

# Machine Learning: Overview

## CS115 - Math for Computer Science

Ngoc Hoang Luong

University of Information Technology (UIT)

*hoangln@uit.edu.vn*

September 8, 2022

# What is machine learning?

*A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$ , and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .*

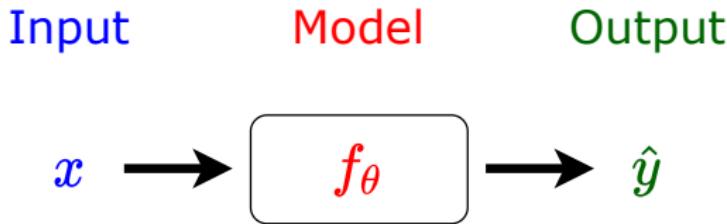
— Tom Mitchell

There are many different kinds of ML, depending on

- the task  $T$  we wish the system to learn
- the performance measure  $P$  we use to evaluate the system
- the training signal or experience  $E$  we give it.

# What is machine learning?

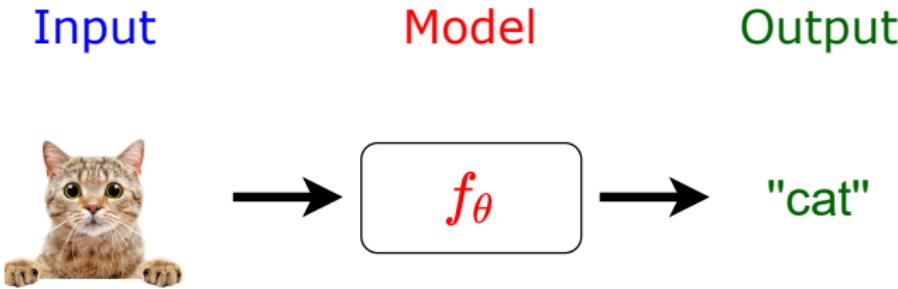
- **Supervised learning.** We get observed inputs  $\mathcal{D} = \{\mathbf{x}_n : n = 1 : N\}$  and corresponding outputs  $y_n$ . We learn a mapping  $f$  from inputs to outputs.



- **Unsupervised learning.** We just get observed inputs  $\mathcal{D}$  without any corresponding outputs. We try to make sense of data.
- **Reinforcement learning.** An **agent** has to learn how to interact with its environment. This can be encoded by a **policy**  $a = \pi(s)$ , which specifies which action  $a$  to perform in response to each possible input  $s$  (environmental state).

# Supervised Learning - Classification

Learning a mapping  $f_{\theta} : \mathbb{R}^{W \times H} \rightarrow \{"\text{cat}", "\text{non-cat}"\}$ .



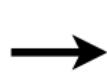
# Supervised Learning - Regression

Learning a mapping  $f_{\theta} : \mathbb{R}^N \rightarrow \mathbb{R}$ .

Input

Model

Output



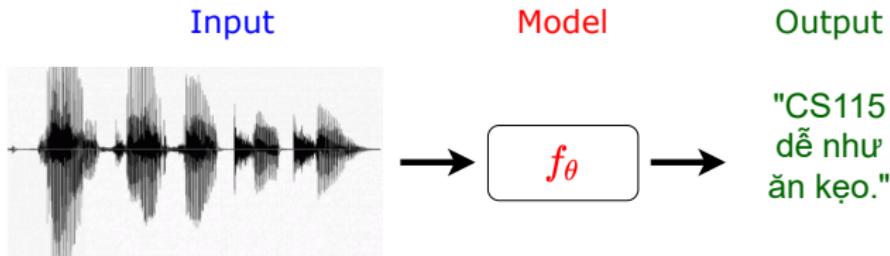
$$f_{\theta}$$



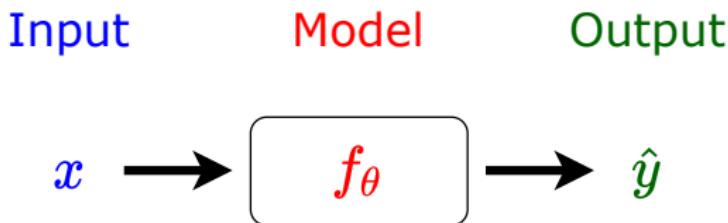
\$50,312

# Supervised Learning - Structured Prediction

Learning a mapping  $f_{\theta} : \mathbb{R}^N \rightarrow \{1, 2, \dots, L\}^M$ .



# Supervised learning



The most common form of ML is **supervised learning**.

