

CS322: Deep Learning

Population & Sample

Sachchida Nand Chaurasia
Assistant Professor

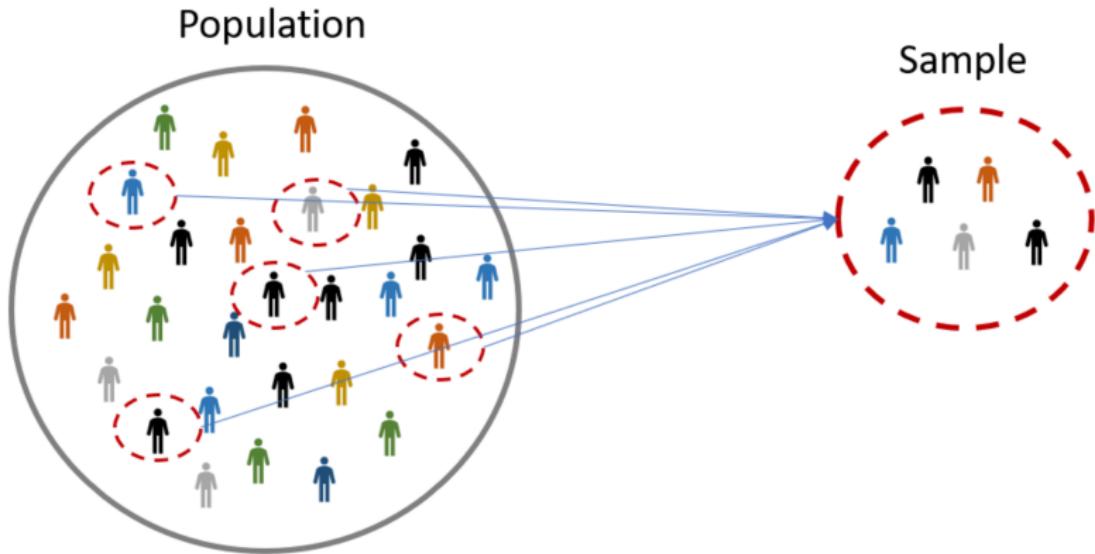
Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



September 30, 2021

Population and Sample I

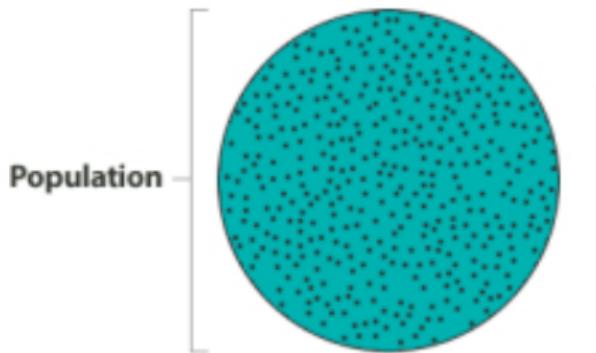


Population and Sample II

We begin a statistical investigation with a research question. The research question is frequently something we want to know about a population.

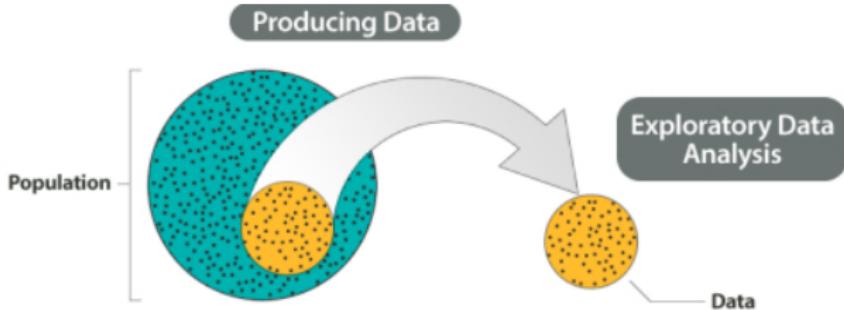
- ✓ The population can be people or other things, such as animals or objects.
- ✓ For example, we might want to know the answer to questions such as:
 - What percentage of U.S. adults supports the death penalty?
(Population: U.S. adults)
 - Do cell phones affect bees? (Population: bees)
 - Do cars get better gas mileage with a new gasoline additive?
(Population: cars)

Population and Sample III



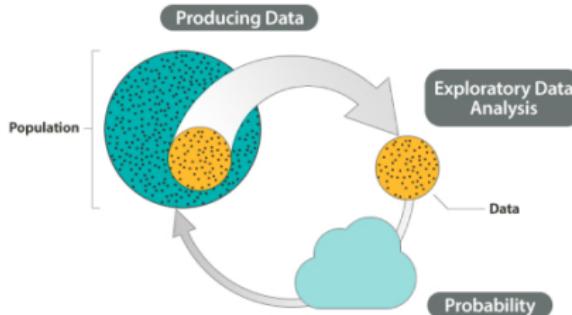
- ✓ In most cases, the population is a large group. Often, the population is so large that we cannot collect information from every individual in the population. So we select a sample from the population.
- ✓ Then we collect data from this sample.
- ✓ This is the first step in the statistical investigation. *We call this step producing data.*

Population and Sample IV



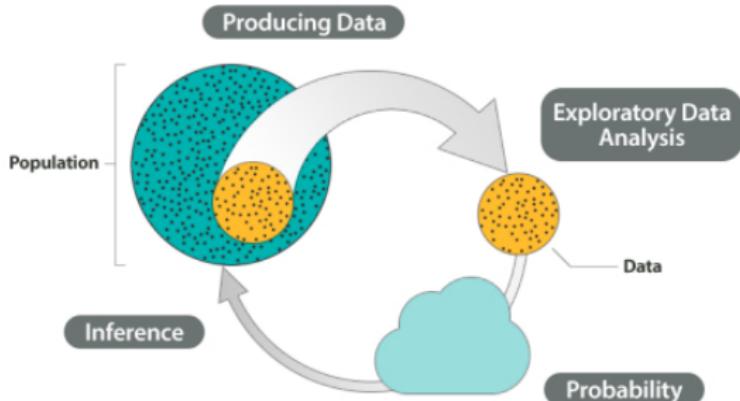
- ✓ Data is often a long list of information.
- ✓ To make sense of the data, we explore it and summarize it using graphs and different numerical measures, such as percentages or averages.
- ✓ *This is called exploratory data analysis.*

Population and Sample V



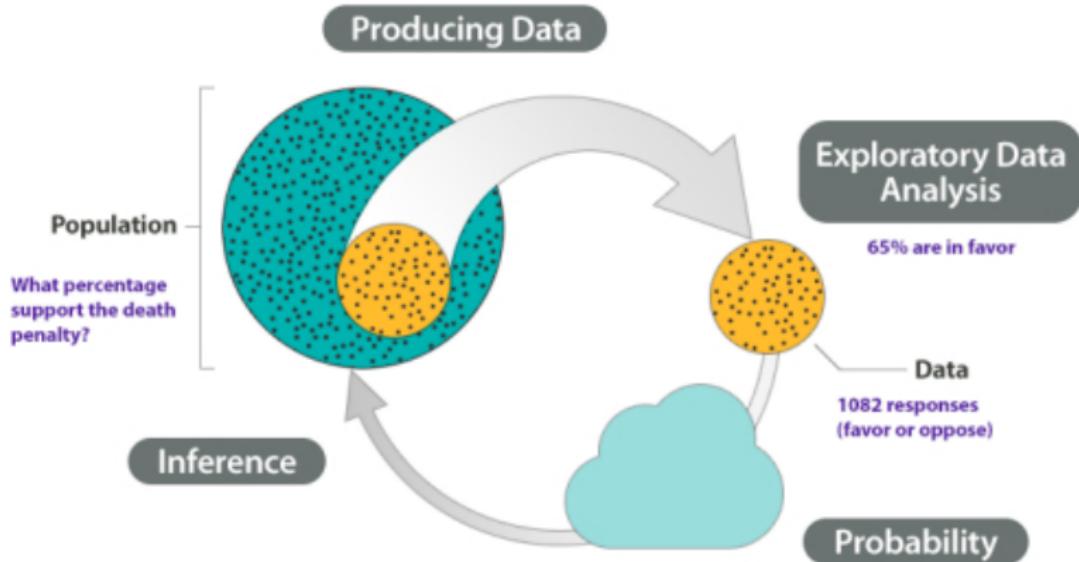
- ✓ Remember, our goal is to answer a question about a population based on a sample. Of course, samples will vary due to chance, and we will need to answer our question in spite of this variability.
- ✓ So we need to understand how sample results will vary and how sample results relate to the population as a whole when chance is involved.
- ✓ *This is where probability comes in.*

Population and Sample VI



- ✓ Probability is the “machinery” behind the last step in the process called **inference**.
- ✓ We infer something about a population based on a sample.
- ✓ This inference is the conclusion we reach from our sample data that answers our original question about the population.

Population and Sample VII



Conclusion: We can be 95% sure that the population percentage is between 62% and 68%.

Population and Sample VIII

- ① Produce Data: Determine what to measure, then collect the data.
 - ✓ The poll selected 1,082 U.S. adults at random. Each adult answered this question: “Do you favor or oppose the death penalty for a person convicted of murder?”
- ② Explore the Data: Analyze and summarize the data.
 - ✓ In the sample, 65% favored the death penalty.
- ③ Draw a Conclusion: Use the data, probability, and statistical inference to draw a conclusion about the population.

Population and Sample IX

- ✓ Our goal is to determine the percentage of the U.S. adult population that supports the death penalty.
- ✓ We know that different samples will give different results.
- ✓ What are the chances that our sample reflects the opinions of the population within 3%?
- ✓ Probability describes the likelihood that our sample is this accurate.
- ✓ So we can say with 95% confidence that between 62% and 68% of the population favor the death penalty.

Population and Sample X

Type of Research Question	Examples
Make an estimate about the population (often an estimate about an average value or a proportion with a given characteristic)	What is the average number of hours that community college students work each week?
	What proportion of all U.S. college students are enrolled at a community college?
Test a claim about the population (often a claim about an average value or a proportion with a given characteristic)	<p>Is the average course load for a community college student greater than 12 units?</p> <p>Do the majority of community college students qualify for federal student loans?</p>
Compare two populations (often a comparison of population averages or proportions with a given characteristic)	<p>In community colleges, do female students have a higher GPA than male students?</p> <p>Are college athletes more likely than non athletes to receive academic advising?</p>
Investigate a relationship between two variables in the population	<p>Is there a relationship between the number of hours high school students spend each week on Facebook and their GPA?</p> <p>Is academic counseling associated with quicker completion of a college degree?</p>

Population and Sample XI

- ✓ An observational study can answer questions about a population. But populations are generally large groups, so we cannot gather data from every individual in the population.
- ✓ Instead, we select a sample and gather data from the sample. We use the data from the sample to make statements about the population.

Here are two examples:

- A political scientist wants to know what percentage of college students consider themselves conservatives. The population is college students. It would be too time consuming and expensive to poll every college student, so the political scientist selects a sample of college students. Of course, the sample must be carefully

Population and Sample XII

selected to represent the political perspectives that are present in the population.

- A government agency plans to test airbags from Honda to determine if the airbags work properly. Testing an airbag means it has to be inflated and punctured, which ruins the airbag, so the researchers certainly cannot test every airbag.
 - ✓ Instead, they test a sample of airbags and draw a conclusion about the quality of airbags from Honda.

Population and Sample XIII

Important Point

- ✓ A goal is to use a sample to make valid conclusions about a population. Therefore, the sample must be representative of the population.
- ✓ A representative sample is a subset of the population that reflects the characteristics of the population.
- ✓ A sampling plan describes exactly how we will choose the sample. A sampling plan is biased if it systematically favors certain outcomes.

Population and Sample XIV

How biased sampling in a survey leads to misleading conclusions about the population:

Example

In 1936, Democrat Franklin Roosevelt and Republican Alf Landon were running for president. Before the election, the magazine Literary Digest sent a survey to 10 million Americans to determine how they would vote. More than 2 million people responded to the poll; 60% supported Landon.

The magazine published the findings and predicted that Landon would win the election. However, Roosevelt defeated Landon in one of the largest landslide presidential elections ever.

Population and Sample XV

What happened?

The magazine used a biased sampling plan. They selected the sample using magazine subscriptions, lists of registered car owners, and telephone directories. The sample was not representative of the American public. In the 1930s, Democrats were much less likely to own a car or have a telephone. The sample therefore systematically underrepresented Democrats. The poll results did not represent the way people in the general population voted.

Population and Sample XVI

Common survey plans that produce unreliable and potentially biased results

How to Sample Badly

Online polls: *The American Family Association (AFA) is a conservative Christian group that opposes same-sex marriage. In 2004, the AFA began a campaign in support of a constitutional amendment to define marriage as strictly between a man and a woman. The group posted a poll on its website asking AFA members to voice their opinion about same-sex marriage. The AFA planned to forward the results to Congress as evidence of America's opposition to same-sex marriage. Almost 850,000 people responded to the poll. In the poll, 60% favored legalizing same-sex marriage.*

Population and Sample XVII

What happened?

Against the wishes of the AFA, the link to the poll appeared in blogs, social-networking sites, and a variety of email lists connected to gay/lesbian/bisexual groups. The AFA claimed that gay rights groups had skewed its poll. Of course, the results of the poll would have been skewed in the other direction had only AFA members been allowed to participate.

This is an example of a voluntary response sample. The people in a voluntary response sample are self-selected, not chosen. For this reason, a voluntary response sample is biased because only people with strong opinions make the effort to participate.

Mall surveys: *Have you ever noticed someone surveying people at a mall? People shopping at a mall are more likely to be teenagers, retired*

Population and Sample XVIII

people, or people who have more money than the typical Indian. In addition, unless interviewers are carefully trained, they tend to interview people with whom they are comfortable talking. For these reasons, mall surveys frequently over-represent the opinions of rich middle-class or retired people. Mall surveys are an example of a convenience sample.

Population and Sample XIX

How to Eliminate Bias in Sampling

✓ In a voluntary response sample, people choose whether to respond. In a convenience sample, the interviewer chooses who will be part of the sample. In both cases, personal choice produces a biased sample.

Random sampling is the best way to eliminate bias.

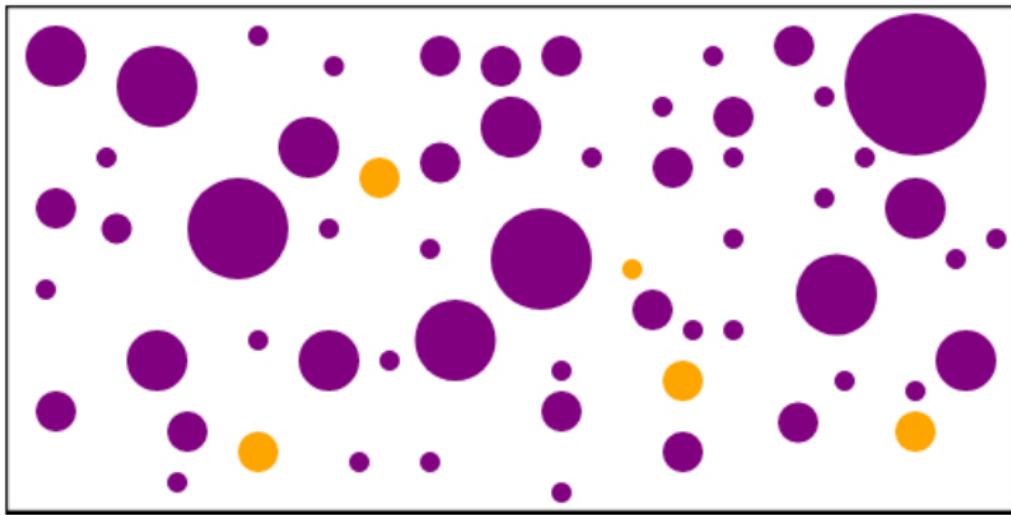
✓ Collecting a random sample is like pulling names from a hat (assuming every individual in the population has a name in the hat!).

✓ In a simple random sample everyone in the population has an equal chance of being chosen.

✓ Reputable polling firms use techniques that are more complicated than pulling names out of a hat. But the goal is the same: eliminate bias by using random chance to decide who is in the sample.

Population and Sample XX

Random Samples



Population and Sample XXI

- ✓ Random selection also guarantees that the sample results do not change unsystematic from sample to sample.
- ✓ When we use random selection, the variability we see in sample results is due to chance.
- ✓ The results obey the mathematical laws of probability.
- ✓ Probability is the machinery for drawing conclusions about a population on the basis of samples. To use this machinery, the sample must be chosen by random chance.

The main points about sampling:

- We draw a conclusion about the population on the basis of the sample.

Population and Sample XXII

- To draw a valid conclusion, the sample must be representative of the population.
- A representative sample is a subset of the population that reflects the characteristics of the population.
- A sample is biased if it systematically favors a certain outcome.
- Random selection eliminates bias.

Population and Sample XXIII

We have not mentioned the size of the sample

Are larger samples more accurate? Well, the answer is *Yes* and *No*

For *No*: Recall the 1936 presidential election. A sample of over 2 million people did not correctly identify the winner of the election. Two million people is a huge sample, yet the results were completely wrong. So a large sample does not guarantee reliable results.

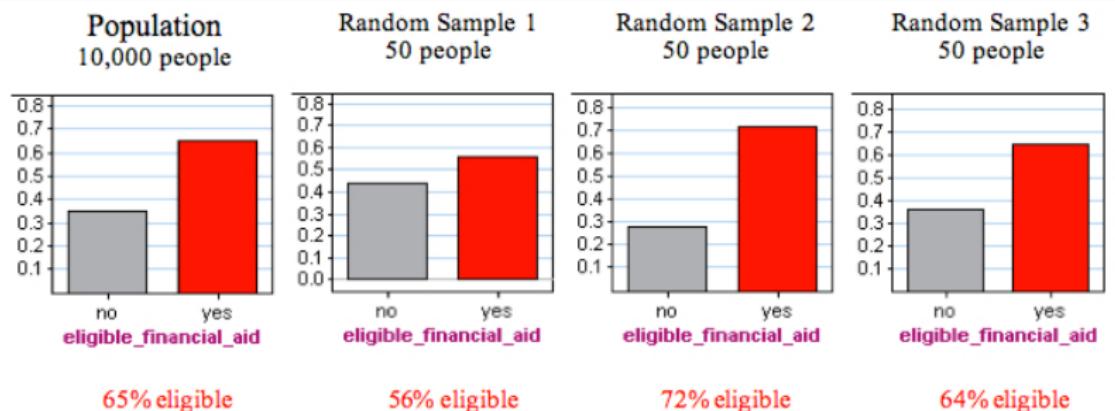
For Random Samples, Size Matters:

Let's compare the accuracy of random samples of different sizes.

✓ Suppose there are 10,000 students at your college. Also suppose that 65% of these students are eligible for financial aid. How accurate are random samples at predicting this population value?

Population and Sample XXIV

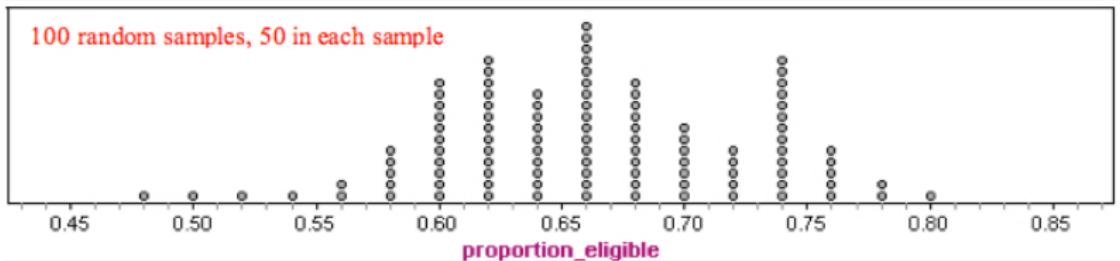
- ✓ To answer this question, we randomly select 50 students and determine the proportion who are eligible for financial aid. We repeat this several times. Here are the results for three random samples:



Population and Sample XXV

- ✓ Notice that each random sample has a different result. Some results are larger than the true population value of 65%; some results are smaller than the true population value.
- ✓ Because there is no bias in random samples, we expect results above and below the true value to occur with similar frequency.
- ✓ A simulation to take many more random samples.
- ✓ Again, each sample is composed of 50 randomly selected people.
- ✓ Here is a dotplot of the proportion who are eligible for financial aid in 100 samples. Each dot is a random sample.

Population and Sample XXVI

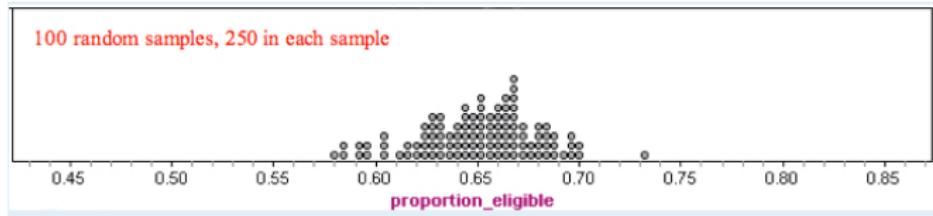


- ✓ We see that the results from random samples vary from 0.48 to 0.80. Typical values range from about 0.58 to 0.74.

Many samples have results below the true population value of 0.65, and many have results above 0.65. This shows that random samples are not biased. For the question Are you eligible for financial aid?, there is no systematic favoring of one response over another. The samples are representative of the population.

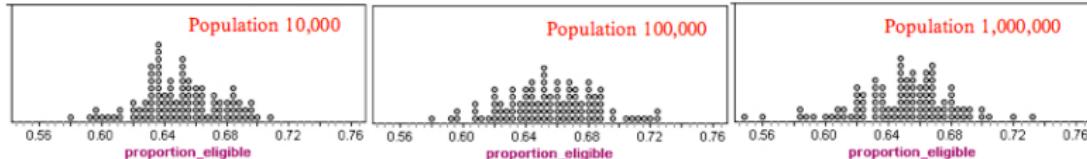
Population and Sample XXVII

What happens when we increase the number of people in the random sample?



- ✓ The precision of the sample results depends on the size of the sample, not the size of the population.

100 random samples, 250 in each sample



Population and Sample XXVIII

- ✓ Dotplots illustrate this point. The selected samples with 250 people in each sample, but we varied the size of the population. Each dotplot contains 100 samples.
- ✓ Notice that the sample results look very similar. For each population, the sample results fall between about 0.58 and 0.73. In each graph, it is common for sample results to fall between about 0.62 and 0.68.

Population and Sample XXIX

Example:

In California in 2000, the ballot included an initiative to add “none of the above” to the list of options in all candidate races. Prior to the election, a Field Poll indicated that support for the measure was 10% below opposition for the measure.

The poll was a telephone survey of 1000 registered voters in California. According to the Field Poll, “The survey was completed by telephone in English and Spanish using random digit dialing methods.”

A spokesperson for the initiative was critical of the poll because only 1000 people were surveyed. He pointed out that there are 23.5 million registered voters in California.

Is his criticism valid?

Population and Sample XXX

- ① No, because he was a spokesperson for the initiative, so he is biased.
- ② Yes, because it is impossible for 1,000 people to be representative of 23,500,000 people.
- ③ No, because the sample was randomly selected.
- ④ Yes, because a telephone survey will not represent the opinions of those without telephones.

Population and Sample XXXI

- ① No, because he was a spokesperson for the initiative, so he is biased.

It is always a good idea to consider the source of a critical statement. But his criticism is not valid because he is saying that the size of the population affects the accuracy of the sample. This is not correct when the sample is randomly chosen.

- ② Yes, because it is impossible for 1,000 people to be representative of 23,500,000 people.

His criticism is not valid because he is saying that size of the population affects the accuracy of the sample. This is not correct when the sample is randomly chosen.

Population and Sample XXXII

- ③ No, because the sample was randomly selected.

The poll used a random sampling method. So the sample is representative of the population. The size of the population does not affect the accuracy of a random sample.

- ④ Yes, because a telephone survey will not represent the opinions of those without telephones.

It is true that a telephone survey will not represent those without telephones. But there is no reason to assume that those without telephones will have a different opinion about this initiative. So the sampling method will not systematically skew the results. This situation is different from the 1936 presidential election. During that time period, many people did not have phones.

Population and Sample XXXIII

Those without phones were Democrats. So sampling from telephone lists produced a biased sample in that situation.

Population and Sample XXXIV

Would his criticism be valid if this was a national initiative and only 1000 people were randomly selected from all registered voters in the nation? Why or why not?

Population and Sample XXXV

Answer: Random sampling should produce a representative sample regardless of the population size. A random sample of 1,000 registered voters has the same level of accuracy representing California as the nation. Of course, we must make sure that the sample is randomly selected from the population of interest.

Population and Sample XXXVI

Comment: If an attempt is made to include every individual from a population in a sample, then the investigation is called a census. Every 10 years, the U.S. Census Bureau conducts a population census. It attempts to collect information about every person living in the United States. However, the population census misses between 1% and 3% of the U.S. population and accidentally counts some people more than once. A full census is possible only for small populations.

Population and Sample XXXVII

	Population	Sample
Distinction	The measurable quality is called a parameter	The measurable quality is called statistic
	The population is a complete set	The sample is subset of the population
	It contains all members of a specified group	It is a subset that represents the entire population
Symbols	N =population	n =sample size
	μ =population mean	x =sample mean
	σ =population standard deviation	s =sample standard deviation
	π =population percentage	p =sample percentage

According to E.R.Babble: A sample is a special subset of population that is observed for purposes of making inference about the nature of the total population itself.

Population and Sample XXXVIII

Common reasons for using samples in research:

- It is impractical to survey every member of a population
- Allows for inferences about the characteristics of a population
- Samples can be helpful representations of the population in question

Population and Sample XXXIX

- We draw a conclusion about the population on the basis of the sample.
- To draw a valid conclusion, the sample must be representative of the population. A representative sample is a subset of the population. It also reflects the characteristics of the population.
- A sample is biased if it systematically favors a certain outcome.
- Random selection eliminates bias.
- Larger samples tend to be more accurate than smaller samples if the samples are chosen randomly.
- The size of the population does not affect the accuracy of a random sample as long as the population is large.

Population and Sample XL

- If an attempt is made to include every individual from a population in a sample, then the investigation is called a census.

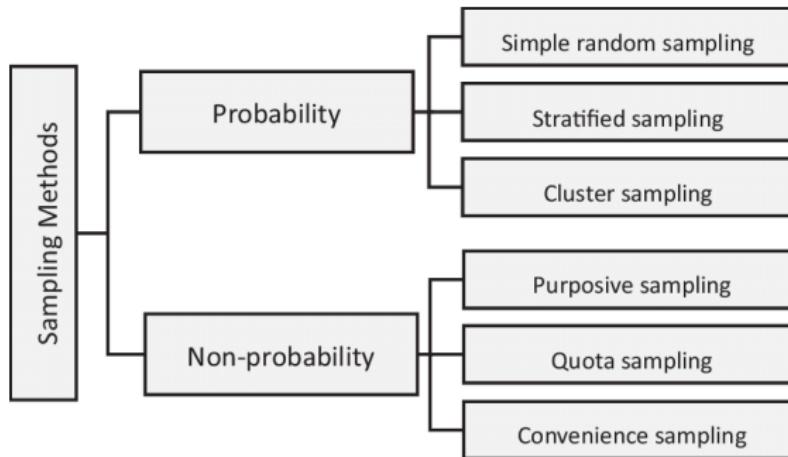
[https://courses.lumenlearning.com/
wm-concepts-statistics/chapter/sampling-2-of-2/](https://courses.lumenlearning.com/wm-concepts-statistics/chapter/sampling-2-of-2/)

[https://courses.lumenlearning.com/
wmopen-concepts-statistics/](https://courses.lumenlearning.com/wmopen-concepts-statistics/)

Population and Sample XLI

Type of sampling:

In psychological research and other types of social research, experiments typically rely on a few different sampling methods.



Population and Sample XLII

① **Probability Sampling:** probability sampling means that every individual in a population stands a chance of being selected.

- ✓ It ensures that every subset of the population has an equal chance of being represented in the sample.
- ✓ This makes probability samples more representative, and researchers are better able to generalize their results to the group as a whole.

Different types of probability sampling

- **Simple random sampling:** Researchers take every individual in a population and randomly select their sample, often using type of compute program or random number generator.

Population and Sample XLIII

- **Stratified random sampling:** It involves separating the population into subgroups and then taking a simple random sample from each of these subgroups.
 - ✓ *For example: research might divide the population up into subgroups based on race, gender, or age and then take a simple random sample of each of these groups.*
 - ✓ *Stratified random sampling often provides greater statistical accuracy than simple random sampling and helps ensure that certain groups are accurately represented in the sample.*

Population and Sample XLIV

- **Cluster sampling:** involves dividing a population into smaller clusters, often based upon geographic location or boundaries. A random sample of these clusters is then selected, and all of the subjects within the cluster are measured.
 *For example, imagine that you are trying to do a study on school principals in your state. Collecting data from every single school principal would be cost-prohibitive and time-consuming. Using a cluster sampling method, you randomly select five counties from your state and then collect data from every subject in each of those five counties.*

Population and Sample XLV

② **Nonprobability Sampling:** It involves selecting participants using methods that do not give every subset of a population an equal chance of being represented.

- **Convenience sampling:** involves using participants in a study who are convenient and easily available.
- **Purposive sampling:** involves seeking out individuals that meet certain criteria.

For example, marketers might be interested in learning how their products are perceived by women between the ages of 18 and 35. They might hire a market research firm to conduct telephone interviews that intentionally seek out and interview women that meet their age criteria.

Population and Sample XLVI

- **Quota sampling:** involves intentionally sampling specific proportions of each subgroup within a population.
✓ For example, political pollsters might be interested in researching the opinions of a population on a certain political issue. If they use simple random sampling, they might miss certain subsets of the population by chance. Instead, they establish criteria to assign each subgroup a certain percentage of the sample. Unlike stratified sampling, researchers use non-random methods to fill the quotas for each subgroup.

Population and Sample XLVII

Sampling Errors

- ✓ Sampling naturally cannot include every single individual in a population, therefore error can occur.
- ✓ **Differences between what is present in a population and what is present in a sample are known as sampling errors.**
- ✓ *In political polls, for example, you might often hear of the margin of errors expressed by certain confidence levels.*
- ✓ In general, the larger the sample size, the smaller the level of error.
- ✓ This is simply because as the sample becomes closer to reaching the size of the total population, the more likely it is to accurately capture all of the characteristics of the population.

Population and Sample XLVIII

- ✓ The only way to completely eliminate sampling error is to collect data from the entire population, which is often simply too cost-prohibitive and time-consuming.
- ✓ Sampling errors can be minimized, however, by using randomized probability testing and a large sample size.

Population and Sample XLIX

What Is Sample Size Formula?

- ✓ The sample size formula helps us find the accurate sample size through the difference between the population and sample.
- ✓ Since it is not possible to survey the whole population, we take a sample from the whole population and then conduct a survey or research.

The sample size formula is determined in two steps.

- ① Calculate the sample size for the infinite population, and
- ② Adjust the sample size to the required population.

Population and Sample L

Formula 1: Sample size for infinite population

$$S = Z^2 \times P \times \frac{(1-P)}{M^2},$$

- S= sample size of infinite population
- Z = Z score
- P= population proportion (Assume as 50% or 0.50)
- M = Margin of error

Formula 2: Adjusted sample size

$$\text{Adjusted Sample Size} = \frac{(S)}{1 + \frac{(S-1)}{\text{Population}}}$$

- ✓ Z score is determined based on the confidence level.

Population and Sample LI

- ✓ Confidence Level: Probability that the value of a parameter falls within a specified range of values. *For example: for 95% confidence level Z score is 1.960.*
- ✓ The margin of error: It is defined as a small amount that is allowed for in case of miscalculation or change of circumstances. Generally, the margin of error is taken as 5% or 0.05.

How to Apply Sample Size Formula?

✓ In order to calculate the required sample size, we need to find several other sets of values and then substitute them into an appropriate formula.

① **Step 1: Determining Key Value:** One of the key values to be determined is the population size which refers to the total number of people within the required demographic.

Population and Sample LIII

② Step 2: Determining the margin of error or confidence interval:

The margin of error is considered to be the amount of error that can be allowed in the study. The margin of error is actually a percentage that shows how close the sample results will be with respect to the true value of the overall population that is considered in the study.

- Usually, you can obtain more accurate answers with a smaller margin of error, but if a small margin of error is chosen, then you may require a larger sample.
- The margin of error usually is represented with a minus or a plus percentage when the results of a survey are presented.

Population and Sample LIV

✓ For instance, 35% of people choose option B, with a margin of error of +/- 5%. In this particular example, the margin of error actually indicates that, if the question was asked to the entire population, then you are confident that between 30% ($35 - 5$) and 40% ($35 + 5$) of the people will agree with option B.

③ **Step 3: Setting the confidence level:** The confidence level is pretty closely related to the margin of error or confidence interval. This value is used to measure the degree of certainty about how well a sample actually represents the entire population within the margin of error chosen for the study.

- When the confidence level is chosen as 95%, then this means that you can be 95% certain that the results will accurately fall within the margin of error chosen by you.

Population and Sample LV

- When a larger confidence level is chosen, it shows a greater degree of accuracy provided that the sample size is larger. Some of the most common confidence levels used in studies are 99% confident, 90% confident, and 95% confident.
- When the confidence level is set to 95%, then it shows that you are 95% confident that 30% to 40% of the total chosen population would definitely agree with option B of the survey.

④ **Step 4: Specifying the standard of deviation:** The standard of deviation shows how much variation can be expected from the responses of the study.

- Compared to the moderate results, you can expect extreme answers to be more accurate.

Population and Sample LVI

- Consider an example where 1% of the survey responses says "No", and then 99% answer "Yes", then it means that the sample actually represents the overall population in an accurate manner.
- In another case, if 55% answer "No" and 45% answer "Yes," then this means that there could be a greater chance of error.

✓ *Since this value is difficult to be calculated in an actual survey, most people choose to use 0.5 (50%) as the value which is actually the worst-case scenario percentage. Thus, using this value will actually guarantee that the calculated sample size is huge enough to show the overall population within the confidence level and the confidence interval in an accurate manner.*

Population and Sample LVII

⑤ **Step 5: Finding the Z-score:** The Z-score can be considered as a constant value that is set automatically depending on the confidence level.

- ✓ Z-score shows the number of standard deviations or the standard normal score between the average/mean of the population and any selected value.
- ✓ Due to the fact that the confidence levels are all standardized, most researchers actually memorize the required z-score for most of the commonly used confidence levels:

Confidence Level	Z-score
80%	1.28
85%	1.44
90%	1.65
95%	1.96
99%	2.58

Population and Sample LVIII

Using the Standard Formula:

✓ If the size of the population is small to moderate, then it is easier to know all the key values and thus the standard formula can be used. The standard formula for calculating the sample size is:

Formula 2: Adjusted sample size

$$\text{Sample size} = \frac{[Z^2 \times P(1-P)]}{e^2 \times N},$$

- N is the population size
- Z is the Z -score
- e is the margin of error
- P is the standard of deviation

Population and Sample LIX

**Example 1: Calculate the sample size for a population of 100000.
Take confidence level as 95% and margin of error as 5%.**

Solution

Given: Z=1.960, P=0.5, M=0.05

$$\begin{aligned} S &= (1.960)^2 \times 0.5 \times \frac{(1-0.5)}{(0.05)^2} \\ &= \frac{3.8416 \times 0.25}{0.0025} \end{aligned}$$

$S = 384.16$ The sample size of infinite population is 384.16

Population and Sample LX

Example 2 : Using the sample size formula, adjust the sample size for the required population in solved example 1.

Solution

Given: Z = 1.960, P = 0.5, M = 0.05

Using sample size formula for adjusted sample size,

$$\text{Adjusted Sample Size} = \frac{384.16}{1 + \frac{384.16 - 1}{100000}}$$

Required sample size = 382.69 or 383

Population and Sample LXI

Example 3: Using the Sample Size Formula, find the sample size for a survey where confidence level = 95%, standard deviation = .5, and margin of error = +/- 5%.

Solution

$$\begin{aligned} S &= \frac{(Z\text{-score})^2 \times SD \times (1 - SD)}{(\text{margin of error})^2} \\ S &= \frac{(1.96)^2 \times 0.5 \times (0.5)}{(0.05)^2} \\ &= \frac{3.8416}{0.0025} \\ &= \frac{0.9604}{0.0025} \\ &= 384.16 \end{aligned}$$

Uniform Distribution I

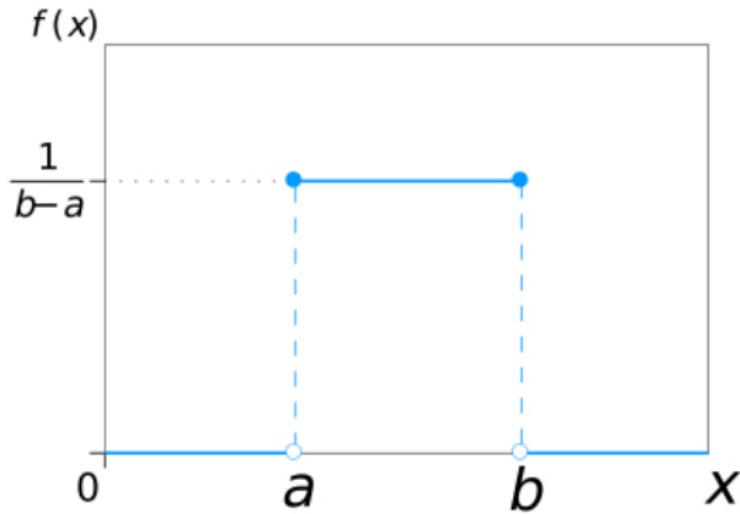
- ✓ Perhaps one of the simplest and useful distribution is the uniform distribution. The probability distribution function of the continuous uniform distribution is:

$$f(x) = \begin{cases} \frac{1}{(b-a)}, & \text{for } a \leq x \leq b \\ 0, & x < a \text{ or } x > b \end{cases}$$

- ✓ Since any interval of numbers of equal width has an equal probability of being observed, the curve describing the distribution is a rectangle, with constant height across the interval and 0 height elsewhere.
- ✓ Since the area under the curve must be equal to 1, the length of the interval determines the height of the curve. The following figure shows

Uniform Distribution II

a uniform distribution in interval (a, b) . Notice since the area needs to be 1. The height is set to $\frac{1}{(b-a)}$.

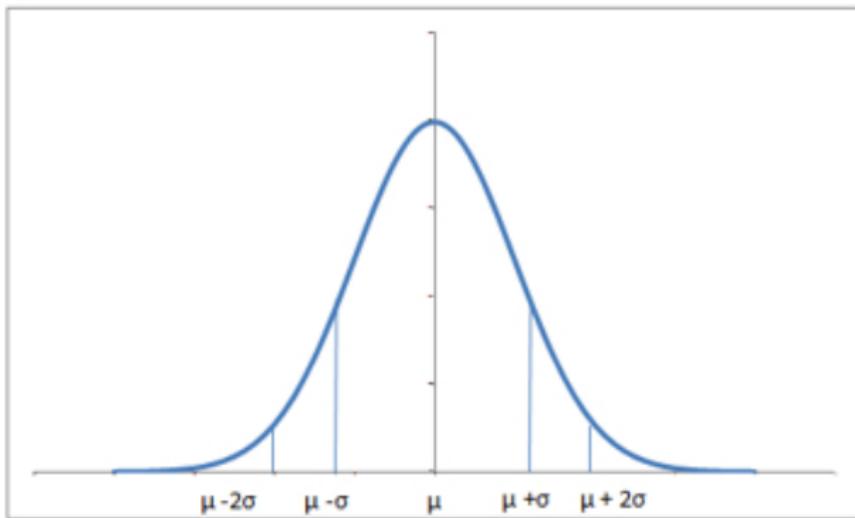


Normal Distribution I

- ✓ Normal Distribution, also known as Gaussian distribution, is ubiquitous in Data Science.
- ✓ You will encounter it at many places especially in topics of statistical inference.
- ✓ It is one of the assumptions of many data science algorithms too.
- ✓ A normal distribution has a bell-shaped density curve described by its mean μ and standard deviation σ .
- ✓ The density curve is symmetrical, centered about its *mean*, with its spread determined by its standard deviation showing that data near the mean are more frequent in occurrence than data far from the mean.
- ✓ The probability distribution function of a normal density curve with mean μ and standard deviation σ at a given point x is given by:

Normal Distribution II

$$f(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

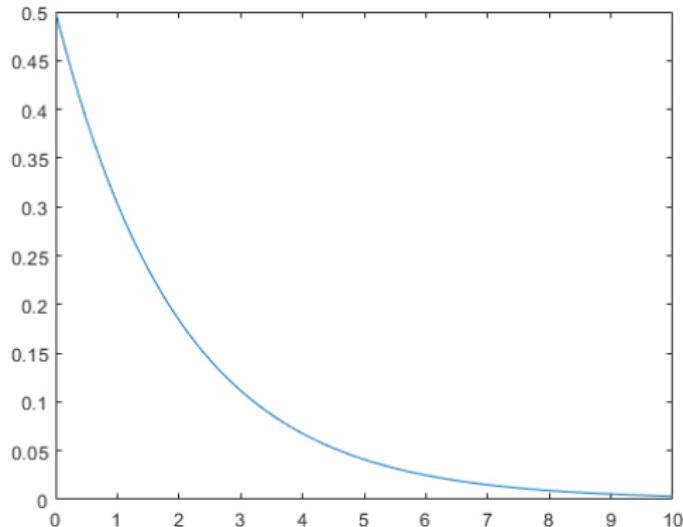


Exponential Distribution Function I

- ✓ The exponential distribution describes the time between events in a Poisson point process, i.e., a process in which events occur continuously and independently at a constant average rate.
- ✓ It has a parameter λ called rate parameter, and its equation is described as :

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Exponential Distribution Function II

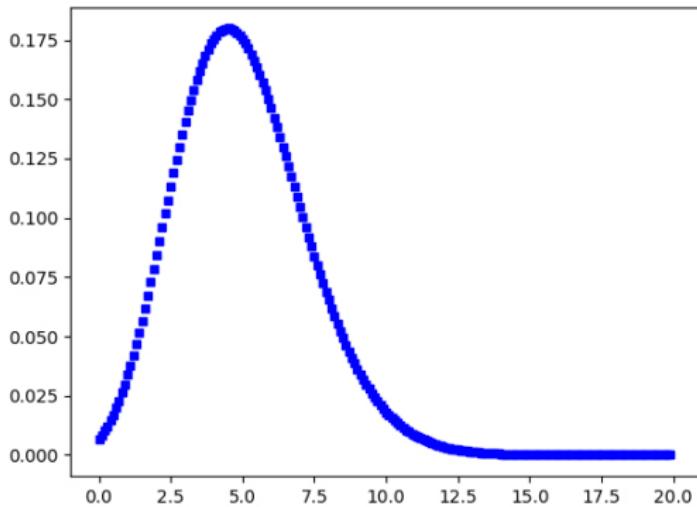


Poisson Distribution I

- ✓ Poisson random variable is typically used to model the number of times an event happened in a time interval.
- ✓ For example, the number of users visited on a website in an interval can be thought of a Poisson process.
- ✓ Poisson distribution is described in terms of the rate (μ) at which the events happen.
- ✓ An event can occur 0, 1, 2, ... times in an interval. The average number of events in an interval is designated λ .
- ✓ λ is the event rate, also called the rate parameter.
- ✓ The probability of observing k events in an interval is given by the equation:

Poisson Distribution II

$$P(k \text{ events in interval}) = e^{-\lambda} \frac{\lambda^K}{k!}$$



Hypothesis Test for a Population Mean I

Conduct and interpret results from a hypothesis test about a population mean.

- Recognize when to use a hypothesis test or a confidence interval to draw a conclusion about a population mean.
- Under appropriate conditions, conduct a hypothesis test about a population mean. State a conclusion in context.

Inference for Means: focus is on inference when the variable is quantitative. here parameters and statistics are means.

Estimating a Population Mean: learned how to use a sample mean to calculate a confidence interval. The confidence interval estimates a population mean.

CS322: Deep Learning

Linear Algebra

Sachchida Nand Chaurasia
Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



December 28, 2021

Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis



Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis



Basic Notations I

- a : A scalar(integer or real)
- \mathbf{a} : A vector
- \mathbf{A} : A matrix
- \mathbf{A} : A tensor
- I_n : Identity matrix with n row and n columns
- I : Identity matrix with dimensionality implied by context
- $e^{(i)}$: standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
- $\text{diag}(\mathbf{a})$: A square, diagonal matrix with diagonal entities given by \mathbf{a}
- a : A scalar random variable



Basic Notations II

- \mathbf{a} : A vector-valued random variable
- \mathbf{A} : A matrix-valued random variable
- \mathbb{A} : A set
- \mathbb{R} : The set of real numbers
- $\mathbb{A} \setminus \mathbb{B}$: Set subtraction
- \mathcal{G} : A graph
- a_i : Element i of vector \mathbf{a} , with indexing starting at 1
- a_{-i} : All elements of vector \mathbf{a} except for element i
- $A_{i,j}$: Element i, j of matrix \mathbf{A}



Basic Notations III

- $A_{i,:}$: Row i of matrix **A**
- $A_{:,i}$: Column i of matrix **A**
- $A_{i,j,k}$: Element i, j, k of a 3-D tensor A
- $A(:,:,i)$: 2-D slice of a 3-D tensor
- a_i : Element i of the random vector **a**
- \mathbf{A}^T : Transpose of matrix **A**
- $A \odot B$: Element-wise (Hadamard) product of **A** and **B**
- $\det(\mathbf{A})$: Determinant of A
- $\frac{dy}{dx}$: Derivative of y with respect to x



Basic Notations IV

- $\frac{\partial y}{\partial x}$: Partial derivative of y with respect to x
- $\nabla_{\mathbf{x}}y$: Gradient derivative with respect to x
- $\nabla_{\mathbf{X}}y$: Matrix derivatives of y with respect to X
- ∇_Xy : Tensor containing derivatives of y with respect X
- $\frac{\partial f}{\partial x}$: Jacobian matrix $J \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
- $\nabla_x^2 f(\mathbf{x})$ or $H(f)(\mathbf{x})$: The Hessian matrix of f at input point x
- $\int f(\mathbf{x})d\mathbf{x}$: Definite integral over the entire domain of x
- $\int_{\mathbb{S}} f(\mathbf{x})d\mathbf{x}$: Definite integral with respect to x over the set \mathbb{S}
- $a \perp b$: The random variables a and b are independent



Basic Notations V

- $a \perp b | c$: They are conditionally independent given c
- $P(a)$: A probability distribution over a discrete variable
- $p(a)$: A probability distribution over a continuous variable, or over a variable whose type has not been specified
- $a \sim P$: Random variable a has distribution P
- $\mathbb{E}_{x \sim P}[f(x)]$ or $\mathbb{E}f(x)$: Expectation of $f(x)$ with respect to $P(x)$
- $\text{Var}(f(x))$: Variance of $f(x)$ under $P(x)$
- $\text{Cov}(f(x), g(x))$: Covariance of $f(x)$ and $g(x)$ under $P(x)$
- $\mathcal{N}(x; \mu, \Sigma)$: Gaussian distribution over x with mean μ and covariance Σ



Basic Notations VI

- $f : \mathbb{A} \rightarrow \mathbb{B}$: The function f with domain \mathbb{A} and range \mathbb{B}
- $f \circ g$: Composition of the function f and g
- $f(\mathbf{x}; \theta)$: A function of \mathbf{x} parameterized by θ
- $\sigma(x)$: Logistic sigmoid, $\frac{1}{1+\exp(-x)}$
- $\|\mathbf{x}\|_p$: L^p norm of \mathbf{x}
- $\|\mathbf{x}\|$: L^2 norm of \mathbf{x}

Datasets and Distributions

- p_{data} : The data generating distribution
- \mathbb{X} ; A set of training examples



Basic Notations VII

- $\mathbf{x}^{(i)}$: The i -th example (input) from dataset
- $y^{(i)}$ or $\mathbf{y}^{(i)}$: The target associated with $\mathbf{x}^{(i)}$ for supervised learning
- \mathbf{X} : The $m \times n$ matrix with example $\mathbf{x}^{(i)}$ in row $\mathbf{X}_i, :$



Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

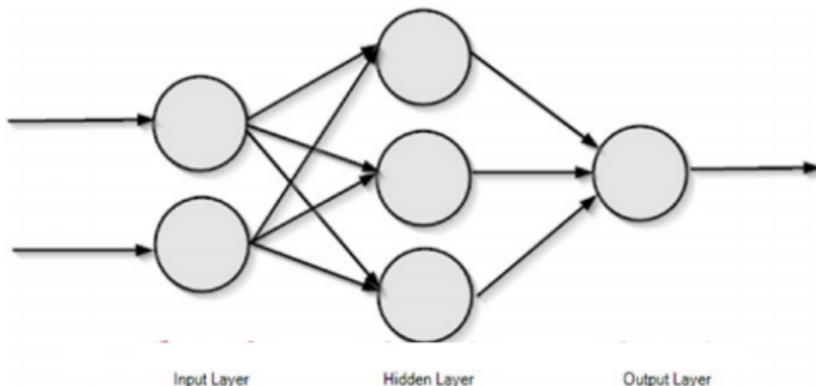
Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis



Linear Algebra in the context of deep learning I



- ✓ The image above shows a simple feed forward neural network that propagates information forward.

