\
&= C\prod_{i=1}^{N}\prod_{j=1}^{J-1}\left(\frac{p_{ij}}{p_{iJ}}\right)^{y_{ij}} p_{iJ}^{n_i}
\end{aligned}$$

Sub in what is for p_{ij} and get:

$$\begin{aligned}
&C\prod_{i=1}^{N}\prod_{j=1}^{J-1}\left(\exp\left(\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)\right)^{y_{ij}}\left(1+\sum_{j=1}^{J-1}\exp\left(\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)\right)^{-n_i}\\
&= C\prod_{i=1}^{N}\prod_{j=1}^{J-1}\left(\exp\left(y_{ij}\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)\right)\left(1+\sum_{j=1}^{J-1}\exp\left(\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)\right)^{-n_i}
\end{aligned}$$

and

$$\ln\mathscr{L}(\beta) = \ln C+\sum_{i=1}^{N}\sum_{J=1}^{J-1}\left(y_{ij}\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)-n_i\ln\left(1+\sum_{j=1}^{J-1}\exp\left(\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)\right)$$

F.O.C. (steps omitted and for exercise)

$$\frac{\partial\ln\mathscr{L}(\beta)}{\partial\beta_{kj}} = \sum_{i=1}^{N} y_{ij}x_{ik} - n_i p_{ij}x_{ik} = 0$$

[so it is logical for the p.] There are $(J-1)(K-1)$ equations that need to solve simultaneously and solve every β_{kj}.

The Hessian matrix is:

$$\begin{aligned}
&\frac{\partial^2\ln L(\theta)}{\partial\beta_{kj}\partial\beta_{k'j'}} = \sum_{i=1}^{N} n_i x_{ik}\frac{\partial}{\partial\beta_{k'j'}}\left(\frac{\exp\left(\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)}{1+\sum_{J=1}^{J-1}\exp\left(\sum_{k=0}^{K}x_{ik}\beta_{kj}\right)}\right)\\
&= \cdots\\
&= \begin{cases}-\sum_{i=1}^{N} n_i x_{ik}p_{ij}(1-p_{ij})x_{ik'} & j'=j\\ \sum_{i=1}^{N} n_i x_{ik}p_{ij}p_{ij'}x_{ik'} & j'\neq j\end{cases}
\end{aligned}$$

§ Ordered

An ordered Logit (or Probit) model can be described as follows:

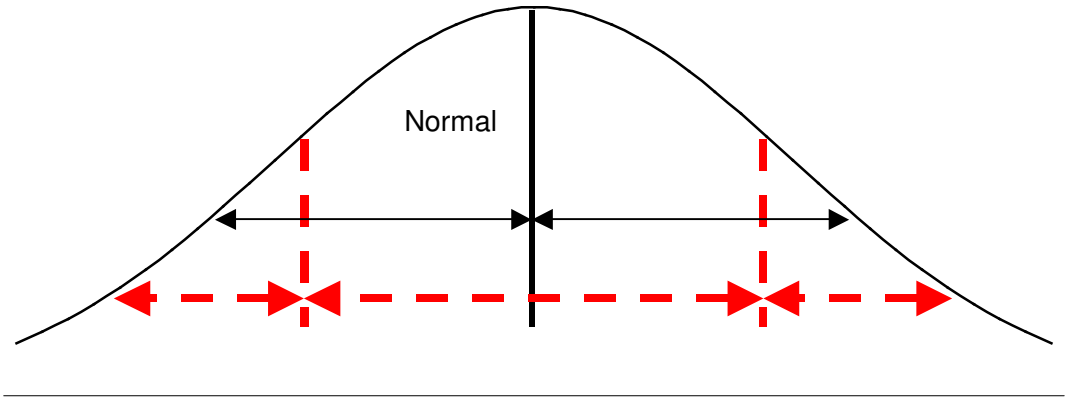


Figure 9.5: Ordered Probit

where the black denotes the binomial and the red denotes the three-group *ordered* Probit.

§ Unordered

Pick a base category and calculate the odds of the other possible outcomes relative to it. For example, say a student can enter a general, vocational, or academic program and use academic as the base category.

Then we will use multinomial logit to estimate Prob(general)/Prob(academic) and Prob(vocational)/Prob(academic). That is, the probability of choosing general or vocational relative to an academic program. These are two separate logit regressions.

The results from a multinomial logit can be interpreted as relative risk ratios (RRR) as:

$$\text{RRR} = \frac{\Pr(\text{general})}{\Pr(\text{academic})}$$

Or they can be interpreted as Odds Ratios:

$$\text{OR} = \frac{\frac{\Pr(\text{general})}{1-\Pr(\text{general})}}{\frac{\Pr(\text{academic})}{1-\Pr(\text{general})}}$$

9.3.10 Example

We study if gender or aptitude has an impact on college admission. The data are:[1]

aptitude	8	7	5	3	3	5	7	8	5	5	4	7	3	2	4	2	3	4	3	2
gender	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
admit	1	0	1	0	0	1	1	1	1	1	0	1	1	0	0	0	0	1	0	0

The results are given in the Appendix.

9.4 Discriminant Analysis

Restor Dent Endod. 2018 Aug; 43(3): e34.
Published online 2018 Aug 9. doi: 10.5395/rde.2018.43.e34
PMCID: PMC6103548
PMID: 30135853
Statistical notes for clinical researchers: simple linear regression 2 – evaluation of regression line
Hae-Young Kimcorresponding author

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6103548/: :text=SSR

Probit/Logit is critically important in the area of default prediction. The left hand side is default/survival and the right hand side includes all the possible explanatory variables (such as leverage, profitability, turnover, etc.) A similar tool that does the same thing is the discriminant analysis.

The two-class discriminant analysis is equivalent to a linear regression. It can be depicted best by the following picture.

Fundamental equations for DA are the same as for MANOVA: First, create cross-products matrices for between-group differences and within-groups differences,

SStotal= SSbg+ SSwg.

[1]Note that in the default case, the admit is easily interpreted as default or survival.

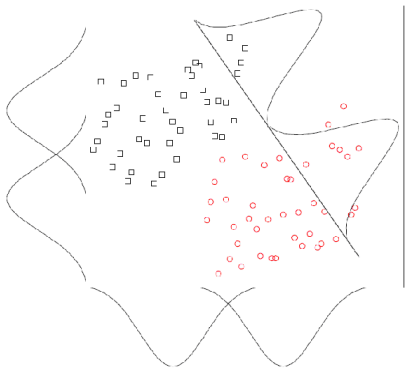


Figure 9.6: Discriminant Analysis

The determinants are calculated for these matrices and used to calculate a test statistic – either Wilks' Lambda or Pillai's Trace.

Wilks' lambda is used to test the null hypothesis that the populations have identical means on D. Wilks' lambda is

$$\Lambda = \frac{SS_{within\_groups}}{SS_{total}}$$

So the smaller the Λ the more doubt cast upon that null hypothesis. We can determine how much of the variance in the grouping variable is explained by our predictor variables by subtracting the Λ from one.