- The task  $T$  is to learn a mapping  $f$  from inputs  $x \in \mathcal{X}$  to outputs  $y \in \mathcal{Y}$ : Estimate **parameters**  $\theta$  from training data  $\{(x_n, y_n)\}_{n=1}^N$ .
- The inputs  $x$  are also called the **features**, or **predictors**.
- The output  $y$  is also known as the **label**, **target**, or **response**.
- The experience  $E$  is a set of  $N$  input-output pairs  $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ , known as the **training set**.  $N$  is called the **sample size**.
- The performance measure  $P$  depends on the type of output we are predicting.

# Classification

In **classification** problems, the output space is

- a set of  $C$  unordered and mutually exclusive labels known as **classes**  
 $\mathcal{Y} = \{1, 2, \dots, C\}$ .
- If there are just two classes,  $y \in \{0, 1\}$  or  $y \in \{+1, -1\}$ , it is called **binary classification**.



(a)



(b)



(c)

**Figure:** Classifying Iris flowers into 3 subspecies: setosa, versicolor and virginica.

# Classification

## In image classification,

- The  $\mathcal{X}$  is the set of images, which is very high-dimensional.
- For a color image with  $C = 3$  channels (RGB) and  $D_1 \times D_2$  pixels, we have  $\mathcal{X} = \mathbb{R}^D$ , where  $D = C \times D_1 \times D_2$ .
- Learning a mapping  $f : \mathcal{X} \rightarrow \mathcal{Y}$  from images to labels is challenging.

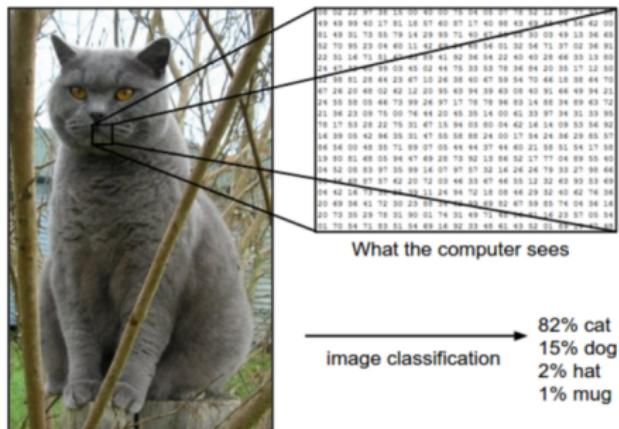


Figure: Illustration of the image classification problem.

# Classification

## Machine Learning:

Sample  
↓  
Label



dog



cat



horse

## Human Learning:

We learn through



Examples

Long Ear Black nose  
↓  
dog



Diagrams

Comparisons

Figure: Machine learning vs. Human learning<sup>1</sup>

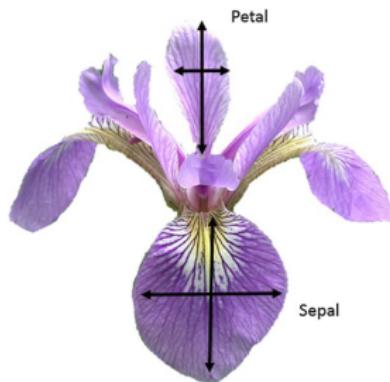
---

<sup>1</sup><https://blog.ml.cmu.edu/2019/03/29/building-machine-learning-models-via-comparisons/>

# Classification

The **Iris dataset** is a collection of 150 labeled examples of Iris flowers, 50 of each type, described by 4 features

- sepal length, sepal width, petal length, petal width.
- The input space is much lower-dimensional  $\mathcal{X} = \mathbb{R}^4$ .
- These numeric features are highly informative.

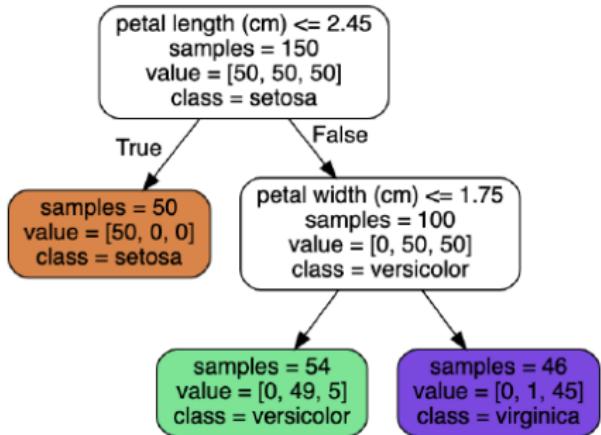


index	sl	sw	pl	pw	label
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
	...				
50	7.0	3.2	4.7	1.4	Versicolor
	...				
149	5.9	3.0	5.1	1.8	Virginica