The image is a beautiful representation of the neural network, but how a computer can understand this?

Linear Algebra in the context of deep learning II

- ✓ In a computer, the layers of the neural network are represented as **vectors**.
- ✓ Consider the input layer as X and the hidden layer as H .
- ✓ The above image shows the operations needed to compute the output for the first and the only hidden layer of the above neural network.
- ✓ Every single column of the network are vectors. Vectors are dynamic arrays that are a collection of data(or features).
- ✓ In the current neural network, the vector x holds the input.
- ✓ The hidden layer H 's output is calculated by performing $H = f(Wx + b)$.
- ✓ Here W is called as the Weight matrix, b is called bias and f is the activation function.
- ✓ The first component is $W \cdot x$; this is a **matrix-vector product**, because W is a matrix and x is a vector.

Linear Algebra in the context of deep learning III

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ vector (input layer)}$$

$$W = \begin{bmatrix} w_1 & w_2 \\ w_4 & w_5 \\ x_3 & w_6 \end{bmatrix} \text{ matrix (the weights for hidden layer 1)}$$

the output is given by

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \\ w_4 & w_5 \\ x_3 & w_6 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ (the product of vector and matrices)}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} + \text{bias}$$

finally,

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = f\left(\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right) \text{ (activation step)}$$



Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis



Scalars I

Scalars: A scalar is just a single number, in contrast to most of the other objects studied in linear algebra, which are usually arrays of multiple numbers.

- ✓ Usually gives scalars lowercase italic variable names.
- ✓ When we introduce them, we specify what kind of number they are. For example, we might say “Let $s \in \mathbb{R}$ be the slope of the line,” while defining a real-valued scalar, or “Let $n \in \mathbb{N}$ be the number of units,” while defining a natural number scalar.



Vectors I

Vectors : A vector is an array of numbers. The numbers are arranged in order. We can identify each individual number by its index in that ordering.

- ✓ Typically we give vectors lowercase names in bold typeface, such as \mathbf{x} .
- ✓ The elements of the vector are identified by writing its name in italic typeface, with a subscript.
- ✓ The first element of \mathbf{x} is x_1 , the second element is x_2 , and so on. We also need to say what kind of numbers are stored in the vector.
- ✓ If each element is in \mathbb{R} , and the vector has n elements, then the vector lies in the set formed by taking the Cartesian product of \mathbb{R}_n times, denoted as \mathbb{R}^n .
- ✓ When we need to explicitly identify the elements of a vector, we write them as a column enclosed in square brackets:

Vectors II

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- ✓ We can think of vectors as identifying points in space, with each element giving the coordinate along a different axis.
- ✓ Sometimes we need to index a set of elements of a vector. In this case, we define a set containing the indices and write the set as a subscript.
- ✓ For example, to access x_1 , x_3 and x_6 , we define the set $\mathbb{S} = \{1, 3, 6\}$ and write $x_{\mathbb{S}}$.
- ✓ We use the - sign to index the complement of a set.
- ✓ For example x_{-1} is the vector containing all elements of x except for x_1 , and $x_{-\mathbb{S}}$ is the vector containing all elements of x except for x_1 , x_3 and x_6 .

Vectors III

A Basis for a Vector Space:

- ✓ Let V be a subspace of \mathbb{R}^n for some n . A collection $B = \{v_1, v_2, \dots, v_r\}$ of vectors from V is said to be a basis for V if B is linearly independent and spans V .
- ✓ If either one of these criterial is not satisfied, then the collection is not a basis for V . If a collection of vectors spans V , then it contains enough vectors so that every vector in V can be written as a linear combination of those in the collection.
- ✓ If the collection is linearly independent, then it doesn't contain so many vectors that some become dependent on the others.

Example 1: The collection $\{i, j\}$ is a basis for \mathbb{R}^2 , since it spans \mathbb{R}^2 and the vectors i and j are linearly independent (because neither is a multiple of the other).

Vectors IV

- ✓ This is called the standard basis for \mathbb{R}^2 . Similarly, the set $\{i, j, k\}$ is called the standard basis for \mathbb{R}^3 , and, in general,

$$e_1 = (1, 0, 0, \dots, 0) \quad (1)$$

$$e_2 = (0, 2, 0, \dots, 0) \quad (2)$$

$$\dots e_n = (0, 0, 0, \dots, 1) \quad (3)$$

is the standard basis for \mathbb{R}^2 .

Example 2: The collection $\{i, i + j, 2j\}$ is not a basis for \mathbb{R}^2 . Although it spans \mathbb{R}^2 , it is not linearly independent. No collection of 3 or more vectors from \mathbb{R}^2 can be independent.

Vectors V

Example 3: The collection $\{i + j, j + k\}$ is not a basis for \mathbb{R}^3 . Although it is linearly independent, it does not span all of \mathbb{R}^3 . For example, there exists no linear combination of $i + j$ and $j + k$ that equals $i + j + k$.

Matrix I

Matrices: A matrix is a 2-D array of numbers, so each element is identified by two indices instead of just one.

- ✓ We usually give matrices uppercase variable names with bold typeface, such as \mathbf{A} . If a real-valued matrix A has a height of m and a width of n , then we say that $A \in \mathbb{R}^{m \times n}$.
- ✓ We usually identify the elements of a matrix using its name in italic but not bold font, and the indices are listed with separating commas.
- ✓ For example, $A_{1,1}$ is the upper left entry of \mathbf{A} and $A_{m,n}$ is the bottom right entry.
- ✓ We can identify all the numbers with vertical coordinate i by writing a “ $:$ ” for the horizontal coordinate. For example, $A_{:,i}$ denotes the horizontal cross section of \mathbf{A} with vertical coordinate i . This is known as the $i - th$ row of \mathbf{A} . Likewise, $\mathbf{A}_{:,i}$ is the $i - th$ **column** of \mathbf{A} .

Matrix II

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

- ✓ For example, $f(A)_{i,j}$ gives element (i, j) of the matrix computed by applying the function f to \mathbf{A} .

Transpose: One important operation on matrices is the transpose.

- ✓ The transpose of a matrix is the mirror image of the matrix across a diagonal line, called the main diagonal, running down and to the right, starting from its upper left corner.

$$(\mathbf{A}^T)_{i,j} = \mathbf{A}_{j,i} \tag{4}$$

- ✓ The transpose of a matrix product has a simple form:

$$(AB)^T = B^T A^T \tag{5}$$

Matrix III

The Rank of a Matrix:

- ✓ The maximum number of linearly independent rows in a matrix A is called the row rank of A , and the maximum number of linearly independent columns in A is called the column rank of A .
- ✓ If A is an m by n matrix, that is, if A has m rows and n columns, then :

$$\text{row rank of } A \leq m \quad (6)$$

$$\text{column rank of } A \leq n \quad (7)$$

$$\text{rank}(A_{m \times n}) \leq \min(m, n) \quad (8)$$

- ✓ A vector r is said to be linearly independent of vectors r_1 and r_2 if it cannot be expressed as a linear combination of r_1 and r_2 .

Matrix IV

$$r \neq ar_1 + br_2$$

$$A = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 6 & 14 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 6 & 14 \end{bmatrix} \quad C = A = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 6 & 14 \end{bmatrix}$$

- In matrix A, row r₂ is a multiple of r₁, r₂ = 2 r₁, so it has only one independent row. Rank(A) = 1
- In matrix B, row r₃ is a sum of r₁ and r₂, r₃ = r₁ + r₂, but r₁ and r₂ are independent. Rank(B) = 2
- In matrix C, all 3 rows are independent of each other. Rank(C) = 3

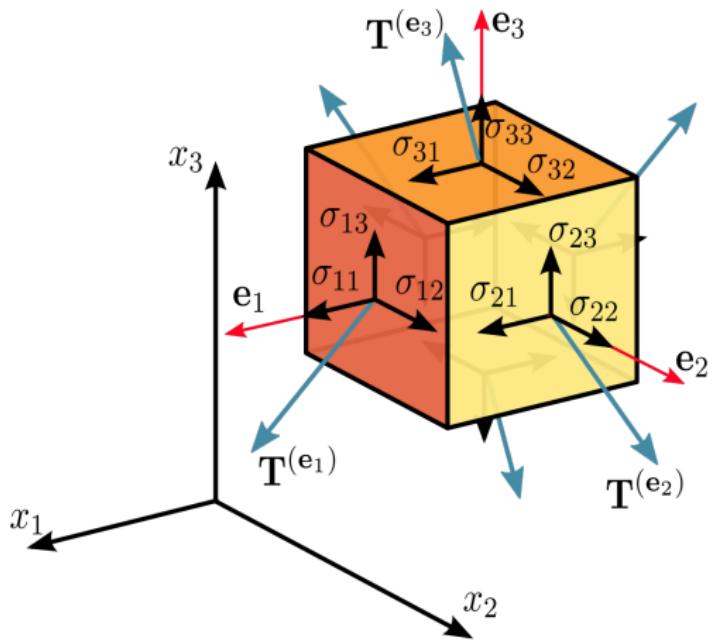
The rank of a matrix can be thought of as a representative of the amount of unique information represented by the matrix. Higher the rank, higher the information.

Tensors I

Tensors : In some cases we will need an array with more than two axes.

- ✓ In the general case, an array of numbers arranged on a regular grid with a variable number of axes is known as a tensor.
- ✓ We denote a tensor named “ \mathbf{A} ” with this typeface: bfa . We identify the element of \mathbf{A} at coordinates (i, j, k) by writing $\mathbf{A}_{i,j,k}$.

Tensors II



Tensors III

Scalar Vector Matrix Tensor

1

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$



Tensors IV

A tensor is an N-dimensional array of data



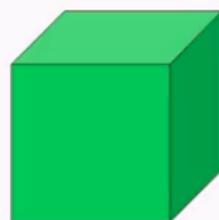
Rank 0
Tensor
scalar



Rank 1
Tensor
vector



Rank 2
Tensor
matrix



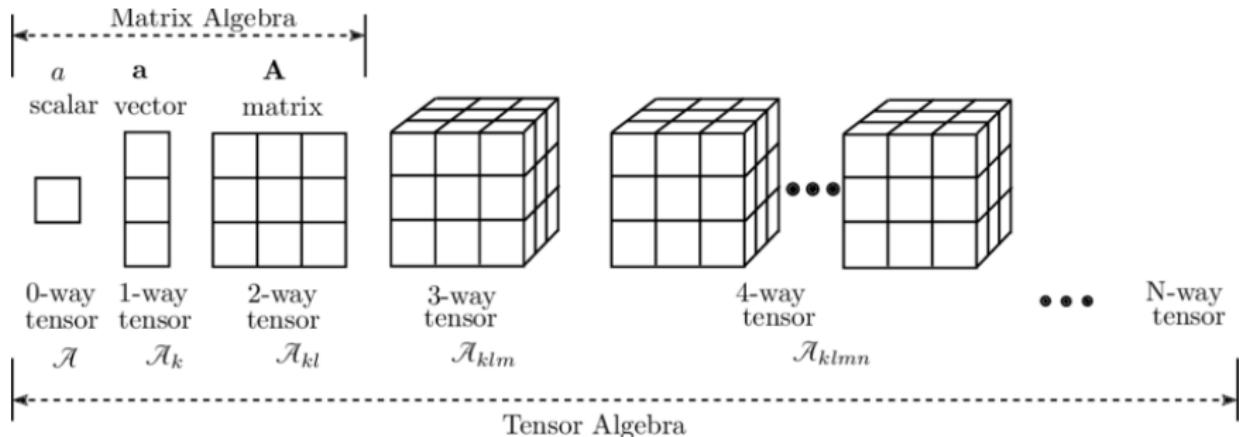
Rank 3
Tensor



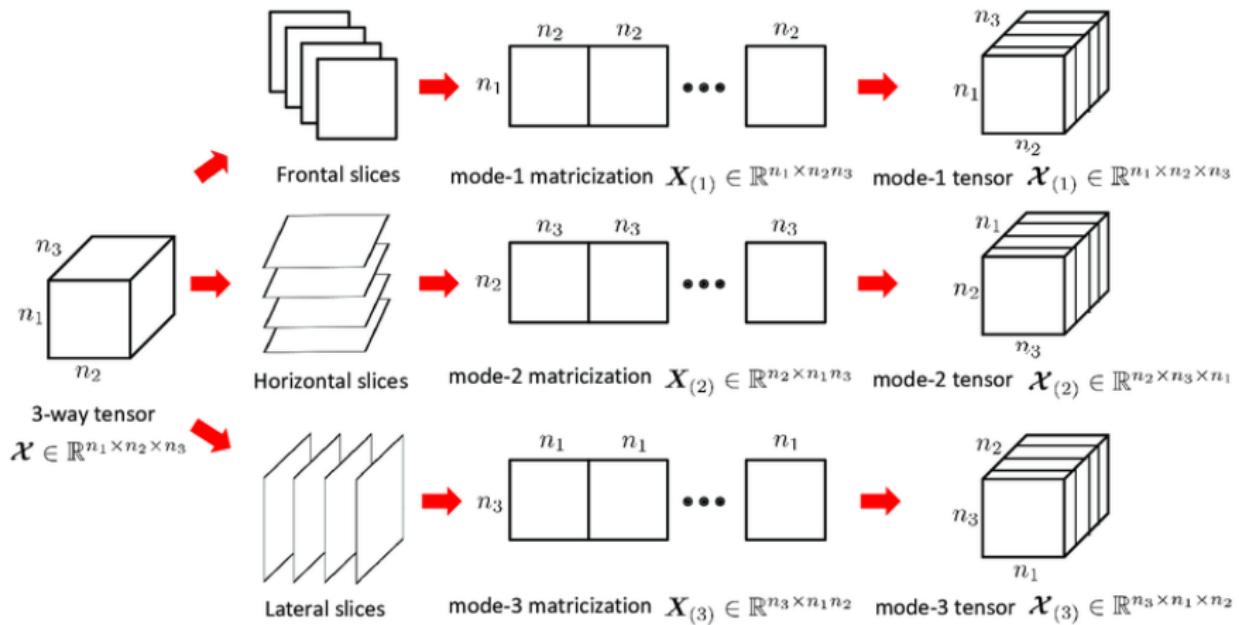
Rank 4
Tensor



Tensors V



Tensors VI



Multiplying Matrices and Vectors I

$A_{m \times n}$ and $B_{n \times p}$, then C is shape of $C_{m \times p}$

$$C = ABC_{i,j} = \sum_b A_{i,j}B_{k,j} \quad (9)$$

$$C_{i,j} = \sum_b A_{i,j}B_{k,j} \quad (10)$$

- ✓ The standard product of two matrices called the element-wise product or Hadamard product, and denoted as $A \odot B$.
- ✓ The dot product between two vectors x and y of the same dimensionality is the matrix product $x^T y$.

Multiplying Matrices and Vectors II

- ✓ Matrix product operations have many useful properties that make mathematical analysis of matrices more convenient. For example, matrix multiplication is distributive:

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC} \quad (\text{Distributive}) \quad (11)$$

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} \quad (\text{Associative}) \quad (12)$$

- ✓ Matrix multiplication is not commutative (the condition $\mathbf{AB} = \mathbf{BA}$ does not always hold), unlike scalar multiplication.
- ✓ However, the dot product between two vectors is commutative:

$$\mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x}. \quad (13)$$

System of linear equations I

$$A\mathbf{x} = \mathbf{b} \quad (14)$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a known matrix,

$\mathbf{b} \in \mathbb{R}^m$ is a known vector, and

$\mathbf{x} \in \mathbb{R}^n$ is a vector of unknown variables that need to be solved for.

Each element x_i of \mathbf{x} is one of these unknown variables.

Each row of \mathbf{A} and each element of \mathbf{b} provide another constraint.

System of linear equations II

Equation (14) can be a form of as:

$$A1,: x = b_1 \quad (15)$$

$$A2,: x = b_2 \quad (16)$$

$$A3,: x = b_3 \quad (17)$$

$$\dots \quad (18)$$

$$A_m,: x = b_m \quad (19)$$

System of linear equations III

- ✓ More explicitly as:

$$A_{1,1}x_1 + A_{1,2}x_2 + \dots + A_{1,n}x_n = b_1 \quad (20)$$

$$A_{2,1}x_1 + A_{2,2}x_2 + \dots + A_{2,n}x_n = b_2 \quad (21)$$

$$\dots \quad (22)$$

$$A_{m,1}x_1 + A_{m,2}x_2 + \dots + A_{m,n}x_n = b_m \quad (23)$$

- ✓ Matrix-vector product notation provides a more compact representation for equations of this form.

Identity and Inverse Matrices I

- ✓ Linear algebra offers a powerful tool called matrix inversion that enables us to analytically solve $Ax = b$ (Equation (14)) for many values of \mathbf{A} .
- ✓ An identity matrix is a matrix that does not change any vector when we multiply that vector by that matrix.
- ✓ We denote the identity matrix that preserves n -dimensional vectors as I_n . Formally, $I_n \in \mathbb{R}^{n \times n}$, and

$$\forall x \in \mathbb{R}^n, I_n x = x \quad (24)$$

$$A^{-1} A = I_n \quad (25)$$



Identity and Inverse Matrices II

$$Ax = b \quad (26)$$

$$A^{-1}Ax = A^{-1}b \quad (27)$$

$$I_n x = A^{-1}b \quad (28)$$

$$x = A^{-1}b \quad (29)$$

- ✓ When A^{-1} exists, several different algorithms can find it in closed form.
- ✓ In theory, the same inverse matrix can then be used to solve the equation many times for different values of b .
- ✓ A^{-1} is primarily useful as a theoretical tool, however, and should not actually be used in practice for most software applications.

Identity and Inverse Matrices III

- ✓ Because A^{-1} can be represented with only limited precision on a digital computer, algorithms that make use of the value of b can usually obtain more accurate estimates of x .



Liner Combination, Dependence and Span I

Linear Combinations

Let v_1, v_2, \dots, v_r be vectors in \mathbb{R}^n . A linear combination of these vectors is any expression of the form:

$$k_1v_1 + k_2v_2, \dots, k_rv_r \quad (30)$$

where the coefficients k_1, k_2, \dots, k_r are scalars.

Example : The vector $v = (-7, -6)$ is a linear combination of the vectors $v_1 = (-2, 3)$ and $v_2 = (1, 4)$, since $v = 2v_1 - 3v_2$.

The zero vector is also a linear combination of v_1 and v_2 , since $0 = 0v_1 + 0v_2$. In fact, it is easy to see that the zero vector in \mathbb{R}^n is always a linear combination of any collection of vectors v_1, v_2, \dots, v_r from \mathbb{R}^n .

Liner Combination, Dependence and Span II

Span

- ✓ The set of all linear combinations of a collection of vectors v_1, v_2, \dots, v_r from \mathbb{R}^n is called the span of $\{v_1, v_2, \dots, v_r\}$.
- ✓ This set, denoted $\text{span}\{v_1, v_2, \dots, v_r\}$, is always a subspace of \mathbb{R}^n , since it is clearly closed under addition and scalar multiplication (because it contains all linear combinations of v_1, v_2, \dots, v_r). If $V = \text{span}\{v_1, v_2, \dots, v_r\}$, then V is said to be spanned by v_1, v_2, \dots, v_r .
- ✓ **span** of a set of vector is the set of all points obtained by linear combination of the original vectors.

Liner Combination, Dependence and Span III

- ✓ A^{-1} to exist, equation $Ax = b$ must have exactly one solution for every value of b .
- ✓ It is also possible for the system of equations to have no solutions or infinitely many solutions for some values of b .
- ✓ It is not possible, however, to have more than one but less than infinitely many solutions for a particular b ; if both x and y are solutions, then

$$z = \alpha x + (1 - \alpha)y \quad (31)$$

- ✓ It has solution for any real α .
- ✓ To analyze how many solutions the equation has, think of the columns of **A** as specifying different directions we can travel in from the **origin** (the point

Liner Combination, Dependence and Span IV

specified by the vector of all zeros), then determine how many ways there are of reaching b .

- ✓ In this view, each element of x specifies how far we should travel in each of these directions, with x_i specifying how far to move in the direction of column i :

$$Ax = \sum_i x_i A_{:,i} \quad (32)$$

- ✓ In general, this kind of operation is called a **linear combination**.
- ✓ A linear combination of some set of vectors $\{v^{(1)}, \dots, v^{(n)}\}$ is given by multiplying each vector $v^{(i)}$ by a corresponding scalar coefficient and adding results:

Liner Combination, Dependence and Span V

$$\sum_i c_i v^{(i)} \quad (33)$$

✓ Let $A = \{v_1, v_2, \dots, v_r\}$ be a collection of vectors from \mathbb{R}^n . If $r > 2$ and at least one of the vectors in A can be written as a linear combination of the others, then A is said to be **linearly dependent**.



Liner Combination, Dependence and Span VI

Linearly Independent Vectors #1

a) Are $\begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -4 \\ 6 \\ 5 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ linearly independent?

⇒ Independent if $\alpha_1 \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} -4 \\ 6 \\ 5 \end{bmatrix} + \alpha_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

has only the trivial solution $\alpha_1 = \alpha_2 = \alpha_3 = 0$

$$\Rightarrow \left[\begin{array}{cccc} 2 & -4 & 1 & 0 \\ 2 & 6 & 0 & 0 \\ 1 & 5 & 0 & 0 \end{array} \right] \xrightarrow{\text{r.r.}} \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right] \Rightarrow \alpha_1 = 0 \\ \alpha_2 = 0 \\ \alpha_3 = 0$$

Augmented
matrix

⇒ Only solution is
the trivial solution ⇒ Independent



Liner Combination, Dependence and Span VII

b) Are $\begin{bmatrix} z \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -4 \\ 6 \\ 5 \end{bmatrix}, \begin{bmatrix} -2 \\ 8 \\ 6 \end{bmatrix}$ linearly independent?

Independent if $x_1 \begin{bmatrix} z \\ 2 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} -4 \\ 6 \\ 5 \end{bmatrix} + x_3 \begin{bmatrix} -2 \\ 8 \\ 6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

has only the trivial solution $x_1 = x_2 = x_3 = 0$

$$\Rightarrow \begin{bmatrix} z & -4 & -2 & 0 \\ 2 & 6 & 8 & 0 \\ 1 & 5 & 6 & 0 \end{bmatrix} \xrightarrow{\text{r.r.}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{array}{l} x_1, x_2 \text{ basic var} \\ x_3 \text{ free} \end{array}$$

$$\Rightarrow \begin{array}{l} X_1 = -X_3 \\ X_2 = -X_3 \\ X_3 \text{ free} \end{array}$$

More than just the
trivial solution

Vectors are linearly dependent.



Liner Combination, Dependence and Span VIII

Example 1: Are the vectors $v_1 = (2, 5, 3)$, $v_2 = (1, 1, 1)$, and $v_3 = (4, -2, 0)$ linearly independent?

Answer:

- ✓ If none of these vectors can be expressed as linear combination of the other two, then the vectors are independent; otherwise, they are dependent.
- ✓ If, for example, v_3 were a linear combination of v_1 and v_2 , then there would exist scalar k_1 and k_2 such that

$$k_1 v_1 + k_2 v_2 = v_3$$

$$k_1(2, 5, 3) + k_2(1, 1, 1) = (4, -2, 0)$$

Liner Combination, Dependence and Span IX

$$2k_1 + k_2 = 4$$

$$5k_1 + k_2 = -2$$

$$3k_1 + k_2 = 0$$

- ✓ This is an inconsistent system. Subtracting the first equation from the third yields $k_1 = -4$, and substituting this value into either the first or third equation gives $k_2 = 12$.
- ✓ $(k_1, k_2) = (-4, 12)$ does not satisfy the second equation.
- ✓ The conclusion is that v_3 is not a linear combination of v_1 and v_2 .
- ✓ **What are the other observations?**

Liner Combination, Dependence and Span X

Second definition

A collection of vectors v_1, v_2, \dots, v_r from \mathbb{R}^n is **linearly independent** if the only scalars that satisfy $k_1v_1 + k_2v_2 + \dots + k_rv_r = 0$ are

$k_1 = k_2 = \dots = k_r = 0$. This is called the **trivial linear combination**. If, on the other hand, there exists a nontrivial linear combination that gives the zero vector, then the **vectors are dependent**.



Liner Combination, Dependence and Span XI

Example 2: Use this second definition to show that the vectors from Example 1, $v_1 = (2, 5, 3)$, $v_2 = (1, 1, 1)$, and $v_3 = (4, -2, 0)$ -are linearly independent.

Answer: Vectors $v_1 = (2, 5, 3)$, $v_2 = (1, 1, 1)$, and $v_3 = (4, -2, 0)$ are linearly independent if the only scalars that satisfy

$$k_1 v_1 + k_2 v_2 + k_3 v_3 = 0 \quad (34)$$

are

$$k_1 = k_2 = k_3 = 0 \quad (35)$$

$$\begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = 0 \quad (36)$$

Liner Combination, Dependence and Span XII

$$\left[\begin{array}{ccc|c} | & | & | & \\ v_1 & v_2 & v_3 & \\ | & | & | & \end{array} \right] \left[\begin{array}{ccc|c} 2 & 1 & 4 & \\ 1 & 1 & 1 & \\ 3 & 1 & 0 & \end{array} \right] \xrightarrow{-2r_1 \text{ added to } r_2} \left[\begin{array}{ccc|c} 2 & 1 & 4 & \\ 1 & -1 & -10 & \\ 3 & 1 & 0 & \end{array} \right]$$
$$\xrightarrow{r_1 + r_2} \left[\begin{array}{ccc|c} 1 & -1 & -10 & \\ 2 & 1 & 4 & \\ 3 & 1 & 0 & \end{array} \right]$$
$$\xrightarrow{-2r_1 \text{ added to } r_2 \text{ and } -3r_1 \text{ added to } r_3} \left[\begin{array}{ccc|c} 1 & -1 & -10 & \\ 0 & 3 & 24 & \\ 0 & 4 & 30 & \end{array} \right]$$
$$\xrightarrow{(-4/3)r_2 \text{ added to } r_3} \left[\begin{array}{ccc|c} 1 & -1 & -10 & \\ 0 & 3 & 24 & \\ 0 & 0 & -2 & \end{array} \right]$$

✓ This echelon form of the matrix makes it easy to see that $k_3 = 0$, from which follow $k_2 = 0$ and $k_1 = 0$. Thus, Equation (36)-and therefore Equation (34)-is satisfied only by $k_1 = k_2 = k_3 = 0$, which proves that the given vectors are linearly independent.

Liner Combination, Dependence and Span XIII

Any collection of vectors from \mathbb{R}^n that contains the zero vector is automatically dependent, for if $\{v_1, v_2, \dots, v_{r-1}, 0\}$ is such a collection, then for any $k \neq 0$,
 $0v_1 + 0v_2 + \dots + 0v_{r-1} + k0$

Liner Combination, Dependence and Span XIV

Question 1: Are the vectors $v_1 = (4, 1, -2)$, $v_2 = (-3, 0, 1)$, and $v_3 = (1, -2, 1)$ linearly independent?

Question 2: There is exactly one value of c such that the vectors $v_1 = (1, 0, 0, -2)$, $v_2 = (0, 1, -1, 0)$, $v_3 = (-1, 0, -1, 0)$, and $v_4 = (1, 1, 1, c)$ are linearly dependent. Find this value of c and determine a nontrivial linear combination of these vectors that equals the zero vector?



Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis



Eigenvalues & Eigenvectors I

- ✓ Eigenvalues and eigenvectors feature prominently in the analysis of linear transformations.
- ✓ In linear algebra, an eigenvector characteristic vector of a linear transformation is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it.
- ✓ The corresponding eigenvalue, often denoted by (λ) , is the factor by which the eigenvector is scaled.
- ✓ Geometrically, an eigenvector, corresponding to a real nonzero eigenvalue, points in a direction in which it is stretched by the transformation and the eigenvalue is the factor by which it is stretched. If the eigenvalue is negative, the direction is reversed. Loosely speaking, in a multidimensional vector space, the eigenvector is not rotated.

Eigenvalues & Eigenvectors II

- ✓ In other words, When a matrix is multiplied by one of its eigenvectors the output is the same eigenvector multiplied by a constant.
- ✓ For example, λ may be negative, in which case the eigenvector reverses direction as part of the scaling, or it may be zero or **complex**.

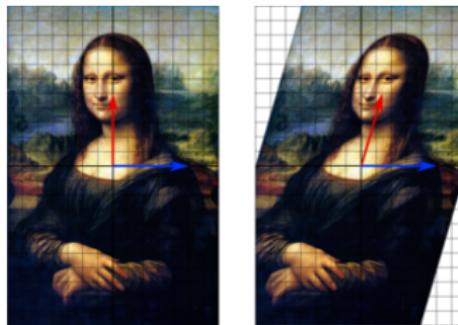


Figure 1: Matrix A acts by stretching the vector x , not changing its direction, so x is an eigenvector of A.

Eigenvalues & Eigenvectors III

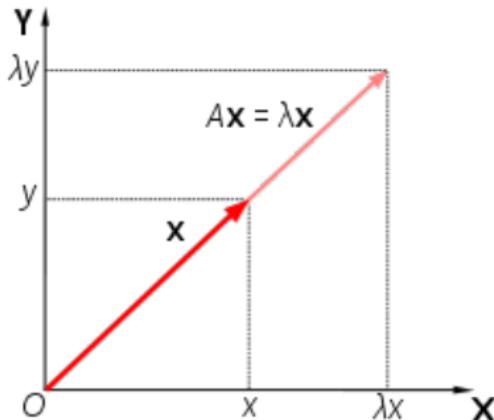


Figure 2: In this shear mapping the red arrow changes direction, but the blue arrow does not. The blue arrow is an eigenvector of this shear mapping because it does not change direction, and since its length is unchanged, its eigenvalue is 1.

Eigenvalues & Eigenvectors IV

The basic equation is

$$A\vec{v} = \lambda\vec{v} \quad (37)$$

- ✓ Left side is **Matrix-vector multiplication**, whereas right side is **Scale multiplication**.

The number λ is an eigenvalue of A .

$$A\vec{v} = (\lambda I)\vec{v} \quad (38)$$

- ✓ After multiplying by unit matrix, both the sides have matrix-vector multiplication.

Eigenvalues & Eigenvectors V

$$A\vec{v} - (\lambda I)\vec{v} = 0 \quad (39)$$

$$(A - \lambda I)\vec{v} = 0 \quad (40)$$

Example : Determine the eigenvectors of the matrix

$$A = \begin{bmatrix} 1 & -2 \\ 3 & -4 \end{bmatrix}$$

The eigenvalues of this matrix are $\lambda = -1, -2$. Therefore, there are nonzero vectors \mathbf{v} such that $A\mathbf{v} = \lambda\mathbf{v}$

For $\lambda=-1$, eigenvector: $A\mathbf{v} = -1\mathbf{v} = \begin{bmatrix} 1-1 & -2 \\ 3 & -4-1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

For $\lambda=-2$, eigenvector: $A\mathbf{v} = -2\mathbf{v} = \begin{bmatrix} 1-2 & -2 \\ 3 & -4-2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



Eigenvalues & Eigenvectors VI

- ✓ The trace of A , defined as the sum of its diagonal elements, is also the sum of all eigenvalues:

$$tr(A) = \sum_{i=1}^n a_{ij} = \sum_{i=1}^n \lambda_i = \lambda_1 + \lambda_2 + \cdots + \lambda_n. \quad (41)$$

- ✓ The determinant of A is the product of all its eigenvalues

$$det(A) = \prod_{i=1}^n \lambda_i = \lambda_1 \lambda_2 \dots \lambda_n. \quad (42)$$



Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis



Singular Value Decomposition(SVD) I

Singular Vector:

- ✓ For any real or complex $m \times n$ matrix M the **left-singular vectors** of M are the eigenvectors of MM^t . They are equal to the columns of the matrix **u** in the singular value decomposition $\{u, w, v\}$ of M .
- ✓ The right-singular vectors of M are the eigenvectors of the matrix **v** in the singular value decomposition of M .

Example: Left-singular vectors of a 2×3 matrix

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad u, w, v = \text{SingularValueDecomposition}[A];$$

$$\text{MatrixForm}[N[\text{Transpose}[u]]] = \begin{bmatrix} 0.386318 & 0.922366 \\ -0.922366 & 0.386318 \end{bmatrix}$$

$$\text{MatrixForm}[M M^t = M.\text{Transpose}[M]] = \begin{bmatrix} 14 & 32 \\ 32 & 77 \end{bmatrix}$$

Singular Value Decomposition(SVD) II

$$\text{MatrixForm}[\text{Eigenvectors}[N[MM^t]]] = \begin{bmatrix} 0.386318 & 0.922366 \\ -0.922366 & 0.386318 \end{bmatrix}$$

✓ Right-singular vectors of a 2×3 matrix $\text{MatrixForm}[M] = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

$\{u, w, v\} = \text{SingularValueDecomposition}[MM];$

$$\text{MatrixForm}[N[\text{Transpose}[v]]] = \begin{bmatrix} 0.428667 & 0.566307 & 0.703947 \\ 0.805964 & 0.112382 & -0.581199 \\ 0.408248 & -0.816497 & 0.408248 \end{bmatrix}$$

$$\text{MatrixForm}[M^t M = \text{Transpose}[M].M] = \begin{bmatrix} 17 & 22 & 37 \\ 22 & 29 & 36 \\ 27 & 36 & 45 \end{bmatrix}$$

$\text{MatrixForm}[\text{Eigenvectors}[N[M^t M]]] =$

$$\begin{bmatrix} 0.428667 & 0.566307 & 0.703947 \\ 0.805964 & 0.112382 & -0.581199 \\ 0.408248 & -0.816497 & 0.408248 \end{bmatrix}$$

Singular Value Decomposition(SVD) III

- ✓ Singular value decomposition (SVD) is a matrix factorization method that generalizes the eigen decomposition of a square matrix ($n \times n$) to any matrix ($n \times m$).
- ✓ SVD is similar to Principal Component Analysis (PCA), but more general. PCA assumes that input square matrix, SVD doesn't have this assumption. General formula of SVD is:

$$M = U \sum V^t, \quad (43)$$

Or

$$M_{[m \times n]} = U_{[m \times r]} \sum_{[r \times r]} V_{[n \times r]}^t, \quad (44)$$

where:



Singular Value Decomposition(SVD) IV

- **M-** is original (input) matrix we want to decompose.

Example: $m \times n$ matrix(e.g. m documents, n terms(words))

- **U-** is left singular matrix(columns are left singular vectors). **U** columns contain eigenvectors of matrix \mathbf{MM}^t .

Example: $m \times r$ matrix (m documents, r concepts)

- Σ - is a diagonal matrix containing singular (eigen) values

Example: $r \times r$ diagonal matrix (strength of each 'concepts')

(r : rank of the matrix M)

- **V-** is right singular matrix (columns are right singular vectors). **V** columns contain eigenvector of matrix \mathbf{MM}^t .

Example: $n \times r$ matrix(n terms, r concepts)



Singular Value Decomposition(SVD) V



$$M = \underset{m \times n}{U} \underset{m \times m}{\Sigma} \underset{m \times n}{V^*} \underset{n \times n}{}$$

Figure 3: SVD matrices

Singular Value Decomposition(SVD) VI

$$\mathbf{A} \approx \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^\top$$

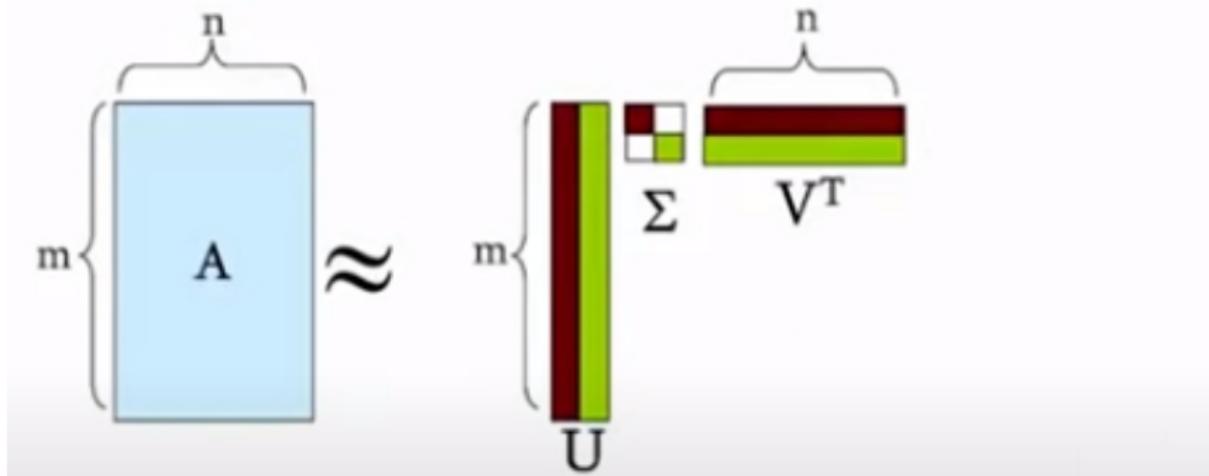


Figure 4: SVD matrices

Singular Value Decomposition(SVD) VII

$$\mathbf{A} \approx \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^\top$$

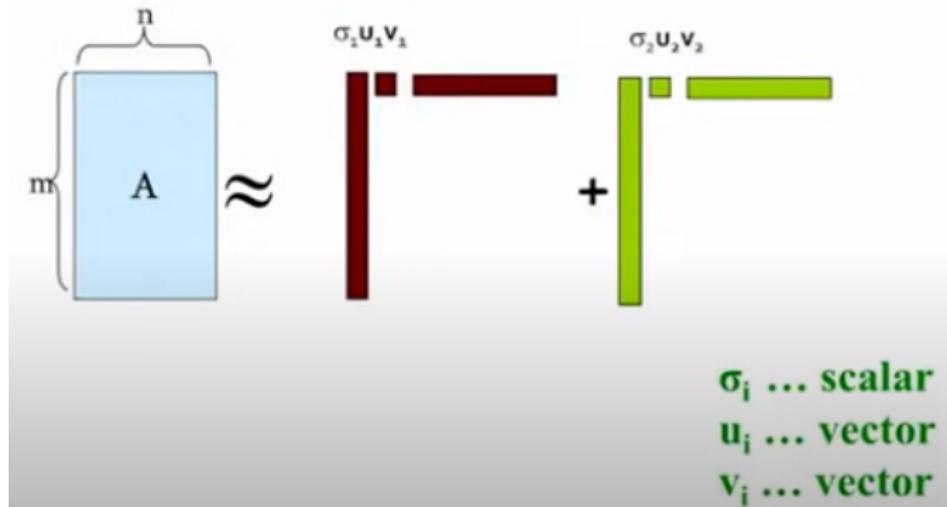


Figure 5: SVD matrices

Singular Value Decomposition(SVD) VIII

It is **always** possible to decompose a real matrix \mathbf{A} into $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$, where

- $\mathbf{U}, \Sigma, \mathbf{V}$: unique
- \mathbf{U}, \mathbf{V} : column orthonormal
 - $\mathbf{U}^T \mathbf{U} = \mathbf{I}; \mathbf{V}^T \mathbf{V} = \mathbf{I}$ (\mathbf{I} : identity matrix)
 - (Columns are orthogonal unit vectors)
- Σ : diagonal
 - Entries (**singular values**) are **positive**, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \dots \geq 0$)

Figure 6: SVD properties

Singular Value Decomposition(SVD) IX

- A = U Σ V^T - example: Users to Movies

The diagram illustrates the SVD decomposition of a user-to-movie matrix. On the left, a 7x5 matrix is shown with rows labeled by user ID (1 to 7) and columns labeled by movie titles (Alien, Serenity, Casablanca, Amelie). The matrix entries represent the likeness of users to movies, with values ranging from 0 to 5. To the right of the matrix is an equals sign followed by the SVD formula A = U Σ V^T. The term U is represented by a vertical stack of 7 green rectangles of increasing height, corresponding to the user rows. The term Σ is represented by a horizontal stack of three colored rectangles (green, white, red) of decreasing width, representing singular values. The term V^T is represented by a horizontal stack of 5 green rectangles of decreasing width, corresponding to the movie columns.

Matrix	Alien	Serenity	Casablanca	Amelie
1	1	1	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
0	2	0	4	4
0	0	0	5	5
0	1	0	2	2

Figure 7: SVD-Example: User-to-Movies

- In the figure, values in the matrix A is likeness of users, i.e, lower value means less vote, whereas higher value means more votes, and zero means no votes.



Singular Value Decomposition(SVD) X

- $A = U \Sigma V^T$ - example: Users to Movies

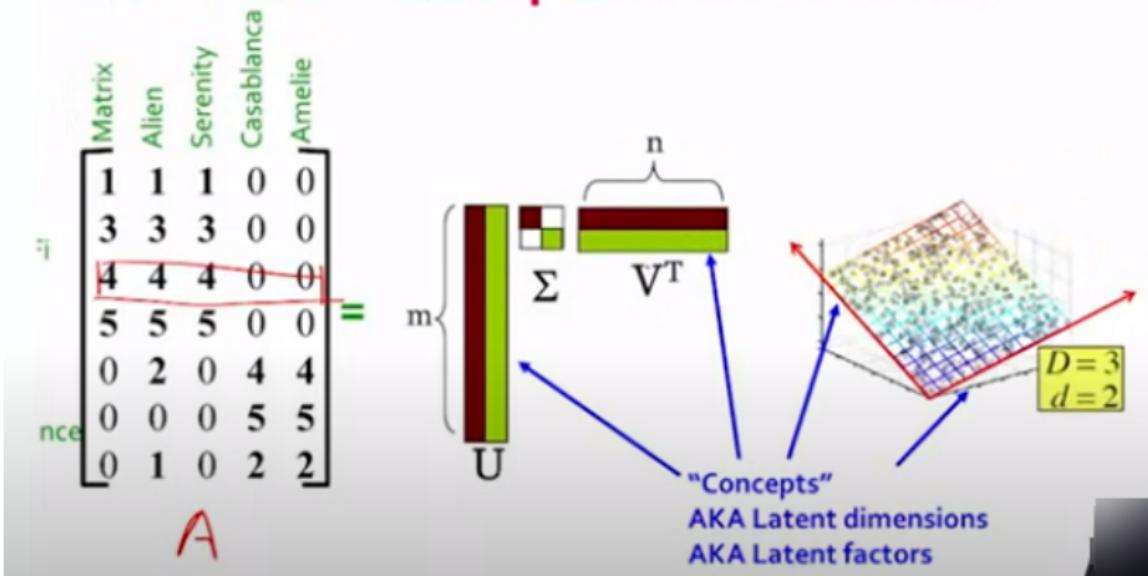


Figure 8: SVD-Example: User-to-Movies

Singular Value Decomposition(SVD) XI

- A = U Σ V^T - example: Users to Movies

$$\begin{matrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ \hline 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{matrix} = \begin{matrix} \text{U} \\ \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \end{matrix} \times \begin{matrix} \Sigma \\ \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \end{matrix} \times \begin{matrix} \text{V}^T \\ \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix} \end{matrix}$$

Figure 9: SVD-Example: User-to-Movies

Singular Value Decomposition(SVD) XII

- ✓ Values in matrices U and V are strength of concepts.
- ✓ U is "user-to-concept" similarity matrix,
- ✓ V is "movie-to-concept" similarity matrix
- ✓ In matrix U , columns first and second have two different concepts; SciFi(science fiction)-concept and Romance-concept, respectively.
- ✓ Similarly, in matrix V , rows first and second have two different concepts; SciFi(science fiction)-concept and Romance-concept, respectively.
- ✓ In matrices U and V , third column and third row, respectively, have very low strength.

'movies', 'users' and 'concepts':

- U : user-to-concept similarity matrix
- V : movie-to-concept similarity matrix
- Σ : its diagonal elements:
‘strength’ of each concept

Figure 10: SVD-Interpretation

Outline

Basic Notations

Linear Algebra in the context of deep learning

Scalars, Vectors, Matrices and Tensors

Scalars

Vectors

Matrix

Tensors

Eigenvalues & Eigenvectors

Singular Value Decomposition

Principal Component Analysis

Principal Component Analysis(PCA) I

- ✓ It is about finding useful information in a data matrix.
- ✓ Start by measuring m properties (m features) of n samples.

Example: A matrix could be grades in m courses for n students. A row for each course, a column for each student.

- ✓ from each row, subtract its average so the sample means are zero.

We look for a **combination of courses** and/or **combination of students** for which the data provides the most information.

- ✓ The information is "distance from randomness" and it is measured by **variance**. A large variance in course grade means greater information than a small variance.
- ✓ Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by

Principal Component Analysis(PCA) II

transforming a large set of variables into a smaller one that still contains most of the information in the large set.

✓ Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

The idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

Principal Component Analysis(PCA) III

HOW DO WE DO A PCA?

- ① Standardize the range of continuous initial variables
- ② Compute the covariance matrix to identify correlations
- ③ Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
- ④ Create a feature vector to decide which principal components to keep
- ⑤ Recast the data along the principal components axes

Principal Component Analysis(PCA) IV

① STANDARDIZATION: The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

Example: If there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

✓ Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{\text{value} - \text{mean}}{\text{standard deviation}}$$

Principal Component Analysis(PCA) V

② COVARIANCE MATRIX COMPUTATION: The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other or to see if there is any relationship between them.

- ✓ Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.
- ✓ The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables x , y , and z , the covariance matrix

is a 3×3 matrix of this form:

$$\begin{bmatrix} Cov(x, x) & Cov(x, y) & Cov(x, z) \\ Cov(y, x) & Cov(y, y) & Cov(y, z) \\ Cov(z, x) & Cov(z, y) & Cov(z, z) \end{bmatrix}$$

Principal Component Analysis(PCA) VI

✓ Since the covariance of a variable with itself is its variance ($Cov(a, a) = Var(a)$), in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. And since the covariance is commutative ($Cov(a, b) = Cov(b, a)$), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?

- ✓ It's actually the sign of the covariance that matters :
- ① if positive then : the two variables increase or decrease together (correlated).
 - ② if negative then : One increases when the other decreases (Inversely correlated).



Principal Component Analysis(PCA) VII

- ✓ The covariance matrix is not more than a table that summarizes the correlations between all the possible pairs of variables.

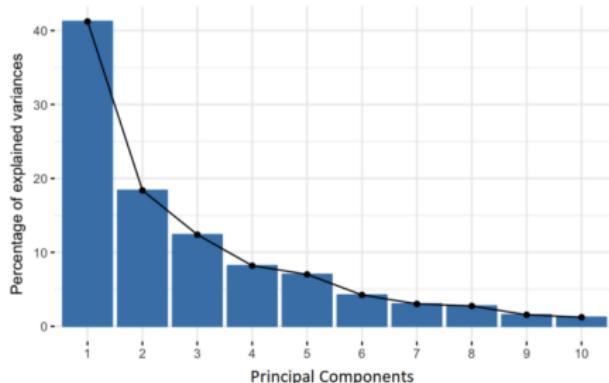
③ COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS:

- ✓ Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the principal components of the data.
- ✓ Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables.
- ✓ These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information

Principal Component Analysis(PCA) VIII

within the initial variables is squeezed or compressed into the first components.

- ✓ So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on.



Principal Component Analysis(PCA) IX

- ✓ Organizing information in principal components this way, will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.
- ✓ An important thing to realize here is that, the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.
- ✓ Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data.
- ✓ The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data

Principal Component Analysis(PCA) X

points along it, and the larger the dispersion along a line, the more the information it has.