If your primary purpose is to predict group membership from the variates (rather than to examine group differences on the variates), you need to do classification. The Bayes' rule:

$$p(G_i|D) = \frac{p(G_i) \times p(D|G_i)}{\sum_{i=1}^{g} p(G_i) \times p(D|G_i)}$$

.

Each subject's discriminant score is used to determine the posterior probabilities of being in each of the two groups. The subject is then classified (predicted) to be in the group with the higher posterior probability.

The discriminant function (which is linear) is:

$$\begin{aligned} Y_i &= a_1 X_{1i} + a_2 X_{2i} + \cdots + a_m X_{mi} \\ &= A'X \end{aligned}$$

where a_i's are coefficients (also known as the eigenvector). Denote $A = [a_1, \cdots, a_m]'$. The discriminant analysis (DA) is to maximize the followint ratio:

$$E = \frac{A'\mathbf{D}\mathbf{D}'A}{A'\mathbf{C}A} = \frac{A'\mathbf{B}A}{A'\mathbf{C}A}$$

where $\mathbf{B}$ is between-group covariance matrix and $\mathbf{C}$ is within-group covariance matrix (to be defined later). To maximize E, we differentiate E with respective to A (each element of A):

$$\begin{aligned}
&\frac{\partial E}{\partial A} = \frac{2\mathbf{B}A(A'\mathbf{C}A) - (A'\mathbf{B}A)(\mathbf{C}A)}{(A'\mathbf{C}A)^2} = 0 \\
&\frac{2(\mathbf{B}A - E\mathbf{C}A)}{A'\mathbf{C}A} = 0 \\
&(\mathbf{B} - E\mathbf{C})A = 0
\end{aligned}$$

and solve the simultaneous equations for a_i's. It is clear that A is the eigenvector.

If $\hat{A}$ maximizes E, then it is also true that $k\hat{A}$ also maximizes E where k is a constant. As a result A *is not unique*.

It can be shown that the following maximization problem also reaches the same result:

$$F = A'DD'A - \lambda(A'\mathbf{C}A - \ell)$$

where λ is the Lagrange multiplier and ℓ is an arbitrary constant. This maximization problem yields the following result:

$$\begin{aligned}
&\frac{\partial F}{\partial A} = 2DD'A - 2\lambda\mathbf{C}A = 0 \\
&\frac{\lambda}{H}\hat{A} = \mathbf{C}^{-1}D
\end{aligned}$$

where $H = D'A$ is 1×1. Alternatively,

$$
\begin{aligned}
&(\mathbf{B} - E\mathbf{C})A = 0 \\
&\mathbf{B}A = E\mathbf{C}A \\
&E\mathbf{B}A + \mathbf{B}A = E\mathbf{B}A + E\mathbf{C}A \\
&(1+E)\mathbf{B}A = E(\mathbf{B}+\mathbf{C})A \\
&\mathbf{B}A = \frac{E}{1+E}(\mathbf{B}+\mathbf{C})A \\
&A'\mathbf{B}A = \frac{E}{1+E}A'(\mathbf{B}+\mathbf{C})A \\
&\frac{A'\mathbf{B}A}{A'\mathbf{S}A} = \frac{E}{1+E} = E^*
\end{aligned}
$$

where $\mathbf{S} = \mathbf{B} + \mathbf{C}$ represents the "total variance". Hence, immediately we have:

$$\frac{\lambda}{H}^* \hat{A} = \mathbf{S}^{-1}D$$

We shall show that this result is identical to a dummy regression. Note that a regression of the following:

$$y = a_1X_1 + a_2X_2 + \cdots + a_kX_k + e$$

where (taken $k = 2$ as an example)

$$y = \begin{pmatrix} Y_1 - \overline{Y} \\ \vdots \\ Y_n - \overline{Y} \end{pmatrix} \quad X = \begin{bmatrix} X_{11} & X_{12} \\ \vdots & \vdots \\ X_{n1} & X_{n2} \end{bmatrix}$$

$$
\begin{aligned}
A = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} &= (X'X)^{-1}(X'y) \\
&= \begin{bmatrix} \sum_{i=1}^n X_{i1}^2 & \sum_{i=1}^n X_{i1}X_{i2} \\ \sum_{i=1}^n X_{i1}X_{i2} & \sum_{i=1}^n X_{i2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^n X_{i1}y_i \\ \sum_{i=1}^n X_{i2}y_i \end{bmatrix} \\
&= \mathbf{S}^{-1}\left(\frac{n_a n_b}{n_a + n_b}D\right)
\end{aligned}
$$

where D is defined as:

$$D = \begin{pmatrix} \overline{X}_{1a} - \overline{X}_{1b} \\ \overline{X}_{2a} - \overline{X}_{2b} \end{pmatrix}$$

and $DD' = \mathbf{B}$ is known as the "between group" covariances. Note that this is because Y is a dichotomous variable (dummy) and there are n_b 0's and n_a 1's and $\overline{Y} = n_a$.

9.5 *k*-means

Hull Chapter 2 (2.2, 2.3)

(https://www.geeksforgeeks.org/k-means-clustering-introduction/) k-means clustering is a technique used to organize data into groups based on their similarity. For example online store uses k-Means to group customers based on purchase frequency and spending creating segments like Budget Shoppers, Frequent Buyers and Big Spenders for personalised marketing.

The algorithm works by first randomly picking some central points called centroids and each data point is then assigned to the closest centroid forming a cluster. After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster. This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group.

See Hull.

Assume the following data in Table **??**. We intentionally simulate the data using two groups. But in reality, especially in high dimensions, we do not know how many groups there are.

	x	y		x	y
1	0.091116	0.056635	6	0.713759	0.464647
2	0.070649	0.230673	7	0.900406	0.720197
3	0.090885	0.436225	8	0.900173	0.950603
4	0.100947	0.217840	9	0.748041	0.855867
5	0.241827	0.175596	10	0.793759	0.779156
			11	0.730267	0.864718
			12	0.918623	0.946683

Table 9.1: Data

To visualize the data, we plot them in Figure 9.7. Again, for better visual, we intentionally draw green circles for one group and orange crosses for the other group. In reality, we will not be able to tell the data.