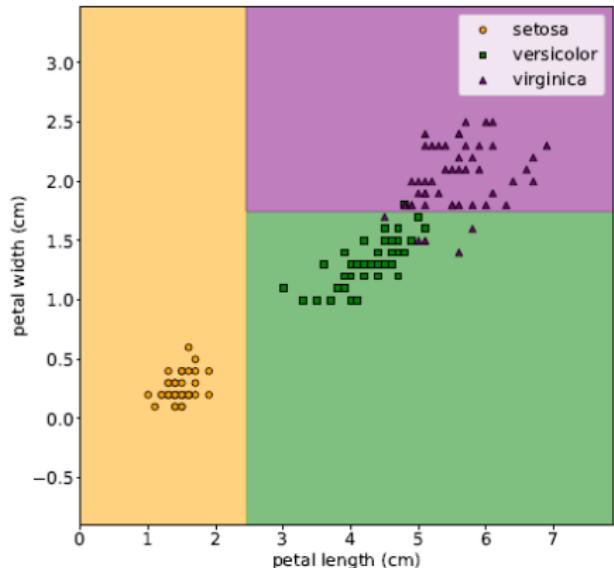
Figure: A subset of the Iris design matrix.

- A small dataset of features can be stored as an  $N \times D$  **design matrix**.

# Learning a classifier



(a)



(b)

Figure: A decision tree on Iris dataset, using just petal length and width features.

# Learning a classifier

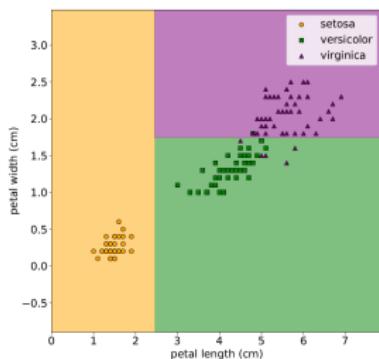


Figure: Decision surface

- We apply a **decision rule** to the Iris dataset

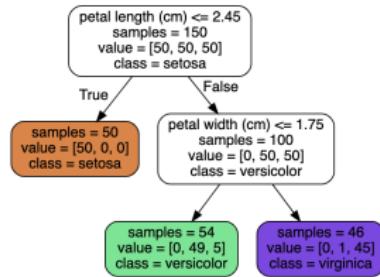
$$f_1(\mathbf{x}; \theta) = \begin{cases} \text{Setosa if petal length} < 2.45 \\ \text{Versicolor or Virginica otherwise} \end{cases}$$

- This rule defines a 1d **decision boundary** at  $x_{\text{petal length}} = 2.45$ , partitioning the input space into 2 regions.
- We add another **decision rule** to inputs that fail the first rule.

$$f_2(\mathbf{x}; \theta) = \begin{cases} \text{Versicolor if petal width} < 1.75 \\ \text{Virginica otherwise} \end{cases}$$

- Both rules induces a 2d **decision surface**.

# Learning a classifier



- We can arrange the rules  $f_1, f_2$  into a tree structure, called a **decision tree**.
- We can represent the tree by storing, for each internal node:
  - the feature index that is used.
  - the corresponding threshold value.
- We denote all these **parameters** by  $\theta$ .

Figure: Decision tree

# Empirical risk minimization

The goal of supervised learning is

- to automatically come up with classification models.
- to reliably predict the labels for any given input.

The **misclassification rate** on the training set can be used to measure performance on this task:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(y_n \neq f(\mathbf{x}_n; \boldsymbol{\theta}))$$

where  $\mathbb{I}(e)$  is the binary **indicator function**:

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

This assumes all errors are equal.

# Empirical risk minimization

- It may happen that some errors are more costly than others.
- Suppose that setosa & Versicolor are tasty but Virginica is poisonous.
- We might use the asymmetric **loss function**  $\ell(y, \hat{y})$  as below.

		Estimate		
		Setosa	Versicolor	Virginica
Truth	Setosa	0	1	1
	Versicolor	1	0	1
	Virginica	10	10	0

The **empirical risk** is the average loss of the predictor on the training set:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n; \boldsymbol{\theta}))$$

The misclassification rate is equal to the empirical risk when we use **zero-one loss**:

$$\ell_{01}(y, \hat{y}) = \mathbb{I}(y \neq \hat{y})$$

# Empirical risk minimization

The problem of **model fitting** or **training** can be defined as finding a setting of parameters that minimizes the empirical risk on the training set:

$$\hat{\theta} = \operatorname{argmin}_{\theta} \mathcal{L}(\theta) = \operatorname{argmin}_{\theta} \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n; \theta))$$

This is called **empirical risk minimization**.