- ✓ To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

How PCA Constructs the Principal Components

- ✓ As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set.



Principal Component Analysis(PCA) XI

- ✓ For example, let's assume that the scatter plot of our data set is as shown in Figure 11.

Can we guess the first principal component ?

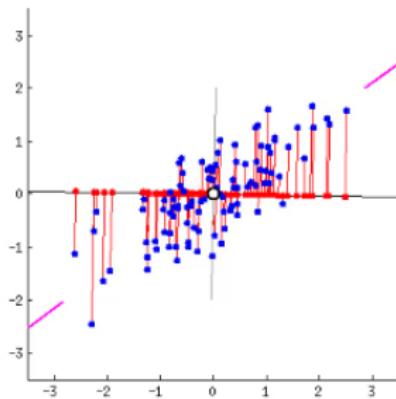


Figure 11: PCA Constructs the Principal Components

Principal Component Analysis(PCA) XII

Answer: Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).

- ✓ The second principal component is calculated in the same way, with the condition that it is **uncorrelated with (i.e., perpendicular to) the first principal component** and that it accounts for the next highest variance. This continues until a total of p principal components have been calculated, equal to the original number of variables.
- ✓ eigenvector and eigenvalue always come in pairs, so that every eigenvector has an eigenvalue. And their number is equal to the number of dimensions of the data.

Principal Component Analysis(PCA) XIII

- ✓ **For example**, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues.
- ✓ *The eigenvectors of the Covariance matrix are actually the directions of the axes where there is the most variance(most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component.*

By ranking the eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

Principal Component Analysis(PCA) XIV

Example:

Let's suppose that our data set is 2-dimensional with 2 variables x,y and that the eigenvectors and eigenvalues of the covariance matrix are as follows:

$$v_1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \quad \lambda_1 = 1.284028$$

$$v_2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \quad \lambda_2 = 0.04908323$$

- ✓ If we rank the eigenvalues in descending order, we get $\lambda_1 > \lambda_2$, which means that the eigenvector that corresponds to the first principal component (PC1) is v_1 and the one that corresponds to the second component (PC2) is v_2 .

Principal Component Analysis(PCA) XV

✓ After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96% and 4% of the variance of the data.

④ FEATURE VECTOR:

✓ As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance.

✓ In this step, to choose whether to keep **all these components or discard those of lesser significance** (of low eigenvalues), and form with the remaining ones a matrix of vectors is called **Feature vector**.

Principal Component Analysis(PCA) XVI

- ✓ The **Feature vector** is simply a matrix that has as columns the eigenvectors of the components that is decided to keep.
- ✓ This makes it the first step towards **dimensionality reduction**, because if we choose to keep only p eigenvectors (components) out of n , the final data set will have only p dimensions.

Example: Continuing with the example from the previous step, we can either form a feature vector with both of the eigenvectors v_1 and v_2 :

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Or discard the eigenvector v_2 , which is the one of lesser significance, and form a feature vector with v_1 only:

Principal Component Analysis(PCA) XVII

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

- ✓ Discarding the eigenvector v_2 will reduce dimensionality by 1, and will consequently cause a loss of information in the final data set. But given that v_2 was carrying only 4% of the information, the loss will be therefore not important and we will still have 96% of the information that is carried by v_1 .

It's up to you to choose whether to keep all the components or discard the ones of lesser significance, depending on what you are looking for.



Principal Component Analysis(PCA) XVIII

⑤ RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES:

- ✓ In the previous steps, apart from standardization, we do not make any changes on the data, we just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).
- ✓ The aim of RECAST the data is to use the feature vector formed using the eigenvectors of the covariance matrix, to **reorient** the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis).
- ✓ This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

Principal Component Analysis(PCA) XIX

$FinalDataSet =$

$FeatureVector^T \times StandardizedOriginalDataSet^T$

Application of PCA

- ✓ **Gene expression data:** Determining the function of genes, and combinations of genes, is a central problem of genetics. Which genes combine to give which properties? Which genes malfunction to give which diseases?
- ✓ The understanding of genetic data (bioinformatics) has become a tremendous application of linear algebra.



Application of linear algebra I

① **Dataset and Data Files:** In machine learning, you fit a model on a dataset.

✓ This is the table-like set of numbers where each row represents an observation and each column represents a feature of the observation.

✓ *Example:*

```
1 5.1,3.5,1.4,0.2,Iris-setosa
2 4.9,3.0,1.4,0.2,Iris-setosa
3 4.7,3.2,1.3,0.2,Iris-setosa
4 4.6,3.1,1.5,0.2,Iris-setosa
5 5.0,3.6,1.4,0.2,Iris-setosa
```



Application of linear algebra II

- ✓ This data is in fact a matrix: a key data structure in linear algebra.
- ✓ Further, when you split the data into inputs and outputs to fit a supervised machine learning model, such as the measurements and the flower species, you have a matrix (X) and a vector (y).
- ✓ The vector is another key data structure in linear algebra.

② Images and Photographs:

③ One-Hot Encoding:

④ Linear Regression:

⑤ Regularization:

⑥ Principal Component Analysis:



Application of linear algebra III

⑦ Singular-Value Decomposition:

⑧ Latent Semantic Analysis:

⑨ Recommender Systems:

⑩ Deep Learning:



CS322: Deep Learning

Vector Calculus

Sachchida Nand Chaurasia
Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



October 18, 2021

Outline

Vector Calculus

Derivative

Partial derivative

Gradients of Vector-Valued Functions

Gradient Descent and it's Variants



Outline

Vector Calculus

Derivative

Partial derivative

Gradients of Vector-Valued Functions

Gradient Descent and it's Variants

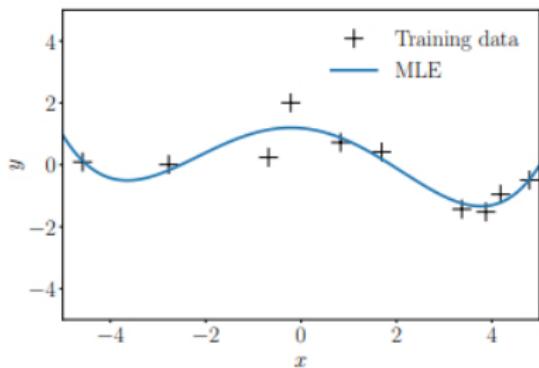


Vector Calculus I

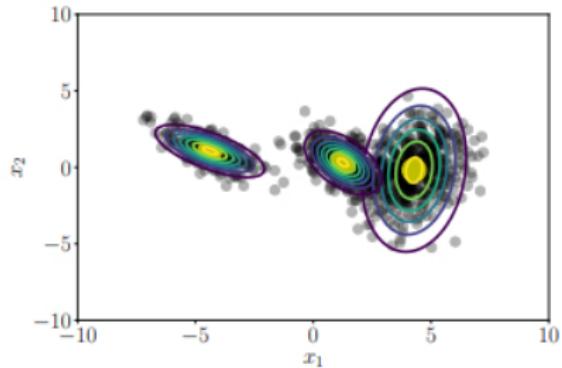
- ✓ Many algorithms in machine learning optimize an objective function with respect to a set of desired model parameters that control how well a model explains the data: Finding good parameters can be phrased as an optimization problem.
- ✓ Curve-fitting problems and optimize linear weight parameters to maximize the likelihood
- ✓ Neural-network auto-encoders for dimensionality reduction and data compression, where the parameters are the weights and biases of each layer, and where we minimize a reconstruction error by repeated application of the chain rule
- ✓ Gaussian mixture models for modeling data distributions, where we optimize the location and shape parameters of each mixture component to maximize the likelihood of the model.



Vector Calculus II



(a) Regression problem: Find parameters, such that the curve explains the observations (crosses) well.



(b) Density estimation with a Gaussian mixture model: Find means and covariances, such that the data (dots) can be explained well.

Outline

Vector Calculus

Derivative

Partial derivative

Gradients of Vector-Valued Functions

Gradient Descent and it's Variants



Differentiation of Univariate Functions I

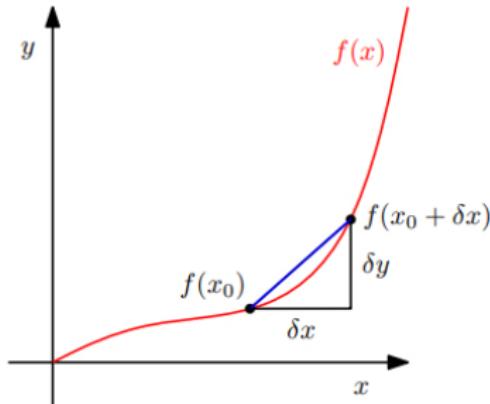


Figure: The average incline of a function f between x_0 and $x_0 + \delta x$ is the incline of the secant (blue) through $f(x_0)$ and $f(x_0 + \delta x)$ and given by $\frac{\delta y}{\delta x}$.

Differentiation of Univariate Functions II

Difference Quotient:

$$\frac{\delta y}{\delta x} := \frac{f(x_0 + \delta x) - f(x)}{\delta x} \quad (1)$$

- ✓ The difference quotient computes the slope of the secant line through two points, x_0 and $x_0 + \delta x$, on the graph of f .
- ✓ The difference quotient can also be considered the average slope of f between x and $x + \delta x$ if we assume f to be a linear function. In the limit for $\delta x \rightarrow 0$, we obtain the tangent of f at x , if f is differentiable.



Differentiation of Univariate Functions III

Derivative: More formally, for $h > 0$ the derivative of f at x is defined as the limit

$$\frac{df}{dx} := \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h} \quad (2)$$

Differentiation of Univariate Functions IV

Derivative of a Polynomial:

- ✓ Given a function $f(x) = x^n, n \in \mathbb{N}$.

$$\begin{aligned}\frac{df}{dx} &= \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \\&= \lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} \\&= \lim_{h \rightarrow 0} \frac{\sum_{i=0}^n \binom{n}{i} x^{n-i} h^i - x^n}{h}.\end{aligned}$$

- ✓ $x^n = \binom{n}{0} x^{n-0} h^0$. By starting the sum at 1, the x^n -term cancels, and we obtain



Differentiation of Univariate Functions V

$$\begin{aligned}\frac{df}{dx} &= \lim_{h \rightarrow 0} \frac{\sum_{i=1}^n \binom{n}{i} x^{n-i} h^i}{h} \\&= \lim_{h \rightarrow 0} \sum_{i=1}^n \binom{n}{i} x^{n-i} h^{i-1} \\&= \lim_{h \rightarrow 0} \binom{n}{1} x^{n-1} + \underbrace{\sum_{i=2}^n \binom{n}{i} x^{n-i} h^{i-1}}_{\rightarrow 0 \text{ as } h \rightarrow 0} \\&= \frac{n!}{1!(n-1)!} x^{n-1} = nx^{n-1}.\end{aligned}$$



Differentiation of Univariate Functions VI

Taylor Polynomial: The Taylor polynomial of degree n of $f : \mathbb{R} \rightarrow \mathbb{R}$ at x_0 is defined as :

$$T_n(x) := \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \quad (3)$$

where $f^{(k)}(x_0)$ is the k -th derivative of f at x_0 (assumed that exists) and $\frac{f^{(k)}(x_0)}{k!}$ are the coefficients of the polynomial.



Differentiation of Univariate Functions VII

Taylor Series: For a smooth function $f \in \mathcal{C}^\infty$, $f : \mathbb{R} \rightarrow \mathbb{R}$, the Taylor Series of f at x_0 is defined as

$$T_\infty(x) := \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \quad (4)$$

$f \in \mathcal{C}^\infty$ means that f is continuously differentiable infinitely many times.
Maclaurin series analytic.

Differentiation of Univariate Functions VIII

We consider the polynomial

$$f(x) = x^4$$

and seek the Taylor polynomial T_6 , evaluated at $x_0 = 1$. We start by computing the coefficients $f^{(k)}(1)$ for $k = 0, \dots, 6$:

$$\begin{aligned} f(1) &= 1 \\ f'(1) &= 4 \\ f''(1) &= 12 \\ f^{(3)}(1) &= 24 \\ f^{(4)}(1) &= 24 \\ f^{(5)}(1) &= 0 \\ f^{(6)}(1) &= 0 \end{aligned}$$

Therefore, the desired Taylor polynomial is

$$\begin{aligned} T_6(x) &= \sum_{k=0}^6 \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \\ &= 1 + 4(x - 1) + 6(x - 1)^2 + 4(x - 1)^3 + (x - 1)^4 + 0. \end{aligned}$$

Multiplying out and re-arranging yields

$$\begin{aligned} T_6(x) &= (1 - 4 + 6 - 4 + 1) + x(4 - 12 + 12 - 4) \\ &\quad + x^2(6 - 12 + 6) + x^3(4 - 4) + x^4 \\ &= x^4 = f(x), \end{aligned}$$

i.e., we obtain an exact representation of the original function.

Differentiation of Univariate Functions IX

Differentiation Rules:

Product rule:

$$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x) \quad (5)$$

Quotient rule:

$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2} \quad (6)$$

Sum rule:

$$(f(x) + g(x))' = f'(x) + g'(x) \quad (7)$$

Chain rule:

$$(g(f(x)))' = (g \circ f)'(x) = g'(f(x)) f'(x) \quad (8)$$

Differentiation of Univariate Functions X

Here $g \circ f$ denotes function composition $x \mapsto f(x) \mapsto g(f(x))$

Example (Chain rule): $h(x) = (2x + 1)^4$ using chain rule.

$$h(x) = (2x + 1)^4 = g(f(x)), \quad (9)$$

$$f(x) = 2x + 1, \quad (10)$$

$$g(f) = f^4, \quad (11)$$

we obtain the derivatives of f and g as

$$f'(x) = 2, \quad (12)$$

$$g'(x) = 4f^3, \quad (13)$$

Such that the derivative of h is given as

$$g'(x) = g'(f)f'(x) = (4f^3) \cdot 2 \stackrel{f=2x+1}{=} 4(2x+1)^3 \cdot 2 = 8(2x+1)^3 \quad (14)$$

Outline

Vector Calculus

Derivative

Partial derivative

Gradients of Vector-Valued Functions

Gradient Descent and it's Variants



Partial derivative I

- ✓ Differentiation applies to functions f of scalar variable $x \in \mathbb{R}$.
- ✓ When a function f depends on one or more variables $x \in \mathbb{R}^n$, e.g., $f(x) = f(x_1, x_2)$.
- ✓ The generalization of the derivative to functions of several variables is the *gradient*.
- ✓ We find the gradient of the function f with respect to x by varying one variable at a time and keeping the others constant.
- ✓ The gradient is then the collection of these *partial derivatives*.

Partial Derivative: For a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$ $\mathbf{x} \in \mathbb{R}^n$ of n variables x_1, \dots, x_n .

Partial derivative II

$$\frac{\partial f}{\partial x_1} = \lim_{h \rightarrow 0} \frac{f(x_1 + h, \dots, x_n) - f(\mathbf{x})}{h} \quad (15)$$

$$\frac{\partial f}{\partial x_2} = \lim_{h \rightarrow 0} \frac{f(x_1, x_2 + h, \dots, x_n) - f(\mathbf{x})}{h} \quad (16)$$

$$\vdots \quad (17)$$

$$\frac{\partial f}{\partial x_n} = \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_{n-1}, x_n + h) - f(\mathbf{x})}{h} \quad (18)$$

✓ The collection of partial derivatives in the row vector

$$\nabla_{\mathbf{x}} f = \text{grad } f = \frac{df}{d\mathbf{x}} = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \frac{\partial f(\mathbf{x})}{\partial x_2} \cdots \frac{\partial f(\mathbf{x})}{\partial x_n} \right] \in \mathbb{R}^{1 \times n}, \quad (19)$$

Partial derivative III

where n is the number of variables and 1 is the dimension of the image/range/codomain of f .

✓ The row vector is called *gradient* of f or the *Jacobian* and is the generalization of the derivative.

Example: Partial derivatives of $f(x, y) = (x + 2y^3)^2$ using chain rule.

$$\frac{\partial f(x, y)}{\partial x} = 2(x + 2y^3) \frac{\partial}{\partial x}(x + 2y^3) = 2(x + 2y^3)$$

$$\frac{\partial f(x, y)}{\partial y} = 2(x + 2y^3) \frac{\partial}{\partial y}(x + 2y^3) = 12(x + 2y^3)y^2$$

Example: For $f(x_1, x_2) = x_1^2x_2 + x_1x_2^3 \in \mathbb{R}$, the partial derivatives of with respect to x_1 and x_2 are

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1x_2 + x_2^3$$

Partial derivative IV

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1x_2^2$$

- ✓ The gradient is

$$\frac{df}{dx} = \begin{bmatrix} \frac{\partial f(x_1, x_2)}{\partial x_1} & \frac{\partial f(x_1, x_2)}{\partial x_2} \end{bmatrix} = [2x_1x_2 + x_2^3 \ x_1^2 + 3x_1x_2^2] \in \mathbb{R}^{1 \times 2}$$

Partial derivative V

Partial Differentiation Rules:

$$\text{Product rule : } \frac{\partial}{\partial \mathbf{x}} (f(\mathbf{x})g(\mathbf{x})) = \frac{\partial f}{\partial \mathbf{x}} g(\mathbf{x}) + f(\mathbf{x}) \frac{\partial g}{\partial \mathbf{x}} \quad (20)$$

$$\text{Sum rule : } \frac{\partial}{\partial \mathbf{x}} (f(\mathbf{x}) + g(\mathbf{x})) = \frac{\partial f}{\partial \mathbf{x}} + \frac{\partial g}{\partial \mathbf{x}} \quad (21)$$

$$\text{Chain rule : } \frac{\partial}{\partial \mathbf{x}} (g \circ f)(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} (g(f(\mathbf{x}))) = \frac{\partial g}{\partial f} \frac{\partial f}{\partial \mathbf{x}} \quad (22)$$



Outline

Vector Calculus

Derivative

Partial derivative

Gradients of Vector-Valued Functions

Gradient Descent and it's Variants

Gradients of Vector-Valued Functions I

- ✓ Partial derivatives and gradients of functions $f : \mathbb{R} \rightarrow \mathbb{R}$ mapping to the real numbers.
- ✓ Partial derivatives and gradient to vector-valued functions (vector fields) $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, where $n \geq 1$ and $m > 1$.
- ✓ For a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$, the corresponding vector of function values is given as

$$f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^m \quad (23)$$

- ✓ The partial derivative of a vector-valued function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with respect to $x_i \in \mathbb{R}$, $i = 1, \dots, n$, is given as the vector

Gradients of Vector-Valued Functions II

$$\frac{\partial \mathbf{f}}{\partial x_i} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{bmatrix} = \begin{bmatrix} \lim_{h \rightarrow 0} \frac{f_1(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots, x_n) - f_1(\mathbf{x})}{h} \\ \vdots \\ \lim_{h \rightarrow 0} \frac{f_m(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots, x_n) - f_m(\mathbf{x})}{h} \end{bmatrix} \in \mathbb{R}^m.$$

✓ The gradient of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with respect to $\mathbf{x} \in \mathbb{R}^n$ by collecting these partial derivatives:

$$\begin{aligned} \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} &= \left[\boxed{\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1}} \dots \boxed{\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n}} \right] \\ &= \left[\begin{array}{c|c} \boxed{\frac{\partial f_1(\mathbf{x})}{\partial x_1}} & \dots & \boxed{\frac{\partial f_1(\mathbf{x})}{\partial x_n}} \\ \vdots & & \vdots \\ \boxed{\frac{\partial f_m(\mathbf{x})}{\partial x_1}} & \dots & \boxed{\frac{\partial f_m(\mathbf{x})}{\partial x_n}} \end{array} \right] \in \mathbb{R}^{m \times n} \end{aligned}$$

Gradients of Vector-Valued Functions III

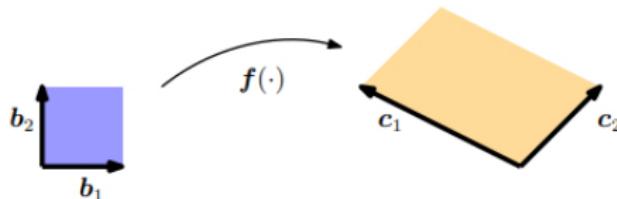
The collection of all first-order partial derivatives of a vector-valued function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is called the Jacobian.

✓ The Jacobian \mathcal{J} is an $m \times n$ matrix is as follows:

$$\begin{aligned}\mathcal{J} &= \nabla_{\mathbf{x}} \mathbf{f} = \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \quad \dots \quad \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \right] \\ &= \left[\begin{array}{ccc} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{array} \right], \\ \mathbf{x} &= \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad J(i, j) = \frac{\partial f_i}{\partial x_j}.\end{aligned}$$

Gradients of Vector-Valued Functions IV

Jacobian of f can be used to compute the magnifier between the blue and orange area.



- ✓ Given two vectors $\mathbf{b}_1 = [1, 0]^T$ and $\mathbf{b}_2 = [0, 1]^T$, the area of this square is

$$\left| \det \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = 1 .$$

- ✓ Given two vectors $\mathbf{c}_1 = [-2, 1]^T$ and $\mathbf{c}_2 = [1, 1]^T$, the area of this parallelogram is

Gradients of Vector-Valued Functions V

$$\left| \det \begin{pmatrix} -2 & 1 \\ 1 & 1 \end{pmatrix} \right| = |-3| = 3,$$

- ✓ The area of parallelogram is exactly three times the area of the unit square.
- ✓ A scaling factor can be find by finding a mapping that transforms the unit square into the other square.
- ✓ A variable transformation from $(\mathbf{b}_1, \mathbf{b}_2)$ to $(\mathbf{c}_1, \mathbf{c}_2)$ can be done in linear algebra.

Partial Derivatives Approach: For this approach, we consider a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that performs a variable transformation.

- ✓ f maps the coordinate representation of any vector $\mathbf{x} \in \mathbb{R}^2$ with respect to $(\mathbf{b}_1, \mathbf{b}_2)$ onto the coordinate representation $\mathbf{y} \in \mathbb{R}^2$ with respect to $(\mathbf{c}_1, \mathbf{c}_2)$.
- ✓ Identifying the mapping so that we can compute how an area (or volume) changes when it is being transformed by f . For this we need to find out how $f(\mathbf{x})$ changes if we modify \mathbf{x} a bit.

Gradients of Vector-Valued Functions VI

- ✓ This question is exactly answered by the Jacobian matrix $\frac{d\mathbf{f}}{dx} \in \mathbb{R}^{2 \times 2}$.

$$y_1 = -2x_1 + x_2 \quad (24)$$

$$y_2 = x_1 + x_2 \quad (25)$$

We obtain the function relationship between x and y , which allows us to get the partial derivatives

$$\frac{\partial y_1}{\partial x_1} = -2, \quad \frac{\partial y_1}{\partial x_2} = 1, \quad \frac{\partial y_2}{\partial x_1} = 1, \quad \frac{\partial y_2}{\partial x_2} = 1$$

and compose the Jacobian as

Gradients of Vector-Valued Functions VII

$$\mathbf{J} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}.$$

- ✓ The Jacobian represents the coordinate transformation we are looking for.



Gradients of Vector-Valued Functions VIII

Example: Chain rule

Consider the function $h : \mathbb{R} \rightarrow \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$\begin{aligned} f : \mathbb{R}^2 &\rightarrow \mathbb{R} \\ g : \mathbb{R} &\rightarrow \mathbb{R}^2 \\ f(\mathbf{x}) &= \exp(x_1 x_2^2), \\ \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= g(t) = \begin{bmatrix} t \cos t \\ t \sin t \end{bmatrix} \end{aligned}$$

and compute the gradient of h with respect to t . Since $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $g : \mathbb{R} \rightarrow \mathbb{R}^2$ we note that

$$\frac{\partial f}{\partial \mathbf{x}} \in \mathbb{R}^{1 \times 2}, \quad \frac{\partial g}{\partial t} \in \mathbb{R}^{2 \times 1}.$$

The desired gradient is computed by applying the chain rule:

$$\begin{aligned} \frac{dh}{dt} &= \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right] \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \\ &= [\exp(x_1 x_2^2) x_2^2 \quad 2 \exp(x_1 x_2^2) x_1 x_2] \begin{bmatrix} \cos t - t \sin t \\ \sin t + t \cos t \end{bmatrix} \\ &= \exp(x_1 x_2^2) (x_2^2 (\cos t - t \sin t) + 2x_1 x_2 (\sin t + t \cos t)), \end{aligned}$$

where $x_1 = t \cos t$ and $x_2 = t \sin t$; see (5.72).

Gradients of Vector-Valued Functions IX

Gradient of Vectors with Respect to Matrices:

- ✓ We will encounter situations where we need to take gradients of matrices with respect to vectors (or other matrices), which results in a multidimensional tensor. We can think of this tensor as a multidimensional array that collects partial derivatives. For example, if we compute the gradient of an $m \times n$ matrix \mathbf{A} with respect to a $p \times p$ matrix \mathbf{B} .
- ✓ The resulting Jacobian would be $(m \times n) \times (p \times p)$, i.e., a four dimensional tensor \mathcal{J} , whose entries are given ad $\mathcal{J}_{ijkl} = \partial A_{ij} / \partial B_{kl}$.

Gradients of Vector-Valued Functions X

Let us consider the following example, where

$$\mathbf{f} = \mathbf{A}\mathbf{x}, \quad \mathbf{f} \in \mathbb{R}^M, \quad \mathbf{A} \in \mathbb{R}^{M \times N}, \quad \mathbf{x} \in \mathbb{R}^N \quad (5.85)$$

and where we seek the gradient $d\mathbf{f}/d\mathbf{A}$. Let us start again by determining the dimension of the gradient as

$$\frac{d\mathbf{f}}{d\mathbf{A}} \in \mathbb{R}^{M \times (M \times N)}. \quad (5.86)$$

By definition, the gradient is the collection of the partial derivatives:

$$\frac{d\mathbf{f}}{d\mathbf{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \mathbf{A}} \end{bmatrix}, \quad \frac{\partial f_i}{\partial \mathbf{A}} \in \mathbb{R}^{1 \times (M \times N)}. \quad (5.87)$$

To compute the partial derivatives, it will be helpful to explicitly write out the matrix vector multiplication:

$$f_i = \sum_{j=1}^N A_{ij}x_j, \quad i = 1, \dots, M, \quad (5.88)$$

and the partial derivatives are then given as

$$\frac{\partial f_i}{\partial A_{iq}} = x_q. \quad (5.89)$$

This allows us to compute the partial derivatives of f_i with respect to a row of \mathbf{A} , which is given as

$$\frac{\partial f_i}{\partial \mathbf{A}_{i,:}} = \mathbf{x}^\top \in \mathbb{R}^{1 \times 1 \times N}, \quad (5.90)$$

Gradients of Vector-Valued Functions XI

$$\frac{\partial f_i}{\partial A_{k \neq i,:}} = \mathbf{0}^\top \in \mathbb{R}^{1 \times 1 \times N} \quad (5.91)$$

where we have to pay attention to the correct dimensionality. Since f_i maps onto \mathbb{R} and each row of A is of size $1 \times N$, we obtain a $1 \times 1 \times N$ -sized tensor as the partial derivative of f_i with respect to a row of A .

We stack the partial derivatives (5.91) and get the desired gradient in (5.87) via

$$\frac{\partial f_i}{\partial A} = \begin{bmatrix} \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \\ \mathbf{x}^\top \\ \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \end{bmatrix} \in \mathbb{R}^{1 \times (M \times N)} \quad (5.92)$$

Gradients of Vector-Valued Functions XII

Gradient of Matrices with Respect to Matrices:

Consider a matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$ and $\mathbf{f} : \mathbb{R}^{M \times N} \rightarrow \mathbb{R}^{N \times N}$ with

$$\mathbf{f}(\mathbf{R}) = \mathbf{R}^\top \mathbf{R} =: \mathbf{K} \in \mathbb{R}^{N \times N}, \quad (5.93)$$

where we seek the gradient $d\mathbf{K}/d\mathbf{R}$.

To solve this hard problem, let us first write down what we already know: The gradient has the dimensions

$$\frac{d\mathbf{K}}{d\mathbf{R}} \in \mathbb{R}^{(N \times N) \times (M \times N)}, \quad (5.94)$$

which is a tensor. Moreover,

$$\frac{dK_{pq}}{d\mathbf{R}} \in \mathbb{R}^{1 \times M \times N} \quad (5.95)$$

for $p, q = 1, \dots, N$, where K_{pq} is the (p, q) th entry of $\mathbf{K} = \mathbf{f}(\mathbf{R})$. Denoting the i th column of \mathbf{R} by \mathbf{r}_i , every entry of \mathbf{K} is given by the dot product of two columns of \mathbf{R} , i.e.,

$$K_{pq} = \mathbf{r}_p^\top \mathbf{r}_q = \sum_{m=1}^M R_{mp} R_{mq}. \quad (5.96)$$

When we now compute the partial derivative $\frac{\partial K_{pq}}{\partial R_{ij}}$ we obtain

$$\frac{\partial K_{pq}}{\partial R_{ij}} = \sum_{m=1}^M \frac{\partial}{\partial R_{ij}} R_{mp} R_{mq} = \partial_{pqij}, \quad (5.97)$$

Gradients of Vector-Valued Functions XIII

$$\partial_{pqij} = \begin{cases} R_{iq} & \text{if } j = p, p \neq q \\ R_{ip} & \text{if } j = q, p \neq q \\ 2R_{iq} & \text{if } j = p, p = q \\ 0 & \text{otherwise} \end{cases}. \quad (5.98)$$

From (5.94), we know that the desired gradient has the dimension $(N \times N) \times (M \times N)$, and every single entry of this tensor is given by ∂_{pqij} in (5.98), where $p, q, j = 1, \dots, N$ and $i = 1, \dots, M$.



Outline

Vector Calculus

Derivative

Partial derivative

Gradients of Vector-Valued Functions

Gradient Descent and it's Variants

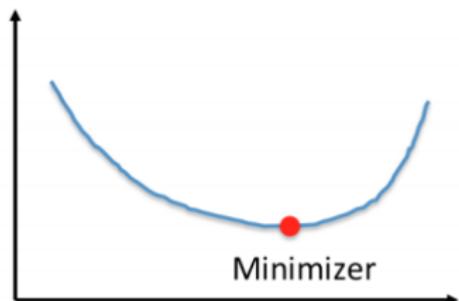


Gradient Descent I

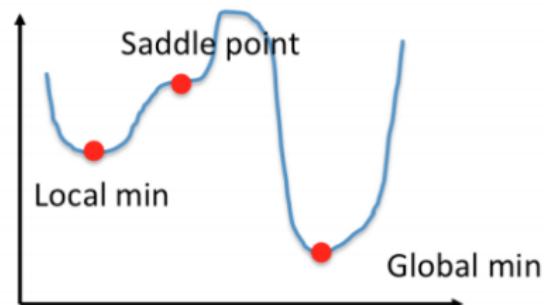
Convex and non-convex functions:

- ✓ A convex optimization problem is a problem where all of the constraints are convex functions, and the objective is a convex function if minimizing, or a concave function if maximizing. Linear functions are convex, so linear programming problems are convex problems.

Convex



Non-Convex

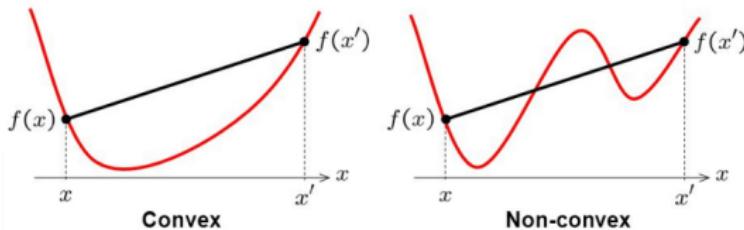


Gradient Descent II

A function $f : A \subseteq X \rightarrow \mathbb{R}$ defined on a convex set A is called convex if

$$f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda)f(x') \quad (26)$$

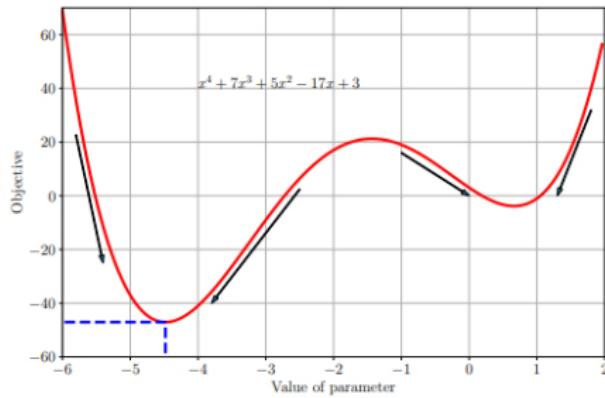
for any $x, x' \in \mathbb{X}$ and $\lambda \in [0, 1]$.



For convex function local minimum = global minimum

Gradient Descent III

- ✓ Gradient descent(GD) is the preferred way to optimize neural networks and many other machine learning algorithms but is often used as a black box.
- ✓ In a Neural Network, the Gradient Descent Algorithm is used during the forward and backward propagation to update the parameters of the model.
- ✓ Gradient descent will eventually converge to a stationary point of the function, regardless of convexity. If the function is convex, this will be a global minimum, but if not, it could be a local minimum or even a saddle point.



Gradient Descent IV

- ✓ if f is convex there are no saddle points and all local minima are also global. Thus GD (with a suitable stepsize) is guaranteed to find a global minimizer.
- ✓ What makes non-convex optimization hard is the presence of saddle points and local minima, where the gradient is $(0, \dots, 0)$ and that have an arbitrarily bad objective value.
- ✓ Finding the global minimizer in such a setting is generally NP-hard and one instead settles with the goal of finding a local minimizer.
- ✓ The probability of GD to get stuck at a saddle is actually 0.
- ✓ However, the presence of saddle points might severely slow GDs progress down because directions of low curvature are exploited too slowly (see here)

Gradient Descent V

Problem of solving for the minimum of a real-valued function:

$$\min_x f(x), \quad (27)$$

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is an objective function that captures the machine learning problem at hand. It is assumed that function is differentiable, and are unable to analytically find a solution in closed form.

Gradient descent is a first-order optimization algorithm

- ✓ Consider multivariate functions. Gradient descent exploits the fact that $f(x_0)$ decreases fastest if one moves from x_0 in the direction of the negative gradient $-((\nabla f)(x_0))^T$ of f at x_0 .

$$x_1 = x_0 - \gamma((\nabla f)(x_0))^T \quad (28)$$

for a small *step-size* $\gamma \geq 0$, then $f(x_1) \leq f(x_0)$.

Gradient Descent VI

- ✓ To find a local optima $f(x_*)$ of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $x \mapsto f(x)$, start with an initial guess x_0 of the parameters and then iterate according to

$$x_{i+1} = x_i - \gamma_i ((\nabla f)(x_i))^T \quad (29)$$

for suitable step-size γ_i , the sequence $f(x_0) \geq f(x_1) \geq \dots$ converge to a local minimum.

Example1: $2x_1^2 + 3x_2^2 + 6x_1x_2$

$$\equiv [x_1 \ x_2] \begin{bmatrix} 2 & 3 \\ 3 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (30)$$

where $A = A'$

Example2: Consider a quadratic function in two dimensions

Gradient Descent VII

$$f(x_1, x_2) = x_1^2 + 10x_2^2 - 5x_1 - 3x_2 + x_1x_2 \quad (31)$$

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (32)$$

with gradient

$$\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T \quad (33)$$

Starting at the initial location $x_0 = [-3, -1]^T$, Equation (29) iteratively applied to obtain a sequence of estimates that converge to the minimum value given in Figure 2.



Gradient Descent VIII

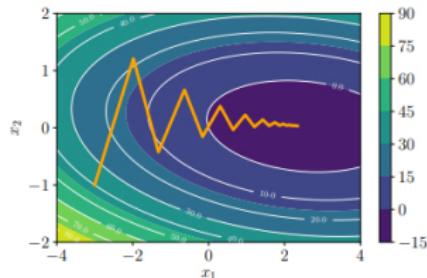


Figure: Gradient descent on a two-dimensional quadratic surface



Gradient Descent IX

Example3: Quadratic function minimization

$$f(x, y, z) = \frac{1}{2} [x \ y \ z] \begin{bmatrix} 4 & 1 & 2 \\ 1 & 8 & 5 \\ 2 & 5 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} - [x \ y \ z] \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{x}^T b$$

$$f(x, y, z) = 2x^2 + 4y^2 + 2z^2 + xy + 5yz + 2zx - 2x - 3y - 4z$$

$$\begin{cases} f_x = 4x + y + 2z - 2 = 0 \\ f_y = x + 8y + 5z - 3 = 0 \\ f_z = 2x + 5y + 4z - 4 = 0 \end{cases}$$

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & 8 & 5 \\ 2 & 5 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$



Cost Function I



Cost Function II



Cost Function III

- ✓ It is a function that measures the performance of a model for any given data.
- ✓ Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number.
- ✓ After making a hypothesis with initial parameters, Cost function is calculated.
- ✓ With a goal to reduce the cost function modify the parameters by using the Gradient descent algorithm over the given data (m , number of observations).

- **Hypothesis:** $h_{\theta}(x) = \theta_0 + \theta_1 x$
- **Parameters:** θ_0, θ_1
- **Cost Function:** $\mathcal{J}(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$
- **Gradient:** $\frac{\partial \mathcal{J}(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot X_j^{(i)}$
- **Goal:** $\min_{\theta_0, \theta_1} \mathcal{J}(\theta_0, \theta_1)$

Cost Function IV

$$\theta_0 := \theta_0 - \alpha \cdot \left(\frac{1}{m} \cdot \sum_{i=1}^m \left(h(\theta^{(i)}) - y^{(i)} \right) \cdot X_0^{(i)} \right) \quad (34)$$

$$\theta_1 := \theta_1 - \alpha \cdot \left(\frac{1}{m} \cdot \sum_{i=1}^m \left(h(\theta^{(i)}) - y^{(i)} \right) \cdot X_1^{(i)} \right) \quad (35)$$

$$\theta_2 := \theta_2 - \alpha \cdot \left(\frac{1}{m} \cdot \sum_{i=1}^m \left(h(\theta^{(i)}) - y^{(i)} \right) \cdot X_2^{(i)} \right) \quad (36)$$

$$\vdots \quad (37)$$

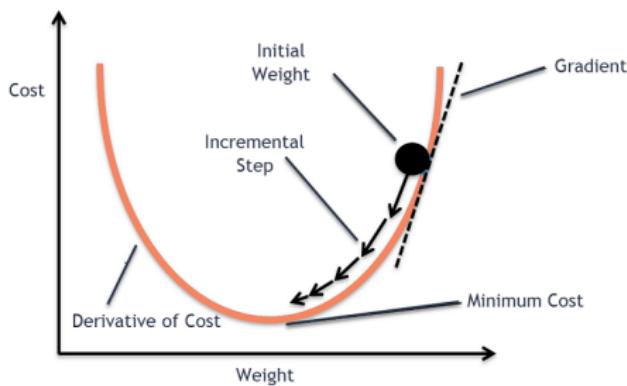
$$\theta_j := \theta_j - \alpha \cdot \left(\frac{1}{m} \cdot \sum_{i=1}^m \left(h(\theta^{(i)}) - y^{(i)} \right) \cdot X_j^{(i)} \right) \quad (38)$$

- ✓ Gradient descent is an iterative optimization algorithm for finding the local minimum of a function.



Cost Function V

- ✓ To find the local minimum of a function using gradient descent, we must take steps proportional to the negative of the gradient (move away from the gradient) of the function at the current point.
- ✓ If we take steps proportional to the positive of the gradient (moving towards the gradient), we will approach a local maximum of the function, and the procedure is called Gradient Ascent.



Cost Function VI

- ✓ Gradient descent was originally proposed by CAUCHY in 1847. It is also known as steepest descent.
- ✓ The goal of the gradient descent algorithm is to minimize the given function (say cost function). To achieve this goal, it performs two steps iteratively:
 - 1 Compute the gradient (slope), the first order derivative of the function at that point
 - 2 Make a step (move) in the direction opposite to the gradient, opposite direction of slope increase from the current point by alpha times the gradient at that point.

Cost Function VII

Algorithm 1 Gradient descent algorithm

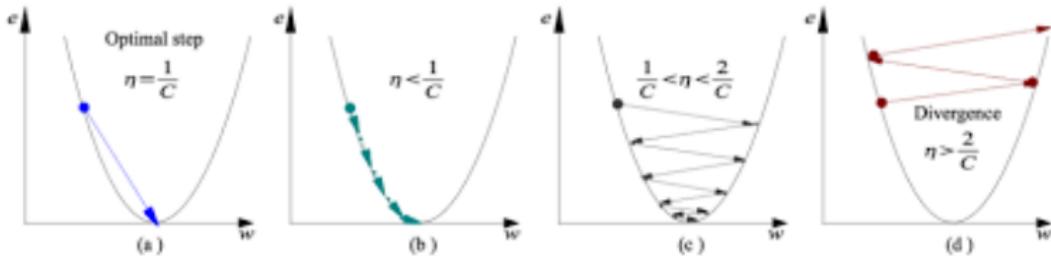
```
1: while (repeat until convergence) do  
2:    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta_0, \theta_1);$   
3: end while
```

Alpha (α) : The learning rate

✓ After having the direction, we must decide the size of the step we must take.

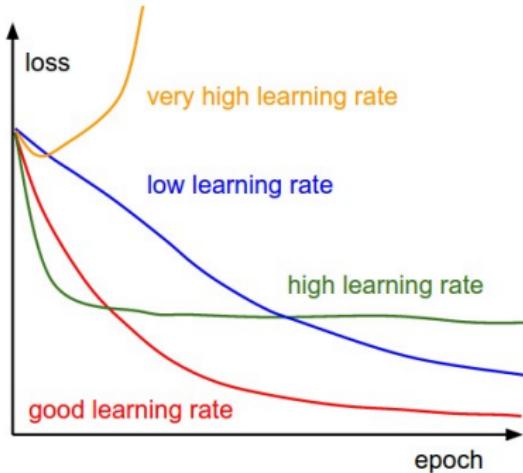
- Step-size must be chosen carefully to end up with local minima.
- If the learning rate is too high, we might OVERSHOOT the minima and keep bouncing, without reaching the minima
- If the learning rate is too small, the training might turn out to be too long

Cost Function VIII



- A Learning rate is optimal, model converges to the minimum
- B Learning rate is too small, it takes more time but converges to the minimum
- C Learning rate is higher than the optimal value, it overshoots but converges ($1/C < \eta < 2/C$)
- D Learning rate is very large, it overshoots and diverges, moves away from the minima, performance decreases on learning

Cost Function IX



As the gradient decreases while moving towards the local minima, the size of the step decreases. So, the learning rate (α) can be constant over the optimization and need not be varied iteratively.

Variants of Gradient Descent I

① Batch Gradient Descent (BGD): If all the observations in data set is used to calculate the cost function.

- ✓ Take the entire training set, perform forward propagation and calculate the cost function.
- ✓ An epoch is when the entire training set is passed through the model, forward propagation and backward propagation are performed and the parameters are updated.

One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.

- ✓ In batch Gradient Descent since the entire training set is used, the parameters will be updated only once per epoch.



Variants of Gradient Descent II

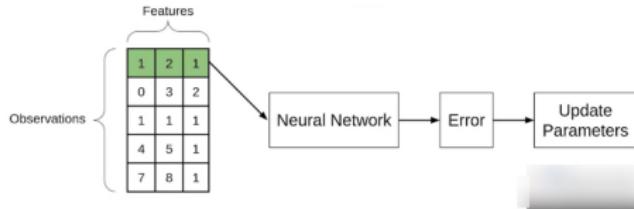
- 2 **Stochastic Gradient Descent(SGD):** A single observation is used to calculate the cost function.

✓ Let's say we have 5 observations and each observation has three features and the values that have taken are completely random.

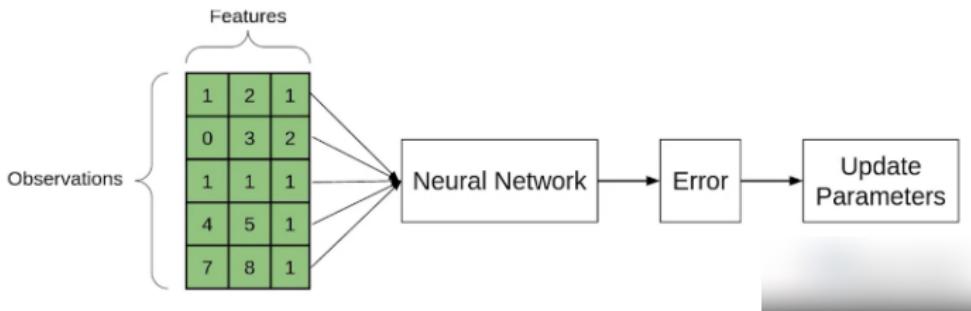
Features			
Observations	1	2	1
	0	3	2
	1	1	1
	4	5	1
	7	8	1

✓ Take the first observation, then pass it through the neural network, calculate the error and then update the parameters.

Variants of Gradient Descent III



- ✓ Then will take the second observation and perform similar steps with it. This step will be repeated until all observations have been passed through the network and the parameters have been updated.



Variants of Gradient Descent IV

- ✓ Each time the parameter is updated, it is known as an Iteration. Here since we have 5 observations, the parameters will be updated 5 times or we can say that there will be 5 iterations.
- ✓ In SGD, there will be ‘m’ iterations per epoch, where ‘m’ is the number of observations in a dataset.

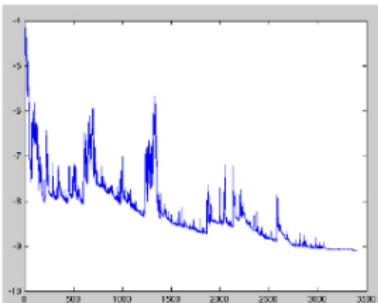


Figure: SGD fluctuation

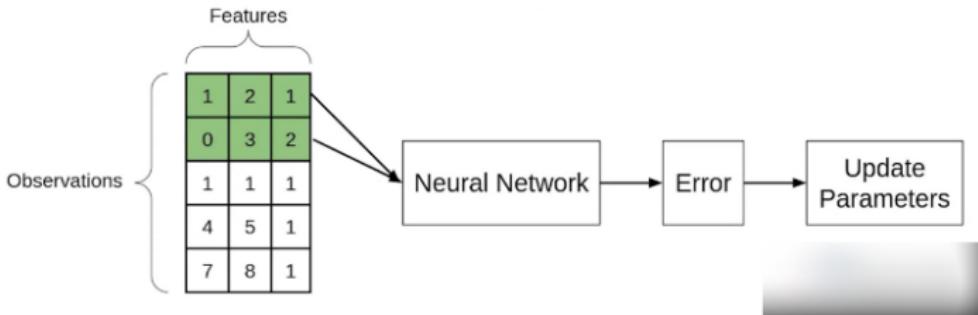
Variants of Gradient Descent V

- ③ **Mini-batch Gradient Descent:** It takes a subset of the entire dataset to calculate the cost function. So if there are ‘m’ observations then the number of observations in each subset or mini-batches will be more than 1 and less than ‘m’.

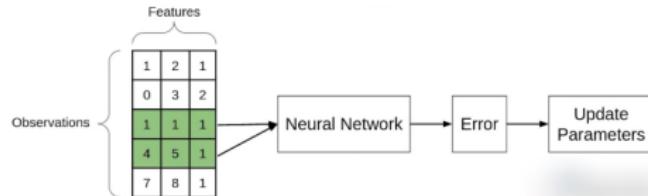
- ✓ Assume that the batch size is 2. So we’ll take the first two observations, pass them through the neural network, calculate the error and then update the parameters.
- ✓ Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

$$\theta = \theta - \eta \cdot \nabla_{\theta} \mathcal{J} \left(\theta; x^{(i:i+n)}; y^{(i:i+n)} \right) \quad (39)$$

Variants of Gradient Descent VI

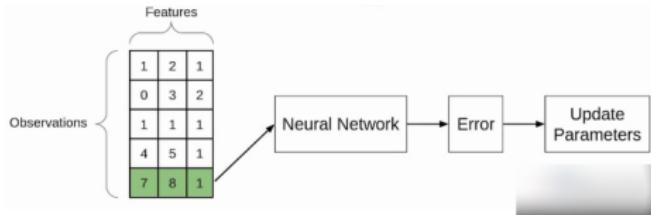


- ✓ Then take the next two observations and perform similar steps i.e will pass through the network, calculate the error and update the parameters.