The first step is to randomly assign centers. Given that we do not know what

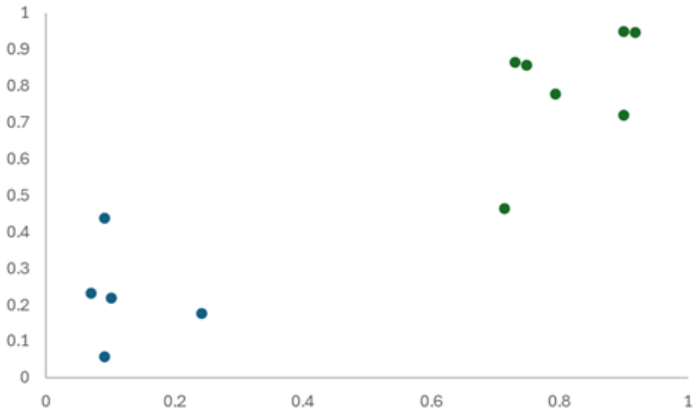


Figure 9.7: Data

the ideal number of clusters is, we must try different values of k. Since we cheat, we let $k = 2$ and randomly select two centers and their coordinates are in Table 9.2. See Figure 9.8 for the visual.

	x	y
C1	0.303512	0.790842
C2	0.528847	0.224935

Table 9.2: 2 Randomly Assigned Centers

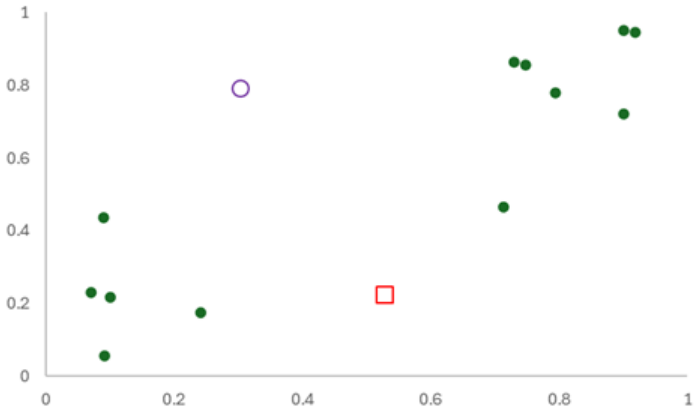


Figure 9.8: 2 Randomly Assigned Centers

The second step is to calculate the distance of each point to the two centers and examine which center is closer to and assign that center to the point. See Table 9.3.

	dist 2 C1	dist 2 C2	nearest C		dist 2 C1	dist 2 C2	nearest C
1	0.764311	0.46897	2	6	0.524124	0.302745	2
2	0.606642	0.458234	2	7	0.601061	0.619145	1
3	0.413477	0.486266	1	8	0.61768	0.815155	1
4	0.607753	0.427959	2	9	0.44926	0.667924	1
5	0.618331	0.29123	2	10	0.490387	0.61428	1
				11	0.433103	0.67074	1
				12	0.634546	0.820272	1

Table 9.3: Distances to Centers

We can see that in group 1, 1 point (#3) is mis-specified as cluster 1 and yet it should belong to cluster 2. Of course, this is because the position of center 1 is not correctly chosen. Similarly, 1 point in group 2 (#6) is mis-specified. See Figure 9.9.

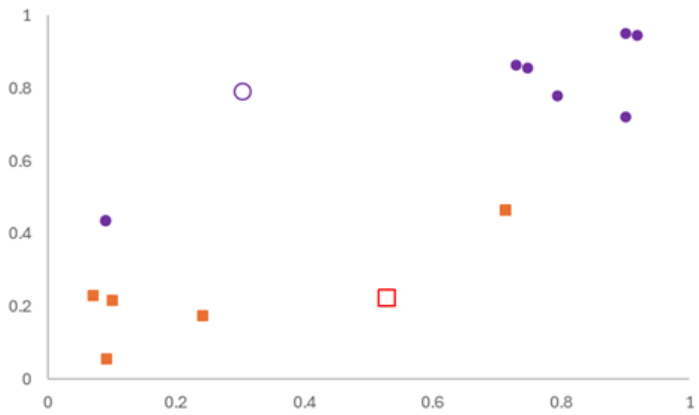


Figure 9.9: Chosen Nearest Center

Now, in step 3, we recalculate the new centers. The new positions of the centers are calculated using the points that belong to each center. See Figure 9.10.

	x	y
C1	0.726022	0.79335
C2	0.24366	0.229078

Table 9.4: Revised Centers

Then, in step 4, we reassign points to the new positions of the two centers. Again, we calculate each distance and use it to determine which center a point belongs to. See Figure 9.11.

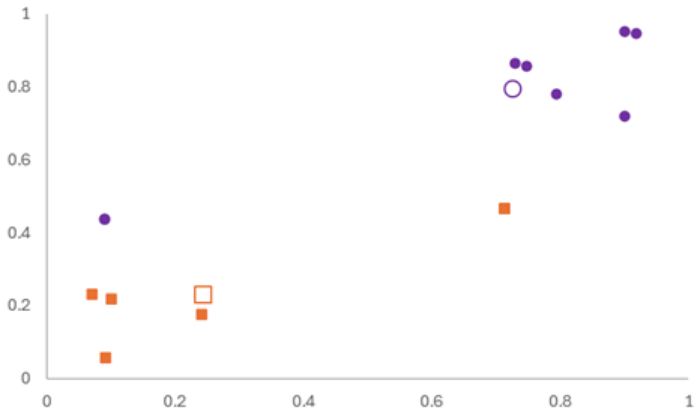


Figure 9.10: Centers Moved

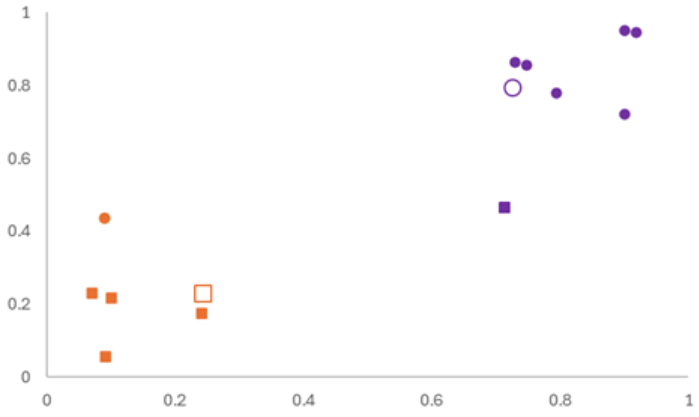


Figure 9.11: Reassign Points to New Centers

In the final and last step, we recalculate the positions of the center again. See Figure 9.12. This is the last move since there will be no mis-specification. We will find that the convergence is reached and there is no more iteration.

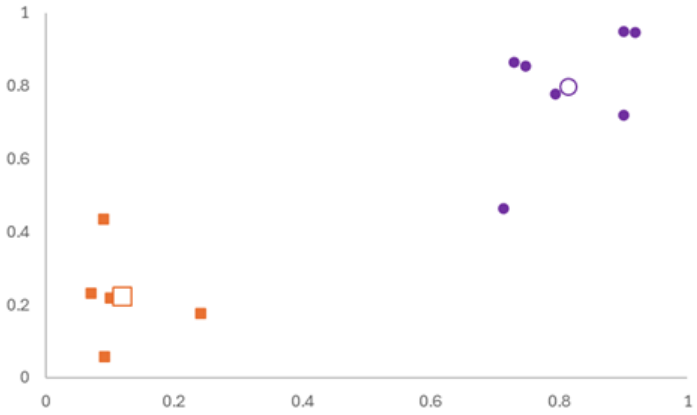


Figure 9.12: Recalculate Centers

As mentioned earlier, we do not know in data how many clusters are a correct number. The way to detect it is to try all numbers (i.e. $k = 2, 3, \cdots$) and then sum up all the distances. In theory, if we have the same number of clusters as the number of points, then we have perfect clustering. But clearly this is meaningless. To seek the right balance of too few clusters (i.e. too much mis-classification) and too many clusters (i.e. inefficient clustering), we use the total distance between points and their corresponding centroids. Inertia is defined as follows, as the sum of all the distances.

$$I = \sum_{i=1}^{n} d_i^2$$

In Figure 9.13, we can see that in general more clusters will reduce the total distance, implying better clustering. And yet, the marginal benefit is diminishing. In the graph, we can see that there isn't much gain after 3 clusters. As a result, we should stop at $k = 3$.