However, our true goal is:

- to minimize the expected loss on **future data** that we have not seen,
- to **generalize**, rather than just do well on the training set.

# Uncertainty

We can't perfectly predict the exact output given the input, due to

- lack of knowledge of the input-output mapping (**model uncertainty**)
- and/or intrinsic (irreducible) stochasticity in the mapping (**data uncertainty**).

We capture our uncertainty using **conditional probability distribution**:

$$p(y = c | \mathbf{x}; \boldsymbol{\theta}) = f_c(\mathbf{x}; \boldsymbol{\theta})$$

where  $f : \mathcal{X} \rightarrow [0, 1]^C$  maps inputs to a probability distribution over the  $C$  possible output labels.

# Uncertainty

Since  $f_c(\mathbf{x}; \boldsymbol{\theta})$  returns the probability of class label  $c$ , we require

$$0 \leq f_c \leq 1 \text{ for each } c, \text{ and } \sum_{c=1}^C f_c = 1$$

To avoid this restriction, it is common to require the model to return unnormalized log-probabilities. We can then use the **softmax function** to convert these to probabilities

$$\mathcal{S}(\mathbf{a}) = \left[ \frac{e^{a_1}}{\sum_{c=1}^C e^{a_c}}, \dots, \frac{e^{a_C}}{\sum_{c=1}^C e^{a_c}} \right]$$

$\mathcal{S} : \mathbb{R}^C \rightarrow [0, 1]^C$ , and satisfied  $0 \leq \mathcal{S}(\mathbf{a})_c \leq 1$  and  $\sum_{c=1}^C \mathcal{S}(\mathbf{a})_c = 1$ .

The inputs to the softmax,  $\mathbf{a} = f(\mathbf{x}; \boldsymbol{\theta})$ , are called **logits**.

$$p(y = c | \mathbf{x}; \boldsymbol{\theta}) = \mathcal{S}_c(f(\mathbf{x}; \boldsymbol{\theta}))$$

# Uncertainty

The overall model is defined as

$$p(y = c|x; \theta) = \mathcal{S}_c(f(x; \theta))$$

A special case is when  $f$  is an **affine function**

$$f(x; \theta) = b + \mathbf{w}^T \mathbf{x} = b + w_1 x_1 + w_2 x_2 + \dots + w_D x_D$$

where  $\theta = (b, \mathbf{w})$  are the parameters of the model.  $\mathbf{w}$  are called the **weights** and  $b$  is called the **bias**. This model is called **logistic regression**. We can define  $\tilde{\mathbf{w}} = [b, w_1, \dots, w_D]$  and  $\tilde{\mathbf{x}} = [1, x_1, \dots, x_D]$  so that

$$\tilde{\mathbf{w}}^T \tilde{\mathbf{x}} = b + \mathbf{w}^T \mathbf{x}$$

This converts the affine function into a **linear function**. For simplicity, we can just write the prediction function as

$$f(x; \theta) = \mathbf{w}^T \mathbf{x}$$

# Maximum likelihood estimation

- When fitting probabilistic models, we can use the **negative log probability** as our loss function:

$$\ell(y, f(\mathbf{x}; \boldsymbol{\theta})) = -\log p(y|f(\mathbf{x}; \boldsymbol{\theta}))$$

- A good model is one that assigns a high probability to the true output  $y$  for each corresponding input  $\mathbf{x}$
- The average negative log probability of the training set is given by

$$\text{NLL}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^N \log p(y_n|f(\mathbf{x}_n; \boldsymbol{\theta}))$$

- This is called the **negative log likelihood**.
- If we minimize this, we can compute the **maximum likelihood estimate** (MLE):

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \operatorname{argmin}_{\boldsymbol{\theta}} \text{NLL}(\boldsymbol{\theta})$$

# Regression

- Suppose we want to predict a real-valued quantity  $y \in \mathbb{R}$  instead of a class label  $y \in \{1, \dots, C\}$ .
- E.g., in the case of Iris flowers,  $y$  might be the average height of the plant, or the degree of toxicity.

This type of problem is known as **regression**.

# Regression

- Regression is very similar to classification, but since the output  $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$  is real-valued, we need to use a different loss function.
- Why don't we use the misclassification rate with the zero-one loss?