Variants of Gradient Descent VII

- ✓ Now since we are left with the single observation in the final iteration, there will be only a single observation and will update the parameters using this observation.



- Reduces the variance of the parameter updates, which can lead to more stable convergence;
- Can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.
- Common mini-batch sizes range between 50 and 256, but can vary for different applications.

Variants of Gradient Descent VIII

- Mini-batch gradient descent is typically the algorithm of choice when training a neural network and the term SGD usually is employed also when mini-batches are used.



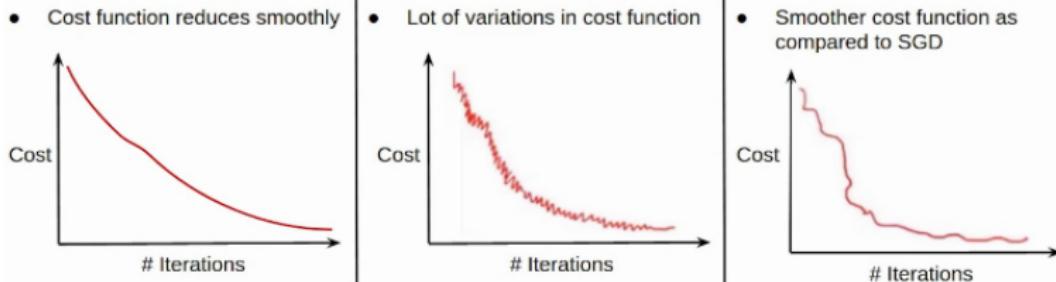
Variants of Gradient Descent IX

Comparison between Batch GD, SGD, and Mini-batch GD:

✓ Comparison: Number of observations used for Updation:

- | Batch Gradient Descent | Stochastic Gradient Descent (SGD) | Mini-Batch Gradient Descent |
|-------------------------------|-----------------------------------|-------------------------------|
| • Entire dataset for updation | • Single observation for updation | • Subset of data for updation |

Comparison: Cost function:



Variants of Gradient Descent X

- Now since we update the parameters using the entire data set in the case of the Batch GD, the cost function, in this case, reduces smoothly.
- On the other hand, this updation in the case of SGD is not that smooth. Since we're updating the parameters based on a single observation, there are a lot of iterations. It might also be possible that the model starts learning noise as well.
- The updation of the cost function in the case of Mini-batch Gradient Descent is smoother as compared to that of the cost function in SGD. Since we're not updating the parameters after every single observation but after every subset of the data.

Variants of Gradient Descent XI

Comparison: Computation Cost and Time

Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
<ul style="list-style-type: none">Entire dataset for updationCost function reduces smoothlyComputation cost is very high	<ul style="list-style-type: none">Single observation for updationLot of variations in cost functionComputation time is more	<ul style="list-style-type: none">Subset of data for updationSmoother cost function as compared to SGDComputation time is lesser than SGDComputation cost is lesser than Batch Gradient Descent

- Since it needs to load the entire data set at a time, perform the forward propagation on that and calculate the error and then update the parameters, the computation cost in the case of Batch gradient descent is very high.

Variants of Gradient Descent XII

- Computation cost in the case of SGD is less as compared to the Batch Gradient Descent since it needs to load every single observation at a time but the Computation time here increases as there will be more number of updates which will result in more number of iterations.
- In the case of Mini-batch Gradient Descent, taking a subset of the data there are a lesser number of iterations or updatations and hence the computation time in the case of mini-batch gradient descent is less than SGD.
 - ✓ Also, since it does not load the entire dataset at a time whereas loading a subset of the data, the computation cost is also less as compared to the Batch gradient descent.
 - ✓ **This is the reason Mini-batch gradient descent is preferred.**

Momentum I

- ✓ SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima.
- ✓ In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.
- ✓ **Momentum** is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a fraction γ of the update vector of the past time step to the current update vector:

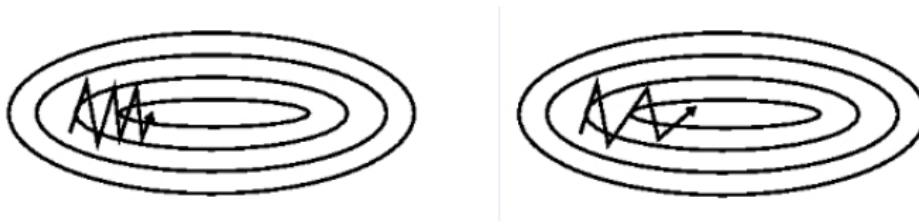


Figure: SGD without momentum and with momentum

Momentum II

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} \mathcal{J}(\theta) \quad (40)$$

$$\theta = \theta - v_t \quad (41)$$



Different gradient descent algorithm I

- ① Nesterov accelerated gradient: A limitation of gradient descent is that the progress of the search can slow down if the gradient becomes flat or large curvature. Momentum can be added to gradient descent that incorporates some inertia to updates. This can be further improved by incorporating the gradient of the projected new position rather than the current position, called Nesterov's Accelerated Gradient (NAG) or Nesterov momentum.
- ② Adagrad(Adaptive Gradient Algorithm): It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features. **For this reason, it is well-suited for dealing with sparse data.**
 - ✓ It greatly improved the robustness of SGD and used it for training large-scale neural nets at Google, which – among other things – learned to recognize cats in Youtube videos.¹

Different gradient descent algorithm II

- ③ Adadelta: Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients.
 - ✓ With Adadelta, we do not even need to set a default learning rate, as it has been eliminated from the update rule.
- ④ RMSprop(Root Mean Squared Propagation): is an extension of gradient descent and the AdaGrad version of gradient descent that uses a decaying average of partial gradients in the adaptation of the step size for each parameter.

Different gradient descent algorithm III

- 5 Adam (Adaptive Moment Estimation): is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t , similar to momentum.

✓ Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface.

Benefits of using Adam on non-convex optimization problems:

- Straightforward to implement.
- Computationally efficient.
- Little memory requirements.
- Invariant to diagonal rescale of the gradients.
- Well suited for problems that are large in terms of data and/or parameters.
- Appropriate for non-stationary objectives.

Different gradient descent algorithm IV

- Appropriate for problems with very noisy/or sparse gradients.
- Hyper-parameters have intuitive interpretation and typically require little tuning.

Parallelizing and distributing SGD:

- Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent.
- Downpour SGD: Large Scale Distributed Deep Networks. It is an asynchronous stochastic gradient descent procedure which leverages adaptive learning rates and supports a large number of model replicas
- TensorFlow

Different gradient descent algorithm V

- Elastic Averaging SGD: the communication and coordination of work among concurrent processes (local workers), is based on an elastic force which links the parameters they compute with a center variable stored by the parameter server (master). The algorithm enables the local workers to perform more exploration, i.e. the algorithm allows the local variables to fluctuate further from the center variable by reducing the amount of communication between local workers and the master.

CS322: Deep Learning

Hypothesis

Sachchida Nand Chaurasia
Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



October 7, 2021

Outline

Hypothesis Testing

Types of errors

Bias & Variance, Best fitting, Overfitting and Underfitting



Outline

Hypothesis Testing

Types of errors

Bias & Variance, Best fitting, Overfitting and Underfitting



Uniform Distribution I

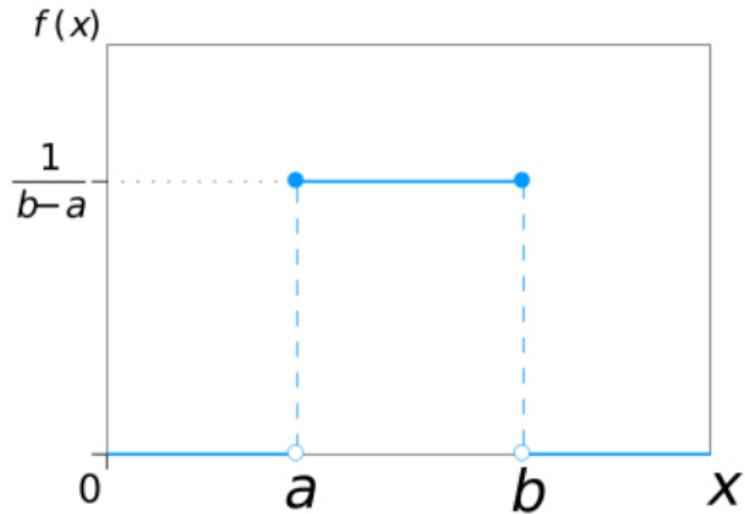
✓ Perhaps one of the simplest and useful distribution is the uniform distribution. The probability distribution function of the continuous uniform distribution is:

$$f(x) = \begin{cases} \frac{1}{(b-a)}, & \text{for } a \leq x \leq b \\ 0, & x < a \text{ or } x > b \end{cases}$$

- ✓ Since any interval of numbers of equal width has an equal probability of being observed, the curve describing the distribution is a rectangle, with constant height across the interval and 0 height elsewhere.
- ✓ Since the area under the curve must be equal to 1, the length of the interval determines the height of the curve. The following figure shows a uniform distribution in interval (a, b) . Notice since the area needs to be 1. The height is set to $\frac{1}{(b-a)}$.



Uniform Distribution II



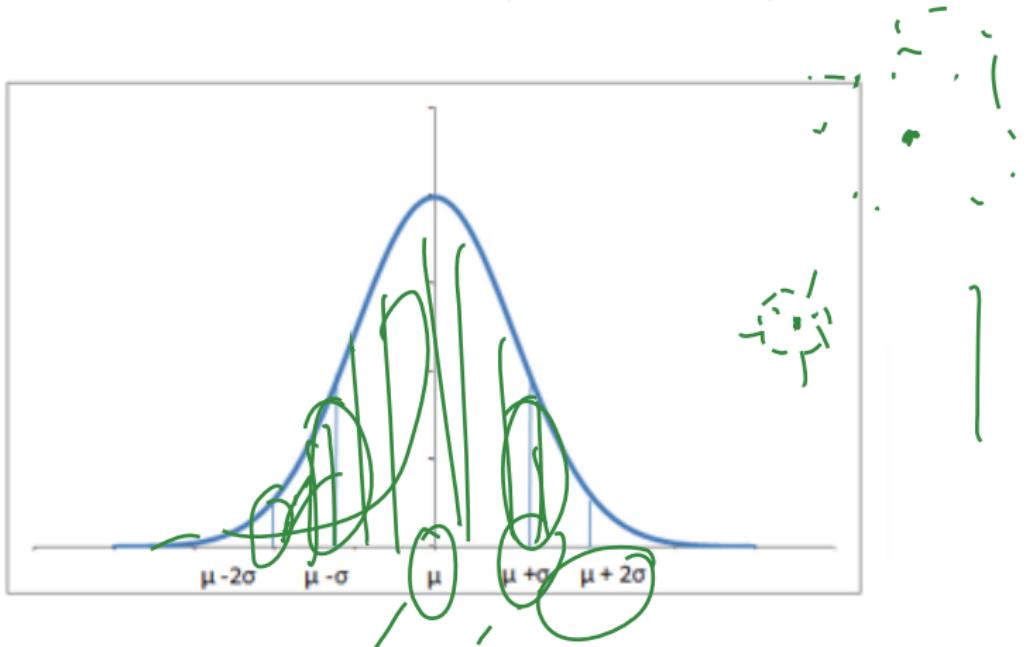
Normal Distribution I

- ✓ Normal Distribution, also known as Gaussian distribution, is ubiquitous in Data Science.
- ✓ You will encounter it at many places especially in topics of statistical inference.
- ✓ It is one of the assumptions of many data science algorithms too.
- ✓ A normal distribution has a bell-shaped density curve described by its mean μ and standard deviation σ .
- ✓ The density curve is symmetrical, centered about its *mean*, with its spread determined by its standard deviation showing that data near the mean are more frequent in occurrence than data far from the mean.
- ✓ The probability distribution function of a normal density curve with mean μ and standard deviation σ at a given point x is given by:



Normal Distribution II

$$f(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{x-\mu}{\sigma}\right)^2\right)$$



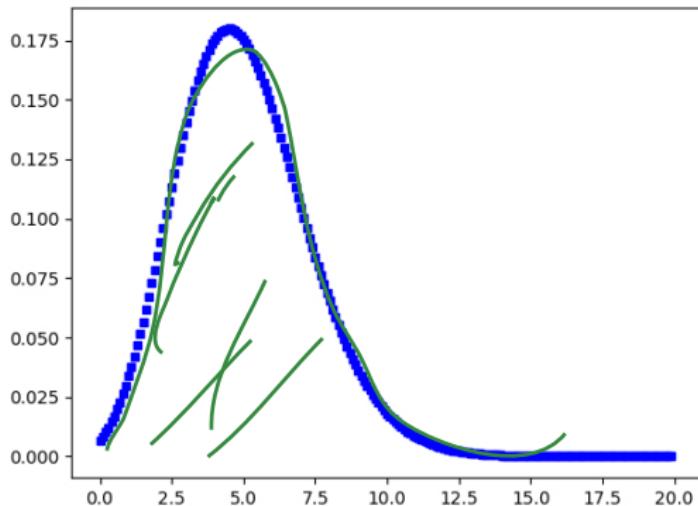
Poisson Distribution I

- ✓ Poisson random variable is typically used to model the number of times an event happened in a time interval.
- ✓ For example, the number of users visited on a website in an interval can be thought of a Poisson process.
- ✓ Poisson distribution is described in terms of the rate (μ) at which the events happen.
- ✓ An event can occur 0, 1, 2, ... times in an interval. The average number of events in an interval is designated λ .
- ✓ λ is the event rate, also called the rate parameter.
- ✓ The probability of observing k events in an interval is given by the equation:

$$P(k \text{ events in interval}) = e^{-\lambda} \frac{\lambda^k}{k!}$$



Poisson Distribution II



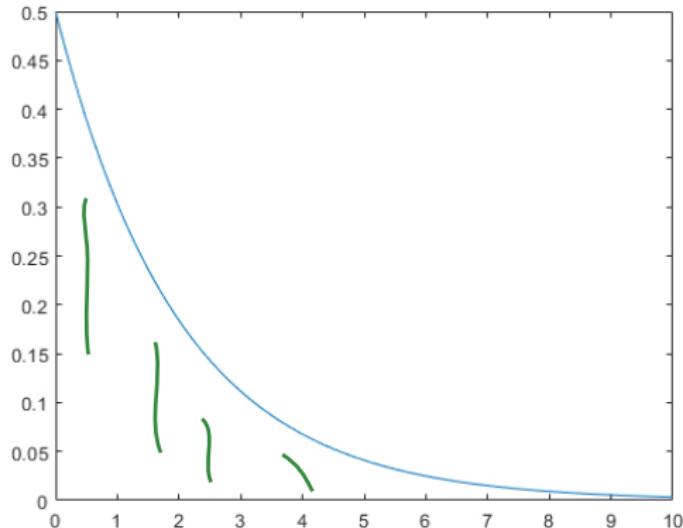
Exponential Distribution Function I

- ✓ The exponential distribution describes the time between events in a Poisson point process, i.e., a process in which events occur continuously and independently at a constant average rate.
- ✓ It has a parameter λ called rate parameter, and its equation is described as :

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$



Exponential Distribution Function II



Hypothesis Test I

Conduct and interpret results from a hypothesis test about a population mean.

- Recognize when to use a hypothesis test or a confidence interval to draw a conclusion about a population mean.
- Under appropriate conditions, conduct a hypothesis test about a population mean. State a conclusion in context.

Hypothesis Test II

Inference for Means: Focus is on inference when the variable is quantitative.

Here parameters and statistics are means.

Estimating a Population Mean: Learned how to use a sample mean to calculate a confidence interval. The confidence interval estimates a population mean.

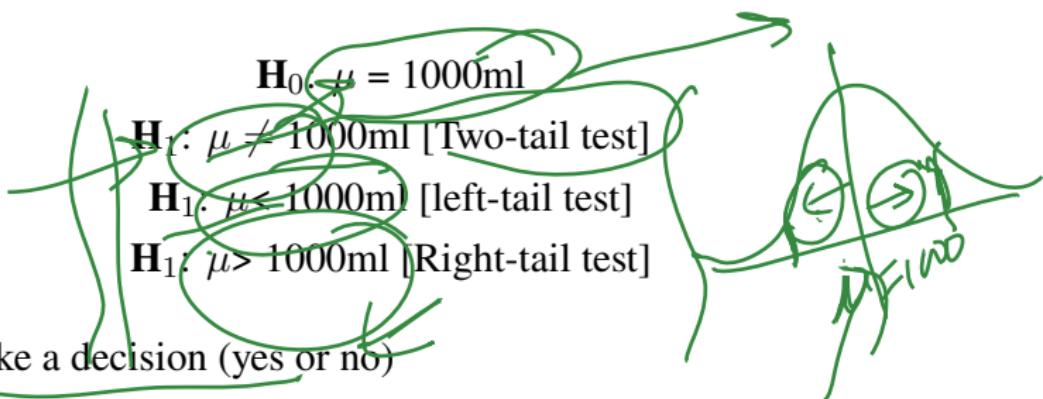
- ✓ Hypothesis testing is a systematic procedure for deciding whether the results of a research study support a particular theory which applies to a population.
- ✓ Hypothesis testing uses sample data to evaluate a hypothesis about a population.
- ✓ The first step in hypothesis test is setting of hypothesis.

① Null hypothesis: It is a claim or statement about population parameter that is assumed to be true until it declared to be false. It is denoted by H_0

Hypothesis Test III

- ② Alternative hypothesis (Research hypothesis): Any hypothesis which is complementary to null hypothesis. It is denoted by H_1 .

Example:



- ✓ H_0 is to make a decision (yes or no)
- ✓ The alternative hypothesis give an answer for a research question about validity of claim. For example: $<1000\text{ml}$ or $>1000\text{ml}$.

Hypothesis Test IV

Computation of test statistic:

Test of significance:

z-test, t-test, Chi-test, f-test are some frequently used testing method.

z-test:

$n \geq 30$, σ = S.D. of population is known, then

$$Z = \frac{\bar{X} - \mu}{\sigma_{\bar{x}}} \sim \mathcal{N}(0, 1) \quad (1)$$

$$\sigma_{\bar{X}} = \text{Standard error of mean} = \frac{\sigma}{\sqrt{n}}$$

$$Z = \frac{\bar{X} - \mu}{\sigma_{\bar{x}}} \sim \mathcal{N}(0, 1) \quad (2)$$

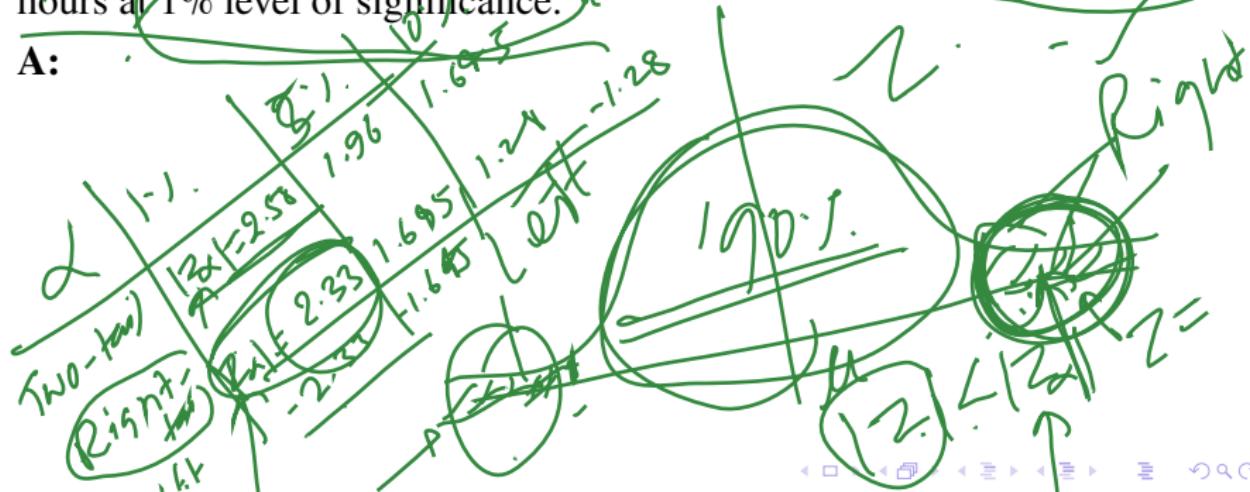


Hypothesis Test V

$$\sigma_{\bar{X}} = \text{Standard error of mean} = \frac{S_\sigma}{\sqrt{n}}, (S_\sigma) \rightarrow \text{Sample S.D.}$$

Q: The mean lifetime of a sample of 400 fluorescent light tube produced by a company is found to be 1570 hours with a standard deviation of 150 hours. Test the hypothesis that the mean lifetime the bulbs produced by the company is 1600 hours against the alternative hypothesis that it is greater than 1600 hours at 1% level of significance.

A:

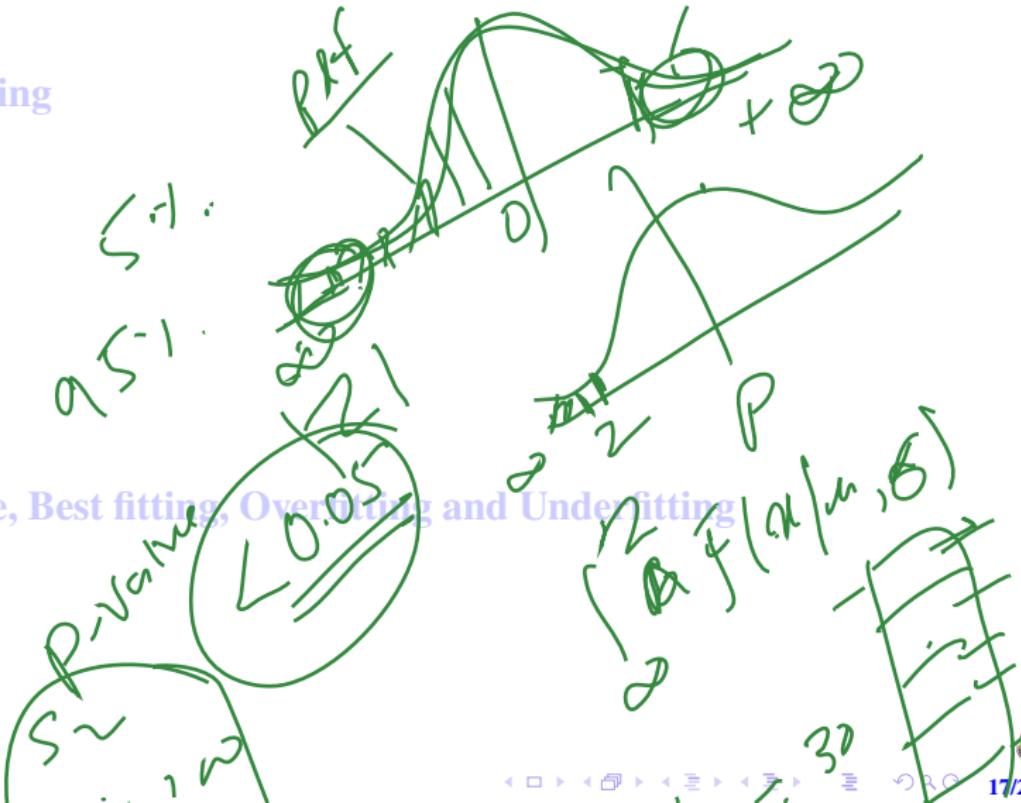


Outline

Hypothesis Testing

Types of errors

Bias & Variance, Best fitting, Overfitting and Underfitting



Type I and Type II errors I

- ✓ In statistical hypothesis testing, a type I error is the **mistaken rejection** of the null hypothesis (also known as a "false positive" finding or conclusion; **Example: "an innocent person is convicted"**)
- ✓ While a type II error is the mistaken acceptance of the null hypothesis (also known as a "false negative" finding or conclusion; **Example: "a guilty person is not convicted"**).
- ✓ When we are testing Null Hypothesis(H_0) against Alternative Hypothesis(H_1) there are four possibilities:

- ① H_0 accepted when H_0 is true [Correct]
- ② H_0 rejected when H_0 is true [Type I error]
- ③ H_0 accepted when H_0 is false [Type II error]

Type I and Type II errors II

- ④ H_0 rejected when H_0 is false [Correct]

Probability of types of errors:

Type I error	Type II error
$\alpha = p(\text{Type I error})$	$\beta = p(\text{Type II error})$
$\alpha = p(\text{reject } H_0 H_1 \text{ is true})$	$\beta = p(\text{accept } H_0 H_1 \text{ is true})$

+ + -

+

Outline

Hypothesis Testing

Types of errors

Bias & Variance, Best fitting, Overfitting and Underfitting



Bias and Variance I

- ✓ It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine learning algorithm.
- ✓ There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of Regularization constant.
- ✓ A proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

Bias:

- ✓ The bias is known as the difference between the prediction of the values by the ML model and the correct value.
- ✓ Being high in biasing gives a large error in training as well as testing data.
- ✓ It is recommended that an algorithm should always be low biased to avoid the problem of underfitting.

Bias and Variance II

- ✓ By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as Underfitting of Data. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.

Variance:

- ✓ The variability of model prediction for a given data point which tells us spread of our data is called the variance of the model.
- ✓ The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.
- ✓ When a model is high on variance, it is then said to as Overfitting of Data. Overfitting is fitting the training set accurately via complex curve and high



Bias and Variance III

order hypothesis but is not the solution as the error with unseen data is high.
While training a data model variance should be kept low.

Bias and variance tradeoff:

- ✓ In statistics and machine learning, the bias–variance trade-off is the property of a model that the variance of the parameter estimated across samples can be reduced by increasing the bias in the estimated parameters.
- ✓ The bias–variance dilemma or bias–variance problem is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set.
- ✓ The bias error is an error from erroneous assumptions in the learning algorithm.
- ✓ *Unfortunately, it is typically impossible to do both simultaneously.*



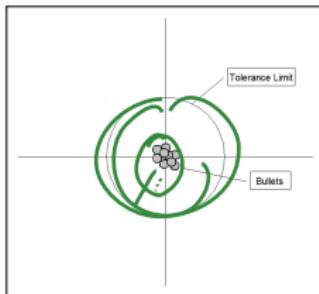
Bias and Variance IV

High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting)

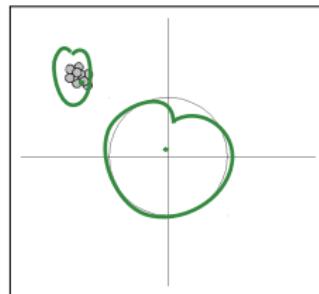
The variance is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting)

- ✓ The bias–variance decomposition is a way of analyzing a learning algorithm's expected generalization error with respect to a particular problem as a sum of three terms, the bias, variance, and a quantity called the irreducible error, resulting from noise in the problem itself.

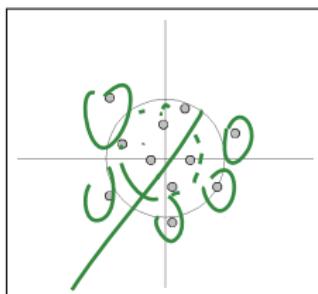
Bias and Variance V



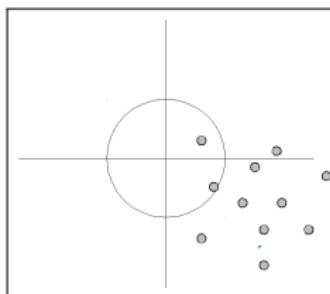
(a) low bias and low variance



(b) high bias and low variance



(c) low bias and high variance



(d) high bias and high variance



Bias and Variance VI

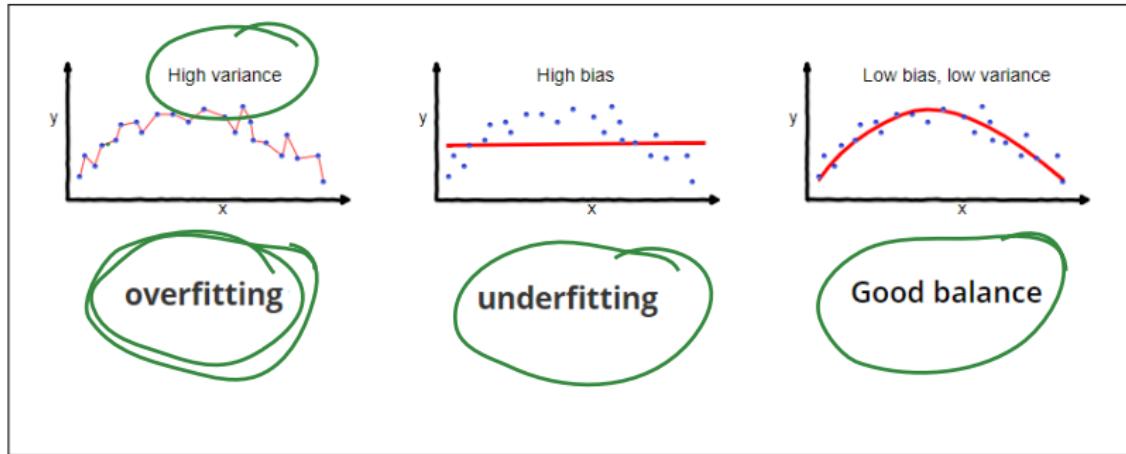


Figure 1: Fitting



Bias and Variance VII

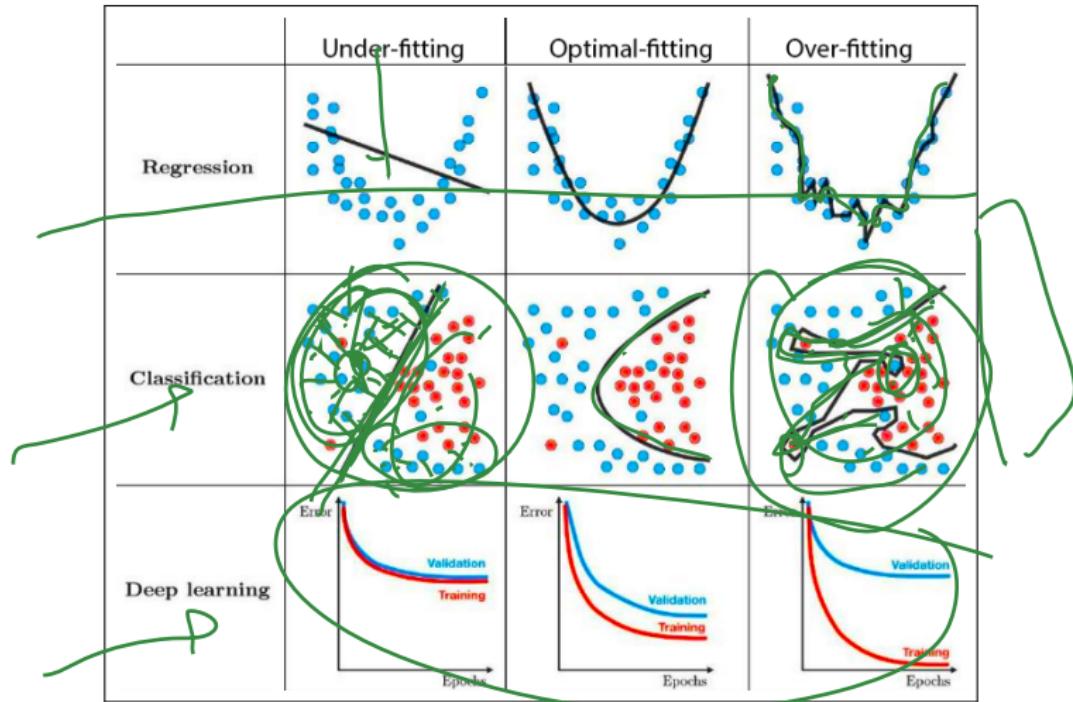


Figure 2: Fitting

Bias and Variance VIII

- ✓ An epoch is a term used in machine learning and indicates the number of passes of the entire training dataset the machine learning algorithm has completed. Datasets are usually grouped into batches (especially when the amount of data is very large).



Bias and Variance IX

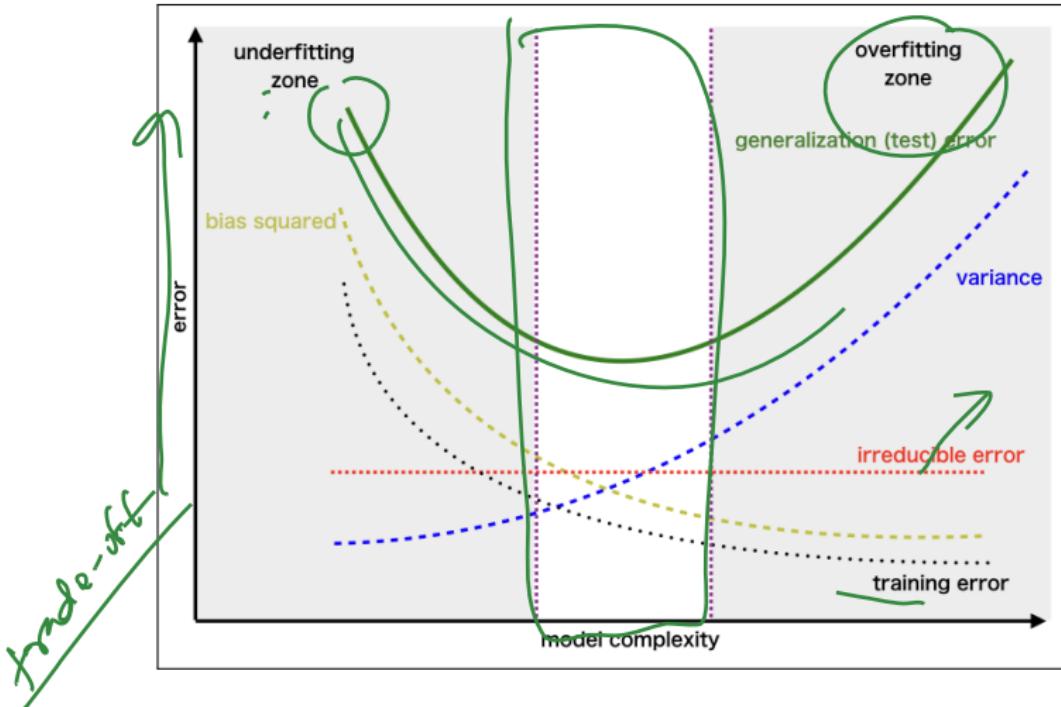


Figure 3: Bias and variance as function of model complexity



CS322: Deep Learning

Logistic Regression Model

Sachchida Nand Chaurasia
Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



October 28, 2021

Maximum likelihood I

Different ways of expressing probability

- Consider a two-outcome probability space, where:

$$\rightarrow p(O_1) = p$$

$$\rightarrow p(O_2) = 1 - p = q$$

- Can express probability of O_1 as

	notation	range equivalent		
standard probability	p	0	0.5	1
odds	p/q	0	1	$+\infty$
log odds (logit)	$\log(p/q)$	$-\infty$	0	$+\infty$

Maximum likelihood II

Log odds

Odds : odds are the chances of success divided by the chances of failure. It is represented in the form of a ratio.

$$Odd\ ratio = \frac{p}{1-p} \quad (1)$$

where, p: success odds, and $(1-p)$ = failure odds.

- ✓ Log odds play an important role in logistic regression as it converts the LR model from probability based to a likelihood based model.



Maximum likelihood III

- ✓ Both probability and log odds have their own set of properties, however log odds makes interpreting the output easier. Thus, using log odds is slightly more advantageous over probability.
- Numeric treatment of outcomes O_1 and O_2 is equivalent:
 - ✓ If neither outcome is favored over the other, then $\log \text{odds} = 0$
 - ✓ If one outcome is favored with $\log \text{odds} = x$, then other outcome is disfavored with $\log \text{odds} = -x$.
- Especially useful in domains where relative probabilities can be tiny
 - ✓ Example: multiple sequence alignment in computational biology.



Maximum likelihood IV

From probability to log odds(and back again)

$$z = \log \left(\frac{p}{1-p} \right) \quad \text{logit function}$$

$$\frac{p}{1-p} = e^z$$

$$p = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}} \quad \text{logistic function}$$

Maximum likelihood V

Standard logistic function

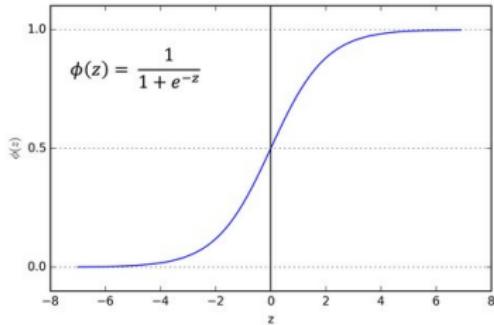
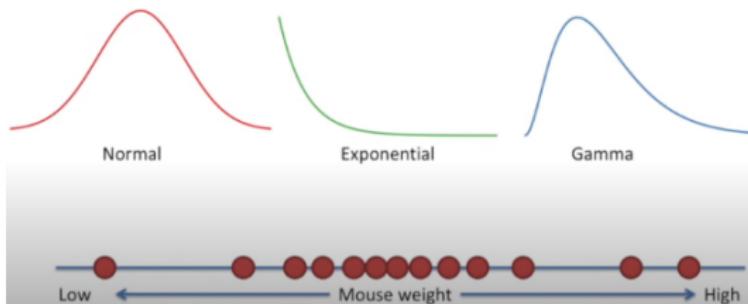


Figure: Standard logistic function

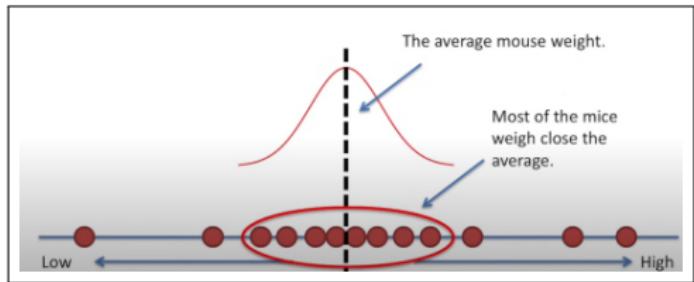
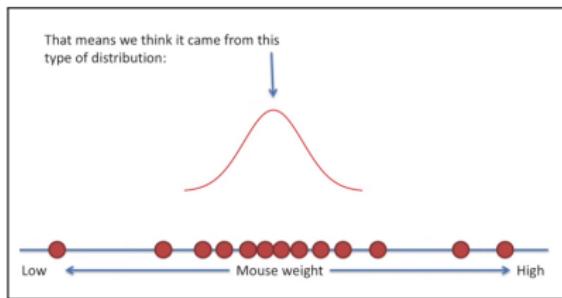
Maximum likelihood VI

- ✓ There are several types of distribution for different type of data.



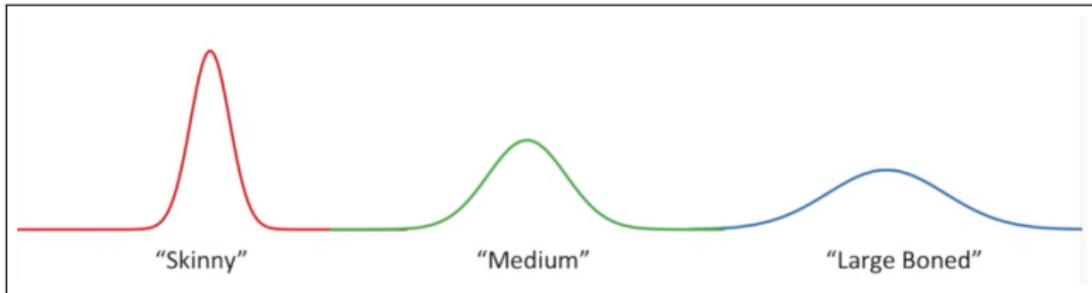
Maximum likelihood VII

- ✓ Normally distributed means a number of things.
- ✓ We expect most of the measurements (mouse weights) to be close to the mean (average).
- ✓ We expect the measurements to be relatively symmetrical around the mean.
- ✓ Although the measurements are not perfectly symmetrical around the mean, they are not skewed to one side, either.



Maximum likelihood VIII

- ✓ Normal distributions come in all kinds of shapes and sizes.



Maximum likelihood IX

This distribution says "most of the values you measure should be near my average!"

Unfortunately, most of the values we measured are far from the distribution's average.

According to a normal distribution with a mean value over here...

...the probability, or "likelihood" of observing all these weights is low.

According to a normal distribution with a mean value here...

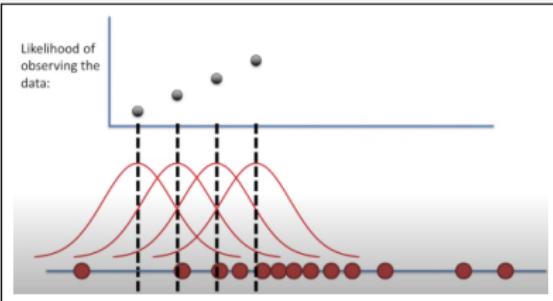
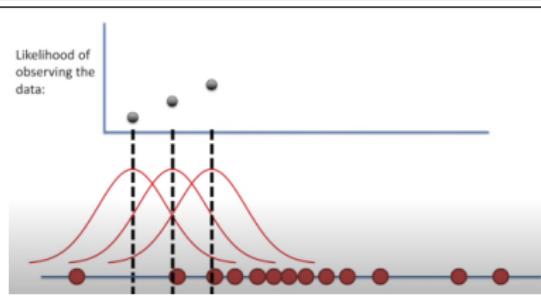
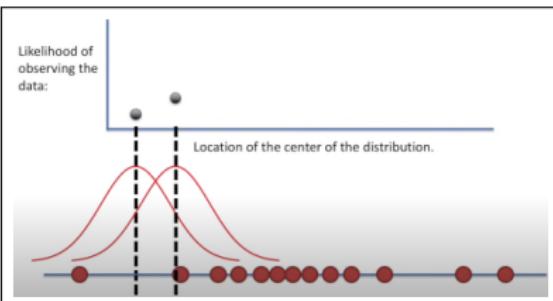
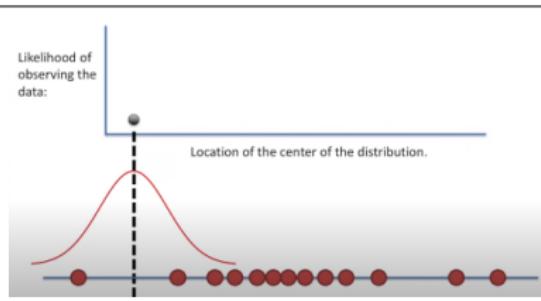
...the probability, or "likelihood" of observing these weights is relatively high.

If we kept shifting the normal distribution over...

... then the probability, or "likelihood", of observing these measurements would go down again.



Maximum likelihood X



Maximum likelihood XI

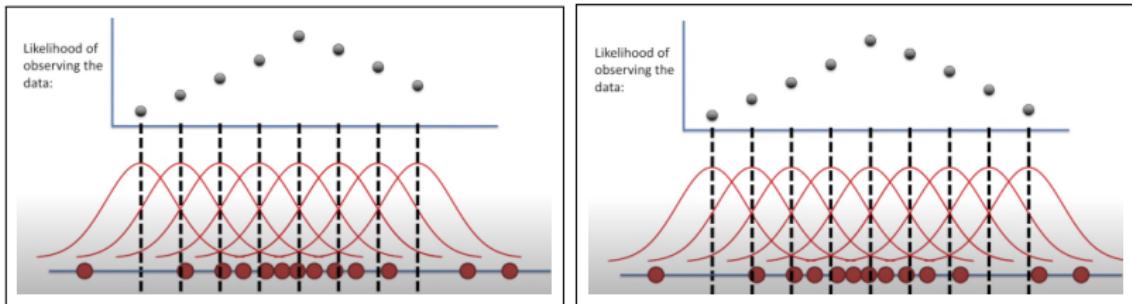
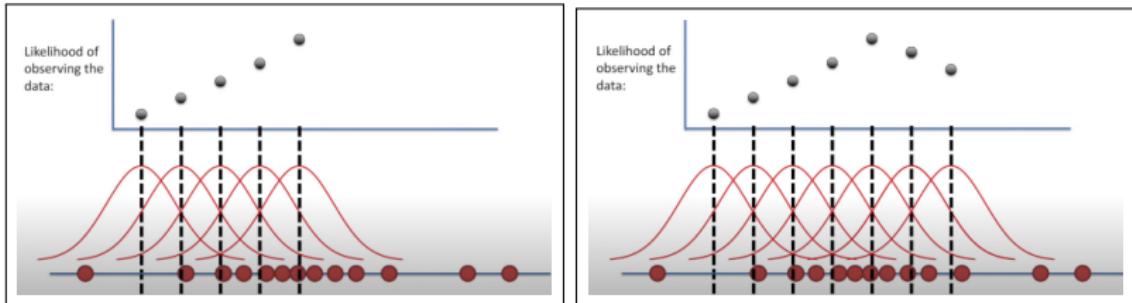
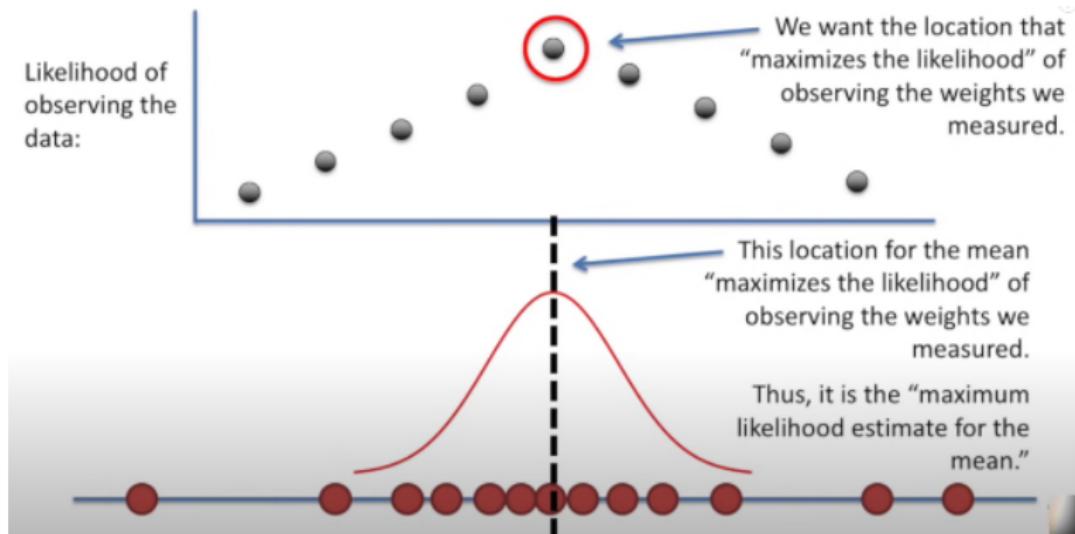


Figure: Different likelihood for different μ

Maximum likelihood XII

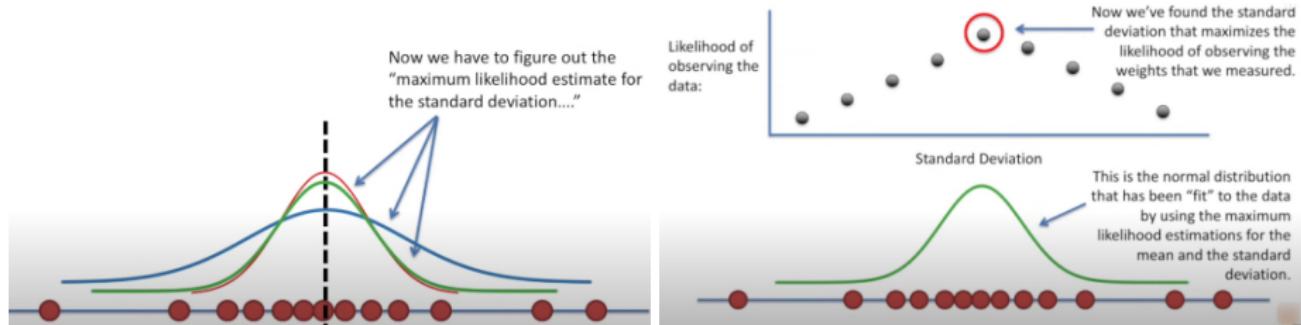


Maximum likelihood XIII



Maximum likelihood XIV

- ✓ Similarly, maximum likelihood estimation for standard deviation can be measured.