9.6 Hierachical Clustering

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

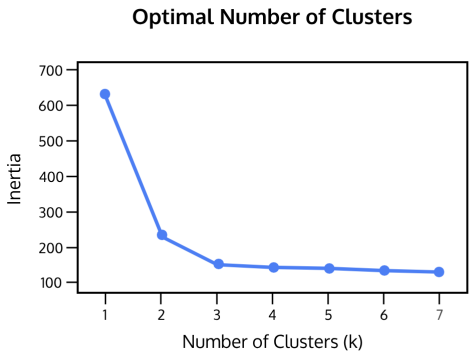


Figure 9.13: Inertia

Take an example in Table 9.5 where 5 individuals have scores in social media and gym usage. We would like to use these two measures (features) to build a hierarchy of groups.

	social media	gym
Alan	2	3
Lisa	5	2
Joe	5	3
Max	1	4
Caro	4	5

Table 9.5: An Example

From Table 9.5, we can calculate pair-wise distances (Euclidean distance), as in Table 9.6. and Joe and Lisa are the closest. So they form a group.

	Alan	Lisa	Joe	Max	Caro
Alan	0	3.16	3.00	1.41	2.83
Lisa		0	**1.00**	4.47	3.16
Joe			0	4.12	2.24
Max				0	3.16
Caro					0

Table 9.6: Pair Distances

To calculate the next cluster, we use the single linkage method (i.e. whoever in the cluster is the closest to others) and in this case Joe is closer to all other persons than Lisa is.

	Alan	Lisa,Joe	Max	Caro
Alan	0	3	**1.41**	2.83
Lisa,Joe		0	4.12	2.24
Max			0	3.16
Caro				0

Table 9.7: Distances from Lisa/Joe to Others

In this case, the next members to consider are Alan and Max. They form a group. Now we update Table 9.7 as follows:

	Lida,Joe	Max,Alan	Caro
Lisa,Joe	0	3.00	**2.24**
Max,Alan		0	2.83

Table 9.8: Distances from Lisa/Joe and Max/Alan to Caro

It is clear that Caro should be grouped with Lisa/Joe in the next hierarchy. Now we have everybody and the final dendrogram is given in Figure 9.14.

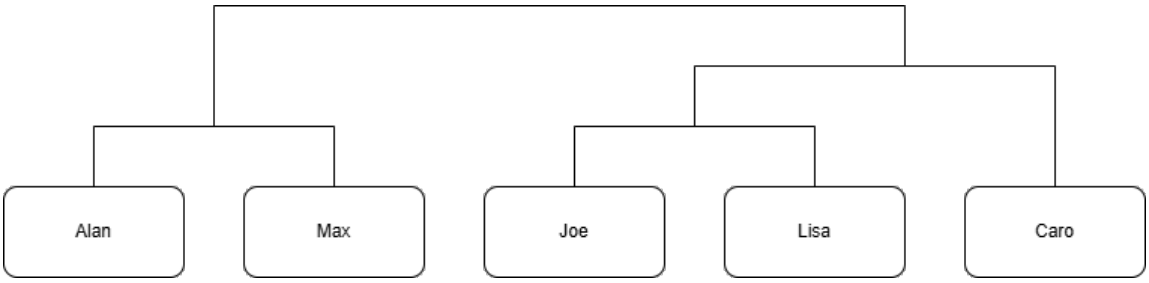


Figure 9.14: Final Dendrogram

Finally, as alternatives to Euclidean distance, we could use Manhattan distance (which is the sum of the x distance and y distance) and maximum distance (which is the larger of the x distance and y distance). Also, as alternatives to single linkage method (which uses the distance between the closest elements in the cluster), we could use complete linkage method (which uses the distance between the most distant elements of the cluster) and average linkage method (which uses the average of all pairwise distances).

9.7 Random Forest

(wiki) Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that works by creating a multitude of decision trees during training. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the output is the average of the predictions of the trees. Random forests correct for decision trees' habit of overfitting to their training set.

9.7.1 CART (Classification and Regression Tree)

Hull Chapter 4

(https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/) CART(Classification And Regression Trees) is a variation of the decision tree algorithm. It can handle both classification and regression tasks. Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees). CART was first produced by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984.

§ Decision Trees

(Stat Quest)

Chest Pain	Blood Circulation	Blocked Arteries	Heart Disease
No	Bad	No	No
Yes	Good	Yes	Yes
Yes	Good	No	No
Yes	Bad	???	Yes
⋮	⋮	⋮	⋮

Table 9.9: Data

Note that here we have missing data. And we skip them. So the total is not 303 patients (only 297 patients).

Given that there are no single cell that is 100% (known as impurity), we need to use Gini index to measure it. Gini index can be calculated using the following

Chest Pain			
(yes)	↙	↘	(no)
Heart Disease		Heart Disease	
Yes	No	Yes	No
105	39	34	125
72.92%	27.08%	21.38%	78.62%

Table 9.10: Chest Pain

Blood Circulation			
(good)	↙	↘	(bad)
Heart Disease		Heart Disease	
Yes	No	Yes	No
37	127	100	33
22.56%	77.44%	75.19%	24.81%

Table 9.11: Blood Circulation

Blocked Arteries			
(yes)	↙	↘	(no)
Heart Disease		Heart Disease	
Yes	No	Yes	No
92	31	45	129
74.80%	25.20%	25.86%	74.14%

Table 9.12: Blocked Arteries

formula:[2]

$$\mathscr{G} = 1 - \sum_j p_j^2$$

where p_j is probability.

In the chest pain case, it is 0.395 for having a chest pain:

$$0.395 = 1 - \left(\tfrac{105}{105+39}\right)^2 - \left(\tfrac{39}{105+39}\right)^2$$

and similarly 0.336 for not having a chest pain. The total Gini index is the weighted average of the two separate Gini indices:

$$0.364 = \left(\tfrac{144}{144+159}\right) \times 0.935 + \left(\tfrac{159}{144+159}\right) \times 0.336$$

Similarly, we can calculate the total Gini index for blood circulation to be 0.360 and for the blocked arteries to be 0.381. Since blood circulation has the lowest impurity (0.360), it will be on the top of the tree (root).