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \ell_{01}(y, \hat{y}) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(y_n \neq f(\mathbf{x}_n; \boldsymbol{\theta}))$$

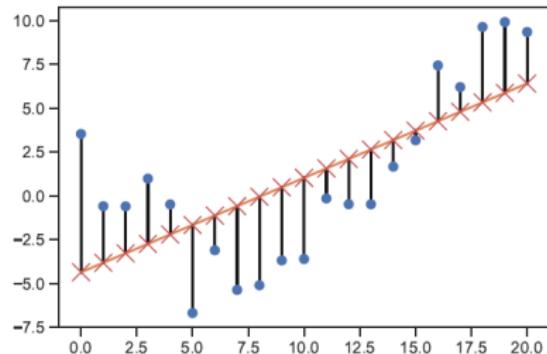
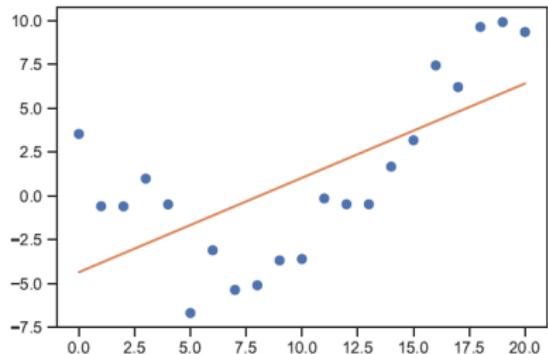
- The most common choice is **quadratic loss**, or  $\ell_2$  loss.

$$\ell_2(y, \hat{y}) = (y - \hat{y})^2$$

- This penalizes large **residuals**  $y - \hat{y}$  more than small ones.
- The empirical risk when using quadratic loss is equal to the **mean squared error** (MSE):

$$\mathcal{L}(\boldsymbol{\theta}) = \text{MSE}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n; \boldsymbol{\theta}))^2$$

# Linear Regression



We can fit the 1d data above using a **simple linear regression** model:

$$f(x; \theta) = b + wx$$

where  $w$  is the **slope**, and  $b$  is the **offset**.  $\theta = (w, b)$  are the parameters of the model. By adjusting  $\theta$ , we can minimize the sum of squared errors until we find the **least squares solution**:

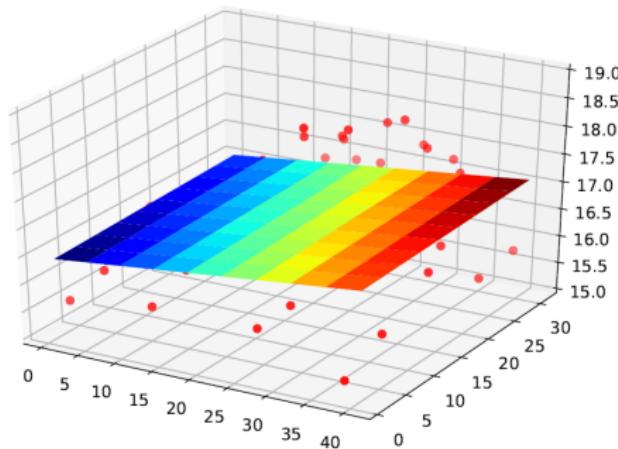
$$\hat{\theta} = \operatorname{argmin}_{\theta} \text{MSE}(\theta)$$

# Linear Regression

If we have multiple input features, we can write

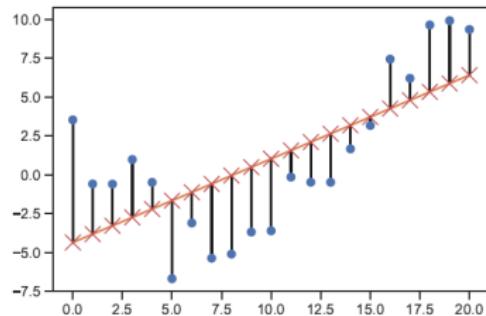
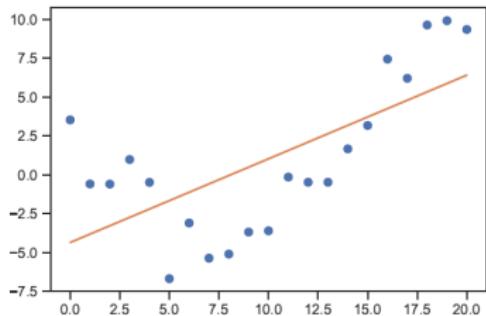
$$f(\mathbf{x}; \boldsymbol{\theta}) = b + w_1x_1 + \dots + w_Dx_D = b + \mathbf{w}^T \mathbf{x}$$

where  $\boldsymbol{\theta} = (\mathbf{w}, b)$ . This is called **multiple linear regression**.