Maximum likelihood for normal distribution I

Motivation: In supervised machine learning, cost functions are used to measure a trained model's performance.

- ✓ The most commonly used cost function for regression is the Mean Squared Error (MSE), and Cross Entropy for binary classification problems.
- ✓ Given a training set of n examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})$ where $x^{(i)}$ is the feature vector for i^{th} training example and $y^{(i)}$ is its target.
- ✓ The goal of supervised learning is to learn a model $f : \mathcal{X} \rightarrow \mathcal{Y}$, a mapping that given $x \in \mathcal{X}$ outputs a prediction $\hat{y} \in \mathcal{Y}$.
- ✓ \mathcal{X} and \mathcal{Y} are called input space and output space respectively.

Maximum likelihood for normal distribution II

- ✓ In order to measure how well the model fits our training data, we define a loss function.
- ✓ For training example $(x^{(i)}, y^{(i)})$, the loss $\mathcal{L}(x^{(i)}, y^{(i)})$ measures how different the model's prediction $\hat{y}^{(i)}$ is from the true label or value.
- ✓ The loss is calculated for all training examples, and its average taken. This value is called the cost function and is given by:

$$Cost = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y^{(i)}, \hat{y}^{(i)}) \quad (2)$$

Maximum likelihood for normal distribution III

- ✓ The model f usually has some unknown parameter θ (In general, θ is a vector of parameters) which we will try to estimate using the training set.
- ✓ We can frame supervised learning as an *optimization problem*.
- ✓ We estimate the value of θ by picking the value that minimizes the cost function.

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y^{(i)}, \hat{y}^{(i)}) \quad (3)$$



Maximum likelihood for normal distribution IV

What is Maximum Likelihood Estimation?

- ✓ Maximum Likelihood Estimation (MLE) is a method of estimating the unknown parameter θ of a model, given observed data.
- ✓ It estimates the model parameter by finding the parameter value that maximises the likelihood function.
- ✓ The parameter estimate is called the maximum likelihood estimate $\hat{\theta}_{MLE}$.
- ✓ This is achieved by maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable.
- ✓ The point in the parameter space that maximizes the likelihood function is called the **maximum likelihood estimate**.

Maximum likelihood for normal distribution V

- ✓ The logic of maximum likelihood is both intuitive and flexible, and as such the method has become a dominant means of statistical inference.

Likelihood Function:

A set of random variables X_1, X_2, \dots, X_n are said to be independent and identically distributed if they have the same probability distribution and are mutually independent.

- ✓ Given a set of independent and identically distributed random variables X_1, X_2, \dots, X_n from a probability distribution P_θ with parameter θ .



Maximum likelihood for normal distribution VI

- ✓ The assumption is that the random variables have a joint probability density function $f(x_1, x_2, \dots, x_n | \theta)$.

Suppose x_1, x_2, \dots, x_n are the observed values of the random variables, then the *likelihood function*, a function of parameter θ as :

$$\mathcal{L}(\theta | x_1, x_2, \dots, x_n) = f(x_1, x_2, \dots, x_n | \theta) \quad (4)$$

- ✓ The likelihood function measures how plausible it is that the observed data was generated by the model with a particular value of θ .
- ✓ For example, if we use θ_1 and θ_2 as values of θ and find that $\mathcal{L}(\theta_1 | x_1, x_2, \dots, x_n) > \mathcal{L}(\theta_2 | x_1, x_2, \dots, x_n)$.

Maximum likelihood for normal distribution VII

- ✓ It can be reasonably conclude that the observed data is more likely to have generated by the model with its parameter being θ_1 .

$$\mathcal{L}(\theta|x_1, x_2, \dots, x_n) = f(x_1|\theta) \times f(x_2|\theta) \times \dots, f(x_n|\theta) = \prod_{i=1}^n f(x_i|\theta) \quad (5)$$

where $f(x_i|\theta)$ is the probability density function of the random variable X_i .

- ✓ Since all the random variables are drawn from the same distribution, their probability density function will be the same.

Maximum likelihood for normal distribution VIII

$$\log (\mathcal{L}(\theta|x_1, x_2, \dots, x_n)) = \log (f(x_1|\theta) \times f(x_2|\theta) \times \dots, f(x_n|\theta)) \quad (6)$$

$$= \log (f(x_1|\theta)) + \log (f(x_2|\theta)) + \dots + \log (f(x_n|\theta)) \quad (7)$$

$$= \sum_{i=1}^n \log (f(x_i|\theta)) \quad (8)$$

- ✓ To find the value of θ that maximizes the likelihood function, find its critical point, the point at which the function's derivative is 0. That is:

Maximum likelihood for normal distribution IX

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = 0 \quad (9)$$

- ✓ Since logarithm is a monotonically increasing function, that is, if $x_1 > x_2$, then $\log(x_1) > \log(x_2)$, *the value that maximizes the likelihood is also the value that maximizes the log-likelihood function.*

$$\hat{\theta}_{MLE} = \arg \max_{\theta} \mathcal{L}(\theta | x_1, x_2, \dots, x_n) = \arg \max_{\theta} \log (\mathcal{L}(\theta | x_1, x_2, \dots, x_n)) \quad (10)$$

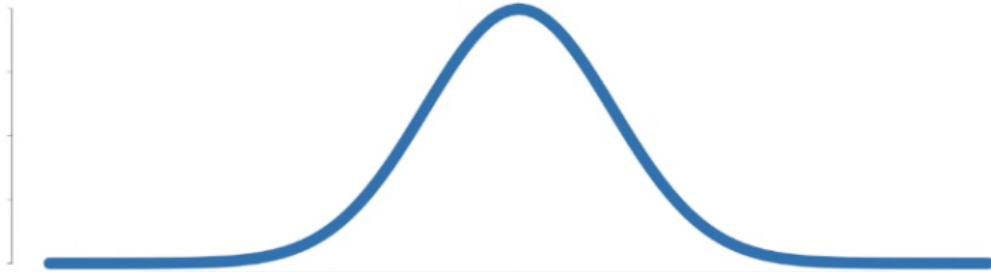


Maximum likelihood for normal distribution X

- ✓ Normal distribution has two parameters μ and σ .

It has two parameters:

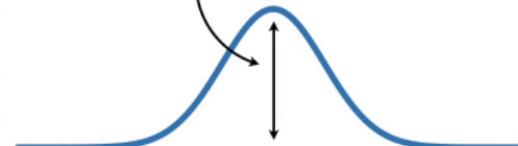
$$pr(x | \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



Maximum likelihood for normal distribution XI

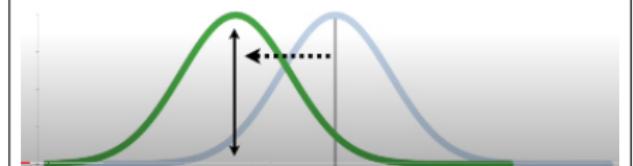
The first parameter, the Greek character μ , determines the location of the normal distribution's **mean**.

$$pr(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



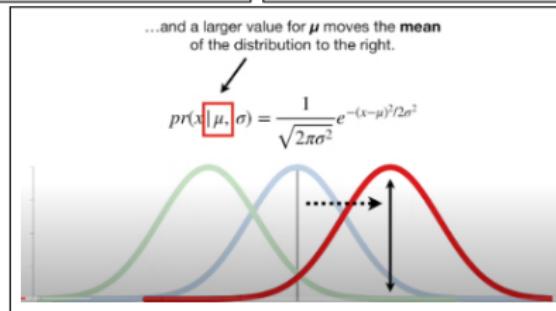
A smaller value for μ moves the **mean** of the distribution to the left...

$$pr(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



...and a larger value for μ moves the **mean** of the distribution to the right.

$$pr(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



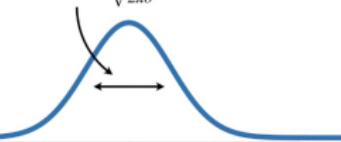
- ✓ The first parameter μ determines the location of the normal curve.



Maximum likelihood for normal distribution XII

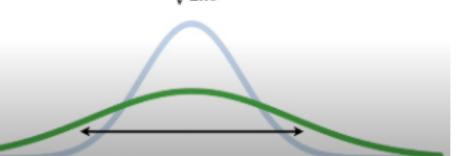
The second parameter, the Greek character σ , is the **standard deviation** and determines the normal distribution's width.

$$pr(x | \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



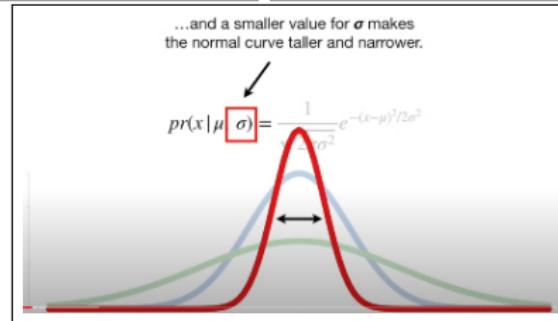
A larger value for σ makes the normal curve shorter and wider...

$$pr(x | \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



...and a smaller value for σ makes the normal curve taller and narrower.

$$pr(x | \mu, \sigma) = \frac{1}{\sqrt{\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$



- ✓ The second parameter σ determines the normal distribution width.



Maximum likelihood for normal distribution XIII

Likelihood of the normal distribution to find the optimal values of μ (the mean) and σ (the SD) for given some data x

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (11)$$

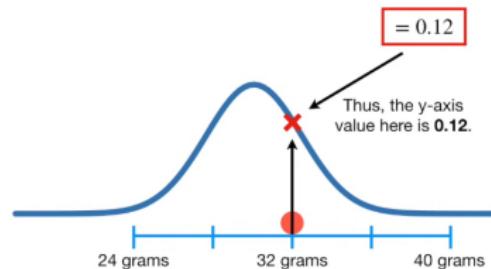
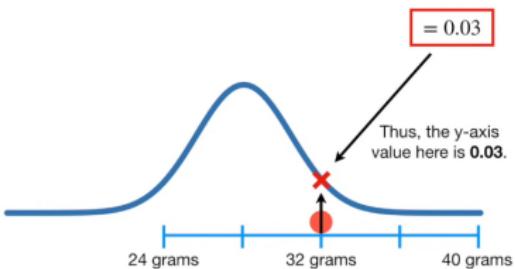
$$L(\mu, \sigma|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (12)$$

Maximum likelihood for normal distribution XIV

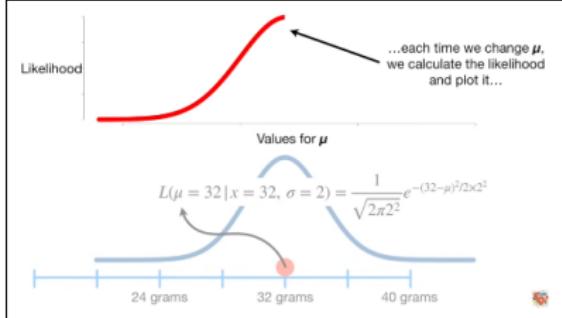
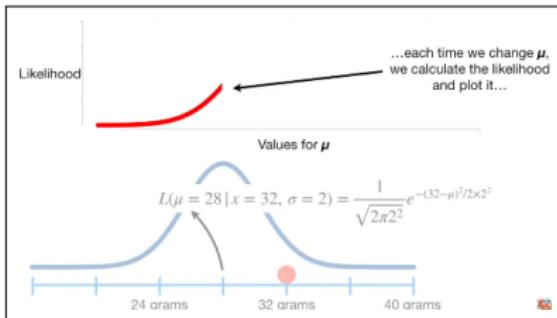
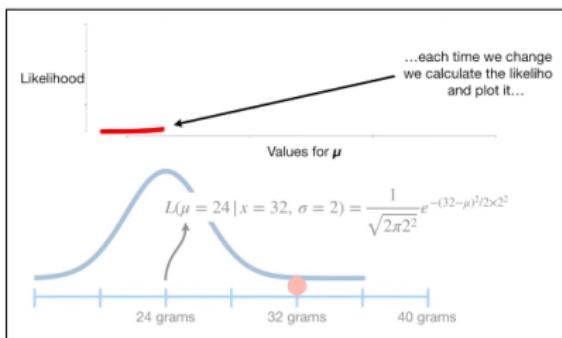
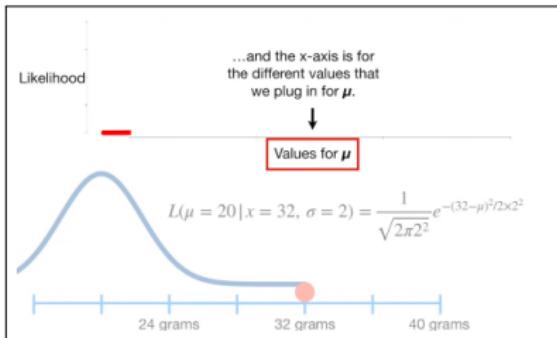
✓ **Example:** To determine the likelihood of a given data $x = 32$ for the given $\mu = 28$ and $\sigma = 2$

$$L(\mu = 28, \sigma = 2 | x = 32) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} = \frac{1}{\sqrt{2\pi2^2}} e^{-(32-28)^2/2\times2^2}$$

$$L(\mu = 30, \sigma = 2 | x = 32) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} = \frac{1}{\sqrt{2\pi2^2}} e^{-(32-30)^2/2\times2^2}$$



Maximum likelihood for normal distribution XV



Maximum likelihood for normal distribution XVI

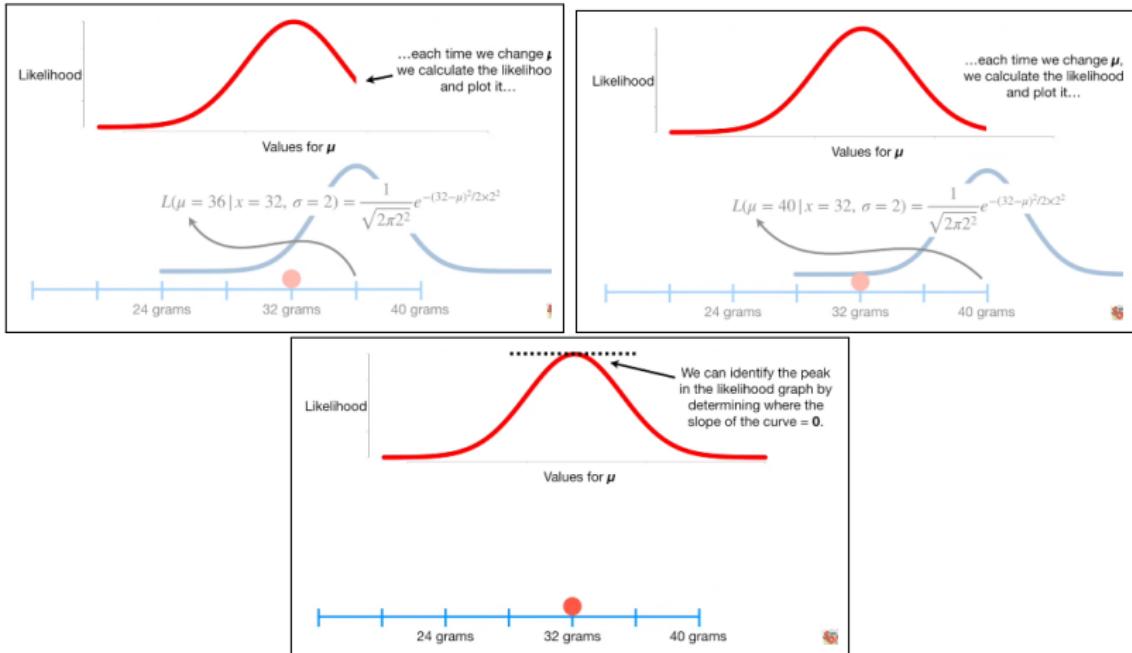
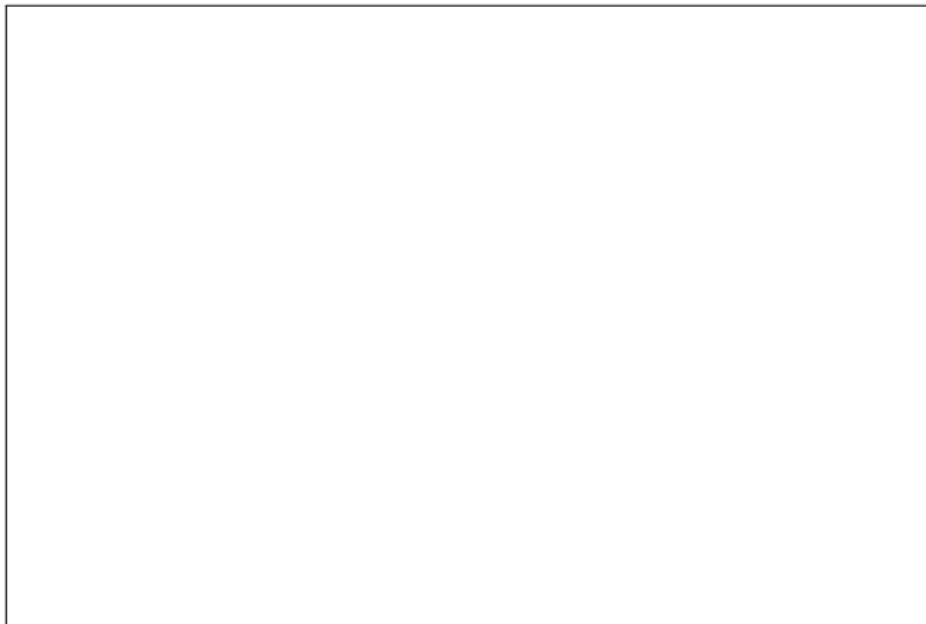


Figure: Different likelihood for different $\mu = 20, 24, 28, 32, 36, 40$ for given $x = 32$

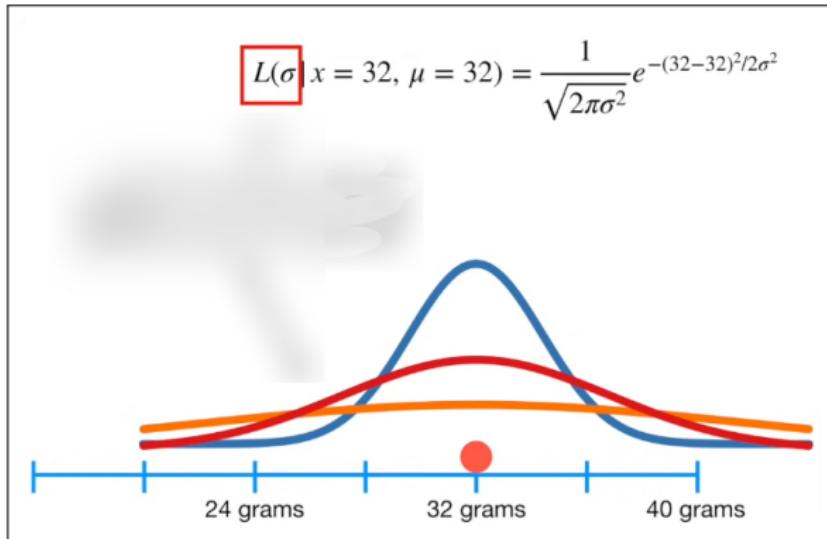
Maximum likelihood for normal distribution XVII

- ✓ Maximum likelihood can be determined by a slop of the curve =0
- ✓ In the given distribution slop of the curve is zero at $\mu = 32$.



Maximum likelihood for normal distribution XVIII

- ✓ After fixing μ , plug different values of σ to determine the maximum likelihood.



Maximum likelihood for normal distribution XIX

- ✓ Need more number of measurements to find the optimal value for σ .
- ✓ To solve for maximum likelihood estimate for μ , treat σ as a constant and, then find where the slope of its likelihood function is zero.
- ✓ To solve for maximum likelihood estimate for σ , treat μ as a constant and, then find where the slope of its likelihood function is zero.



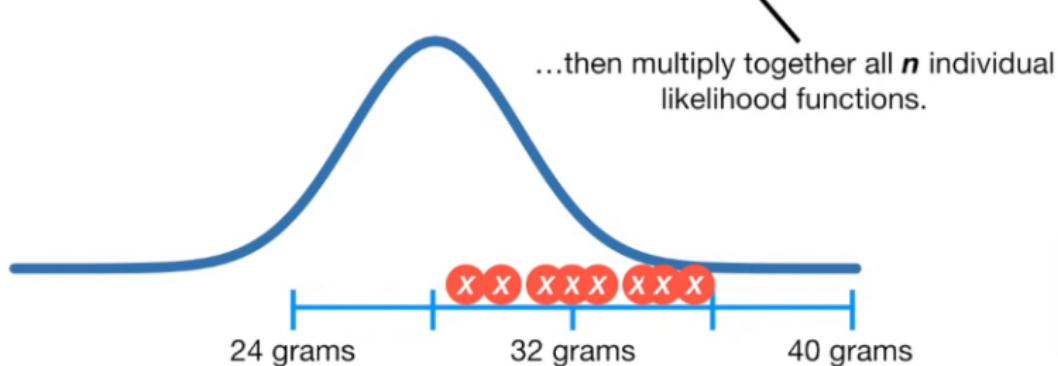
Maximum likelihood for normal distribution XX

General formula for getting maximum likelihood for n data points

$$L(\mu, \sigma | x_1, x_2, \dots, x_n) = L(\mu, \sigma | x_1) \times \dots \times L(\mu, \sigma | x_n)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1-\mu)^2/2\sigma^2} \times \dots \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_n-\mu)^2/2\sigma^2}$$

...then multiply together all n individual likelihood functions.



Maximum likelihood for normal distribution XXI

- ✓ The maximum likelihood function without any values specified for μ and σ .

$$L(\mu, \sigma | x_1, x_2, \dots, x_n) = L(\mu, \sigma | x_1) \times L(\mu, \sigma | x_2) \times \dots \times L(\mu, \sigma | x_n)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2}\right) \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_2-\mu)^2}{2\sigma^2}\right) \times \dots \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right)$$

- ✓ To find the maximum likelihood of the equation, take two different derivatives of the equation.

Maximum likelihood for normal distribution XXII

- ✓ One derivative will be with respect to μ , when σ is treated like a constant. We can find the **maximum likelihood estimate** for μ by finding where this derivative=0.
- ✓ The other derivative will be with respect to σ , when we treat μ as a constant.

Maximum likelihood for normal distribution XXIII

$$\begin{aligned} & \ln [L(\mu, \sigma | x_1, x_2, \dots, x_n)] \\ &= \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2}\right) \times \dots \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right) \right) \\ &= \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2}\right) \right) + \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_2-\mu)^2}{2\sigma^2}\right) \right) + \dots \\ &\quad + \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right) \right) \\ &= \left[\ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \ln \left(\exp\left(-\frac{(x_1-\mu)^2}{2\sigma^2}\right) \right) \right] + \dots + \left[\ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \ln \left(\exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right) \right) \right] \\ &= \left[-\frac{1}{2} \ln(2\pi) - \ln(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2} \right] + \dots + \left[-\frac{1}{2} \ln(2\pi) - \ln(\sigma) - \frac{(x_n-\mu)^2}{2\sigma^2} \right] \\ &= -\frac{n}{2} \ln(2\pi) - n \ln(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2} - \dots - \frac{(x_n-\mu)^2}{2\sigma^2} \end{aligned}$$



Maximum likelihood for normal distribution XXIV

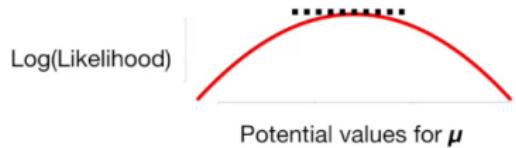
$$\ln [L(\mu, \sigma|x_1, x_2, \dots, x_n)] = -\frac{n}{2} \ln (2\pi) - n \ln (\sigma) - \frac{(x_1 - \mu)^2}{2\sigma^2} - \dots - \frac{(x_n - \mu)^2}{2\sigma^2}$$

- ✓ Taking the derivative with respect to μ . This derivative is the slope function for the Log(likelihood) curve.

$$\frac{\partial}{\partial \mu} \ln [L(\mu, \sigma|x_1, x_2, \dots, x_n)] \quad (13)$$



Maximum likelihood for normal distribution XXV



$$\frac{\partial}{\partial \mu} \ln [L(\mu, \sigma | x_1, x_2, \dots, x_n)] = 0 - 0 + \frac{(x_1 - \mu)}{\sigma^2} + \dots + \frac{(x_n - \mu)}{\sigma^2} \quad (14)$$

$$= \boxed{\frac{1}{\sigma^2} [(x_1 + \dots + x_n) - n\mu]} \quad (15)$$



Maximum likelihood for normal distribution XXVI

The derivative of the log-likelihood function with respect to σ :

$$\frac{\partial}{\partial \sigma} \ln [L(\mu, \sigma | x_1, x_2, \dots, x_n)] = \boxed{-\frac{n}{\sigma} + \frac{(x_1 - \mu)^2}{\sigma^3} + \dots + \frac{(x_n - \mu)^2}{\sigma^3}} \quad (16)$$

- ✓ To find the maximum likelihood estimate for μ , we need to solve for where the derivative with respect to $\mu=0$, because the slope is 0 at the peak of the curve.

Maximum likelihood for normal distribution XXVII

$$\frac{1}{\sigma^2} [(x_1 + \cdots + x_n) - n\mu] = 0 \quad (17)$$

$$\boxed{\mu = \frac{(x_1 + \cdots + x_n)}{n}} \quad (18)$$

Maximum likelihood estimate for μ is the mean of the measurements

- ✓ Likewise, to find the maximum likelihood estimate for σ , we need to solve for where the derivative with respect to $\sigma=0$, because the slope is 0 at the peak of the curve.

Maximum likelihood for normal distribution XXVIII

$$-\frac{n}{\sigma} + \frac{(x_1 - \mu)^2}{\sigma^3} + \cdots + \frac{(x_n - \mu)^2}{\sigma^3} = 0 \quad (19)$$

$$\sigma^2 = \frac{(x_1 - \mu)^2 + \cdots + (x_n - \mu)^2}{n} \quad (20)$$

$$\boxed{\sigma = \sqrt{\frac{(x_1 - \mu)^2 + \cdots + (x_n - \mu)^2}{n}}} \quad (21)$$

Maximum likelihood estimate for σ is the standard deviation of the measurements



Maximum likelihood for normal distribution XXIX

- ✓ If the likelihood function is differentiable, the derivative test for determining maxima can be applied.
- ✓ In some cases, the first-order conditions of the likelihood function can be solved explicitly; for instance, the ordinary least squares estimator maximizes the likelihood of the linear regression model.
- ✓ Under most circumstances, however, numerical methods will be necessary to find the maximum of the likelihood function.
- ✓ This maximum log-likelihood can be shown to be the same for more general least squares, even for non-linear least squares.

Maximum likelihood for normal distribution XXX

- ✓ This is often used in determining likelihood-based approximate confidence intervals and confidence regions, which are generally more accurate than those using the asymptotic normality discussed above.



Mean Squared Error I

Deriving Mean Squared Error (MSE) using MLE:

Mean Squared Error is the cost function commonly used for regression. Given a set of n training examples of the form $(x^{(i)}, y^{(i)})$, MSE is given by:

$$MSE = \frac{1}{n} \sum_{i=1}^n (x^{(i)}, y^{(i)})^2 \quad (22)$$

where $x^{(i)}$ is the feature vector, $y^{(i)}$ is the true output, and $\hat{y}^{(i)}$ is the regression model's prediction for the i^{th} training example.

Mean Squared Error II

Regression Model:

Given a vector of predictor variables $X = (X_1, X_2, \dots, X_p)$, and quantitative outcome variable Y , linear regression assumes that there is a linear relationship between the population mean of the outcome and the predictor variables.

- ✓ The relationship can be expressed by:

$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \quad (23)$$

$\mathbb{E}(Y|X)$ measures the population mean of Y given a particular value of X and it is often called *true regression line*.

Mean Squared Error III

- ✓ Now, the observed data will not lie exactly on this true regression line. They will be spread about the true regression line.
- ✓ For example, if we are trying to predict height from age of a population, we will find that people of the same age will have different heights. Each person's height will differ from the population mean $\mathbb{E}(Y|X)$ by certain amount. It is mathematically represented as:

$$Y = \mathbb{E}(Y|X) + \epsilon \quad (24)$$

$$= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \quad (25)$$

- ✓ The spread about the true regression line is what the ϵ term captures. 

Mean Squared Error IV

Estimating linear regression's model parameters:

- ✓ The true regression line and its model parameters $\beta_0, \beta_1, \dots, \beta_p$ in Equation (25) are unknown, so we will try to estimate them using training data $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})$, where $x^{(i)}$ is a vector of p predictor variables (x_1, x_2, \dots, x_p) and $y^{(i)}$ is the target value.
- ✓ The model (the estimate of the true regression line) is:

$$y^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \dots + \beta_p x_p^{(i)} + \epsilon \quad (26)$$

$$= \beta^\top x^{(i)} + \epsilon \quad (27)$$

$$= \hat{y}^{(i)} + \epsilon \quad (28)$$

Mean Squared Error V

- ✓ The ϵ term is called the residual and it measures the difference between the observed value $y^{(i)}$ and the predicted value $\hat{y}^{(i)}$ from the regression equation.
- ✓ Estimating $\beta_0, \beta_1, \dots, \beta_p$ using the training data is an optimization problem that we can solve using MLE by defining a likelihood function.
- ✓ Since ϵ is normally distributed $\epsilon \sim \mathcal{N}(0, \sigma^2)$, our outcome variable y will also be normally distributed.
- ✓ So, $y \sim \mathcal{N}(\beta^\top x, \sigma^2)$ The probability density function of the normal distribution (parameterised by μ : mean, and σ^2 : variance) is given by:

Mean Squared Error VI

$$f(y \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \quad (29)$$

Therefore, if we replace the parameter μ with the mean of y , we get:

$$f(y \mid \beta^T x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^T x)^2}{2\sigma^2}} \quad (30)$$

The log-likelihood function will then be:

Mean Squared Error VII

$$\log(\mathcal{L}(\beta|x^{(1)}, x^{(2)}, \dots, x^{(n)})) = \sum_{i=1}^n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y - \beta^\top x^{(i)})^2}{2\sigma^2}}\right) \quad (31)$$

$$= n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^n \frac{(y - \beta^\top x^{(i)})^2}{2\sigma^2} \quad (32)$$

$$\frac{\partial}{\partial \beta} [\log(\mathcal{L}(\beta|x^{(1)}, x^{(2)}, \dots, x^{(n)}))] = - \sum_{i=1}^n (y - \beta^\top x^{(i)})^2 \quad (33)$$

$$= - \sum_{i=1}^n (y - \hat{y}^{(i)})^2 \quad (34)$$



Mean Squared Error VIII

- ✓ The value of β that maximises equation is the maximum likelihood estimate $\hat{\beta}_{MSE}$. That is,

$$\hat{\beta}_{MSE} = \arg \max_{\beta} \left[- \sum_{i=1}^n (y - \hat{y}^{(i)})^2 \right] \quad (35)$$

- ✓ **Maximizing a function is the same as minimizing its negative**, so we can rewrite equation

Mean Squared Error IX

$$\hat{\beta}_{MSE} = \arg \max_{\beta} \left[- \sum_{i=1}^n (y - \hat{y}^{(i)})^2 \right] \quad (36)$$

$$= \arg \min_{\beta} \sum_{i=1}^n (y - \hat{y}^{(i)})^2 \quad (37)$$

Taking the average across all n training examples, we get:

$$\hat{\beta}_{MSE} = \arg \min_{\beta} \frac{1}{n} \sum_{i=1}^n (y - \hat{y}^{(i)})^2 \quad (38)$$

Mean Squared Error X

which is exactly the mean squared error (MSE) cost function as defined in equation

$$MSE = \frac{1}{n} \sum_{i=1}^n (x^{(i)}, y^{(i)})^2 \quad (39)$$

Logistic Regression I

- ✓ Logistic regression is a model for binary classification predictive modeling.
- ✓ The parameters of a logistic regression model can be estimated by the probabilistic framework called **maximum likelihood estimation**.
- ✓ Under this framework, a probability distribution for the target variable (**class label**) must be assumed and then a likelihood function defined that calculates the probability of observing the outcome given the input data and the model.
- ✓ This function can then be optimized to find the set of parameters that results in the largest sum likelihood over the training dataset.

Logistic Regression II

- ✓ Classification predictive modeling problems are those that require the prediction of a class label (e.g. 'red', 'green', 'blue') for a given set of input variables.
- ✓ Binary classification refers to those classification problems that have two class labels, e.g. true/false or 0/1.
- ✓ Logistic regression has a lot in common with linear regression, although linear regression is a technique for predicting a numerical value, not for classification problems.
- ✓ Both techniques model the target variable with a line (or hyperplane, depending on the number of dimensions of input).

Logistic Regression III

- ✓ Linear regression fits the line to the data, which can be used to predict a new quantity, whereas logistic regression fits a line to best separate the two classes.
- ✓ Logistic regression involves a more probabilistic view of classification.
- ✓ Binary logistic regression is used to model the relationship between a categorical target variable Y and a predictor vector $X = (X_1, X_2, \dots, X_p)$.
- ✓ The target variable will have two possible values, such as whether a student passes an exam or not, or whether a visitor to a website subscribes to the website's newsletter or not.

Logistic Regression IV

- ✓ The two possible categories are coded as ‘1’, called the positive class, and ‘0’, called the negative class.
- ✓ Binary logistic regression estimates the probability that the response variable Y belongs to the positive class given X .

$$p(X) = Pr(Y = 1|X) \quad (40)$$

- ✓ In linear regression, we model the expected value (the mean μ) of the continuous target variable Y as a linear combination of the predictor

Logistic Regression V

vector X and estimate the weight parameters $\beta_1, \beta_2, \dots, \beta_p$ using our training data.

$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p \quad (41)$$

- ✓ In this case where our target variable Y is categorical and has two possible values coded as 0 and 1, the expected value or mean of Y is the probability $p(X)$ of observing the positive . It seems sensible then to model the expected value of our categorical Y variable using the equation, as in linear regression.



Logistic Regression VI

$$\mathbb{E}(Y|X) = p(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \quad (42)$$

- ✓ The problem with modelling the probability $p(X)$ as a linear combination of the predictor variables is that probability $p(X)$ has a range $[0, 1]$, but the right-hand side of the equation outputs values in the range $(-\infty, +\infty)$. In other words, we will get meaningless estimates of the probability if we use that equation.
- ✓ The solution is to use a function of probability $p(X)$ that provides a suitable relationship between the linear combination of the predictor variables X and $p(X)$, the mean of the response variable.

Logistic Regression VII

- ✓ This function is called a link function, and it maps the probability range $[0, 1]$ to $(-\infty, +\infty)$.
- ✓ The most commonly used link function for binary logistic regression is the logit function (or log-odds, given as:

$$\text{logit}\left(p(X)\right) = \log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \quad (43)$$

How do we then go from the logit function to getting the estimate of the probability $p(X)$ of observing the positive class?

Logistic Regression VIII

- ✓ logit is a function of probability, we can take its inverse to map arbitrary values in the range $(-\infty, +\infty)$ back to the probability range $[0, 1]$.
- ✓ if we take the inverse of equation, we get:

$$\frac{p(X)}{1 - p(X)} = e^{(\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p)} \quad (44)$$

If we solve for $p(X)$ in equation, we get



Logistic Regression IX

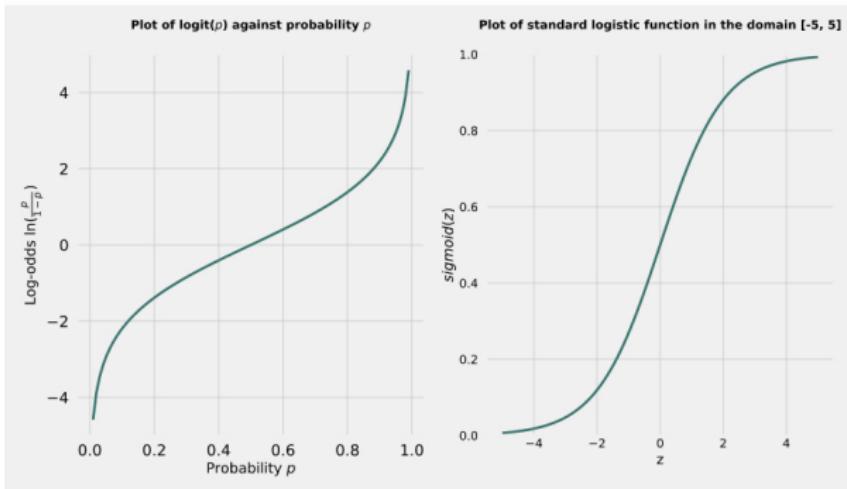
$$p(X) = \frac{e^{(\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p)}}{e^{(\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p)} + 1} \quad (45)$$

$$= \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p)}} \quad (46)$$

- ✓ Equation is the logistic (or sigmoid) function, and it maps values in the logit range $(-\infty, +\infty)$ back into the range $[0, 1]$ of probabilities.



Logistic Regression X



Logistic Regression XI



Logistic Regression XII

Deriving Cost Entropy/ log loss using MLE:

Given a set of n training examples

$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$, binary cross-entropy is given by:

$$\text{Cross-Entropy} = - \left[\frac{1}{n} \sum_{i=1}^n \left(y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log (1 - p^{(i)}) \right) \right] \quad (47)$$



Logistic Regression XIII

where $x^{(i)}$ is the feature vector, $y^{(i)}$ is the true label (0 or 1) for the i^{th} training example, and $p^{(i)}$ is the predicted probability that the i^{th} training example belongs to the positive class, that is,

$$Pr(Y = 1 | X = x^{(i)}) \quad (48)$$

- ✓ The derivation of cross-entropy follows from using MLE to estimate the parameters $\beta_0, \beta_1, \dots, \beta_p$ of our logistic model on our training data.
- ✓ We start by describing the random process that generated $y^{(i)}$.
 $y^{(i)}$ is a realization of the Bernoulli random variable Y . The Bernoulli distribution is parameterized by p , and its **probability mass function (pmf)** is given by:

Logistic Regression XIV

$$Pr(Y = y^{(i)}) = \begin{cases} p & \text{if } y^{(i)} = 1, \\ 1 - p & \text{if } y^{(i)} = 0. \end{cases} \quad (49)$$

which can be written in the more compact form:

$$Pr(Y = y^{(i)}) = p^{y^{(i)}} (1 - p)^{1-y^{(i)}} \text{ for } y^{(i)} \in \{0, 1\} \quad (50)$$

We then define our Likelihood function. The estimates of $\beta_0, \beta_1, \dots, \beta_p$ we choose will be the ones that maximise the likelihood function. The



Logistic Regression XV

likelihood function is a function of our parameter p given our training data:

$$\mathcal{L}(p | (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})) = \prod_{i=1}^n f(y^{(i)}|p) \quad (51)$$

$$= \prod_{i=1}^n p^{y^{(i)}} (1-p)^{1-y^{(i)}} \quad (52)$$

- ✓ It is easier to work with the log of the likelihood function, called the log-likelihood, so if we take the natural logarithm of the equation, we get:

Logistic Regression XVI

$$\log \left(\mathcal{L}(p|y^{(1)}, y^{(2)}, \dots, y^{(n)}) \right) = \log \left(\prod_{i=1}^n p^{y^{(i)}} (1-p)^{1-y^{(i)}} \right) \quad (53)$$

$$= \sum_{i=1}^n \log \left(p^{y^{(i)}} (1-p)^{1-y^{(i)}} \right) \quad (54)$$

$$= \sum_{i=1}^n \left(y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log (1 - p^{(i)}) \right) \quad (55)$$

- ✓ The training data, $p^{(i)}$ in equation is the predicted probability of the i^{th} training example gotten from the logistic function, so it is a function of the parameters $\beta_0, \beta_1, \dots, \beta_p$.

Logistic Regression XVII

- ✓ The maximum likelihood estimate $\hat{\beta}$ is therefore the value of the parameters that maximises the log-likelihood function.

$$\hat{\beta} = \arg \max_{\beta} \left[\sum_{i=1}^n \left(y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log (1 - p^{(i)}) \right) \right] \quad (56)$$

- ✓ We also know that maximizing a function is the same as minimizing its negative.

$$\hat{\beta} = \arg \min_{\beta} \left[- \sum_{i=1}^n \left(y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log (1 - p^{(i)}) \right) \right] \quad (57)$$

Logistic Regression XVIII

- ✓ Taking the average across our n training examples, we get:

$$\hat{\beta} = \arg \min_{\beta} \left[-\frac{1}{n} \sum_{i=1}^n \left(y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log(1 - p^{(i)}) \right) \right] \quad (58)$$

which is the cross-entropy.

Logistic Regression XIX

Softmax function:

- ✓ The softmax function, also known as softargmax or normalized exponential function is a generalization of the logistic function to multiple dimensions.
- ✓ It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes, based on Luce's choice axiom.
- ✓ The softmax function takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.



Logistic Regression XX

- ✓ That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval $[0, 1]$, and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities.
- ✓ The standard (unit) softmax function $\sigma : \mathbb{R}^K \rightarrow [0, 1]^K$ is defined by the formula

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (59)$$

Logistic Regression XXI

for $i = 1, \dots, K$ and $z = (z_1, \dots, z_k) \in \mathbb{R}^K$.

Application of Softmax:

- ✓ The softmax function is used in various multiclass classification methods, such as multinomial logistic regression (also known as softmax regression), multiclass linear discriminant analysis, naive Bayes classifiers, and artificial neural networks.
- ✓ Specifically, in multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions, and the predicted probability for the j^{th} class given a sample vector x and a weighting vector w is:

Logistic Regression XXII

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^\top \mathbf{w}_k}} \quad (60)$$

This can be seen as the composition of K linear functions

$\mathbf{x} \mapsto \mathbf{x}^\top \mathbf{w}_1, \dots, \mathbf{x} \mapsto \mathbf{x}^\top \mathbf{w}_K$ and the softmax function (where $\mathbf{x}^\top \mathbf{w}$ denotes the inner product of \mathbf{x} and \mathbf{w}).

- ✓ The operation is equivalent to applying a linear operator defined by \mathbf{w} to vectors \mathbf{x} , thus transforming the original, probably highly-dimensional, input to vectors in a K -dimensional space \mathbb{R}^K .

Logistic Regression XXIII

Neural networks

✓ The softmax function is often used in the final layer of a neural network-based classifier. Such networks are commonly trained under a log loss (or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression.

Since the function maps a vector and a specific index i to a real value, the derivative needs to take the index into account:

$$\frac{\partial}{\partial q_k} \sigma(\mathbf{q}, i) = \sigma(\mathbf{q}, i)(\delta_{ik} - \sigma(\mathbf{q}, k)). \quad (61)$$

Logistic Regression XXIV

- ✓ This expression is symmetrical in the indexes i, k and thus may also be expressed as

$$\frac{\partial}{\partial q_k} \sigma(\mathbf{q}, i) = \sigma(\mathbf{q}, k)(\delta_{ik} - \sigma(\mathbf{q}, i)). \quad (62)$$

Here, the Kronecker delta is used for simplicity (cf. the derivative of a sigmoid function, being expressed via the function itself).

- ✓ If the function is scaled with the parameter β , then these expressions must be multiplied by β .

Logistic Regression XXV

Gradient Ascent Optimization:

Initialize: $\theta_j = 0$ for all $0 \leq j \leq m$

Repeat many times:

gradient[j] = 0 for all $0 \leq j \leq m$

For each training example (x, y) :

For each parameter j :

$$\text{gradient}[j] += x_j \left(y - \frac{1}{1 + e^{-\theta^T x}} \right)$$

$\theta_j += \eta * \text{gradient}[j]$ for all $0 \leq j \leq m$

Pros and Cons of Activation Functions I

Type of Function	Pros	Cons
Linear	<p>It gives a range of activations, so it is not binary activation.</p> <p>It can definitely connect a few neurons together and if more than 1 fire, take the max and decide based on that.</p>	<p>It is a constant gradient and the descent is going to be on a constant gradient.</p> <p>If there is an error in prediction, the changes made by backpropagation are constant and not depending on the change in input.</p>
Sigmoid	<p>It is nonlinear in nature. Combinations of this function are also nonlinear.</p> <p>It will give an analog activation, unlike the step function.</p>	<p>Sigmoids saturate and kill gradients. It gives rise to a problem of “vanishing gradients”</p> <p>The network refuses to learn further or is drastically slow.</p>
Tanh	The gradient is stronger for tanh than sigmoid i.e. derivatives are steeper.	Tanh also has a vanishing gradient problem.
ReLU	<p>It avoids and rectifies the vanishing gradient problem.</p> <p>ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.</p>	<p>It should only be used within hidden layers of a Neural Network Model.</p> <p>Some gradients can be fragile during training and can die.</p> <p>It can cause a weight update which will make it never activate on any data point again. Thus, ReLU could even result in Dead Neurons.</p>
Leaky ReLU	Leaky ReLUs is one attempt to fix the “dying ReLU” problem by having a small negative slope	<p>As it possesses linearity, it can't be used for complex Classification.</p> <p>It lags behind the Sigmoid and Tanh for some of the use cases.</p>
ELU	Unlike ReLU, ELU can produce negative outputs.	For $x > 0$, it can blow up the activation with the output range of $[0, \infty]$.

Vanishing gradient problem I

- ✓ In machine learning, the vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods and backpropagation.
- ✓ In such methods, each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training.
- ✓ The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training.



Vanishing gradient problem II

- ✓ As one example of the problem cause, traditional activation functions such as the hyperbolic tangent function have gradients in the range $(0,1]$, and backpropagation computes gradients by the chain rule. This has the effect of multiplying n of these small numbers to compute gradients of the early layers in an n -layer network, meaning that the gradient (error signal) decreases exponentially with n while the early layers train very slowly.
- ✓ Back-propagation allowed researchers to train supervised deep artificial neural networks from scratch, initially with little success.

Vanishing gradient problem III

- ✓ Hochreiter's diplom thesis of 1991 formally identified the reason for this failure in the "**vanishing gradient problem**", which not only affects many-layered feedforward networks, but also recurrent networks.
- ✓ The latter are trained by unfolding them into very deep feedforward networks, where a new layer is created for each time step of an input sequence processed by the network. (The combination of unfolding and backpropagation is termed backpropagation through time.)
- ✓ When activation functions are used whose derivatives can take on larger values, one risks encountering the related exploding gradient problem.

Beyond Logistic Regression I

Logistic regression is a fundamental classification technique. It's a relatively uncomplicated linear classifier. Despite its simplicity and popularity, there are cases (especially with highly complex models) where logistic regression doesn't work well. In such circumstances, one can use other classification techniques:

- k-Nearest Neighbors
- Naive Bayes classifiers
- Support Vector Machines
- Decision Trees
- Random Forests
- Neural Networks



CS322: Deep Learning

Neural Network

Sachchida Nand Chaurasia
Assistant Professor

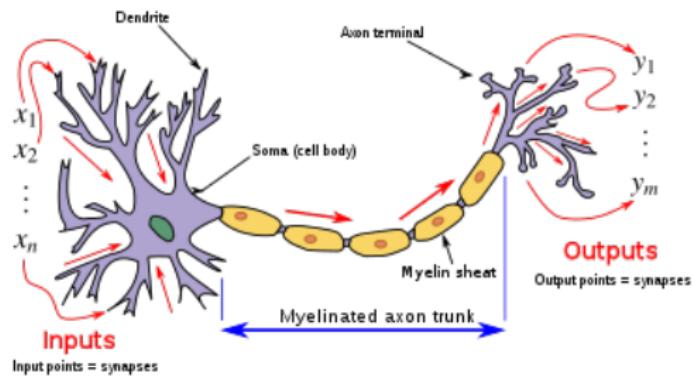
Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



December 11, 2021

Artificial Neuron I



Artificial Neuron II

- ✓ An artificial neuron is a connection point in an artificial neural network. Artificial neural networks, like the human body's biological neural network, have a layered architecture and each network node (connection point) has the capability to process input and forward output to other nodes in the network.
- ✓ In both artificial and biological architectures, the nodes are called neurons and the connections are characterized by synaptic weights, which represent the significance of the connection.
- ✓ As new data is received and processed, the synaptic weights change and this is how learning occurs.