To select the next level, we continue to compare different Gini indices. After blood circulation been chosen, we have either blocked arteries or chest pain to be the next level. The results are given in Table 9.13.

blood circ yes	
37	127

blocked art yes		blocked art no	
13	98	24	29

blood circ yes	
37	127

chest pain yes		chest pain no	
24	25	13	102

Table 9.13: Compare Blocked Arteries and Chest Pain

We compare their Gini indices: 0.30 for blocked arteries and 0.29 for chest pain and choose blocked arteries in that it has a lower Gini score. Then, the last level is chest pain. Finally, we have Table 9.14. Figure 9.15.

We next construct the right side of the tree. See Table 9.15. Note that the tree is shorter. This is because further splitting the tree gives a higher Gini score. As a result, as we can see in Table 9.15, the final leaf stops at chest pain yes since further splitting it into blocked arteries yields a higher Gini score.[3] Figure **??**.

§ Regression Trees

Here the variables are real numbers. See Figure 9.16

[2]https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb

[3]The Gini score for chest pain yes is 0.06. And further splitting it into blocked arteries is 0.29.

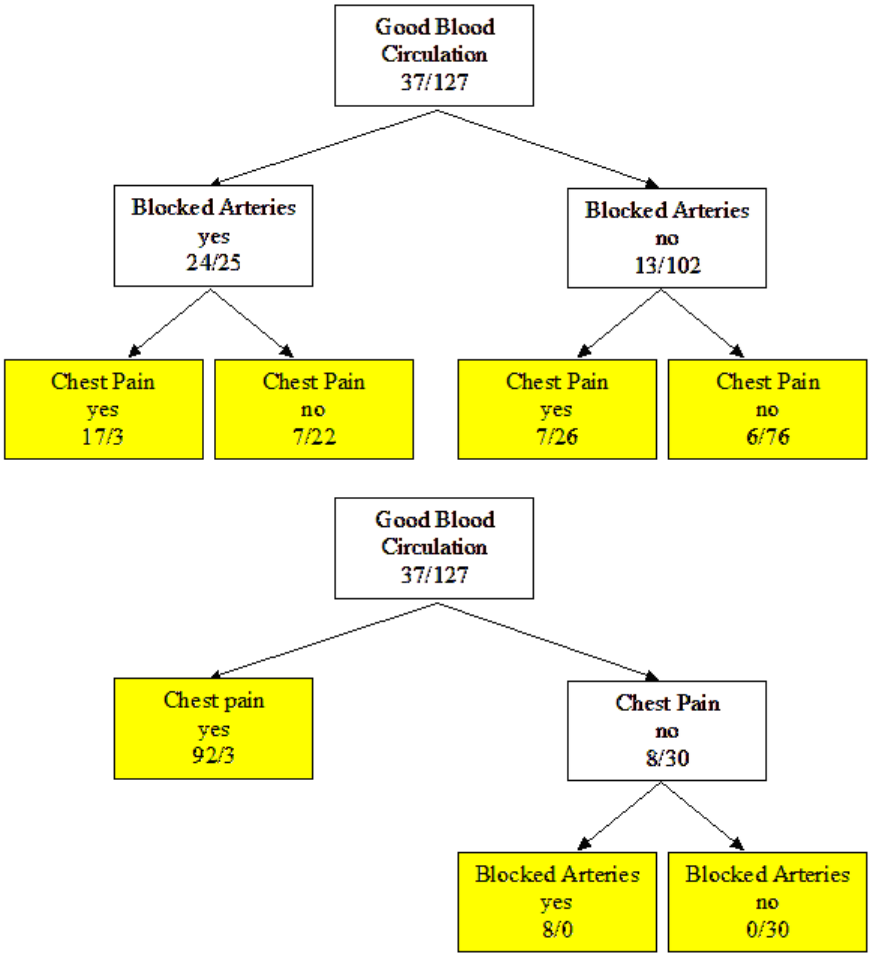


Figure 9.15: Decision Tree under Good Blood Circulation

good blood circ			
37		127	
blocked art yes		blocked art no	
24	25	13	102
chest pain yes	chest pain no	chest pain yes	chest pain no
17	3	7	22	7	26	6	76

Table 9.14: Decision Tree 1

good blood circ	
37	127

↙ ↘

chest pain yes	
92	3

chest pain no	
8	30

↙ ↘

blocked art yes		blocked art no	
8	0	0	30

Table 9.15: Decision Tree 2

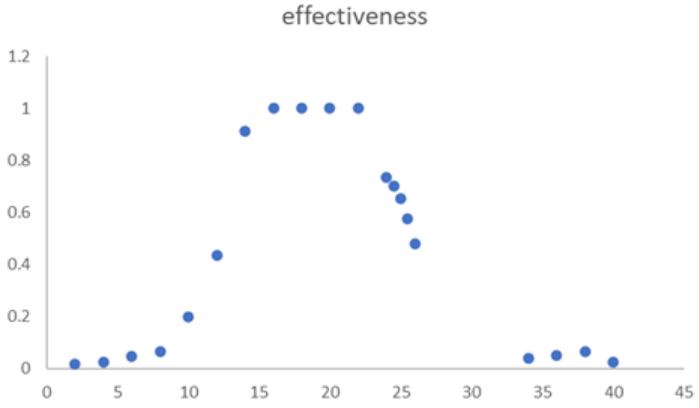


Figure 9.16: Data

To find the correct cutoff of the top level, we need to move one data point at a time. The first try is dosage greater than or less than 3. Then we calculate the square errors.

§ CART

Here variables are both. See an Excel example

9.7.2 Random Forest

Random forest is a way to randomize CART. The major problem with CART is that the result (i.e. tree) is very sensitive data (i.e. a slight change or data can result in drastically different tree, hence prediction). For a model like this, we call it "unintelligent". To make it intelligent, we need to randomize it.

By now (given that this is the last chapter), you should already know that so called artificial *intelligent*, lies in "randomness" in the system. In genetic algorithm, mutation (i.e. generated by random numbers) is the key of positive evolution. In swarm intelligence, exploration (i.e. generated by random numbers) is how a better solution can be found. In reinforcement learning (Q learning), a better route can only be discovered via random attempts (i.e. generated by random numbers).

The same idea is applied here. To implement a random forest, we simply draw *randomly* from the data:

1. randomly draw a sub-sample,
   (a) only draw a sub-set of the features
   (b) usually a square root of the original number of features
2. run a CART on the sub-sample and identify the result
3. repeat previous steps 1 and 2

9.8 Support Vector Machines

Hull Chapter 5

SVM is a tool that can best set a (soft) margin so that its prediction is best when the sample contains outliers.

Given an already clustered sample, we would like to draw a margin so that a new observation can be classified correctly. If we use the "mid point" between any two clusters, then this mid point is very sensitive to outliers. So how to accurately draw the margin is known as SVM.

In a one-dimensional line, the SVM is a point. In a two-dimensional plane, the SVM is a line. In a three-dimensional cube, the SVM is a plane. In a high-dimensional space, the SVM is a hyper cube.