**Figure:** We predict the temperature as a function of 2d location in a room. Linear model:  $f(\mathbf{x}; \boldsymbol{\theta}) = b + w_1x_1 + w_2x_2$

# Polynomial regression



We can improve the fit by a **polynomial regression** model of degree  $D$ :

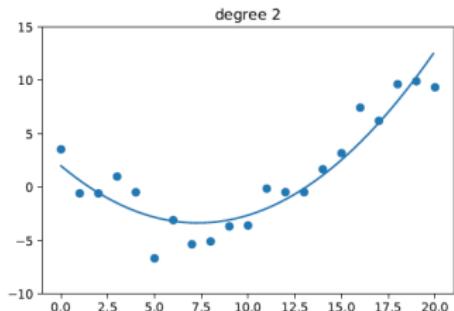
$$f(x; \mathbf{w}) = \mathbf{w}^T \phi(x)$$

where  $\phi(x)$  is a feature vector derived from the input:

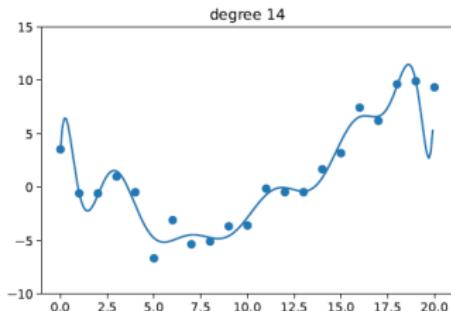
$$\phi(x) = [1, x, x^2, \dots, x^D]$$

This is an example of **feature engineering**.

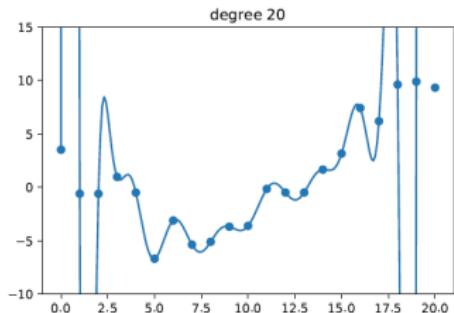
# Polynomial regression



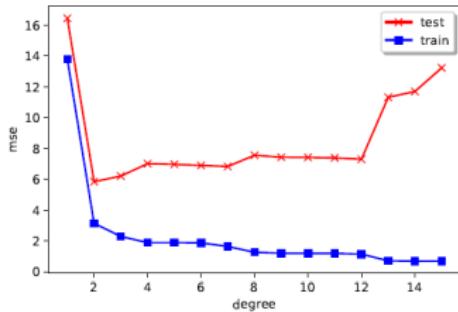
(a)



(b)

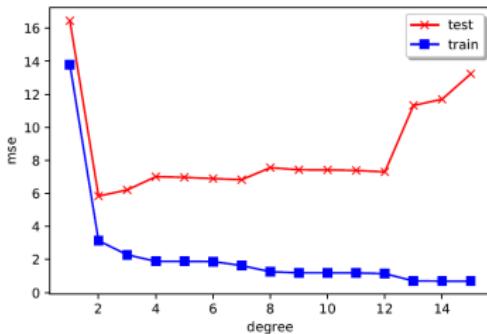


(c)



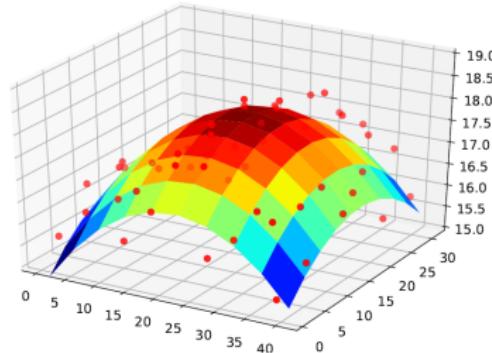
(d)

# Polynomial regression



- Using  $D = 2$  results in a much better fit.
- We can keep increasing  $D$ , and hence the number of parameters in the model, until  $D = N - 1$ .
- In this case, we have one parameter per data point, so we can perfectly **interpolate** the data. The resulting model will have 0 MSE.
- However, the resulting function will not be a good predictor.
- A model that perfectly fits the training data, but which is too complex, is said to suffer from **overfitting**.

# Polynomial regression



- We can apply polynomial regression to multi-dimensional inputs.
- For example, we use a quadratic expansion of the inputs as the prediction function for the temperature model

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$$

- The prediction function is still a linear function of the parameters  $\mathbf{w}$ , even though it is a nonlinear function of the original input  $\mathbf{x}$ .