Artificial Neuron III

- ✓ In both artificial and biological networks, when neurons process the input they receive, they decide whether the output should be passed on to the next layer as input.
- ✓ The decision of whether or not to send information on is called bias and it's determined by an activation function built into the system.
- ✓ For example, an artificial neuron may only pass an output signal on to the next layer if its inputs (which are actually voltages) sum to a value above some particular threshold value. Because activation functions can either be linear or non-linear, neurons will often have a wide range of convergence and divergence.

Artificial Neuron IV

- ✓ Divergence is the ability for one neuron to communicate with many other neurons in the network and convergence is the ability for one neuron to receive input from many other neurons in the network.
- ✓ An **artificial neuron** is a mathematical function conceived as a model of biological neurons, a neural network.
- ✓ Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs and sums them to produce an output (or activation, representing a neuron's action potential which is transmitted along its axon).

Artificial Neuron V

- ✓ Usually each input is separately weighted, and the sum is passed through a non-linear function known as an activation function or transfer function.
- ✓ The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions.
- ✓ They are also often monotonically increasing, continuous, differentiable and bounded.
- ✓ The thresholding function has inspired building logic gates referred to as threshold logic; applicable to building logic circuits resembling brain processing.

Artificial Neuron VI

- ✓ For example, new devices such as memristors have been extensively used to develop such logic in recent times.

Basic structure of artificial neural network

For a given artificial neuron k , let there be $m + 1$ inputs with signals x_0 through x_m and weights w_{k0} through w_{km} .

- ✓ Usually, the x_0 input is assigned the value +1, which makes it a bias input with $w_{k0} = b_k$. This leaves only m actual inputs to the neuron: from x_1 to x_m .
- ✓ The output of the i^{th} neuron is:

Artificial Neuron VII

$$y_k = \varphi \left(\sum_{j=0}^m w_{kj} x_l \right) \quad (1)$$

where φ (phi) is the transfer function (commonly a threshold function).



Artificial Neuron VIII

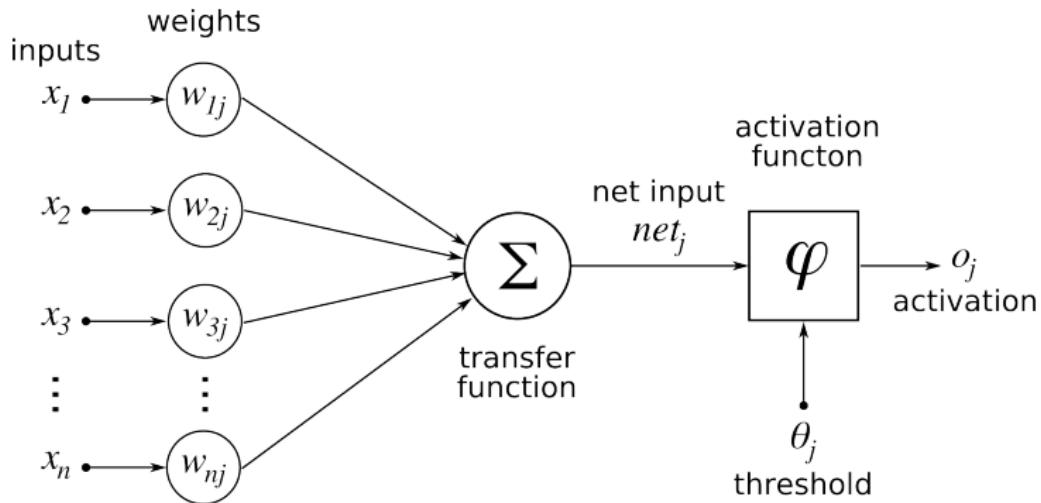


Figure: Basic Artificial Neuron

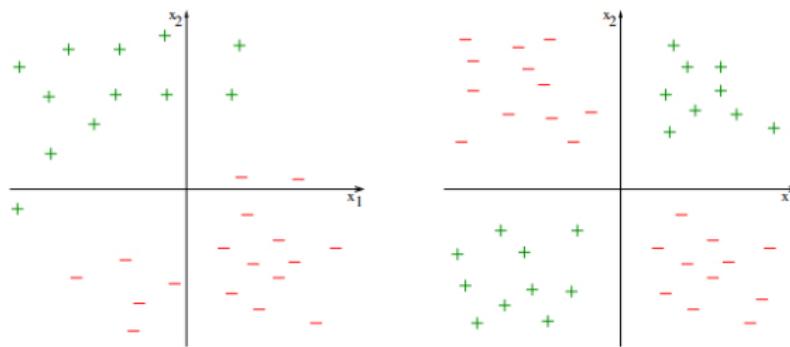
Artificial Neuron IX

- ✓ The output is analogous to the axon of a biological neuron, and its value propagates to the input of the next layer, through a synapse.
- ✓ It may also exit the system, possibly as part of an output vector.
- ✓ It has no learning process as such. Its transfer function weights are calculated and threshold value are predetermined.

single and multilayer perceptrons I

Perceptron convergence theorem:

The 'Perceptron convergence theorem' states that for any data set which is linearly separable the Perceptron learning rule is guaranteed to find a solution in a finite number of steps.



single and multilayer perceptrons II

Figure: Labeled points in 2-dimensional space. The data set on the left is linearly separable, whereas the data set on the right is not

Perceptron Learning Algorithm:

- ✓ Perceptron Algorithm is used in a supervised machine learning domain for classification.
- ✓ In classification, there are two types of linear classification and no-linear classification.
- ✓ Linear classification is nothing but if we can classify the data set by drawing a simple straight line then it can be called a linear binary classifier.

single and multilayer perceptrons III

- ✓ Whereas if we cannot classify the data set by drawing a simple straight line then it can be called a non-linear binary classifier.

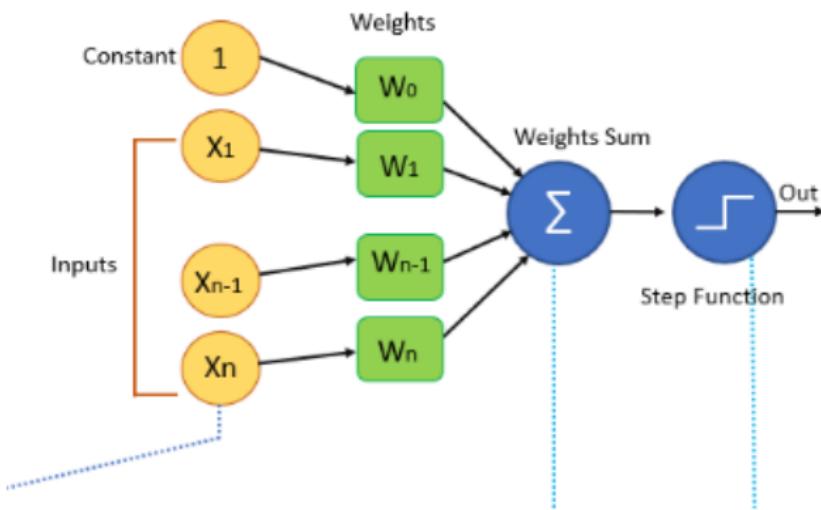


Figure: Perceptron Algorithm Block Diagram

single and multilayer perceptrons IV

- Perceptron Algorithm Block Diagram

- ① Input: All the features of the model we want to train the neural network will be passed as the input to it, Like the set of features $[X_1, X_2, X_3 \dots X_n]$. Where n represents the total number of features and X represents the value of the feature.
- ② Weights: Initially, we have to pass some random values as values to the weights and these values get automatically updated after each training error that is the values are generated during the training of the model. In some cases, weights can also be called as weight coefficients.

single and multilayer perceptrons V

- ③ Weights Sum: Each input value will be first multiplied with the weight assigned to it and the sum of all the multiplied values is known as a **weighted sum**.
- **Step or Activation Function:**
 - ✓ Activation function applies step rule which converts the numerical value to 0 or 1 so that it will be easy for data set to classify.
 - ✓ Based on the type of value we need as output we can change the activation function. Sigmoid function, if we want values to be between 0 and 1 we can use a sigmoid function that has a smooth gradient as well.

single and multilayer perceptrons VI

- ✓ Sign function, if we want values to be +1 and -1 then we can use sign function.
- ✓ The hyperbolic tangent function is a zero centered function making it easy for the multilayer neural networks.
- ✓ Relu function is highly computational but it cannot process input values that approach zero. It is good for the values that are both greater than and less than a Zero.

Bias:

- ✓ The passed value one as input in the starting and W_0 in the weights section W_0 is an element that adjusts the boundary away from origin to move the activation function left, right, up or down.

single and multilayer perceptrons VII

- ✓ Since we want this to be independent of the input features, we add constant one in the statement so the features will not get affected by this and this value is known as Bias.
- ✓ Perceptron algorithms can be divided into two types they are single layer perceptrons and multi-layer perceptron's.
- ✓ In single-layer perceptron's neurons are organized in one layer whereas in a multilayer perceptron's a group of neurons will be organized in multiple layers.
- ✓ Every single neuron present in the first layer will take the input signal and send a response to the neurons in the second layer and so on.

single and multilayer perceptrons VIII

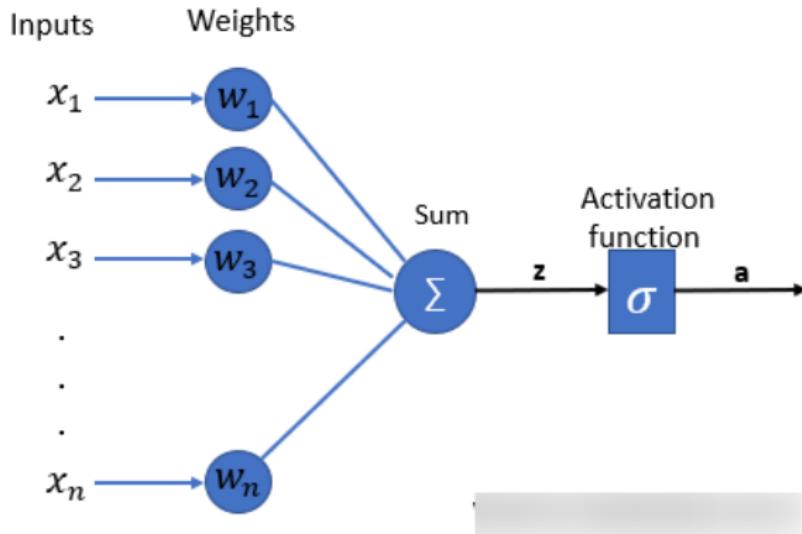


Figure: Single Layer Perceptron

single and multilayer perceptrons IX

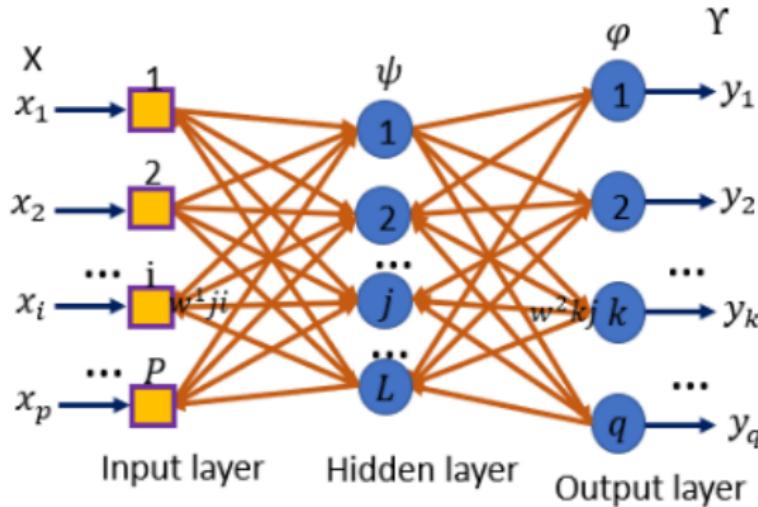


Figure: Multi-Layer Perceptron

- Perceptron Learning Steps

single and multilayer perceptrons X

- 1 Features of the model we want to train should be passed as input to the perceptrons in the first layer.
- 2 These inputs will be multiplied by the weights or weight coefficients and the production values from all perceptrons will be added.
- 3 Adds the Bias value, to move the output function away from the origin.
- 4 This computed value will be fed to the activation function (chosen based on the requirement, if a simple perceptron system activation function is step function).
- 5 The result value from the activation function is the output value.

single and multilayer perceptrons XI

- ✓ The input to the Perceptron Learning Algorithm is a data set of $n \geq 1$ points (each d -dimensional), and their associated labels (+ 'or' -).
- ✓ For mathematical convenience, we associate the 'label with the number '+1, and the '-' label with the number '-1'.
- ✓ we may take our input to be: $(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$, where for $i \in \{1, 2, \dots, n\}$, $x^i \in \mathbb{R}^d$ and $y^i \in \{-1, 1\}$.

Assumption 1 (Linear Separability): There exists some $\mathbf{w}^* \in \mathbb{R}^d$ such that $\|\mathbf{w}^*\| = 1$ and for some $\gamma > 0$ for all $i \in \{1, 2, \dots, n\}$.

Assumption 2 (Bounded coordinates): There exists $R \in \mathbb{R}$ such that for $i \in \{1, 2, \dots, n\}$, $\|x^i\| \leq R$.

Number of hidden layers and nodes in a feedforward neural network? I

- ✓ Once the network is initialized, you can iteratively tune the configuration during training using a number of ancillary algorithms.
- ✓ one family of these works by pruning nodes based on (small) values of the weight vector after a certain number of training epochs—in other words, eliminating unnecessary/redundant nodes.
- ✓ Every Neural Network (NN) has three types of layers: input, hidden, and output.
- ✓ Creating the NN architecture therefore means coming up with values for the number of layers of each type and the number of nodes in each of these layers.

Number of hidden layers and nodes in a feedforward neural network? II

The Input Layer

- ✓ With respect to the number of neurons comprising this layer, this parameter is completely and uniquely determined once you know the shape of your training data. Specifically, the number of neurons comprising that layer is equal to the number of features (columns) in your data.
- ✓ Some NN configurations add one additional node for a bias term.

The Output Layer

- ✓ Like the Input layer, every NN has exactly one output layer. Determining its size (number of neurons) is simple; it is completely determined by the chosen model configuration.

Number of hidden layers and nodes in a feedforward neural network? III

- ✓ ✓ If the NN is a regressor, then the output layer has a single node.
- ✓ ✓ If the NN is a classifier, then it also has a single node unless softmax is used in which case the output layer has one node per class label in your model.

The Hidden Layers:

- ✓ How many hidden layers? *If your data is linearly separable (which you often know by the time you begin coding a NN) then you don't need any hidden layers at all. Of course, you don't need an NN to resolve your data either, but it will still do the job.*
- ✓ Beyond that, there is a consensus is the performance difference from adding additional hidden layers: the situations in which performance

Number of hidden layers and nodes in a feedforward neural network? IV

improves with a second (or third, etc.) hidden layer are very few. *One hidden layer is sufficient for the large majority of problems.*

- ✓ So what about size of the hidden layer(s)—how many neurons? There are some empirically-derived rules-of-thumb, of these, the most commonly relied on is '*the optimal size of the hidden layer is usually between the size of the input and size of the output layers*'.
- ✓ In sum, for most problems, one could probably get decent performance (even without a second optimization step) by setting the hidden layer configuration using just two rules:
 - ① Number of hidden layers equals one; and

Number of hidden layers and nodes in a feedforward neural network? V

- ② The number of neurons in that layer is the mean of the neurons in the input and output layers.

Optimization of the Network Configuration:

- ✓ Pruning describes a set of techniques to trim network size (by nodes not layers) to improve computational performance and sometimes resolution performance.
- ✓ The gist of these techniques is removing nodes from the network during training by identifying those nodes which, if removed from the network, would not noticeably affect network performance (i.e., resolution of the data).

Number of hidden layers and nodes in a feedforward neural network? VI

- ✓ Even without using a formal pruning technique, you can get a rough idea of which nodes are not important by looking at your weight matrix after training; look weights very close to zero—it's the nodes on either end of those weights that are often removed during pruning.
- ✓ Obviously, if you use a pruning algorithm during training then begin with a network configuration that is more likely to have excess (i.e., 'prunable') nodes.
- ✓ There's one additional rule of thumb that helps for supervised learning problems. You can usually prevent over-fitting if you keep your number of neurons below:

Number of hidden layers and nodes in a feedforward neural network? VII

$$N_h = N_s(\alpha * (N_i + N_o)) \quad (2)$$

N_i = number of input neurons.

N_o = number of output neurons.

N_s = number of samples in training data set.

α = an arbitrary scaling factor usually 2-10.

Back Propagation I

- ✓ After performing the first pass (based on the input and randomly given inputs) error will be calculated and the back propagation algorithm performs an iterative backward pass and try to find the optimal values for weights so that the error value will be minimized.
- ✓ To minimize the error back propagation algorithm will calculate partial derivatives from the error function till each neuron's specific weight, this process will give us complete transparency from total error value to a specific weight that is responsible for the error.

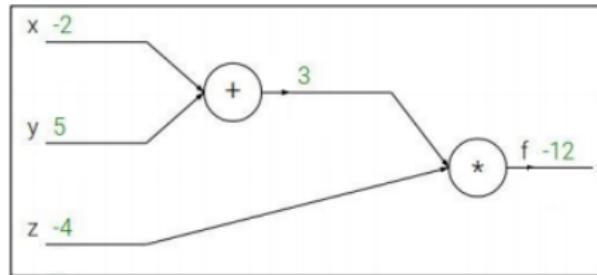
Back Propagation II

Example1:

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y = 5$, $z = -4$



Back Propagation III

Backpropagation: a simple example

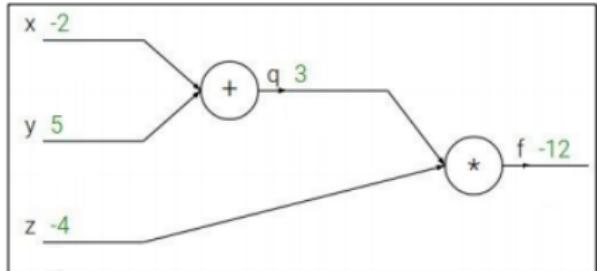
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Back Propagation IV

Backpropagation: a simple example

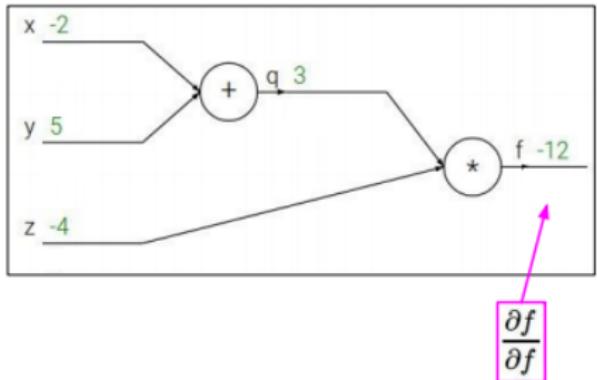
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Back Propagation V

Backpropagation: a simple example

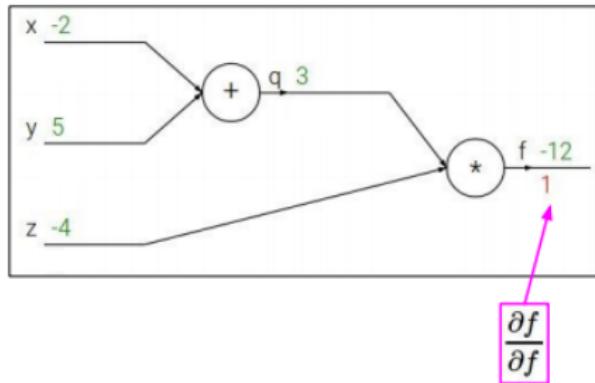
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Back Propagation VI

Backpropagation: a simple example

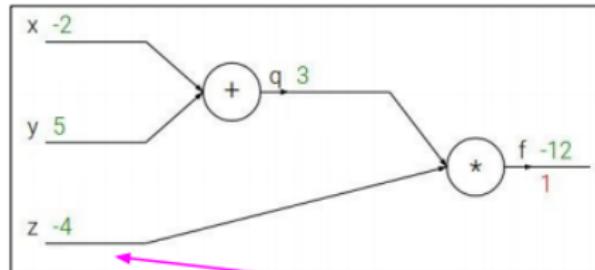
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

Back Propagation VII

Backpropagation: a simple example

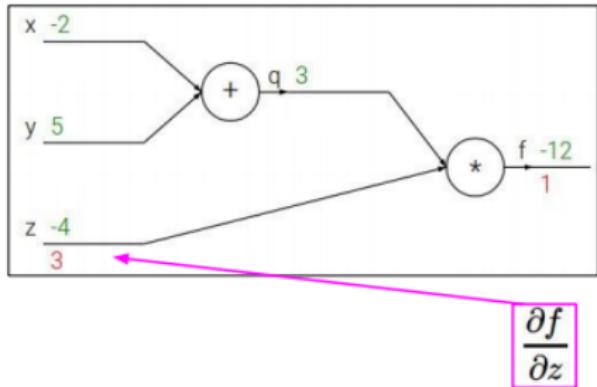
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

Back Propagation VIII

Backpropagation: a simple example

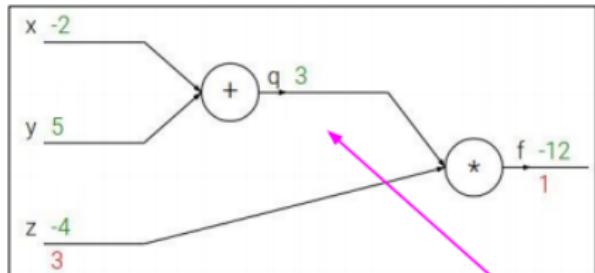
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

Back Propagation IX

Backpropagation: a simple example

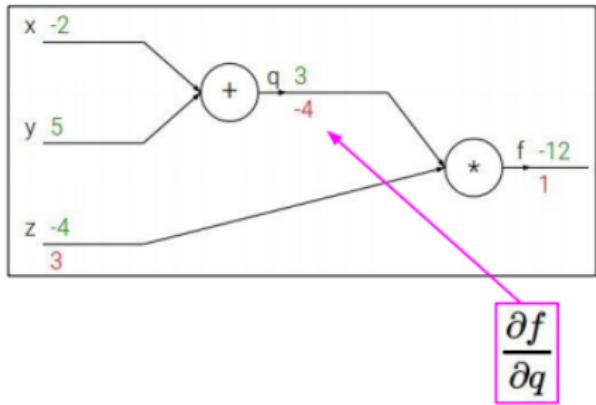
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Back Propagation X

Backpropagation: a simple example

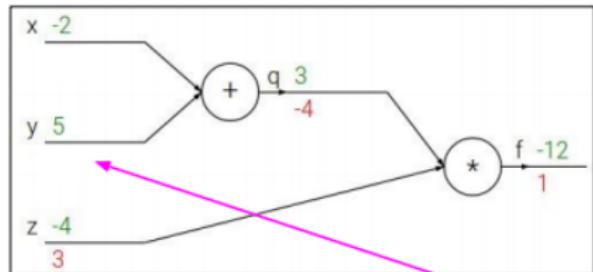
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

Back Propagation XI

Backpropagation: a simple example

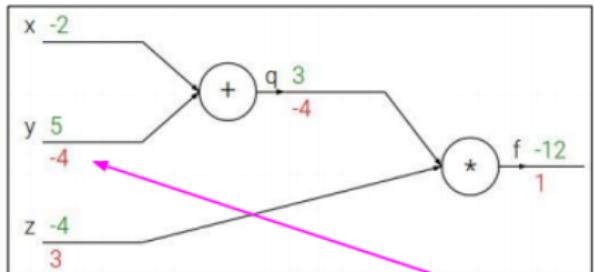
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y = 5$, $z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

$$\frac{\partial f}{\partial y}$$

Back Propagation XII

Backpropagation: a simple example

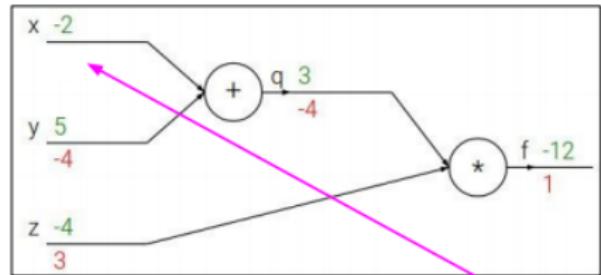
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y = 5$, $z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial x}$$

Back Propagation XIII

Backpropagation: a simple example

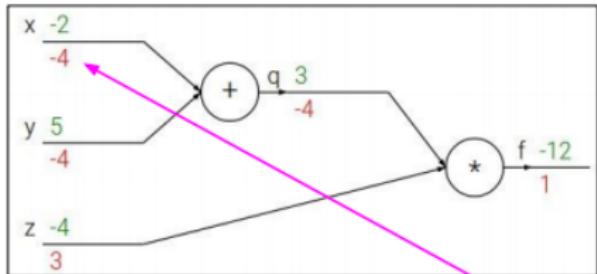
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

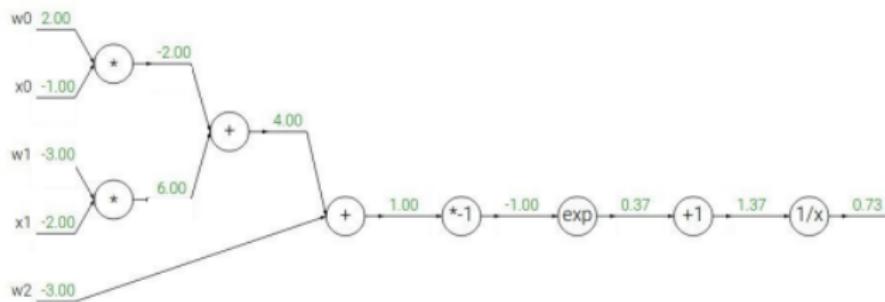
$$\frac{\partial f}{\partial x}$$

Back Propagation XIV

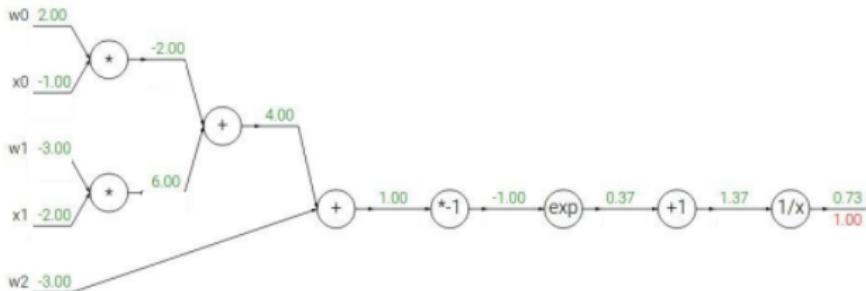
Example2:

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Back Propagation XV



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

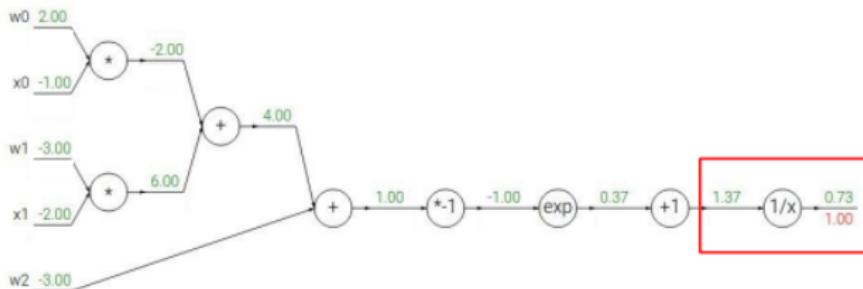
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Back Propagation XVI



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

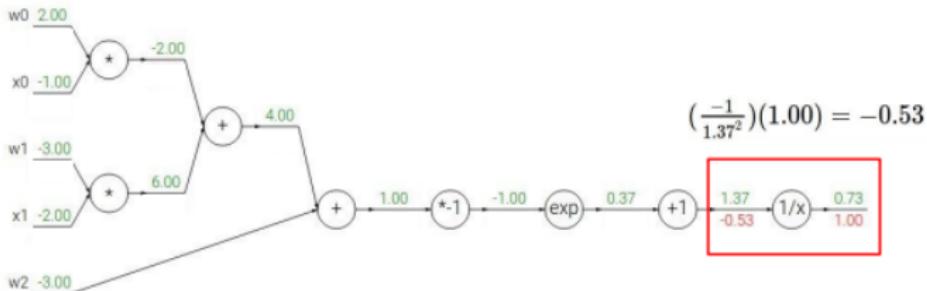
$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Back Propagation XVII



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Normalization vs Standardization I

- ✓ ML that had multiple features spanning varying degrees of magnitude, range, and units.
- ✓ This is a significant obstacle as a few machine learning algorithms are highly sensitive to these features.
- ✓ For example, one feature is entirely in kilograms while the other is in grams, another one is liters, and so on. How can we use these features when they vary so vastly in terms of what they're presenting?
- ✓ This is where the concept of feature scaling comes. It's a crucial part of the data preprocessing stage.
- ✓ Feature scaling – it improves (significantly) the performance of some machine learning algorithms and does not work at all for others.

Normalization vs Standardization II

Normalization and standardization

- ✓ These are two of the most commonly used feature scaling techniques in machine learning but a level of ambiguity exists in their understanding. When should you use which technique?

Why Should we Use Feature Scaling?

- ✓ The first question we need to address – why do we need to scale the variables in our dataset? Some machine learning algorithms are sensitive to feature scaling while others are virtually invariant to it.

Normalization vs Standardization III

Gradient Descent Based Algorithms

- ✓ Machine learning algorithms like linear regression, logistic regression, neural network, etc. that use gradient descent as an optimization technique require data to be scaled.
- ✓ The presence of feature value X in the formula will affect the step size of the gradient descent. The difference in ranges of features will cause different step sizes for each feature.
- ✓ To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

Normalization vs Standardization IV

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad (3)$$

Having features on a similar scale can help the gradient descent converge more quickly towards the minima.

Distance-Based Algorithms

Distance algorithms like KNN, K-means, and SVM are most affected by the range of features. This is because behind the scenes they are using distances between data points to determine their similarity.

Normalization vs Standardization V

For example, let's say we have data containing high school CGPA scores of students (ranging from 0 to 5) and their future incomes (in thousands Rupees):

Student	CGPA	Salary '000
0	1	60
1	2	40
2	3	40
3	4	50
4	5	52

Normalization vs Standardization VI

Since both the features have different scales, there is a chance that higher weightage is given to features with higher magnitude. This will impact the performance of the machine learning algorithm and obviously, we do not want our algorithm to be biased towards one feature.

Therefore, we scale our data before employing a distance based algorithm so that all the features contribute equally to the result.

Normalization vs Standardization VII

	Student	CGPA	Salary '000
0	1	-1.184341	1.520013
1	2	-1.184341	-1.100699
2	3	0.416120	-1.100699
3	4	1.216350	0.209657
4	5	0.736212	0.471728

The effect of scaling is conspicuous when we compare the Euclidean distance between data points for students A and B, and between B and C, before and after scaling as shown below:

$$\text{Distance AB before scaling} \Rightarrow \sqrt{(40 - 60)^2 + (3 - 3)^2} = 20$$

$$\text{Distance BC before scaling} \Rightarrow \sqrt{(40 - 40)^2 + (4 - 3)^2} = 1$$

$$\text{Distance AB after scaling} \Rightarrow \sqrt{(1.1 - 1.5)^2 + (1.18 - 1.18)^2} = 2.6$$

Normalization vs Standardization VIII

Distance BC after scaling => $\sqrt{(1.1 - 1.1)^2 + (0.41 - 1.18)^2} = 1.59$

Scaling has brought both the features into the picture and the distances are now more comparable than they were before we applied scaling.

Tree-Based Algorithms

Tree-based algorithms, on the other hand, are fairly insensitive to the scale of the features. Think about it, a decision tree is only splitting a node based on a single feature. The decision tree splits a node on a feature that increases the homogeneity of the node. This split on a feature is not influenced by other features.

So, there is virtually no effect of the remaining features on the split. This is what makes them invariant to the scale of the features!

Normalization vs Standardization IX

What is Normalization?

- ✓ Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4)$$

Here, X_{max} and X_{min} are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0

Normalization vs Standardization X

- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

What is Standardization?

- ✓ Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation.
- ✓ This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Normalization vs Standardization XI

$$X' = \frac{X - \mu}{\sigma} \quad (5)$$

μ is the mean of the feature values and σ is the standard deviation of the feature values. **The values are not restricted to a particular range.**

The Big Question – Normalize or Standardize?

- ✓ Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.

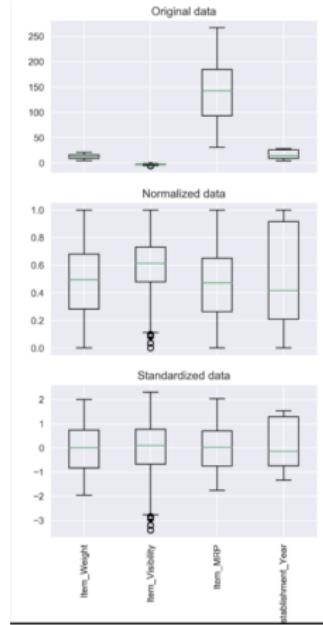
Normalization vs Standardization XII

- ✓ Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true.
- ✓ Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.
- ✓ However, the choice of using normalization or standardization will depend on your problem and the machine learning algorithm you are using.
- ✓ There is no hard and fast rule to tell you when to normalize or standardize your data.

Normalization vs Standardization XIII

- ✓ You can always start by fitting your model to raw, normalized and standardized data and compare the performance for best results.
- ✓ It is a good practice to fit the scaler on the training data and then use it to transform the testing data. This would avoid any data leakage during the model testing process. Also, the scaling of target values is generally not required.

Normalization vs Standardization XIV

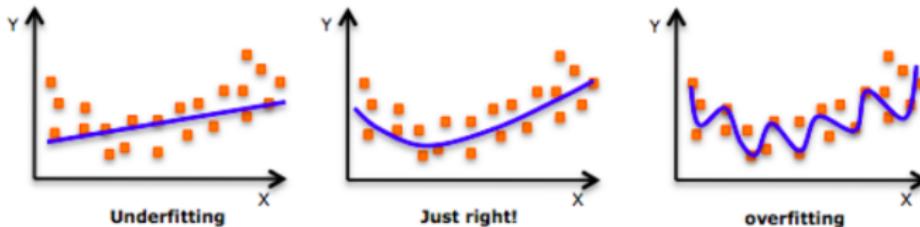


Regularization Techniques in Deep Learning I

- ✓ One of the most common problems data science professionals face is to avoid overfitting.
- ✓ A situation where a model performed exceptionally well on train data but was not able to predict test data.
- ✓ Avoiding overfitting can single-handedly improve our model's performance.
- ✓ Regularization helps in overcoming the same problem.

What is Regularization?

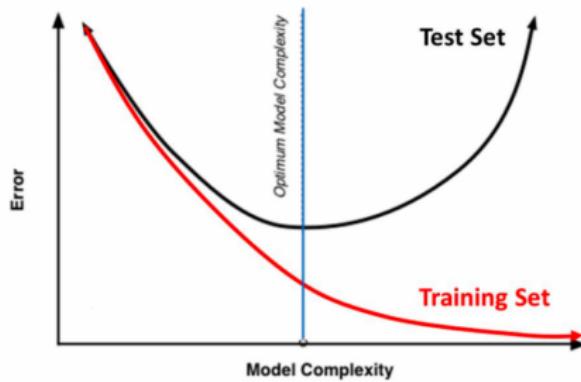
Regularization Techniques in Deep Learning II



- ✓ As we move towards the right in this image, the model tries to learn too well the details and the noise from the training data, which ultimately results in poor performance on the unseen data.
- ✓ In other words, while going towards the right, the complexity of the model increases such that the training error reduces but the testing error doesn't.

Regularization Techniques in Deep Learning III

Training Vs. Test Set Error

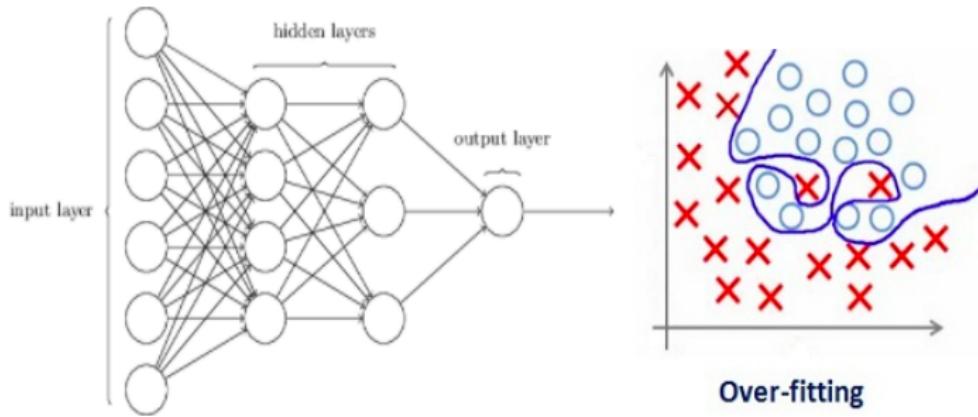


Regularization Techniques in Deep Learning IV

- ✓ A neural network makes model more prone to overfitting.
- ✓ Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.

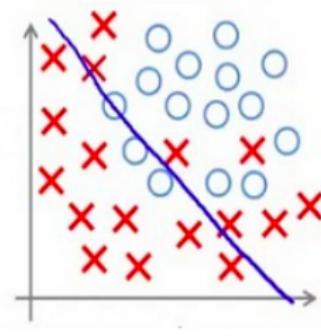
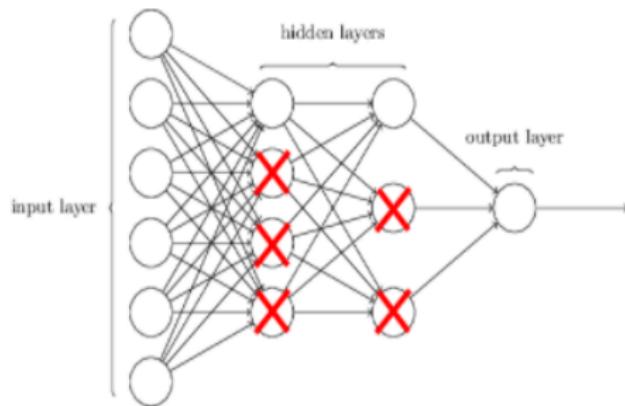
How does Regularization help reduce Overfitting?

Regularization Techniques in Deep Learning V



Regularization Techniques in Deep Learning VI

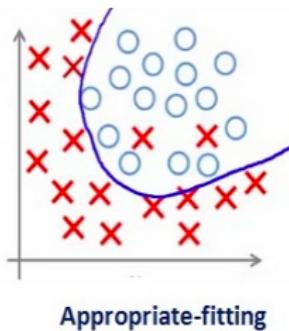
- ✓ In machine learning, a fair idea that regularization penalizes the coefficients. In deep learning, it actually penalizes the weight matrices of the nodes.
- ✓ Assume that our regularization coefficient is so high that some of the weight matrices are nearly equal to zero.



Under-fitting

Regularization Techniques in Deep Learning VII

- ✓ This will result in a much simpler linear network and slight underfitting of the training data.
- ✓ Such a large value of the regularization coefficient is not that useful. We need to optimize the value of regularization coefficient in order to obtain a well-fitted model as shown in the image below.



Different Regularization Techniques in Deep Learning

- ✓ A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression.

L2 & L1 regularization

- ✓ L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the *regularization term*.

Cost function = Loss (say, binary cross entropy) + Regularization term

- ✓ Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller

Regularization Techniques in Deep Learning IX

weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

However, this regularization term differs in L2.

In L2:

$$\text{Cost function} = Loss + \frac{\lambda}{2m} * \sum \| w \|^2 \quad (6)$$

Here, λ is the regularization parameter. It is the hyperparameter whose value is optimized for better results.

- ✓ L2 regularization is also known as weight decay as it forces the weights to decay towards zero (but not exactly zero).

Regularization Techniques in Deep Learning X

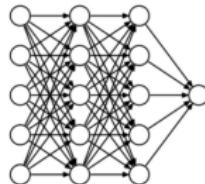
In L1:

$$\text{Cost function} = Loss + \frac{\lambda}{2m} * \sum \| w \| \quad (7)$$

- ✓ In this, we penalize the absolute value of the weights. Unlike L2, the weights may be reduced to zero here. Hence, it is very useful when we are trying to compress our model. Otherwise, we usually prefer L2 over it.

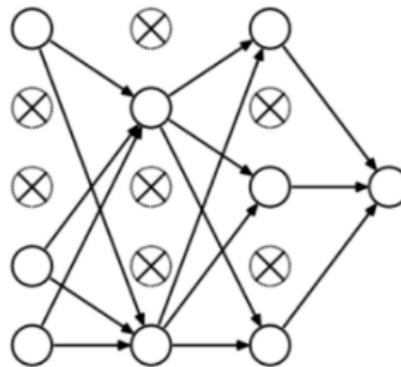
Dropout:

- ✓ This is the one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.
- ✓ To understand dropout, let's say our neural network structure is akin to the one shown below:



Regularization Techniques in Deep Learning XII

- ✓ At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below.



Regularization Techniques in Deep Learning XIII

- ✓ So each iteration has a different set of nodes and this results in a different set of outputs. It can also be thought of as an ensemble technique in machine learning.
- ✓ Ensemble models usually perform better than a single model as they capture more randomness. Similarly, dropout also performs better than a normal neural network model.
- ✓ This probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function. Dropout can be applied to both the hidden layers as well as the input layers.

Regularization Techniques in Deep Learning XIV



Regularization Techniques in Deep Learning XV

- ✓ *Due to these reasons, dropout is usually preferred when we have a large neural network structure in order to introduce more randomness.*

Data Augmentation: The simplest way to reduce overfitting is to increase the size of the training data. In machine learning, it is not possible to increase the size of training data as the labeled data was too costly.

- ✓ For example, dealing with images. In this case, there are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc.
- ✓ In the below image, some transformation has been done on the handwritten digits dataset.

Regularization Techniques in Deep Learning XVI



✓ This technique is known as data augmentation. This usually provides a big leap in improving the accuracy of the model. It can be considered as a mandatory trick in order to improve the predictions.

Early stopping: Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set. When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

Regularization Techniques in Deep Learning XVII



- ✓ In the above image, we will stop training at the dotted line since after that our model will start overfitting on the training data.

Weight Initialization I

Why do we need it?

- ✓ There are three main reasons why it has been used.
 - ① One of the reasons is that it can avoid Vanishing, exploding gradients problem in the late part of training.
 - ② Vanishing gradients problems refer to the problem which occurs when weights don't change during the training (occurs when weights are initialized very much less than 1).
 - ③ Exploding gradients problem refers to the problem which occurs when weights keep on increasing rapidly (occurs when weights are initialized greater than 1)

Weight Initialization II

- ④ Another reason is it can speed up the training process as it can learn faster with optimal initialization.

Key Points:

- ① Weights should be small
- ② Weights should not be same
- ③ Weights should follow the same distribution
- ④ Weights should have good variance

Different ways to initialize the weight

- ① Uniform Distribution:

$$W_{ij} = \text{Uniform}\left[\frac{-1}{\sqrt{f_n}}, \frac{1}{\sqrt{f_n}}\right], f_n \text{ means number of inputs.}$$

This works well for the sigmoid activation function.

Weight Initialization III

② Xavier/Gorat Distribution:

- ① Here the weight is initialization using the normal distribution:

$W_{ij} = N(\text{mean}, \text{std})$, $\text{mean} = 0$, $\text{std} = \sqrt{\frac{2}{(f_i + f_o)}}$, f_i and f_o are number of inputs and outputs, respectively.

- ② Here the weight is initialization using a uniform distribution:

$W_{ij} = \text{Uniform} \left[-\frac{\sqrt{6}}{\sqrt{(f_i + f_o)}}, \frac{\sqrt{6}}{\sqrt{(f_i + f_o)}} \right]$, f_i and f_o are number of inputs and outputs, respectively.

③ He init:

- ① He uniform : Here the weight is initialization using the normal distribution.

$W_{ij} = \text{Uniform} \left[-\sqrt{\frac{6}{f_i}}, \sqrt{\frac{6}{f_i}} \right]$, f_i is number of inputs.

Weight Initialization IV

- ② He Normal : Here the weight is initialization using the normal distribution.

$$W_{ij} \sim N(\text{mean}, \text{std}), \text{mean} = 0, \text{std} = \sqrt{\frac{2}{f_i}}, f_i \text{ is number of inputs.}$$

This works well for the ReLU activation function.

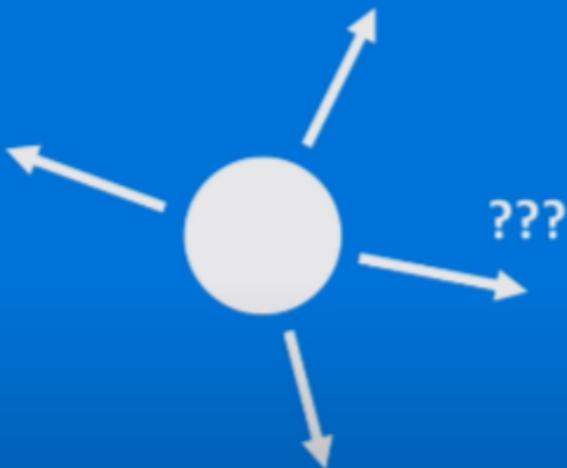
Recurrent Neural Network I

Given an image of a ball,
can you predict where it will go next?



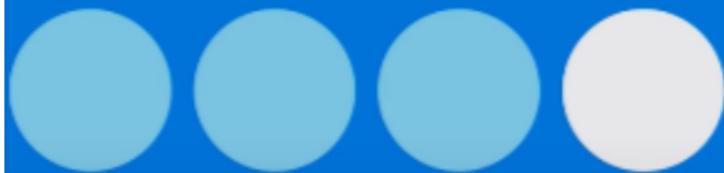
Recurrent Neural Network II

Given an image of a ball,
can you predict where it will go next?



Recurrent Neural Network III

Given an image of a ball,
can you predict where it will go next?



Recurrent Neural Network IV

Given an image of a ball,
can you predict where it will go next?



Sequences in the Wild



Audio

Recurrent Neural Network VI

“working love learning we on deep”, did this make any sense to you? Not really –

read this one – “We love working on deep learning”. Made perfect sense!

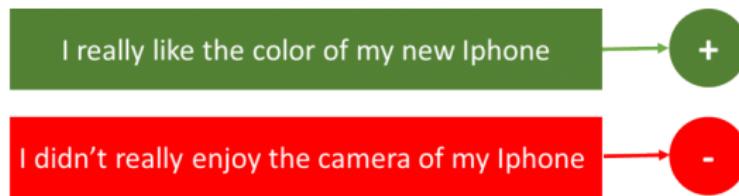
A little jumble in the words made the sentence incoherent.

- ✓ Can we expect a neural network to make sense out of it? Not really!
- ✓ If the human brain was confused on what it meant I am sure a neural network is going to have a tough time deciphering such text.

The beauty of recurrent neural networks lies in their diversity of application. When we are dealing with RNNs they have a great ability to deal with various input and output types.

Recurrent Neural Network VII

- **Sentiment Classification** – This can be a task of simply classifying tweets into positive and negative sentiment. So here the input would be a tweet of varying lengths, while output is of a fixed type and size.



- **Image Captioning** – Here, let's say we have an image for which we need a textual description. So we have a single input – the image, and a series or sequence of words as output. Here the image might be of a fixed size, but the output is a description of varying lengths

Recurrent Neural Network VIII



A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.

Recurrent Neural Network IX

- **Language Translation** – This basically means that we have some text in a particular language let's say English, and we wish to translate it in French. Each language has its own semantics and would have varying lengths for the same sentence. So here the inputs as well as outputs are of varying lengths.