Note that when SVM is drawn, the outlier will be mis-classified, hence an error. The balance is the tolerate such errors in order to make correct predictions of new samples. To do that we employ cross validations.

1. start with the data in a relatively low dimension
2. move the data into a higher dimension
3. find a SVC that separates the higher dimensional data into two groups

Step 2 above raises a question: how to decide a desired higher dimension? Is it x^2? Or x^3? Or $\frac{\pi}{4}\sqrt{x}$? To answer this question, SVM use "kernel functions" to systematically find SVC in higher dimensions.

For example, one can use a "polynomial kernel", which has a parameter d, which stands for the degree of the polynomial. When $d = 1$, the polynomial kernel computes the relationships between each pair of observations in 1 dimension, and these relationships are used to find a SVC. when $d = 2$, we get a 2nd dimension based on x^2, and the polynomial kernel computes the 2-dimensional relationships between each pair of observations. When $d = 3$, then we would get a 3rd dimension based on x^3 and the polynomial kernel computes the 2-dimensional relationships between each pair of observations.

The polynomial kernel takes the form of $(a \times b + r)^d$ where a and b are two observations in the data and r determines the coefficient of the polynomial. As mentioned earlier, d sets the degree. For example, set $r = \cdot$ and $d = 2$. Then $(a \times b + r)^d$

$$\begin{aligned}\left(a \times b + \tfrac{1}{2}\right)^2 &= a^2b^2 + ab + \tfrac{1}{4} \\ &= ab + a^2b^2 + \tfrac{1}{4} \\ &= \left(a, a^2, \tfrac{1}{2}\right) \cdot \left(b, b^2, \tfrac{1}{2}\right)\end{aligned}$$

Using the inner product, we can see that the first term is the x-axis, and the second term is the y-axis. Naturally the third term is the z-axis but since it is a constant (½), we simply can ignore it.

If we set $r = 1$ and $d = 2$, then it becomes:

$$\left(\sqrt{2}a, a^2, 1\right) \cdot \left(\sqrt{2}b, b^2, 1\right)$$

This scales the x-axis.

If we want to know the relationship between dosage 9 mg and dosage 14 mg, then we simply plug the two numbers in the equation and get $(9 \times 14 + 1/2)^2 = 16002.25$.

Another common kernel is radial kernel, also known as radial basis function (RBF) kernel. It is like a weighted nearest neighbor model. The closest observations have a lot of influence on how we classify the new observation. Vice versa, the observations that are further away have little influence.

Radial kernel takes the form of $e^{-\gamma(a-b)^2}$ where γ is determined by cross validation. It scales the squared distance.

It is interesting to see the connection between the polynomial kernel and the radial kernel. Let $r = 0$ in the polynomial kernel and keep adding dimensions. This leads to

$$\left(a, a^2, \cdots, a^\infty\right) \cdot \left(b, b^2, \cdots, b^\infty\right)$$

which turns out to be the same as the radial kernel.

9.9 Homework

Homework

In the k-mean example, use $k = 3$ and redo the example.

Project

Default prediction.

Chapter 10

Genetic Algorithm

I crossed the goldenrod with poison ivy once. What do you think I got? Hay fever and the seven-year itch.

Josette ,1938 (a line by May Morris played by Joan Davis).

10.1 Introduction

https://www.youtube.com/watch?app=desktop&v=XP8R0yzAbdo

Chapter 11

GPU Computing

Smart people focus on the right things.

Jensen Huang, CEO of Nvidia

11.1 Introduction

GPU, or Graphics Processing Unit, plays an essential role in AI's success. Not exaggerating, without GPU, we would not have today's AI.

There are several versions of GPU chips. The most popular one is NVidia. Yet, Apple (M chips), Google, Meta, etc. all are developing their own GPU chips.

11.2 From Games to AI

As we all know, former generations of GPU are nothing more than a graphic display card.

11.2.1 CUDA

11.2.2 OpenCL

11.3 An Example by William Huang

Use Colab by Google, which is free.

11.3.1 Environment Setup

Colab has installed the most of the packaged we needed. The following packages are needed if environment is set up locally (local machine must have Nvidia GPU).

1. GPU Driver
2. Python 3.X
3. C++ Compilier (i.e., Visual Studio for Windows)
4. CUDA Toolkits
5. PyCuda Python Package

   The first command is to install `pycuda`:

   ```
   !pip install pycuda
   ```

   Then, import `pycuda`:

   ```
    import pycuda
import pycuda.driver as drv
drv.init()
   ```

   Then,

   key Info

- Compute Capability: version number of GPU architecture
- Streaming Multiprocessors(SMs): a GPU will have serveral SMs and each individual SM has a particular number of CUDA cores.
- Total Memory: decide the limit of data can be loaded

```
print('CUDA device query (PyCUDA version) \n')

print('Detected {} CUDA Capable device(s) \n'.format(drv.Device.count()))

for i in range(drv.Device.count()):

    gpu_device = drv.Device(i)
    print('Device {}: {}'.format( i, gpu_device.name() ) )
    compute_capability = float( '%d.%d' % gpu_device.compute_capability() )
    print('\t Compute Capability: {}'.format(compute_capability))
    print('\t Total Memory: {} megabytes'.format(gpu_device.total_memory()//(1024**2)))

    # The following will give us all remaining device attributes as seen
    # in the original deviceQuery.
    # We set up a dictionary as such so that we can easily index
    # the values using a string descriptor.

    device_attributes_tuples = gpu_device.get_attributes().items()
    device_attributes = {}

    for k, v in device_attributes_tuples:
        device_attributes[str(k)] = v

    num_mp = device_attributes['MULTIPROCESSOR_COUNT']

    # Cores per multiprocessor is not reported by the GPU!
    # We must use a lookup table based on compute capability.
    # See the following:
    # http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#compute-capabilities

    cuda_cores_per_mp = { 5.0 : 128, 5.1 : 128, 5.2 : 128, 6.0 : 64, 6.1 : 128, 6.2 : 128, 7.5:64}[compute_capability]

    print('\t ({}) Multiprocessors, ({}) CUDA Cores / Multiprocessor: {} CUDA Cores'.format(num_mp, cuda_cores_per_mp, num_mp*cuda_cores_per_mp))

    device_attributes.pop('MULTIPROCESSOR_COUNT')

    for k in device_attributes.keys():
        print('\t {}: {}'.format(k, device_attributes[k]))
```

Figure 11.1: Code

11.3.2 Data Communication between CPU and GPU

GPU has its own memory apart from computer's memory, which is called device memory. The prerequisite of GPU computing is load data into device memory.