# Deep neural networks

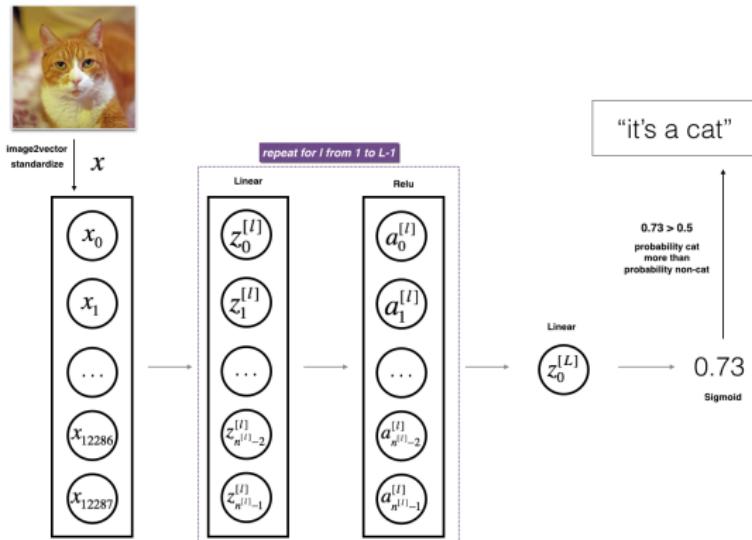
- Previously, we **manually** specify the transformation of the input features (feature engineering). For example, in polynomial regression

$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, \dots]$$

- We can create more powerful models by learning to do such *nonlinear feature extraction* **automatically**.
- We let  $\phi(\mathbf{x})$  have its own set of parameters  $\mathbf{V}$ :

$$f(\mathbf{x}; \mathbf{w}, \mathbf{V}) = \mathbf{w}^T \phi(\mathbf{x}; \mathbf{V})$$

# Deep neural networks

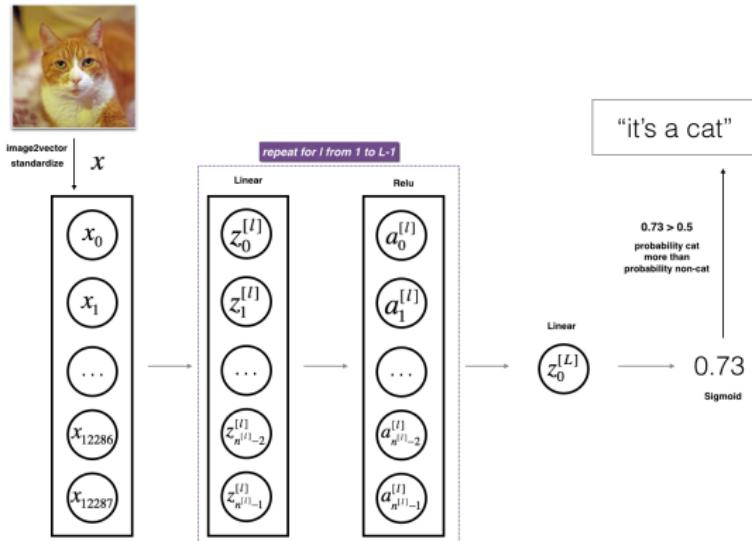


The **feature extractor**  $\phi(\mathbf{x}; \mathbf{V})$  is a composition of simpler functions.  
The resulting model is a stack of  $L$  nested functions:

$$f(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{x}; \mathbf{w}, \mathbf{V}) = f_L(f_{L-1}(\dots(f_1(\mathbf{x}))\dots))$$

where  $f_\ell(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}_\ell)$  is the function at layer  $\ell$ .

# Deep neural networks



$$f(\mathbf{x}; \mathbf{w}, \mathbf{V}) = \mathbf{w}^T \phi(\mathbf{x}; \mathbf{V})$$

Typically, the final layer  $L$  is *linear* and has the form

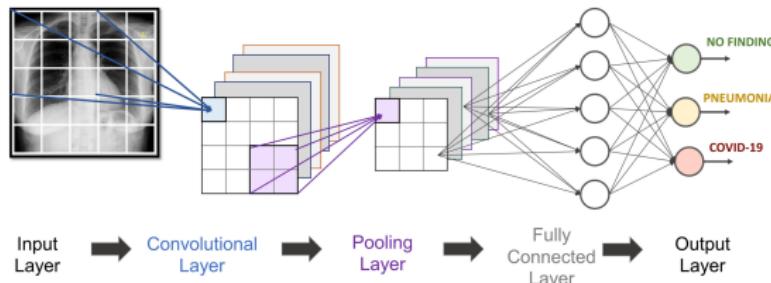
$$f_L(\mathbf{x}) = \mathbf{w}^T f_{1:L-1}(\mathbf{x})$$

where  $f_{1:L-1}(\mathbf{x})$  is the **learned** feature extractor  $\phi(\mathbf{x}; \mathbf{V})$ .