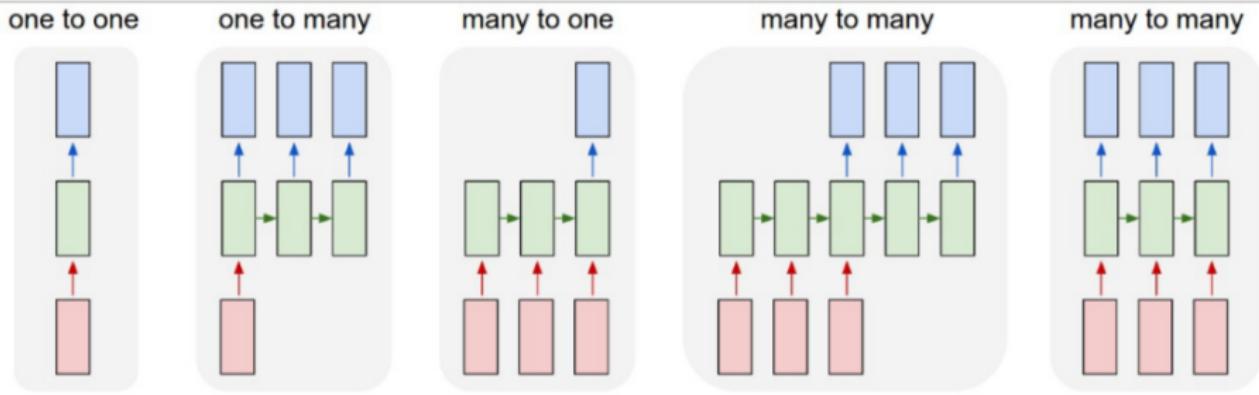
Recurrent Neural Network X

French was the official language of the colony of French Indochina, comprising modern-day Vietnam, Laos, and Cambodia. It continues to be an administrative language in Laos and Cambodia, although its influence has waned in recent years.

Translate into : French

Le français était la langue officielle de la colonie de l'Indochine française, comprenant le Vietnam d'aujourd'hui, le Laos et le Cambodge. Il continue d'être une langue administrative au Laos et au Cambodge, bien que son influence a décliné au cours des dernières années.

Recurrent Neural Network XI



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: (1) Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). (2) Sequence output (e.g. image captioning takes an image and outputs a sentence of words). (3) Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). (4) Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). (5) Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.

Recurrent Neural Network XII

Examples of sequence data

Speech recognition



"The quick brown fox jumped
over the lazy dog."

Music generation



Sentiment classification

"There is nothing to like
in this movie."



DNA sequence analysis

AGCCCCCTGTGAGGAACTAG



AGCCC**C**TGTGAGGAACTAG

Machine translation

Voulez-vous chanter avec
moi?



Do you want to sing with
me?

Video activity recognition



Running

Name entity recognition

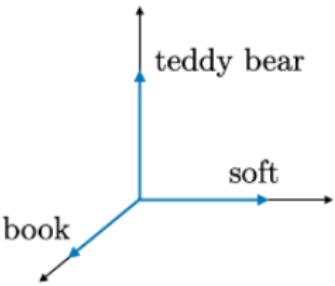
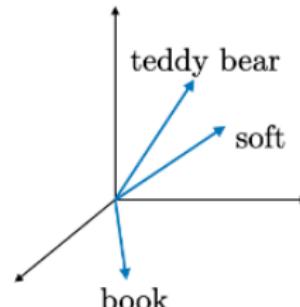
Yesterday, Harry Potter
met Hermione Granger.



Yesterday, **Harry** Potter
met **Hermione** Granger.

Recurrent Neural Network XIII

✓ **Word Representation techniques:** The two main ways of representing words.

1-hot representation	Word embedding
	
<ul style="list-style-type: none">• Noted o_w• Naive approach, no similarity information	<ul style="list-style-type: none">• Noted e_w• Takes into account words similarity

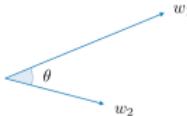
Recurrent Neural Network XIV

- ✓ **Embedding matrix:** For a given word w , the embedding matrix E is a matrix that maps its 1-hot representation o_w tp its embedding e_w as follows:

$$e_w = E o_w \quad (8)$$

Comparing words:

- ✓ Cosine similarity : The cosine similarity between words w_1 and w_2 is expressed as follows: $\text{similarity} = \frac{w_1 \cdot w_2}{\|w_1\| \cdot \|w_2\|} = \cos \theta$



Recurrent Neural Network XV

x:

Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$

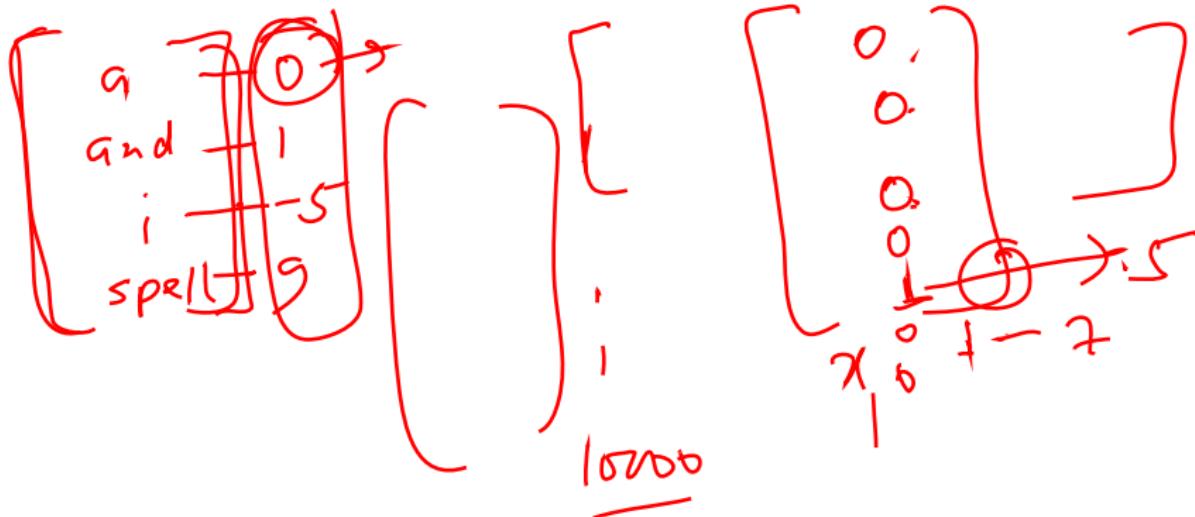
$x^{<2>}$

$x^{<3>}$

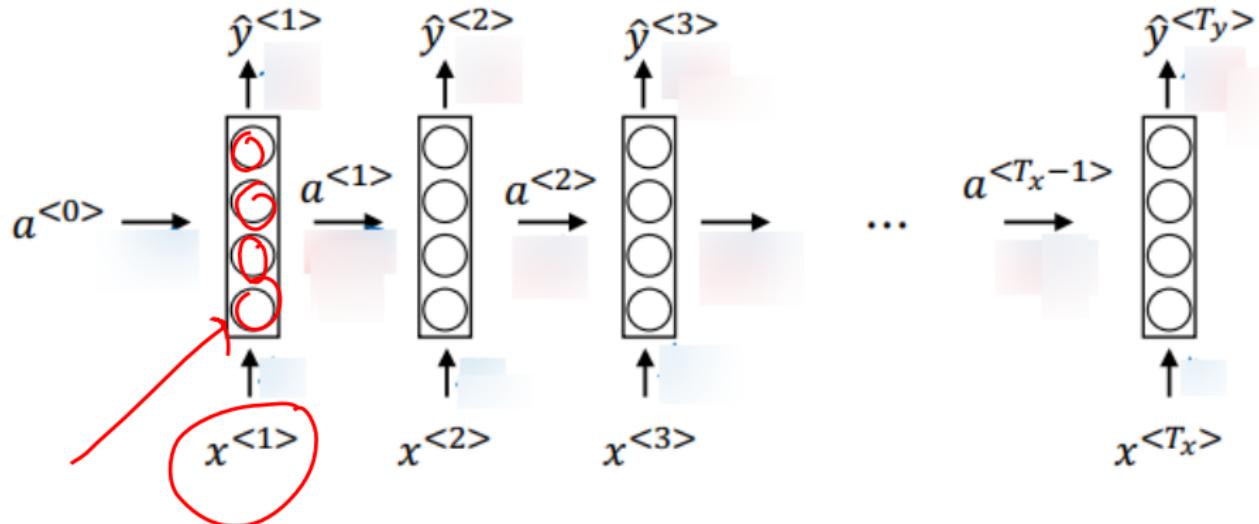
...

$x^{<9>}$

1



Recurrent Neural Network XVI

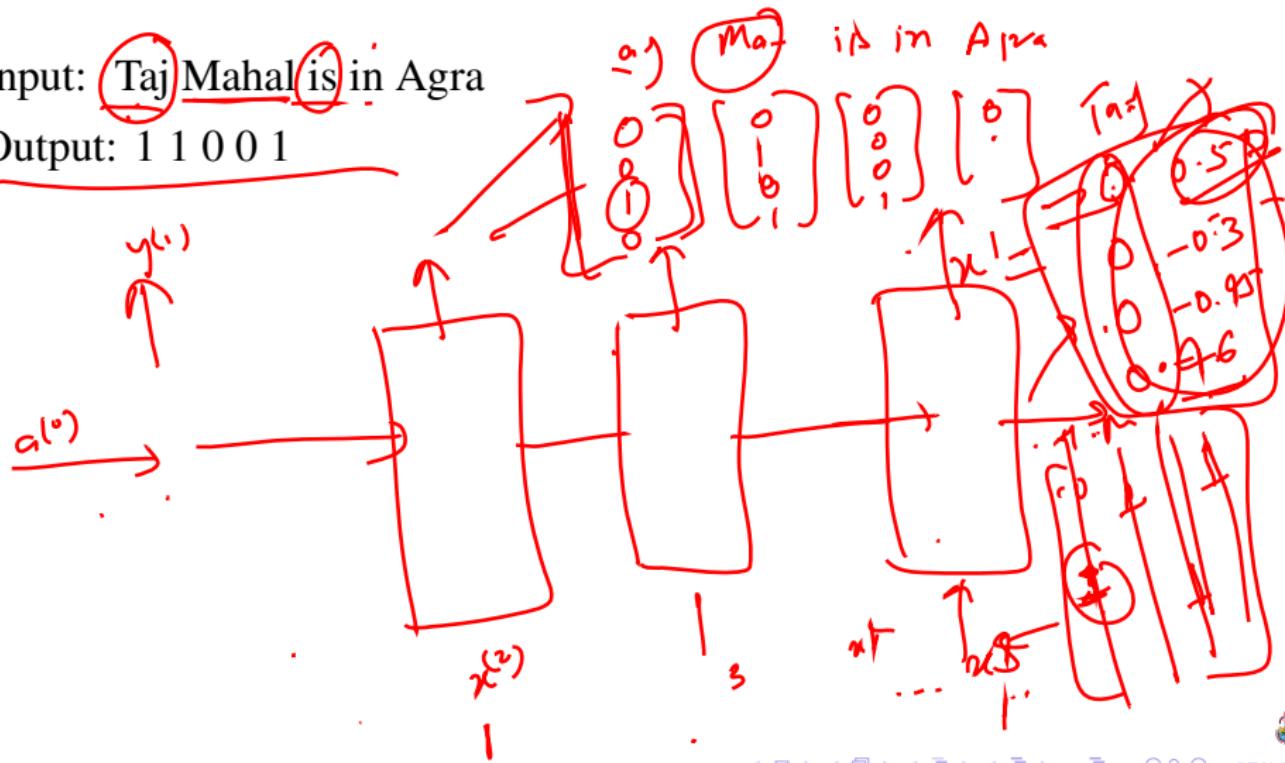


Recurrent Neural Network XVII

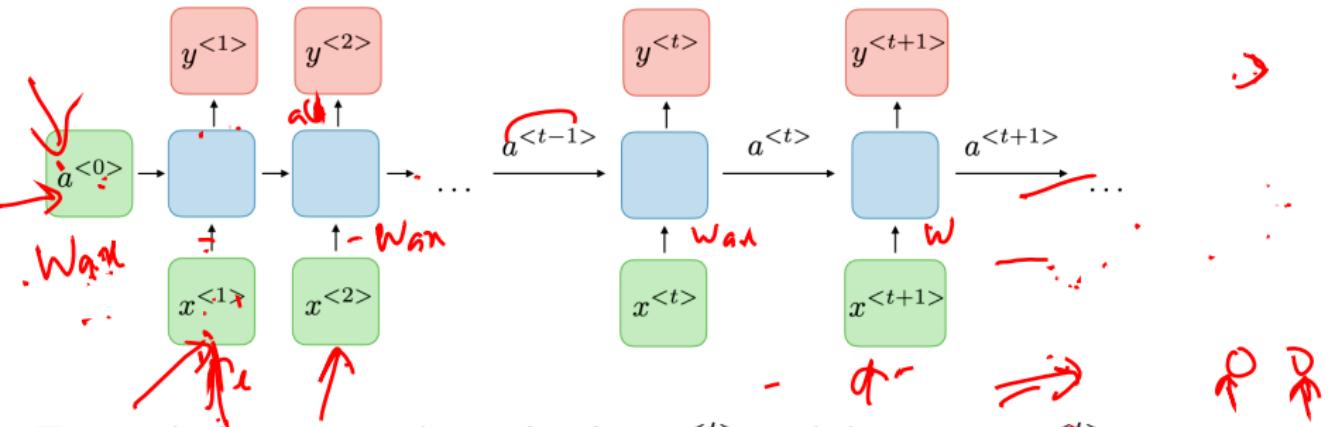
Name Entity Recognition Through DNNs

Input: Taj Mahal is in Agra

Output: 1 1 0 0 1



Recurrent Neural Network XVIII



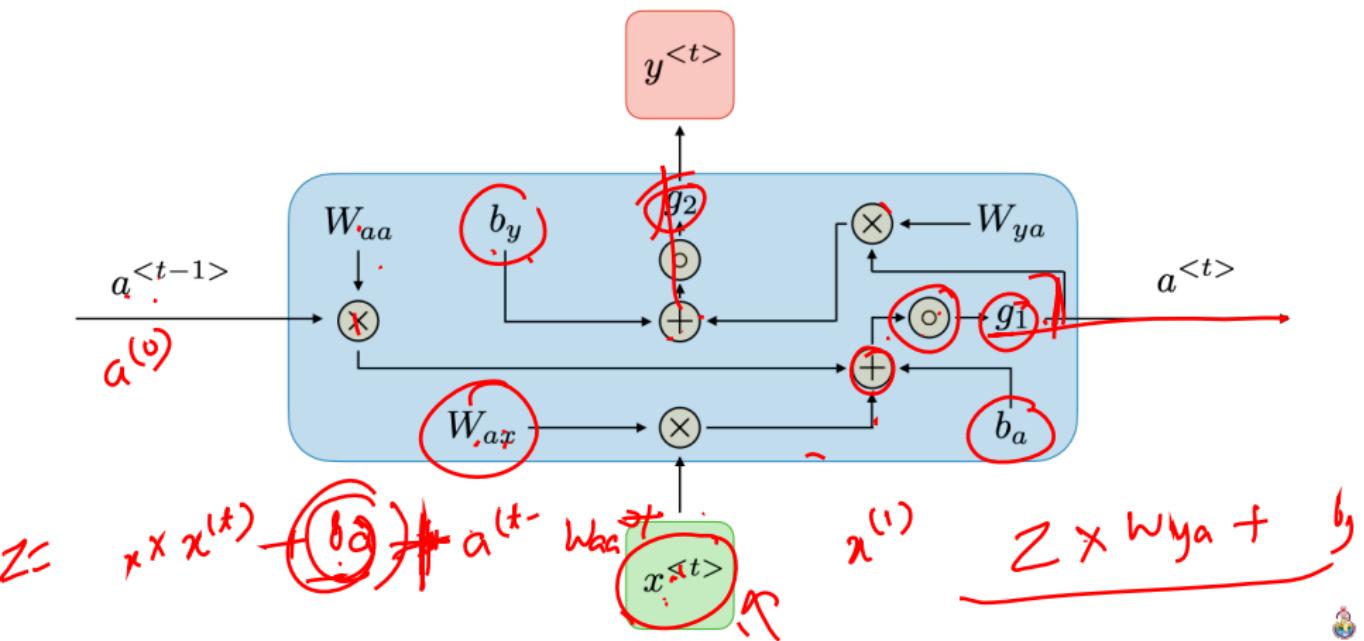
For each timestep t , the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1 (W_{ax}x^{<t>} + W_{aa}a^{<t-1>} + b_a) \quad (9)$$

$$y^{<t>} = g_2 (W_{ya}a^{<t>} + b_y) \quad (10)$$

Recurrent Neural Network XIX

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.



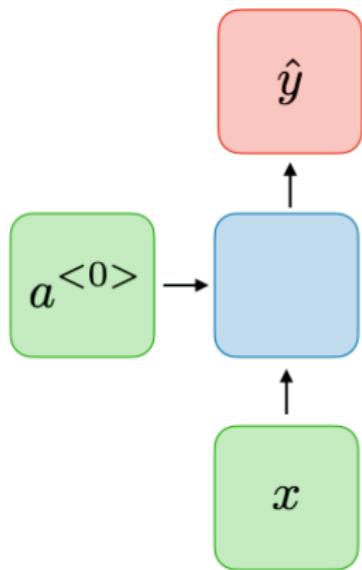
Recurrent Neural Network XX

The pros and cons of a typical RNN architecture are summed up in the table below:

Advantages	Drawbacks
<ul style="list-style-type: none">• Possibility of processing input of any length• Model size not increasing with size of input• Computation takes into account historical information• Weights are shared across time	<ul style="list-style-type: none">• Computation being slow• Difficulty of accessing information from a long time ago• Cannot consider any future input for the current state

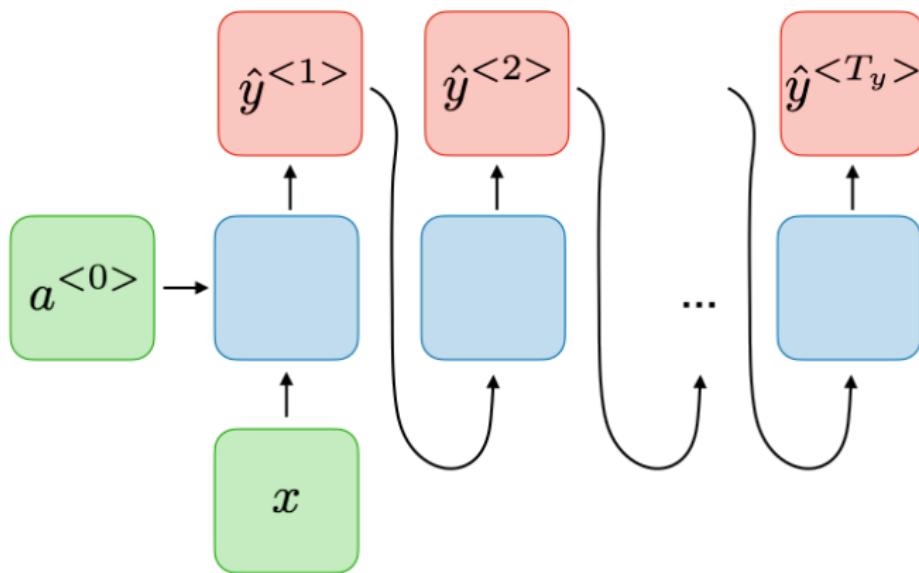
Applications of RNNs—RNN models are mostly used in the fields of natural language processing and speech recognition. The different applications are summed up in the table below:

One-to-one: $T_x = T_y = 1$, **Example:** Traditional neural network



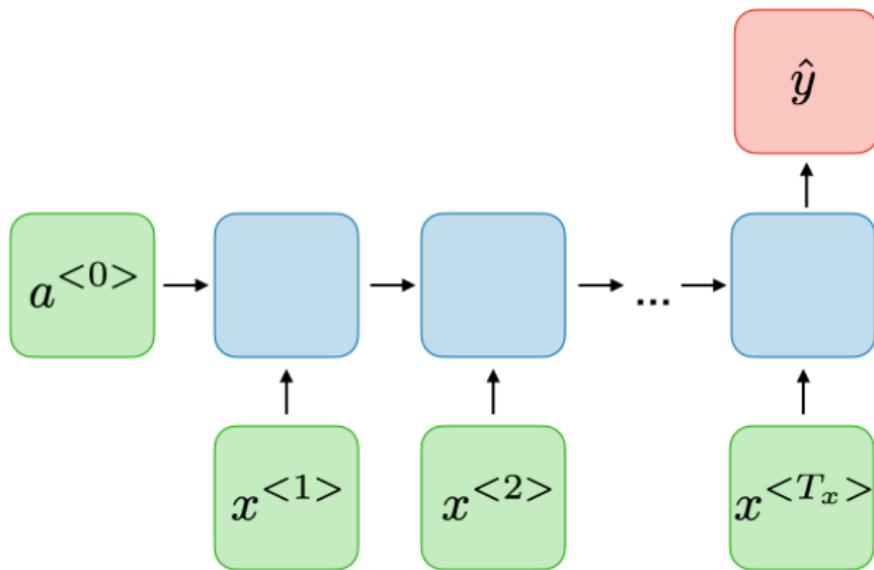
Recurrent Neural Network XXII

One-to-many: $T_x = 1, T_y > 1$, **Example:** Music generation



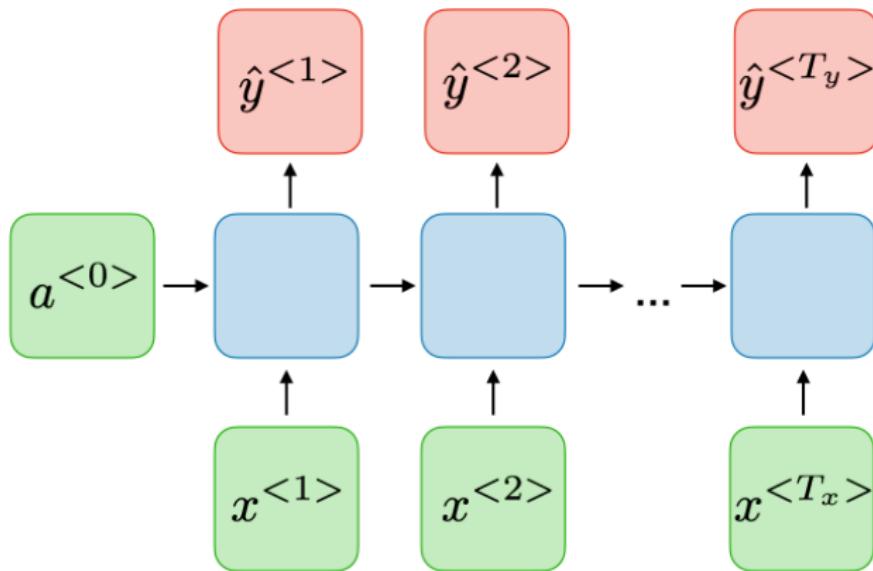
Recurrent Neural Network XXIII

Many-to-one: $T_x > 1, T_y = 1$, **Example:** Sentiment classification

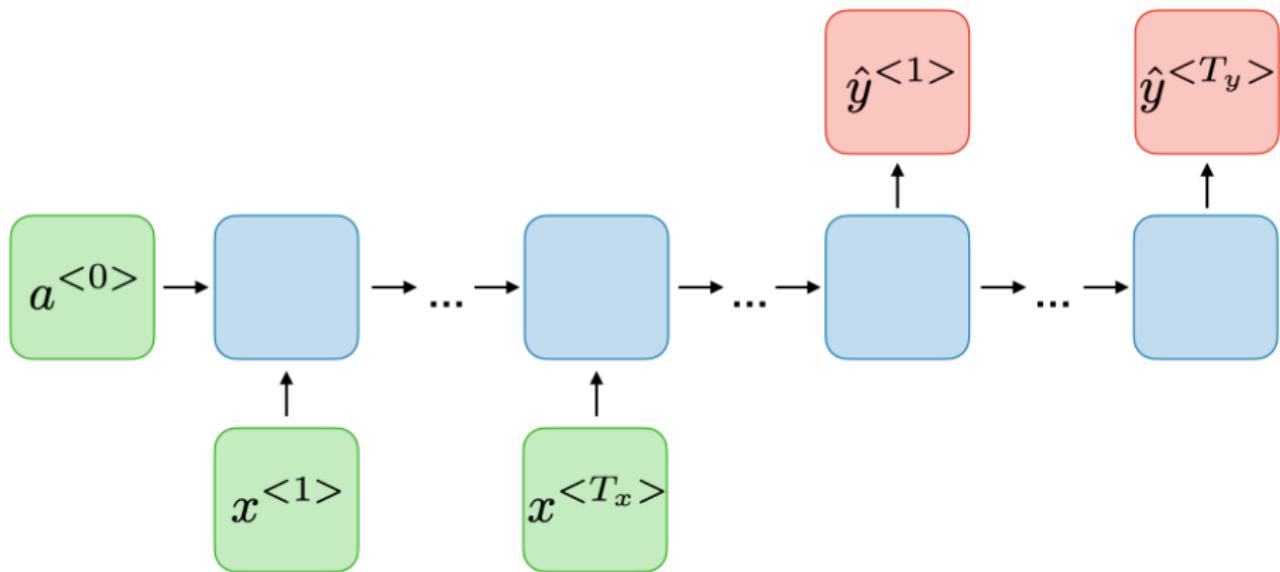


Recurrent Neural Network XXIV

Many-to-many: $T_x = T_y$, Example: Name entity recognition



Many-to-many: $T_x \neq T_y = 1$, **Example:** Machine translation



Recurrent Neural Network XXVI

Loss function: In the case of a recurrent neural network, the loss function \mathcal{L} of all time steps is defined based on the loss at every time step as follows:

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{}, y^{}) \quad (11)$$

Recurrent Neural Network XXVII

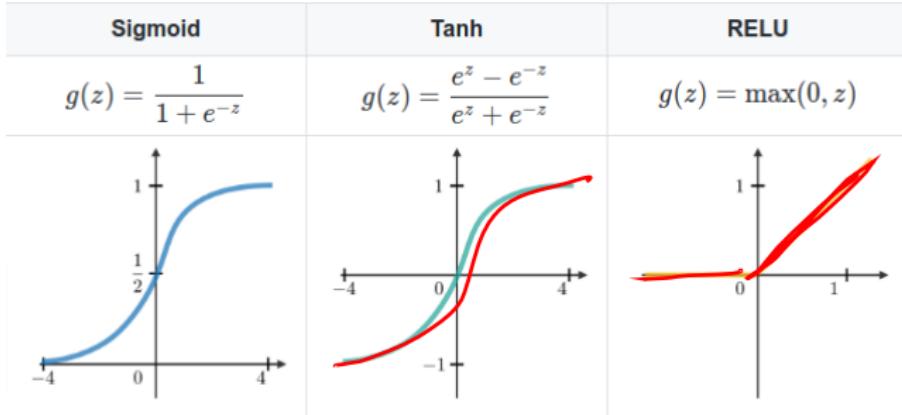
Backpropagation through time– Backpropagation is done at each point in time. At timestep T , the derivative of the loss \mathcal{L} with respect to weight matrix W is expressed as follows:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T_y} \frac{\partial \mathcal{L}^{(T)}}{\partial W}|_{(t)} \quad (12)$$

Recurrent Neural Network XXVIII

Handling long term dependencies:

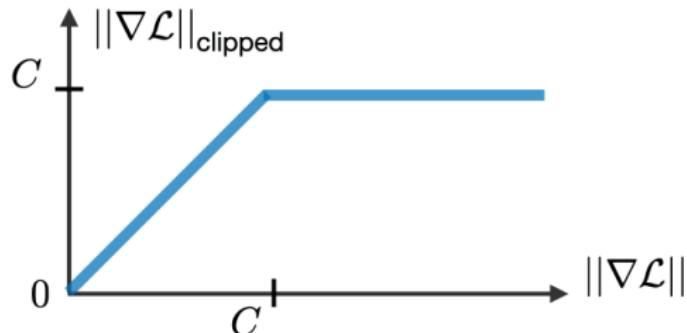
✓ **Commonly used activation functions:** The most common activation functions used in RNN modules are described below:



Recurrent Neural Network XXIX

- ✓ **Vanishing/exploding gradient:** The vanishing and exploding gradient phenomena are often encountered in the context of RNNs. The reason why they happen is that it is difficult to capture long term dependencies because of multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers.
- ✓ **Gradient clipping:** It is a technique used to cope with the exploding gradient problem sometimes encountered when performing backpropagation. By capping the maximum value for the gradient, this phenomenon is controlled in practice.

Recurrent Neural Network XXX



✓ **Types of gates:** In order to remedy the vanishing gradient problem, specific gates are used in some types of RNNs and usually have a well-defined purpose. They are usually noted Γ and are equal to:

$$\Gamma = \sigma(Wx^{} + Ua^{} + b) \quad (13)$$

Recurrent Neural Network XXXI

where W , U , b are coefficients specific to the gate and σ is the sigmoid function. The main ones are summed up in the table below:

Type of gate	Role	Used in
Update gate Γ_u	How much past should matter now?	GRU, LSTM
Relevance gate Γ_r	Drop previous information?	GRU, LSTM
Forget gate Γ_f	Erase a cell or not?	LSTM
Output gate Γ_o	How much to reveal of a cell?	LSTM

Recurrent Neural Network XXXII

✓ **GRU/LSTM:** Gated Recurrent Unit (GRU) and Long Short-Term Memory units (LSTM) deal with the vanishing gradient problem encountered by traditional RNNs, with LSTM being a generalization of GRU. Below is the characterizing equations of each architecture:

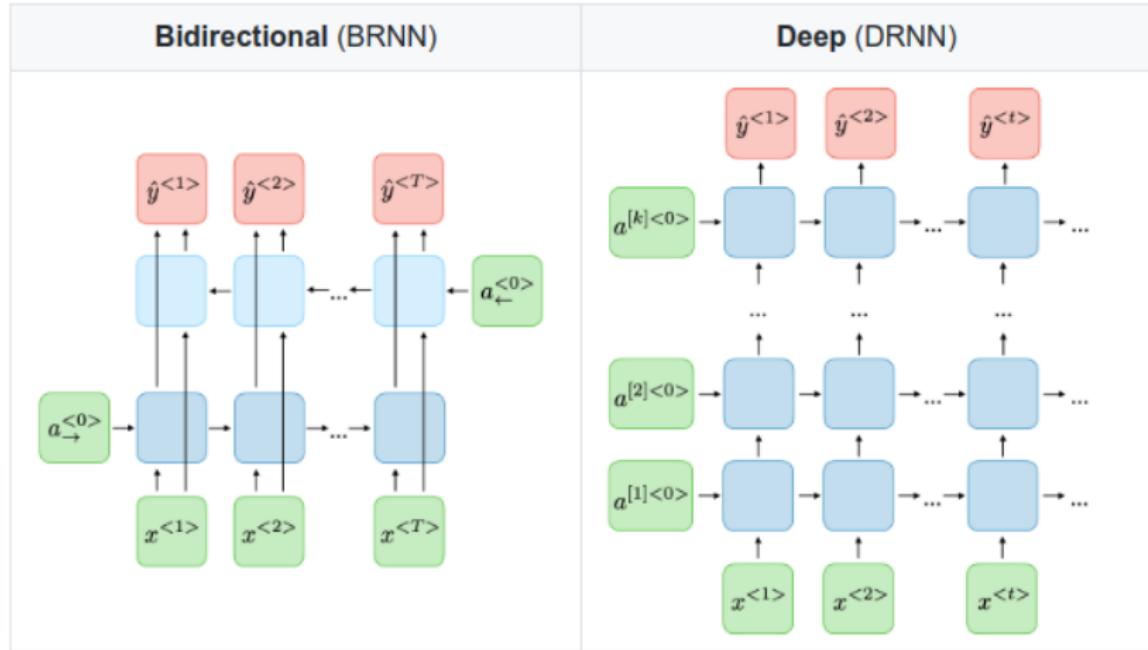
Recurrent Neural Network XXXIII

Characterization	Gated Recurrent Unit (GRU)	Long Short-Term Memory (LSTM)
$\tilde{c}^{}$	$\tanh(W_c[\Gamma_r \star a^{}, x^{}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{}, x^{}] + b_c)$
$c^{}$	$\Gamma_u \star \tilde{c}^{} + (1 - \Gamma_u) \star c^{}$	$\Gamma_u \star \tilde{c}^{} + \Gamma_f \star c^{}$
$a^{}$	$c^{}$	$\Gamma_o \star c^{}$
Dependencies		

Remark: the sign \star denotes the element-wise multiplication between two vectors.

Recurrent Neural Network XXXIV

Variants of RNNs: The commonly used RNN architectures:



Gated Recurrent Unit

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al.

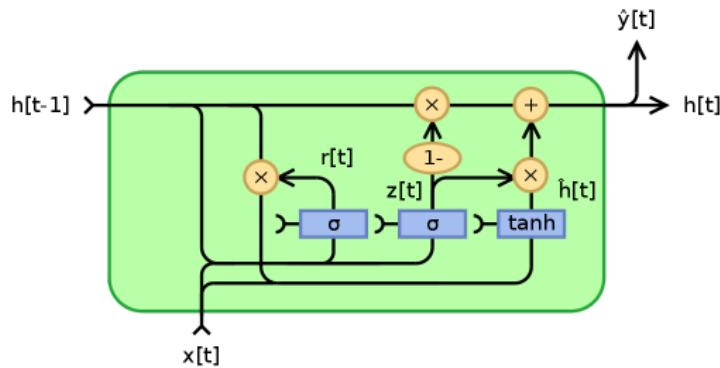
- ✓ The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it **lacks an output gate**.
- ✓ GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM.
- ✓ GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets.

Recurrent Neural Network XXXVI

Architecture:

There are several variations on the full gated unit, with gating done using the previous hidden state and the bias in various combinations, and a simplified form called minimal gated unit.

① Fully gated unit:



Recurrent Neural Network XXXVII

Initially, for $t = 0$, the output vector is $h_0 = 0$.

$$\sigma_g(W_z x_t + U_z h_{t-1} + b_z) \quad (14)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \quad (15)$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \quad (16)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \quad (17)$$

x_t : input vector

h_t : output vector

\hat{h}_t : candidate activation vector

Recurrent Neural Network XXXVIII

z_t : update gate vector

r_t : reset gate vector

W , U and b : parameter matrices and vector

The operator \odot denotes the Hadamard product

For two matrices A and B of the same dimension $m \times n$, the Hadamard product $A \circ B$ (or $A \odot B$) is a matrix of the same dimension as the operands, with elements given by:

$$(A \circ B)_{ij} = (A \odot B)_{ij} = (A)_{ij}(B)_{ij} \quad (18)$$

Recurrent Neural Network XXXIX

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}$$

Activation functions:

- ✓ σ_g : The original is a sigmoid function.
- ✓ ϕ_h : The original is a hyperbolic tangent.
- ✓ Alternative activation functions are possible, provided that $\sigma_g(x) \in [0, 1]$.

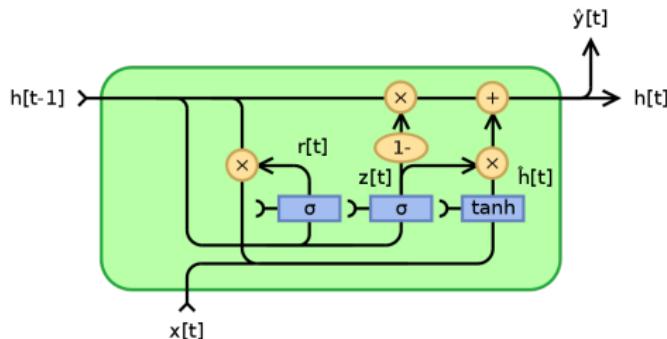
Alternate forms can be created by changing $z_t z_t$ and $r_t r_t$

Recurrent Neural Network XL

- Type 1, each gate depends only on the previous hidden state and the bias.

$$z_t = \sigma_g(U_z h_{t-1} + b_z) \quad (19)$$

$$r_t = \sigma_g(U_r h_{t-1} + b_r) \quad (20)$$

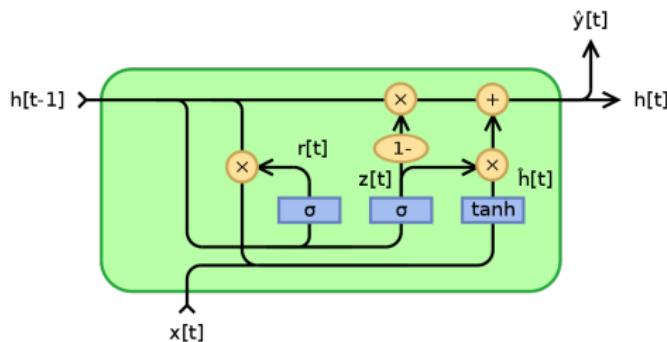


Recurrent Neural Network XLI

- Type 2, each gate depends only on the previous hidden state.

$$z_t = \sigma_g(U_z h_{t-1}) \quad (21)$$

$$r_t = \sigma_g(U_r h_{t-1}) \quad (22)$$

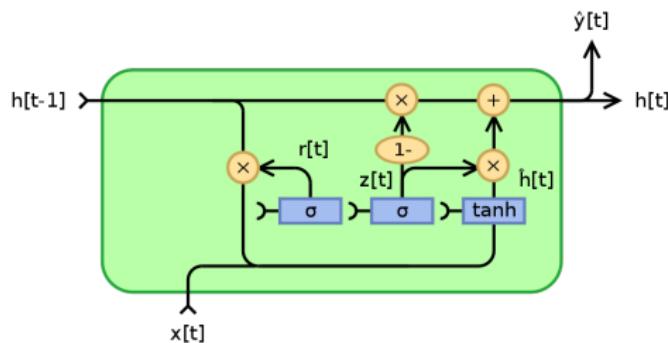


Recurrent Neural Network XLII

- Type 3, each gate is computed using only the bias.

$$z_t = \sigma_g(b_z) \quad (23)$$

$$r_t = \sigma_g(b_r) \quad (24)$$



Recurrent Neural Network XLIII

② Minimal gated unit: The minimal gated unit is similar to the fully gated unit, except the update and reset gate vector is merged into a forget gate. This also implies that the equation for the output vector must be changed:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (25)$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(f_t \odot h_{t-1}) + b_h) \quad (26)$$

$$h_t = (1 - f_t) \odot h_{t-1} + f_t \odot \hat{h}_t \quad (27)$$

Variables

Recurrent Neural Network XLIV

x_t : input vector

h_t : output vector

\hat{h}_t : candidate activation vector

f_t : forget vector

W , U and b : parameter matrices and vector

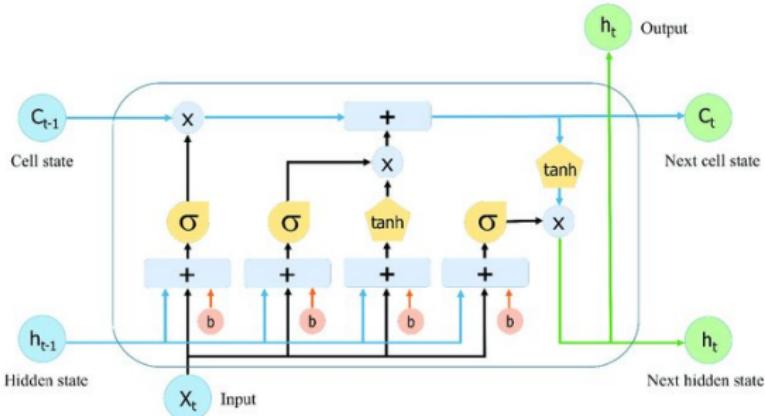
Long Short Term Memory (LSTM) Unit

- ✓ Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.
- ✓ They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people.
- ✓ LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!
- ✓ All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

Recurrent Neural Network XLVI

- ✓ A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.
- ✓ LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.
- ✓ LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

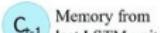
Recurrent Neural Network XLVII



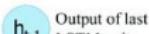
Inputs:



X_t Current input

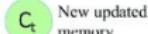


C_{t-1} Memory from last LSTM unit



h_{t-1} Output of last LSTM unit

Outputs:



C_t New updated memory



h_t Current output

Nonlinearities:



σ Sigmoid layer



\tanh Tanh layer



b Bias

Vector operations:



x Scaling of information



$+$ Adding information

Recurrent Neural Network XLVIII

LSTM with a forget gate

The compact forms of the equations for the forward pass of an LSTM cell with a forget gate are:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (28)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (29)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (30)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \quad (31)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (32)$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (33)$$

Recurrent Neural Network XLIX

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator \circ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

Variables

$\vec{x}_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0, 1)^h$: forget gate's activation vector

$i_t \in (0, 1)^h$: input/update gate's activation vector

$o_t \in (0, 1)^h$: output gate's activation vector

$h_t \in (-1, 1)^h$: hidden state vector also known as output vector of the LSTM unit

$\tilde{c}_t \in (-1, 1)^h$: cell input activation vector

Recurrent Neural Network L

$c_t \in \mathbb{R}^h$: cell state vector

$W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector
parameters which need to be learned during training

where the superscripts d and h refer to the number of input features and
number of hidden units, respectively.

Activation functions

σ_g : sigmoid function.

σ_c : hyperbolic tangent function.

σ_h : hyperbolic tangent function.

Recurrent Neural Network LI

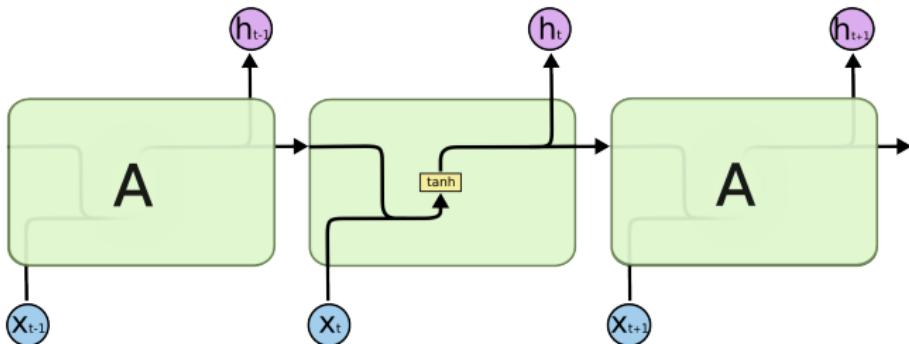


Figure: The repeating module in a standard RNN contains a single layer

- ✓ LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

Recurrent Neural Network LII

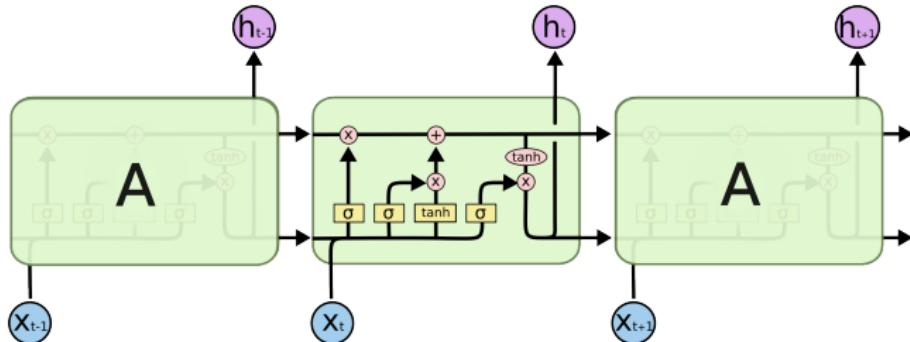


Figure: The repeating module in an LSTM contains four interacting layers

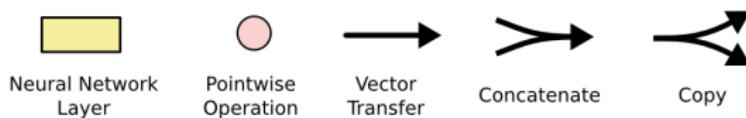
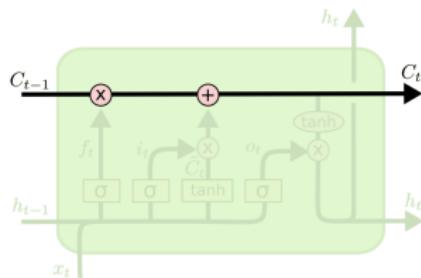


Figure: Notation

Recurrent Neural Network LIII

Steps in LSTM:



- ✓ The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.
- ✓ The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

Recurrent Neural Network LIV

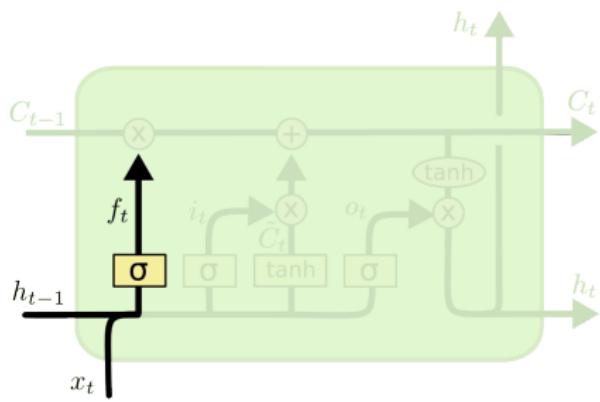
- ✓ The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Step-by-Step LSTM Walk Through

- ✓ The first step in our LSTM is to decide what information we're going to throw away from the cell state.
- ✓ This decision is made by a sigmoid layer called the forget gate layer. It looks at h_{t-1} and x_t , and outputs a number between 0 and 1 for each number in the cell state C_{t-1} .
- ✓ A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

Recurrent Neural Network LV

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (34)$$

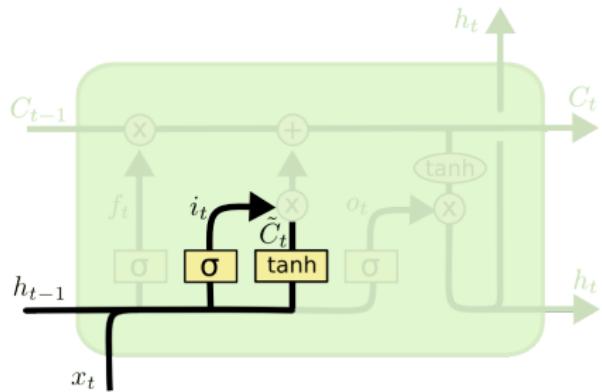


$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

Recurrent Neural Network LVI

- ✓ The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update.
- ✓ Next, a tanh layer creates a vector of new candidate values, \tilde{C}_t , that could be added to the state. In the next step, we'll combine these two to create an update to the state.

Recurrent Neural Network LVII



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

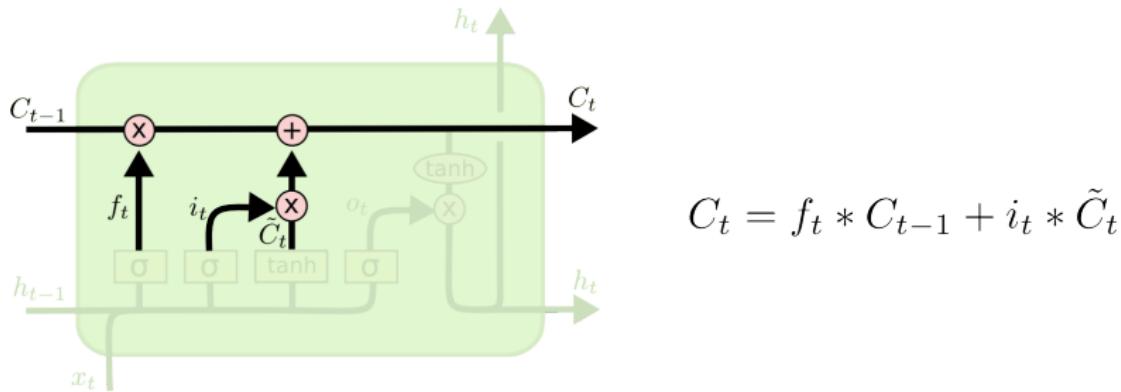
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (35)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (36)$$

Recurrent Neural Network LVIII

- ✓ It's now time to update the old cell state, C_{t-1} , into the new cell state C_t . The previous steps already decided what to do, we just need to actually do it.
- ✓ We multiply the old state by f_t , forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$.
- ✓ This is the new candidate values, scaled by how much we decided to update each state value.