Simple Example

Simple Calculation Speed Test

```
 import numpy as np
from time import time
import pycuda.autoinit
from pycuda import gpuarray
import matplotlib.pyplot as plt
import math
from scipy.stats import norm
```

11.3.3 Monte Carlo Simulation using GPU

Assume the stock price follows geometric brownian motion under risk neutral measure:

$$dS = rSdt + \sigma SdW$$

Solving the SDE,

$$ln(St) = ln(S0) + (r - \frac{\sigma^2}{2}t) + \sigma Wt$$

CPU Version

GPU Version

In the stock simulation, every path is independent which is pefect for GPU parallization.

ElementwiseKernel is a PyCuda wrapper for user to implement more complicated calculation.

- First input is a list of parameters feed into the function.
- Second input is imlementation. Note that PyCuda has automatically sets up the integer index i. Calculation will parallelize over it.

- Third parameter is the kernel name under C namespace.

  Note that the parameter and implementation is written in C++ syntax.

One-step Simulation Speed Test

11.4 A Case Study

Chapter 12

Quantum Computing

We live on an island surrounded by a sea of ignorance. As our island of knowledge grows, so does the shore of our ignorance.

John Archibald Wheeler, July 9, 1911 – April 13, 2008

12.1 Introduction

Microsoft has recently announced its advanced quantum chip – Majorana 1 (February 19, 2025). If true, then this is a revolutionary milestone in quantum mechanics.

(PC World) After 17 years of research, Microsoft has finally developed the Majorana 1 chip, its first quantum computing processor based on a brand-new material and architecture. At the heart of a quantum computer are quantum bits (qubits), which handle data in a way similar to today's binary bits but with significantly higher computational potential. Companies such as IBM, Microsoft, and Google have long tried to make qubits more stable, as they're sensitive to disturbances that can cause errors or data loss. With Majorana 1, up to a million qubits can fit on a single circuit, about the size of a desktop computer's processor. Instead of using electrons for calculations, the chip utilizes the Majorana particle, first described by Italian physicist Ettore Majorana in 1937.[1]

Before Microsoft, Google announced in 2024 its quantum chip but it is limited to 105 qubits, which is not permissible for large scale computing. Yet, it already demonstrated a remarkable error-correction ability.

[1] https://www.pcworld.com/article/2616121/microsoft-makes-quantum-computing-breakthrough-with-new-majorana-1-chip.html

It would be amazing in the next few years how AI can advance due to the power of quantum computing. In this chapter, we discuss briefly what quantum mechanics is and how quantum computing is the future of AI.

12.2 Understanding Light

Perhaps the most well-known concept about light is its wave-particle duality. Newton (1643 – 1727) first thought that light was particle. But then the *double-slit experiment* in 1801 by Thomas Young (1773 – 1829) overthrew Newton's conjecture.

(Canon Science Lab) The theory of light being a particle completely vanished until the end of the 19th century when Albert Einstein revived it. Now that the dual nature of light as "both a particle and a wave" has been proved, its essential theory was further evolved from electromagnetics into quantum mechanics. Einstein believed light is a particle (photon) and the flow of photons is a wave. [2]

Photoelectric Effect. (Canon Science Lab) Albert Einstein conducted research on the photoelectric effect, in which electrons fly out of a metal surface exposed to light. The strange thing about the photoelectric effect is the energy of the electrons (photoelectrons) that fly out of the metal does not change whether the light is weak or strong. (If light were a wave, strong light should cause photoelectrons to fly out with great power.) Another puzzling matter is how photoelectrons multiply when strong light is applied. Einstein explained the photoelectric effect by saying that "light itself is a particle," and for this he received the Nobel Prize in Physics. (Some say that this is a compensation for not giving him the prize for relativity.)

12.2.1 Time Dilation

Time travel could be a myth (or science fiction). But time dilation is real. Here time is just distance divided by speed. Given that the speed of light is a constant, time dilation provides two (or many) clocks in two difference places. One place's 1 minute could be 1 hour in another place (1 minute in heaven is 1 day on earth). It is exactly the discovery of time dilation that we could have GPS today.

$$d = v \times t$$

where d is distance, v is velocity (speed), and t is time.

[2] https://global.canon/en/technology/s_labo/light/001/11.html#:~:text=Einstein 20believed 20light 20is 20a,related 20to 20its 20oscillation 20frequency.

If we reverse the equation:

$$t = d/v$$

and if the speed is constant

12.2.2 Twins Paradox

One twin is sitting on earth while the other twin flies a rocket out to space and back. Due to time dilation, the space twin is younger than the earth twin – twins paradox.

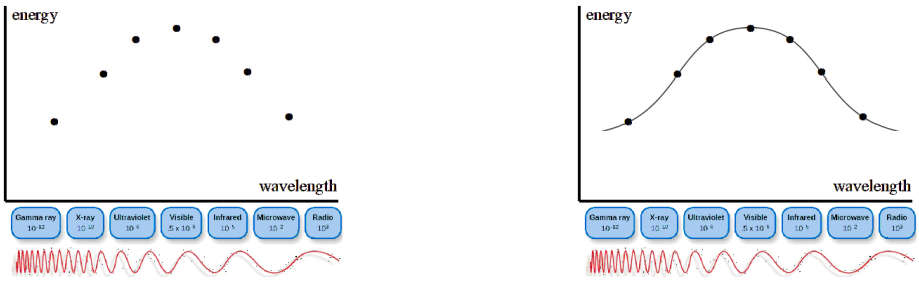
12.2.3 Special Relativity

Einstein first recognized that the faster you move through space, the slower you move through time.

12.3 Quantum Mechanics

Max Planck (April 23, 1858 – October 4, 1947), the father of quantum physics, gave the name quantum.

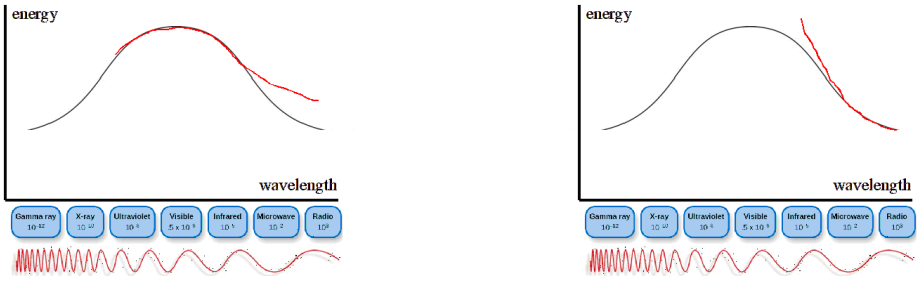
Physists prioer to Planck studied electromagnetic waves and their energy and draw a diagram as follows:



(a) Data (b) Guessed Function

Figure 12.1: Wavelength and Energy

Experimental physicists observed such an outcome and now it was up to the theoretical physicists to find a model for it. But the problem is that no matter how hard those physicists tried, they could not fit the the curve. Either they fit the left



(a) Right Tail Miss (b) Left Tail Miss

Figure 12.2: Fitting Miss

side and missed the right side, as in Figure 12.2a. Or they could fit the right tail, but missed the left side, as in Figure 12.2b.

Furthermore, the left is exploding, known as the ultraviolet catastrophe. See Figure 12.3.

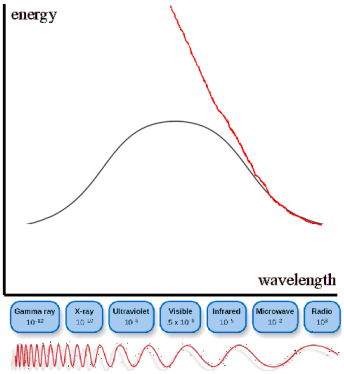


Figure 12.3: Ultraviolet Catastrophe

Planck solved this problem. At the time, people believed energy should be continuous. But Planck proposed that electromagnetic waves release energy in a discrete manner. If the energy is discrete, then how do we call it? Planck called it "a quantum" of energy. At the time, Planck did not understand why his formula worked – until Einstein who discovered photoelectric effect, which gave birth to another term: photon.

His formula:

$$B_\lambda(T) = \frac{\frac{2hc^2}{\lambda^5}}{e^{\frac{hc}{\lambda kT}} - 1}$$

12.3.1 Wheeler's Delayed Experiment

(wiki) John Archibald Wheeler's (July 9, 1911 – April 13, 2008) delayed-choice experiment describes a family of thought experiments in quantum physics proposed by John Archibald Wheeler, with the most prominent among them appearing in 1978 and 1984. These experiments illustrate the central point of quantum theory: *It is wrong to attribute a tangibility to the photon in all its travel from the point of entry to its last instant of flight.* These experiments close a loophole in the traditional double-slit experiment demonstration that quantum behavior depends on the experimental arrangement. The loophole has been called a "conspiracy" model where light somehow "senses" the experimental apparatus, adjusting its behavior to particle or wave behavior. By altering the apparatus after the photon is supposed to be in "flight" the loophole is closed. Cosmic versions of the delayed-choice use photons emitted billions of years ago; the results are unchanged. The concept of delayed choice has been productive of many revealing experiments. New versions of the delayed choice concept use quantum effects to control the "choices", leading to quantum delayed-choice experiments.

12.3.2 Niels Henrik David Bohr's Complementarity Principle

Complementarity principle, in physics, tenet that a complete knowledge of phenomena on atomic dimensions requires a description of both wave and particle properties. The principle was announced in 1928 by the Danish physicist Niels Bohr. Depending on the experimental arrangement, the behavior of such phenomena as light and electrons is sometimes wavelike and sometimes particle-like; i.e., such things have a wave-particle duality (q.v.). It is impossible to observe both the wave and particle aspects simultaneously. Together, however, they present a fuller description than either of the two taken alone.

12.3.3 Erwin Schrodinger's Cat, 1926

(wiki) In quantum mechanics, Schrödinger's cat is a thought experiment concerning quantum superposition. In the thought experiment, a hypothetical cat may be considered simultaneously both alive and dead, while it is unobserved in a closed box, as a result of its fate being linked to a random subatomic event that may or may not occur. This experiment viewed this way is described as a paradox. This thought experiment was devised by physicist Erwin Schrödinger in 1935 in a discussion with Albert Einstein to illustrate what Schrödinger saw as the problems

of the Copenhagen interpretation of quantum mechanics.

12.3.4 Werner Heisenberg's indeterminacy principle (uncertainty principle) 1927

(wiki) The uncertainty principle, also known as Heisenberg's indeterminacy principle, is a fundamental concept in quantum mechanics. It states that there is a limit to the precision with which certain pairs of physical properties, such as position and momentum, can be simultaneously known. In other words, the more accurately one property is measured, the less accurately the other property can be known.

12.3.5 Quantum Eraser Experiment

(wiki) In quantum mechanics, a quantum eraser experiment is an interferometer experiment that demonstrates several fundamental aspects of quantum mechanics, including quantum entanglement and complementarity. The quantum eraser experiment is a variation of Thomas Young's classic double-slit experiment. It establishes that when action is taken to determine which of two slits a photon has passed through, the photon cannot interfere with itself. When a stream of photons is marked in this way, then the interference fringes characteristic of the Young experiment will not be seen. The experiment also creates situations in which a photon that has been "marked" to reveal through which slit it has passed can later be "unmarked." A photon that has been "unmarked" will interfere with itself once again, restoring the fringes characteristic of Young's experiment.

12.3.6 Quantum Entanglement

(wiki) Quantum entanglement is the phenomenon of a group of particles being generated, interacting, or sharing spatial proximity in a manner such that the quantum state of each particle of the group cannot be described independently of the state of the others, including when the particles are separated by a large distance. The topic of quantum entanglement is at the heart of the disparity between classical physics and quantum physics: entanglement is a primary feature of quantum mechanics not present in classical mechanics.

12.4 Quantum Computing

We use an example of digital encryption. Image a number of a long series of digits. To crack the encryption, usually it employs the method of prime number factorization. A 300-digit encryption takes the traditional computer 150,000 years and yet only takes the quantum computer 1 second.

Let's take an easy example of a code of 1529. The prime number factorization goes as follows. We first divide 1529 by 3 (smallest prime number) and the remainder is 2. Then by the next prime number 5 and the remainder is 4. Then by the next prime number 7 and the remainder is 3. Finally by the prime number 11 and the remainder is 0. Afterwards, we can easily figure out (a few tries only) the code to be 1529.

The quantum method is different. First we translate the numbers into the binary representation.

$$\begin{aligned} 3 &= 0011 \\ 5 &= 0101 \\ 7 &= 0111 \\ 11 &= 1011 \end{aligned}$$

Then we define a quantum state φ_1 as:

$$|\varphi_1\rangle = \tfrac{1}{2}(|0011\rangle + |0101\rangle + |0111\rangle + |1011\rangle)$$

where $|x\rangle$ is a Dirac notation (known as a ket) in quantum language representing a matrix.

Do the same for $1529 = 10111111001$ and define another state φ_2 as:

$$|\varphi_2\rangle = |10111111001\rangle$$

Then, we divide φ_1 by φ_2 and get:

$$|\varphi_2\rangle \div |\varphi_1\rangle = |1011\rangle$$

which is 11.

The trick is that all the elements in φ_1 are quantum entangled (chip must be designed this way). This allows the division to be parallelly performed.

12.5 Finance Never Leaves Physics

Finance began its connection with physics back in 1900 when Louis Bachelier (March 11, 1870 – April 28, 1946) first modeled stock prices with Brownian motion. Even since, stochastic processes, differential equations, measure theory, among others have become part of finance.

For the past half of decade, finance has been so deeply involved with computation methods used in physics. Every day, millions of positions are closed and executed based upon risk neutral pricing mathematics (a,k.a. Q-measure pricing). As machine learning became popular, gradually industry has shifted from Q-measure pricing back to P-measure pricing that is consistent with statistical methods. Continuous time finance courses have gradually been abandoned, replaced by machine learning and data mining.

The recent announcement by Microsoft certainly brings back physics to finance again. Only this time it jumps onto quantum physics.

Index