Recurrent Neural Network LIX

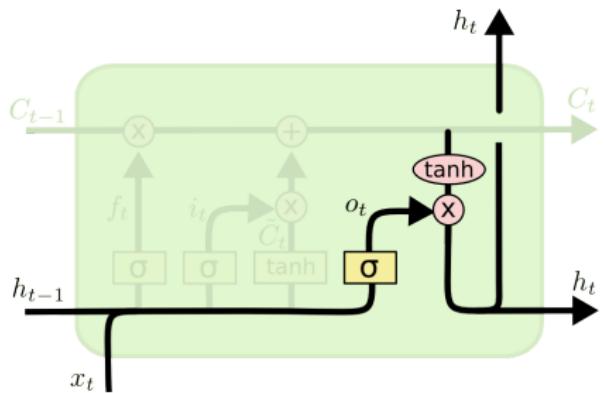


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (37)$$

- ✓ Put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



Recurrent Neural Network LX



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o) \quad (38)$$

$$h_t = o_t * \tanh(C_t) \quad (39)$$

Applications of LSTM:

- Robot control
- Time series prediction
- Speech recognition
- Rhythm learning
- Music composition
- Grammar learning
- Handwriting recognition
- Human action recognition
- Sign language translation

Recurrent Neural Network LXII

- Protein homology detection
- Predicting subcellular localization of proteins
- Time series anomaly detection
- Several prediction tasks in the area of business process management
- Prediction in medical care pathways
- Semantic parsing
- Object co-segmentation
- Airport passenger management
- Short-term traffic forecast

Recurrent Neural Network LXIII

- Drug design
- Market Prediction

Vanishing/Exploding Gradients in Deep Neural Networks I

- ✓ In Machine Learning, the Vanishing Gradient Problem is encountered while training Neural Networks with gradient-based methods (example, Back Propagation).
- ✓ This problem makes it hard to learn and tune the parameters of the earlier layers in the network.
- ✓ The vanishing gradients problem is one example of unstable behaviour that you may encounter when training a deep neural network.
- ✓ It describes the situation where a deep multilayer feed-forward network or a recurrent neural network is unable to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.

Vanishing/Exploding Gradients in Deep Neural Networks II

- ✓ The result is the general inability of models with many layers to learn on a given dataset, or for models with many layers to prematurely converge to a poor solution.

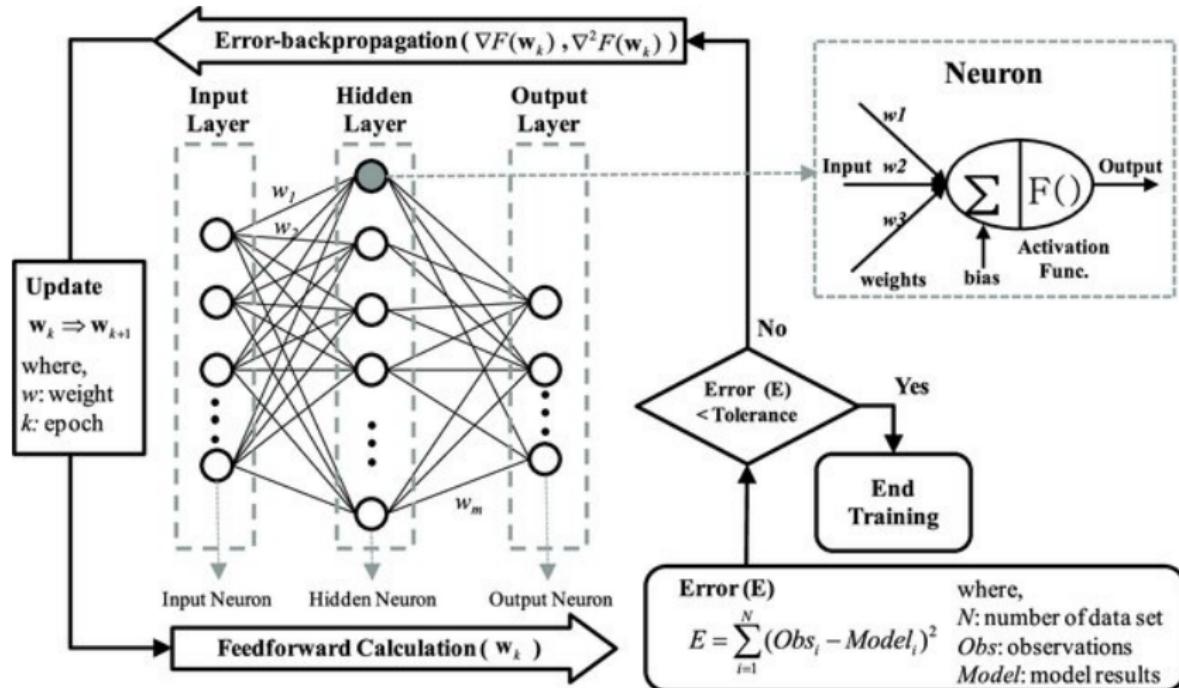
Why it's significant:

- ✓ For shallow network with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively.
- ✓ Gradients of neural networks are found using backpropagation. Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one.

Vanishing/Exploding Gradients in Deep Neural Networks III

- ✓ By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.
- ✓ However, 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.
- ✓ A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

Vanishing/Exploding Gradients in Deep Neural Networks IV



Vanishing/Exploding Gradients in Deep Neural Networks V

- After propagating the input features forward to the output layer through the various hidden layers consisting of different/same activation functions, we come up with a predicted probability of a sample belonging to the positive class (generally, for classification tasks).
- Now, the backpropagation algorithm propagates backward from the output layer to the input layer calculating the error gradients on the way.

Vanishing/Exploding Gradients in Deep Neural Networks VI

- Once the computation for gradients of the cost function w.r.t each parameter (weights and biases) in the neural network is done, the algorithm takes a gradient descent step towards the minimum to update the value of each parameter in the network using these gradients.
- 1** Vanishing – As the backpropagation algorithm advances downwards(or backward) from the output layer towards the input layer, the gradients often get smaller and smaller and approach zero which eventually leaves the weights of the initial or lower layers nearly unchanged. As a result, the gradient descent never

Vanishing/Exploding Gradients in Deep Neural Networks VII

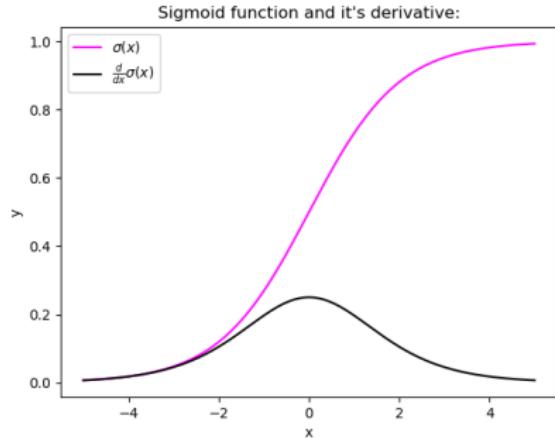
converges to the optimum. This is known as the **vanishing gradients problem**.

- ② Exploding – On the contrary, in some cases, the gradients keep on getting larger and larger as the backpropagation algorithm progresses. This, in turn, causes very large weight updates and causes the gradient descent to diverge. This is known as the **exploding gradients problem**.

Why do the gradients even vanish/explode?

- ✓ Certain activation functions, like the logistic function (sigmoid), have a very huge difference between the variance of their inputs and the outputs.
- ✓ In simpler words, they shrink and transform a larger input space into a smaller output space that lies between the range of [0,1].

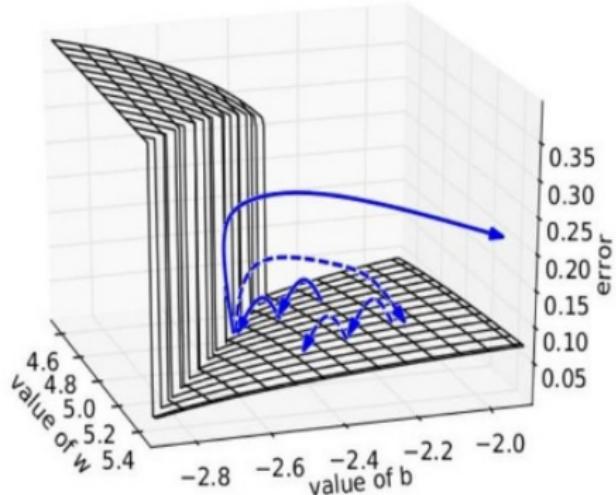
Vanishing/Exploding Gradients in Deep Neural Networks IX



Vanishing/Exploding Gradients in Deep Neural Networks X

✓ Observing the above graph of the Sigmoid function, it can be seen that for larger inputs (negative or positive), it saturates at 0 or 1 with a derivative very close to zero. Thus, when the backpropagation algorithm chips in, it virtually has no gradients to propagate backward in the network, and whatever little residual gradients exist keeps on diluting as the algorithm progresses down through the top layers. So, this leaves nothing for the lower layers.

Vanishing/Exploding Gradients in Deep Neural Networks XI



Vanishing/Exploding Gradients in Deep Neural Networks XII

✓ Similarly, in some cases suppose the initial weights assigned to the network generate some large loss. Now the gradients can accumulate during an update and result in very large gradients which eventually results in large updates to the network weights and leads to an unstable network. The parameters can sometimes become so large that they overflow and result in NaN values.

How to know if our model is suffering from the Exploding/Vanishing gradient problem?

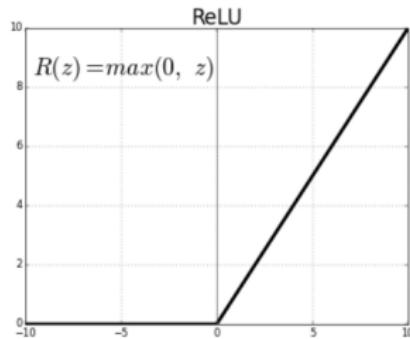
Vanishing/Exploding Gradients in Deep Neural Networks XIII

Exploding	Vanishing
<p>There is an exponential growth in the model parameters.</p>	<p>The parameters of the higher layers change significantly whereas the parameters of lower layers would not change much (or not at all).</p>
<p>The model weights may become NaN during training.</p>	<p>The model weights may become 0 during training.</p>
<p>The model experiences avalanche learning.</p>	<p>The model learns very slowly and perhaps the training stagnates at a very early stage just after a few iterations.</p>

Methods proposed to overcome vanishing gradient problem

- ① Proper Weight Initialization
- ② Long short – term memory:
- ③ Faster hardware
- ④ Residual neural networks (ResNets)
- ⑤ ReLU (Rectified Linear Unit):

Vanishing/Exploding Gradients in Deep Neural Networks XV

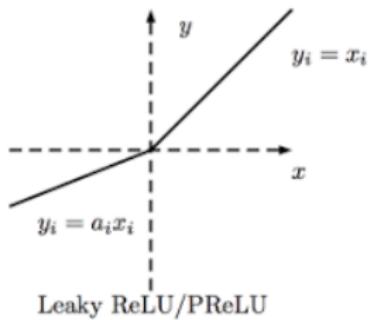


- $Relu(z) = \max(0, z)$
- Outputs 0 for any negative input.
- Range: $[0, \infty]$

Vanishing/Exploding Gradients in Deep Neural Networks XVI

✓ Unfortunately, the ReLU function is also not a perfect pick for the intermediate layers of the network “in some cases”. It suffers from a problem known as dying ReLus wherein some neurons just die out, meaning they keep on throwing 0 as outputs with the advancement in training.

⑥ LReLU (Leaky ReLU):

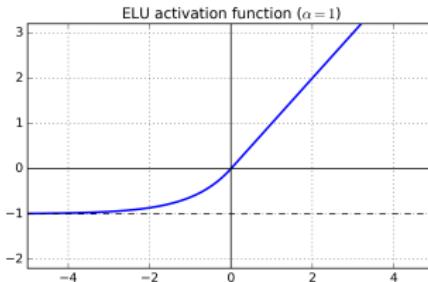


Vanishing/Exploding Gradients in Deep Neural Networks XVII

- LeakyReLU $\alpha(z) = \max(\alpha z, z)$
- The amount of "leak" is controlled by the hyperparameter α , it is the slope of the function for $z < 0$.
- The smaller slope for the leak ensures that the neurons powered by leaky ReLU never die; although they might venture into a state of coma for a long training phase they always have a chance to eventually wake up.
- α can also be trained, that is, the model learns the value of α during training. This variant wherein α is now considered a parameter rather than a hyperparameter is called parametric leaky ReLU (PReLU).

7 ELU (Exponential Linear Unit):

Vanishing/Exploding Gradients in Deep Neural Networks XVIII



- For $z < 0$, it takes on negative values which allow the unit to have an average output closer to 0 thus alleviating the vanishing gradient problem
- For $z < 0$, the gradients are non zero. This avoids the dead neurons problem.
- For $\alpha = 1$, the function is smooth everywhere, this speeds up the gradient descent since it does not bounce right and left around $z = 0$.

Vanishing/Exploding Gradients in Deep Neural Networks XIX

- A scaled version of this function (SELU: Scaled ELU) is also used very often in Deep Learning.
- ⑧ Batch Normalization: Batch Normalization to address the problem of vanishing/exploding gradients.
- It consists of adding an operation in the model just before or after the activation function of each hidden layer.
 - This operation simply zero-centers and normalizes each input, then scales and shifts the result using two new parameter vectors per layer: one for scaling, the other for shifting.
 - In other words, the operation lets the model learn the optimal scale and mean of each of the layer's inputs.

Vanishing/Exploding Gradients in Deep Neural Networks XX

- To zero-center and normalize the inputs, the algorithm needs to estimate each input's mean and standard deviation.
 - It does so by evaluating the mean and standard deviation of the input over the current mini-batch (hence the name “Batch Normalization”).
- ⑨ Gradient Clipping : Another popular technique to mitigate the exploding gradients problem is to clip the gradients during backpropagation so that they never exceed some threshold. This is called Gradient Clipping.
- This optimizer will clip every component of the gradient vector to a value between -1.0 and 1.0 .

Vanishing/Exploding Gradients in Deep Neural Networks XXI

- Meaning, all the partial derivatives of the loss w.r.t each trainable parameter will be clipped between -1.0 and 1.0

CS322: Deep Learning

Convolutional Neural Networks

Sachchida Nand Chaurasia
Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



December 23, 2021

Introduction I

Computer Vision Problems

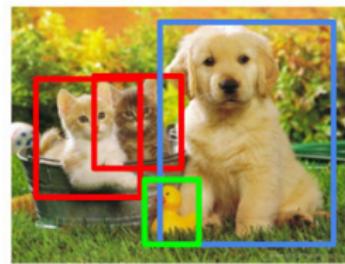
1. Image classifications
2. Object detection
3. Neural Style Transfer

Classification



CAT

Object Detection

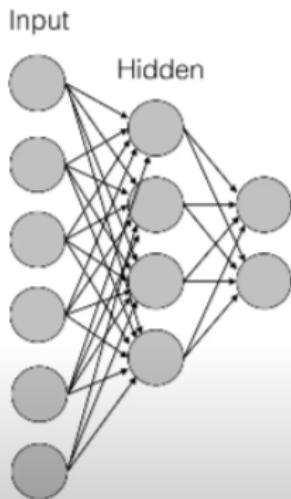


CAT, DOG, DUCK

Introduction II



Introduction III

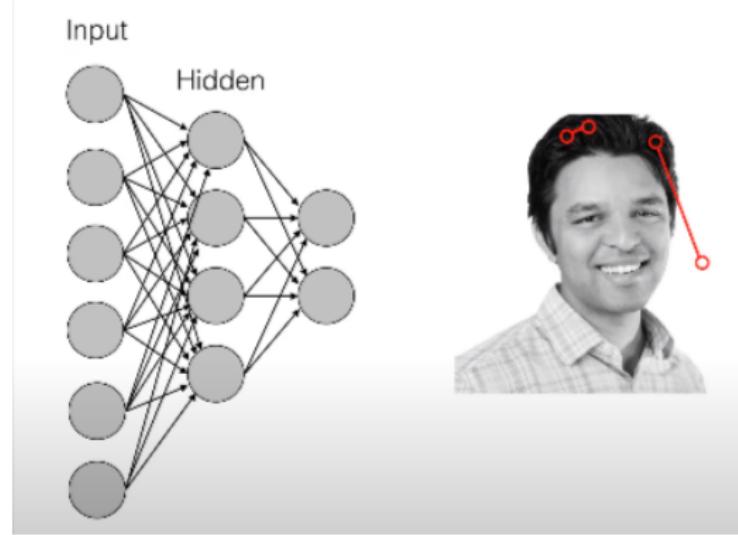


MNIST dataset: 28 x 28 pixels (784 pixels)
First layer weights: ~78k parameters

Typical Image: 256 x 256 (56,000 pixels)
First layer weights: 560k parameters !

Too many parameters, unscalable to real images

Introduction IV



- ✓ These kind of networks do not take into account or understanding of relationship between space and pixels in the images.

Introduction V

Background of CNNs: CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc.

- ✓ The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources.
- ✓ This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.
- ✓ In 2012 Alex Krizhevsky uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets

Introduction VI

with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

What exactly is a CNN?

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery.

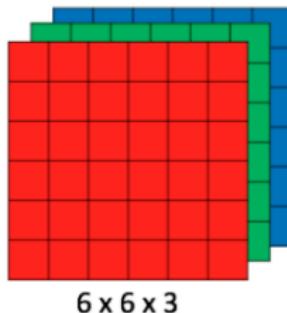
✓ When we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called *Convolution*.

Bottom line is that the role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

Introduction VII

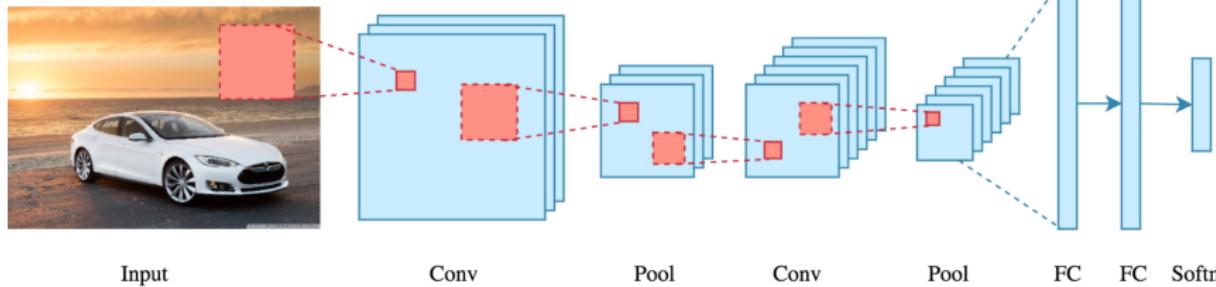
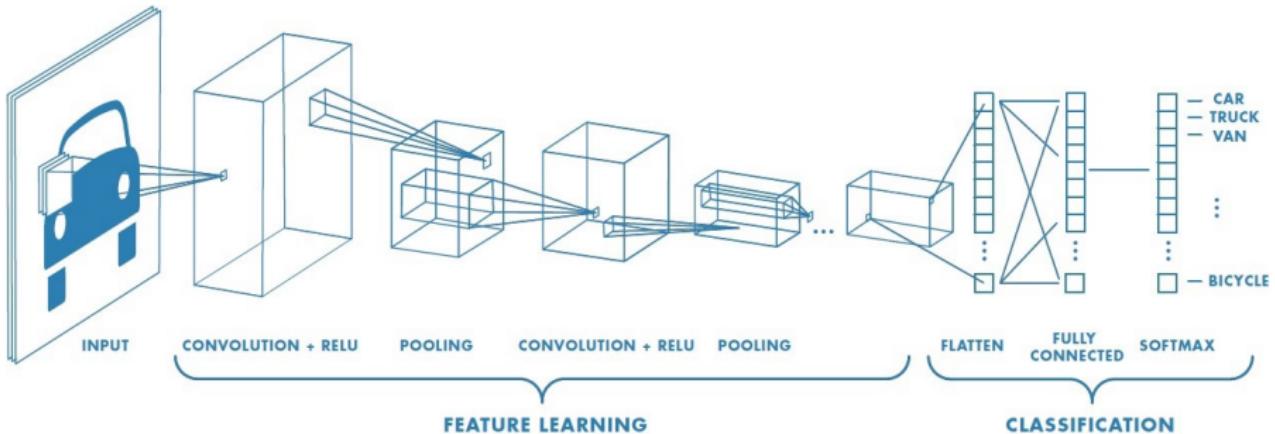
- ✓ In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications.
- ✓ Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.
- ✓ CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion).
- ✓ Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see $h \times w \times d$ (h = Height, w = Width, d = Dimension).
- ✓ An image of $6 \times 6 \times 3$ array of matrix of RGB (3 refers to RGB values) and an image of $4 \times 4 \times 1$ array of matrix of grayscale image.

Introduction VIII

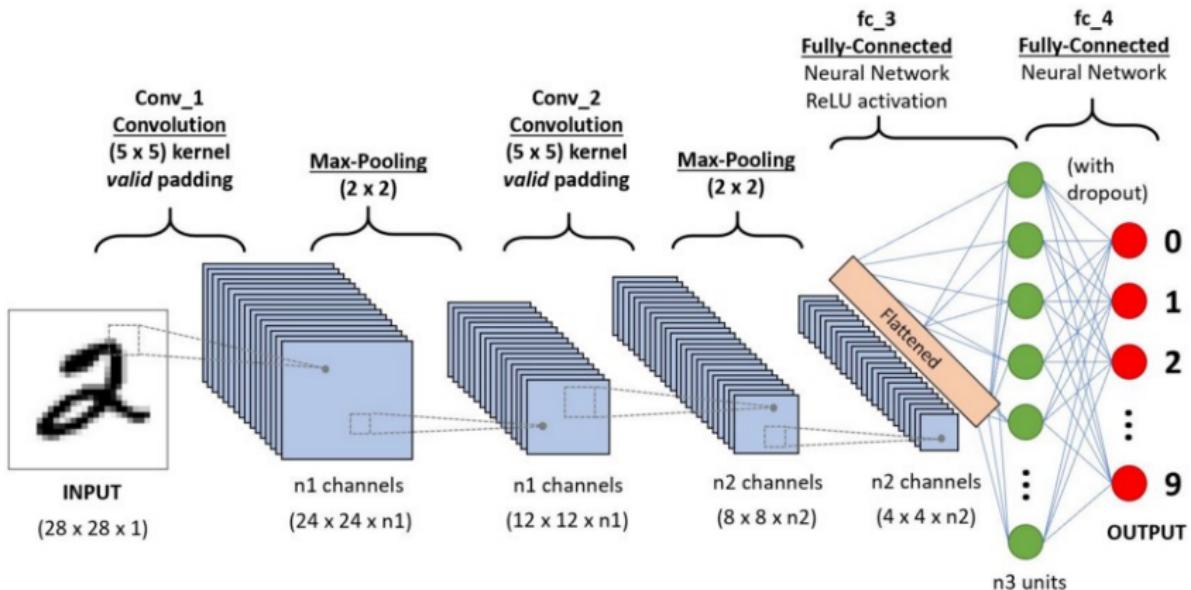


- ✓ Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

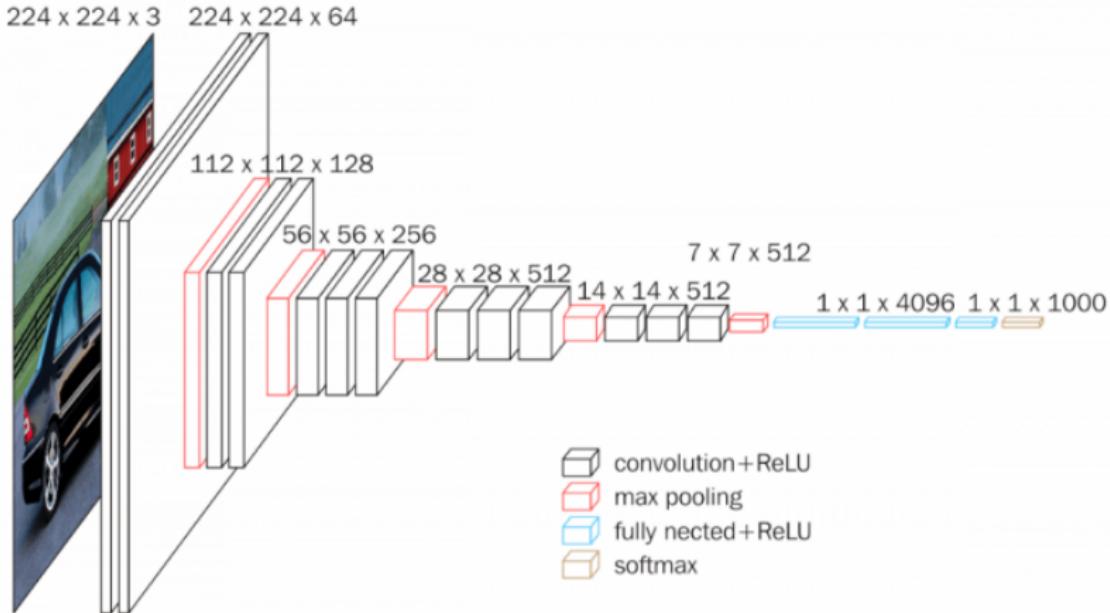
Introduction IX



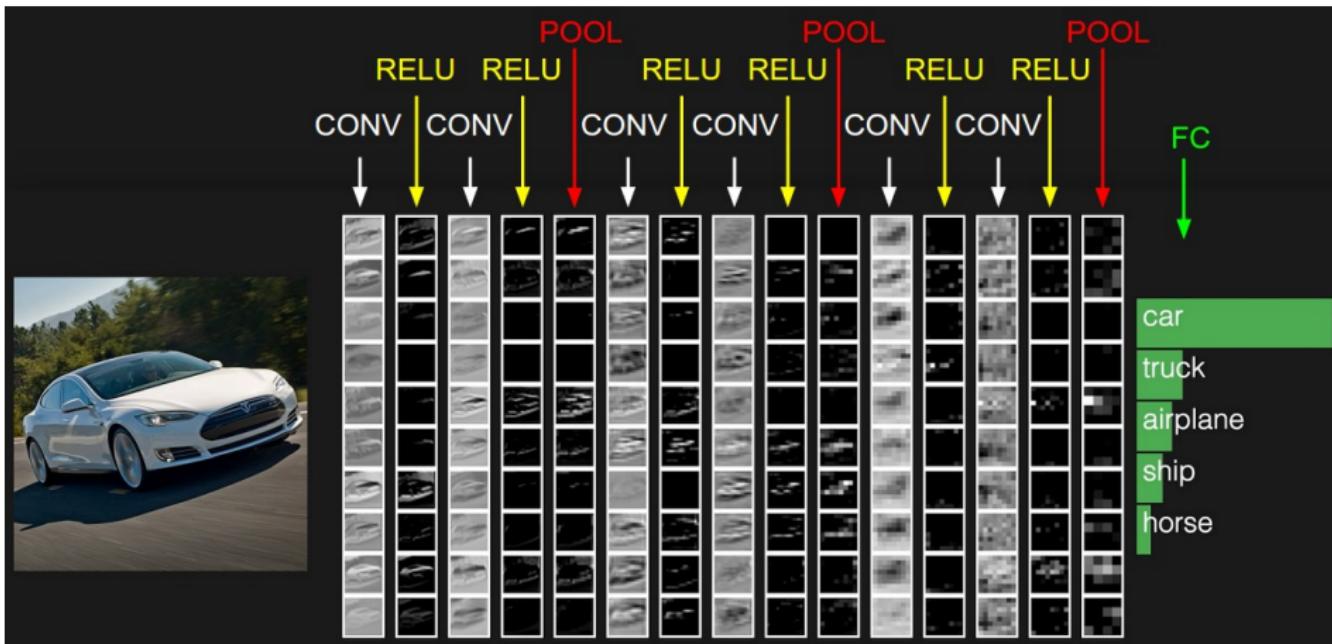
Introduction X



Introduction XI



Introduction XII



Introduction XIII

What is Convolution?

Convolution is a mathematical operation on two functions (f and g) that produces a third function ($f * g$) that expresses how the shape of one is modified by the other.

- ✓ The term convolution refers to both the result function and to the process of computing it.
- ✓ The fundamental difference between a densely connected layer and a convolution layer is this: Dense layers learn global patterns in their input feature space (for example, for a MNIST digit, patterns involving all pixels), whereas convolution layers learn local patterns: in the case of images, patterns found in small 2D windows of the inputs.

Introduction XIV

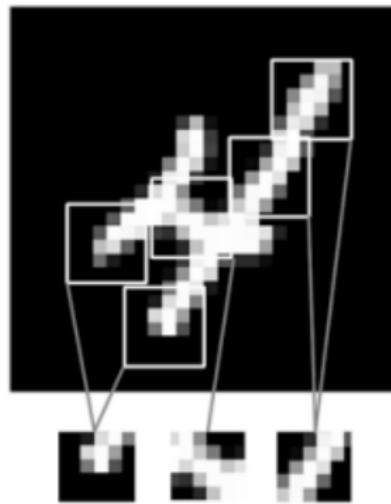


Figure 1: Images can be broken into local patterns such as edges, textures, and so on.

Introduction XV

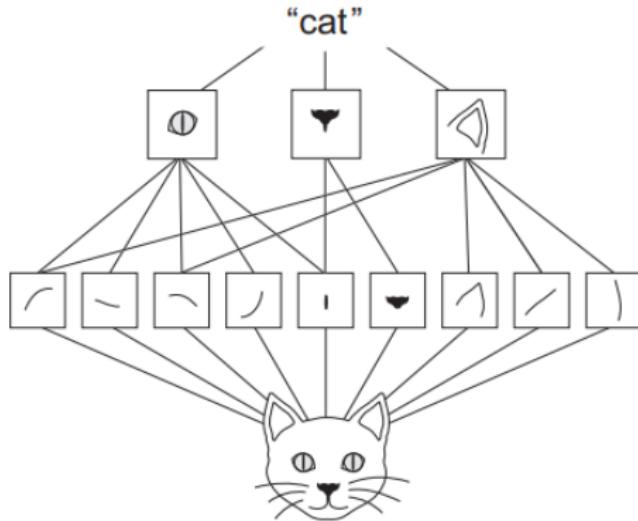


Figure 2: The visual world forms a spatial hierarchy of visual modules: hyperlocal edges combine into local objects such as eyes or ears, which combine into high-level concepts such as “cat.”

Introduction XVI

- ✓ The convolution of f and g is written $f * g$, denoting the operator with the symbol $*$. It is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of integral transform:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau \quad (0.1)$$

An equivalent definition is :

$$(f * g)(t) := \int_{-\infty}^{\infty} f(t - \tau)g(\tau) d\tau \quad (0.2)$$

Introduction XVII

A common engineering notational convention is:

$$f(t) * g(t) := \underbrace{\int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau}_{(f*g)(t)} \quad (0.3)$$

- ① Express each function in terms of a dummy variable τ .
- ② Reflect one of the functions: $g(\tau) \rightarrow g(-\tau)$.
- ③ Add a time-offset, t , which allows $g(t - \tau)$ to slide along the τ -axis.

Introduction XVIII

- ④ Start t at $-\infty$ and slide it all the way to $+\infty$. Wherever the two functions intersect, find the integral of their product. In other words, at time t , compute the area under the function $f(\tau)$ weighted by the weighting function $g(t - \tau)$.

Introduction XIX

Example:

Convolution of two functions.

Example

Find the convolution of $f(t) = e^{-t}$ and $g(t) = \sin(t)$.

Solution: By definition: $(f * g)(t) = \int_0^t e^{-\tau} \sin(t - \tau) d\tau$.

Integrate by parts twice: $\int_0^t e^{-\tau} \sin(t - \tau) d\tau =$

$$\left[e^{-\tau} \cos(t - \tau) \right] \Big|_0^t - \left[e^{-\tau} \sin(t - \tau) \right] \Big|_0^t - \int_0^t e^{-\tau} \sin(t - \tau) d\tau,$$

$$2 \int_0^t e^{-\tau} \sin(t - \tau) d\tau = \left[e^{-\tau} \cos(t - \tau) \right] \Big|_0^t - \left[e^{-\tau} \sin(t - \tau) \right] \Big|_0^t,$$

$$2(f * g)(t) = e^{-t} - \cos(t) - 0 + \sin(t).$$

We conclude: $(f * g)(t) = \frac{1}{2} [e^{-t} + \sin(t) - \cos(t)]$. □

Introduction XX

Properties of convolutions

For every piecewise continuous functions f , g , and h , hold:

- ① Commutativity: $f * g = g * f$;
- ② Associativity: $f * (g * h) = (f * g) * h$;
- ③ Distributivity: $f * (g + h) = f * g + f * h$;
- ④ Neutral element: $f * 0 = 0$;
- ⑤ Identity element: $f * \delta = f$;

Introduction XXI

Applications of convolution:



Introduction XXII



Introduction XXIII

- In image processing:
 - ✓ In digital image processing convolutional filtering plays an important role in many important algorithms in edge detection and related processes.
 - ✓ In optics, an out-of-focus photograph is a convolution of the sharp image with a lens function.
 - ✓ In image processing applications such as adding blurring.
- In digital data processing:
 - ✓ In analytical chemistry, Savitzky–Golay smoothing filters are used for the analysis of spectroscopic data. They can improve signal-to-noise ratio with minimal distortion of the spectra.
 - ✓ In statistics, a weighted moving average is a convolution.

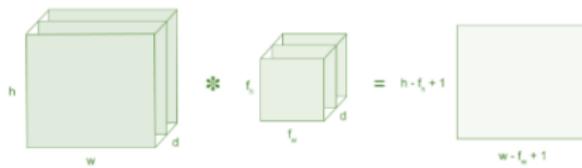
Introduction XXIV

Convolution Layer: Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data.

- ✓ It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.
- ✓ Initially use fewer filters and gradually increase and monitor the error rate to see how it is varying.
- ✓ Very small filter sizes will capture very fine details of the image. On the other hand having a bigger filter size will leave out minute details in the image.

Introduction XXV

- An image matrix (volume) of dimension $(h \times w \times d)$
 - A filter $(f_h \times f_w \times d)$
 - Outputs a volume dimension $(h - f_h + 1) \times (w - f_w + 1) \times 1$
-
- An image matrix (volume) of dimension **$(h \times w \times d)$**
 - A filter **$(f_h \times f_w \times d)$**
 - Outputs a volume dimension **$(h - f_h + 1) \times (w - f_w + 1) \times 1$**



✓ Consider a 5×5 whose image pixel values are 0, 1 and filter matrix 3×3 :

Introduction XXVI

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

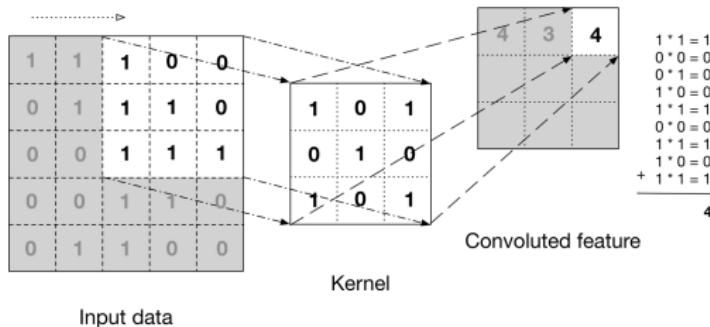


1	0	1
0	1	0
1	0	1

5 × 5 – Image Matrix

3 × 3 – Filter Matrix

- ✓ Then the convolution of 5×5 image matrix multiplies with 3×3 filter matrix which is called “Feature Map” as output shown in below:



Introduction XXVII

✓ Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels).

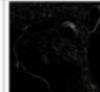
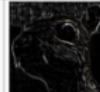
Feature detection

- Edge detection: Canny, Deriche, Differential, Sobel, Prewitt, Roberts cross
- Corner detection: Harris operator, Shi and Tomasi, Level curve curvature, Hessian feature strength measures, SUSANFAST
- Blob detection: Laplacian of Gaussian (LoG), Difference of Gaussians (DoG), Determinant of Hessian (DoH), Maximally stable extremal regions, PCBR

Introduction XXVIII

- Ridge detection: Hough transform, Hough transform, Generalized Hough transform
- Structure tensor: Structure tensor, Generalized structure tensor
- Affine invariant feature detection: Affine shape adaptation, Harris affine, Hessian affine
- Feature description: SIFT, SURF, GLOH, HOG
- Scale space: Scale-space axioms, Implementation details, Pyramids

Introduction XXIX

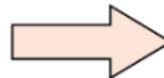
	Operation	Filter	Convolved Image
Identity		$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection		$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
		$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
		$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
	Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)		$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)		$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Introduction XXX

Strides: Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

1	2	3	4	5	6	7
11	12	13	14	15	16	17
21	22	23	24	25	26	27
31	32	33	34	35	36	37
41	42	43	44	45	46	47
51	52	53	54	55	56	57
61	62	63	64	65	66	67
71	72	73	74	75	76	77

Convolve with 3x3
filters filled with ones



108	126	
288	306	

Introduction XXXI

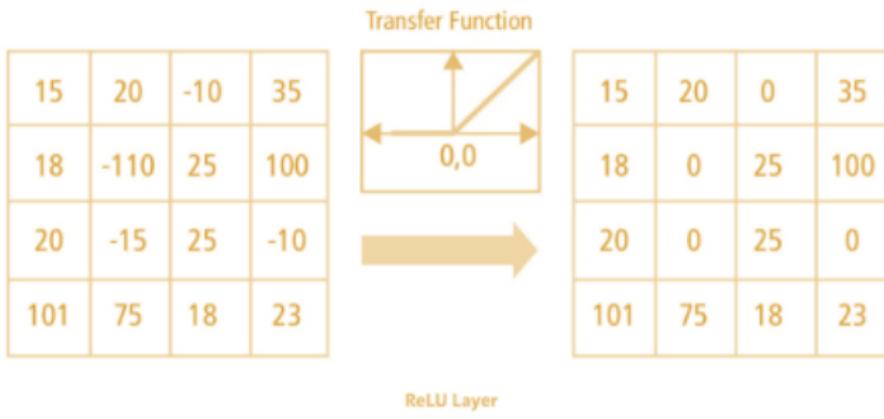
Padding: Sometimes filter does not perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Introduction XXXII

Non Linearity (ReLU): ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $(x) = \max(0, x)$.

Why ReLU is important : ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.



Introduction XXXIII

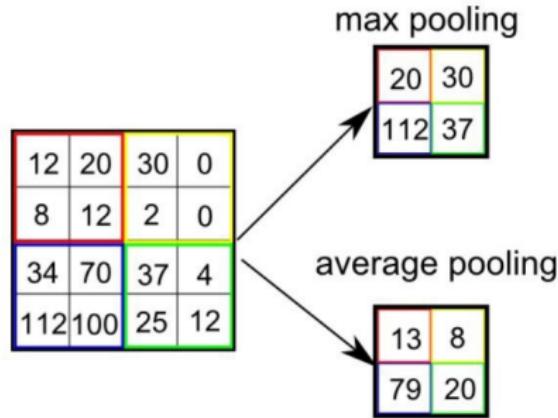
Pooling Layer: Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains important information.

- ✓ This is to decrease the computational power required to process the data by reducing the dimensions.
- ✓ Spatial pooling can be of different types:
 - Max Pooling
 - Average Pooling
 - Sum Pooling

Introduction XXXIV

- ✓ Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.
- ✓ Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism.
- ✓ We can say that Max Pooling performs a lot better than Average Pooling.

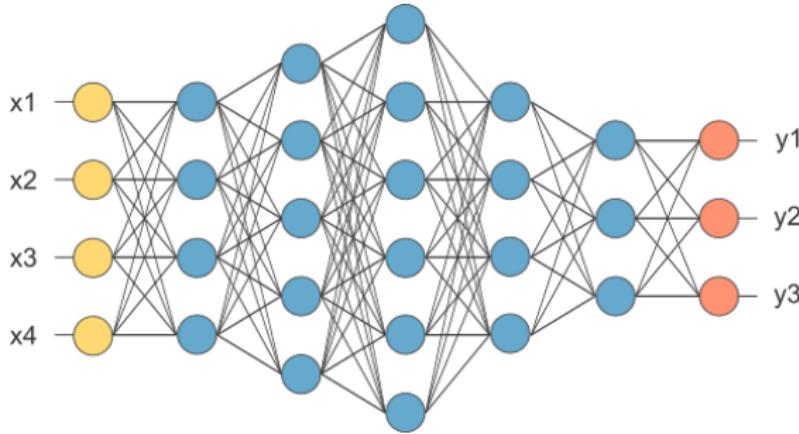
Introduction XXXV



Introduction XXXVI

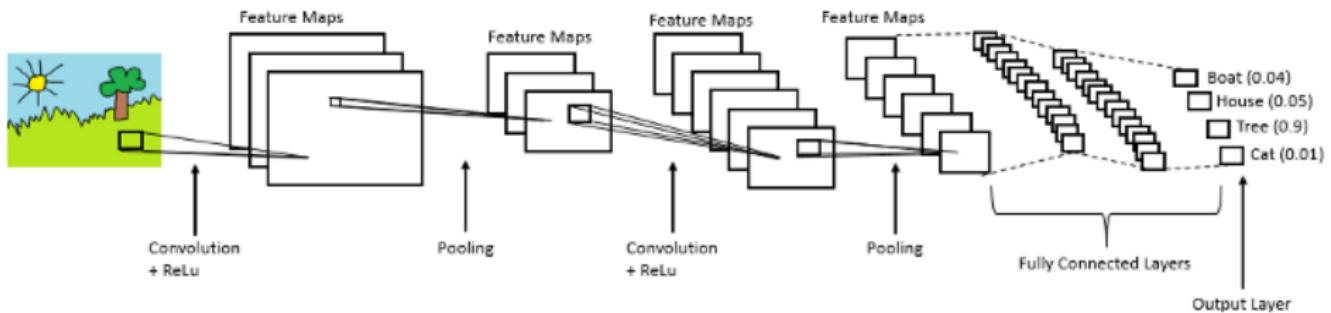
Fully Connected Layer

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.



Introduction XXXVII

- ✓ In the above diagram, the feature map matrix will be converted as vector (x_1, x_2, x_3, \dots). With the fully connected layers, we combined these features together to create a model.
- ✓ Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.



Introduction XXXVIII

Cross-correlation:

Given an input image I and a filter (kernel) K of dimensions $k_1 \times k_2$, the cross-correlation operation is given by:

$$(I \otimes K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n)K(m, n) \quad (1)$$

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i-m, j-n)K(m, n) \quad (0.4)$$

$$= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n)K(-m, -n) \quad (0.5)$$

Introduction XXXIX

From the equation it is easy to see that convolution is the same as cross-correlation with a flipped kernel i.e: for a kernel K where $K(-m, -n) == K(m, n)$.

- ✓ CNNs consists of convolutional layers which are characterized by an input map I , a bank of filters K and biases b .
- ✓ In the case of images, we could have as input an image with height H , width W and $C = 3$ channels (red, blue and green) such that $I \in \mathbb{R}^{H \times W \times C}$. Subsequently for a bank of D filters we have $K \in \mathbb{R}^{k_1 \times k_2 \times C \times D}$ and biases $b \in \mathbb{R}^D$, one for each filter.

The output from this convolution procedure is:

Introduction XL

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^C K_{m,n,c} \cdot I_{i+m,j+n,c} + b \quad (0.6)$$

For a grayscale image:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} K_{m,n} \cdot I_{i+m,j+n} + b \quad (0.7)$$

Notation

- ① l is the l^{th} layer where $l=1$ is the first layer and $l = L$ is the last layer.

Introduction XLI

- ② Input x is of dimension $H \times W$ and has i by j as the iterators.
- ③ Filter or kernel w is of dimension $k_1 \times k_2$ has m by n as the iterators.
- ④ $w_{m,n}^l$ is the weight matrix connecting neurons of layer l with neurons of layer $l - 1$.
- ⑤ b^l is the bias unit at layer l .
- ⑥ $x_{i,j}^l$ is the convolved input vector at layer l plus the bias represented as

$$x_{i,j}^l = \sum_m \sum_n w_{m,n}^l o_{i+m, j+n}^{l-1} + b^l \quad (0.8)$$

Introduction XLII

- ⑦ $o_{i,j}^l$ is the output at layer l given by

$$o_{i,j}^l = f(x_{i,j}^l) \quad (0.9)$$

- ⑧ $f(\cdot)$ is the activation function. Application of the activation layer to the convolved input vector at layer l is $f(x_{i,j}^l)$

Forward Propagation

The convolution equation of the input at layer 1 is given by:

$$x_{i,j}^l = \text{rot}_{180^\circ} \{w_{m,n}^l\} * o_{i,j}^{l-1} + b_{i,j}^l \quad (0.10)$$

$$x_{i,j}^l = \sum_m \sum_n w_{m,n}^l o_{i+m,j+n}^{l-1} + b_{i,j}^l \quad (0.11)$$

$$o_{i,j}^l = f(x_{i,j}^l) \quad (0.12)$$

Introduction XLIII

Error

For a total of P predictions, the predicted network outputs y_p and their corresponding targeted values t_p the the mean squared error is given by:

$$E = \frac{1}{2} \sum_p (t_p - y_p)^2 \quad (0.13)$$

- ✓ Learning will be achieved by adjusting the weights such that y_p is as close as possible or equals to corresponding t_p .
- ✓ In the classical backpropagation algorithm, the weights are changed according to the gradient descent direction of an error surface E .

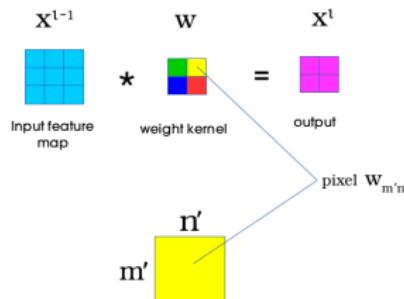
Backpropagation

Introduction XLIV

For backpropagation there are two updates performed, for the weights and the deltas. Lets begin with the weight update.

Gradient w.r.t weights is the measurement of how the change in a single pixel $w_{m',n'}$ in the weight kernel affects the loss function E .

$$\frac{\partial E}{\partial w_{m',n'}^l}$$



Introduction XLV

During forward propagation, the convolution operation ensures that the yellow pixel $w_{m',n'}$ in the weight kernel makes a contribution in all the products (between each element of the weight kernel and the input feature map element it overlaps).

This means that pixel $w_{m',n'}$ will eventually affect all the elements in the output feature map.

Convolution between the input feature map of dimension $H \times W$ and the weight kernel of dimension $k_1 \times k_2$ produces an output feature map of size $(H - k_1 + 1)$ by $(W - k_2 + 1)$. The gradient component for the individual weights can be obtained by applying the chain rule in the following way:

Introduction XLVI

$$\frac{\partial E}{\partial w_{m',n'}^l} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \frac{\partial E}{\partial x_{i,j}^l} \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} \quad (0.14)$$

$$= \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta_{i,j}^l \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} \quad (0.15)$$

In the above equation $x_{i,j}^l$ is equivalent to

$$\{w_{m,n}^l\} o_{i+m,j+n}^{l-1} + b^l \quad (0.16)$$

Introduction XLVII

$$\frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} = \frac{\partial}{\partial w_{m',n'}^l} \left(\sum_m \sum_n w_{m,n}^l o_{i+m,j+n}^{l-1} + b^l \right) \quad (0.17)$$

Further expanding the summations in the above equation and taking the partial derivatives for all the components results in zero values for all except the components where $m = m'$ and $n = n'$ in $w_{m,n}^l o_{i+m,j+n}^{l-1}$ as follows:

Introduction XLVIII

$$\frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} = \frac{\partial}{\partial w_{m',n'}^l} (w_{0,0}^l o_{i+0,j+0}^{l-1} + \cdots + w_{m',n'}^l o_{i+m',j+n'}^{l-1} + \cdots + b^l) \quad (0.18)$$

$$= \frac{\partial}{\partial w_{m',n'}^l} (w_{m',n'}^l o_{i+m',j+n'}^{l-1}) \quad (0.19)$$

$$= o_{i+m',j+n'}^{l-1} \quad (0.20)$$

Introduction XLIX

Substituting Equation (0.18) in Equation (0.14) gives us the following results:

$$\frac{\partial E}{\partial w_{m',n'}^l} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta_{i,j}^l o_{i+m',j+n'}^{l-1} \quad (13)$$

$$= \text{rot}_{180^\circ} \{ \delta_{i,j}^l \} * o_{m',n'}^{l-1} \quad (14)$$

Introduction L

Summary

- Provide input image into convolution layer.
- Choose parameters, apply filters with strides, padding if required.
Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size.
- Add as many convolution layers until satisfied.
- Flatten the output and feed into a fully connected layer (FC layer).
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.