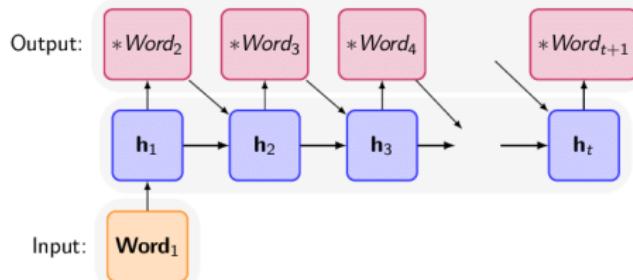
# Deep neural networks

Common DNN variants are:

- Convolutional neural networks (CNNs) for images.



- Recurrent neural networks (RNNs) for sequences.



# Overfitting and Generalization

- We can rewrite the empirical risk equation as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n; \boldsymbol{\theta}))$$

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_{\text{train}}} \ell(y, f(\mathbf{x}; \boldsymbol{\theta}))$$

where  $\mathcal{D}_{\text{train}}$  is the size of the training set  $\mathcal{D}_{\text{train}}$ .

- This makes explicit which dataset the loss is being evaluated on.
- With a complex enough model (or simply by memorizing all training data), we can achieve **zero** training loss.
- But we care about prediction accuracy on new data, that are not in the training set.
- A model that perfectly fits the training data, but which is too complex, is said to suffer from **overfitting**.

# Overfitting and Generalization

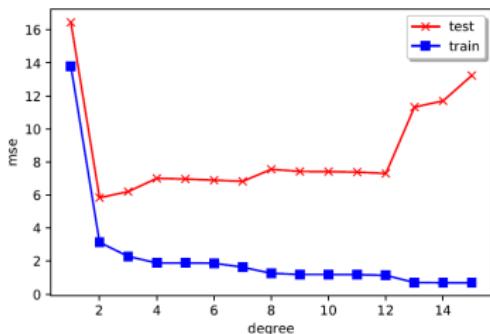
- Let  $p^*(x, y)$  denote the true (and **unknown**) distribution of the training data. Instead of the empirical risk, we have the **population risk** (or the **theoretical** expected loss):

$$\mathcal{L}(\boldsymbol{\theta}; p^*) = \mathbb{E}_{p^*(\mathbf{x}, y)} [\ell(y, f((\mathbf{x}); \boldsymbol{\theta}))]$$

- The difference  $\mathcal{L}(\boldsymbol{\theta}; p^*) - \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}})$  is the **generalization gap**.
- If a model has a large generalization gap (i.e., low empirical risk but high population risk), it is a sign of overfitting.
- However, in practice, we don't know  $p^*$ .
- We partition the data we have into two subsets: the training set and the **test set**. We approximate the population risk using the **test risk**:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{test}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_{\text{test}}} \ell(y, f(\mathbf{x}; \boldsymbol{\theta}))$$

# Overfitting and Generalization



- The training error goes to 0 as the model becomes more complex.
- The test error has a **U-shaped curve**:
  - On the left, where  $D = 1$ , the model is **underfitting**.
  - On the right, where  $D \gg 1$ , the model is **overfitting**.
  - When  $D = 2$ , the model complexity is “just right”.
- The most complex model has the most **degrees of freedom** → minimum training loss.
- We should pick the model with the minimum test loss.

# Overfitting and Generalization

- In practice, we partition the data into 3 sets: a training set, a test set, and a **validation set**.
- The training set is used for model fitting.
- The validation set is used for model selection.
- The test set is used to estimate future performance (i.e., the population risk).
- The test set is NOT used for model fitting or model selection.

# No free lunch theorem

*All models are wrong, but some models are useful.*

— George Box

- There is no single best model that works optimally for all kinds of problems.
- A set of assumptions (**inductive bias**) that works well in one domain may work poorly in another.
- We pick a suitable model based on domain knowledge, and/or trial and error (using model selection techniques).
- It is important to have many models and algorithmic techniques to choose from.

# CS 115: Maths for Computer Science

## Probability

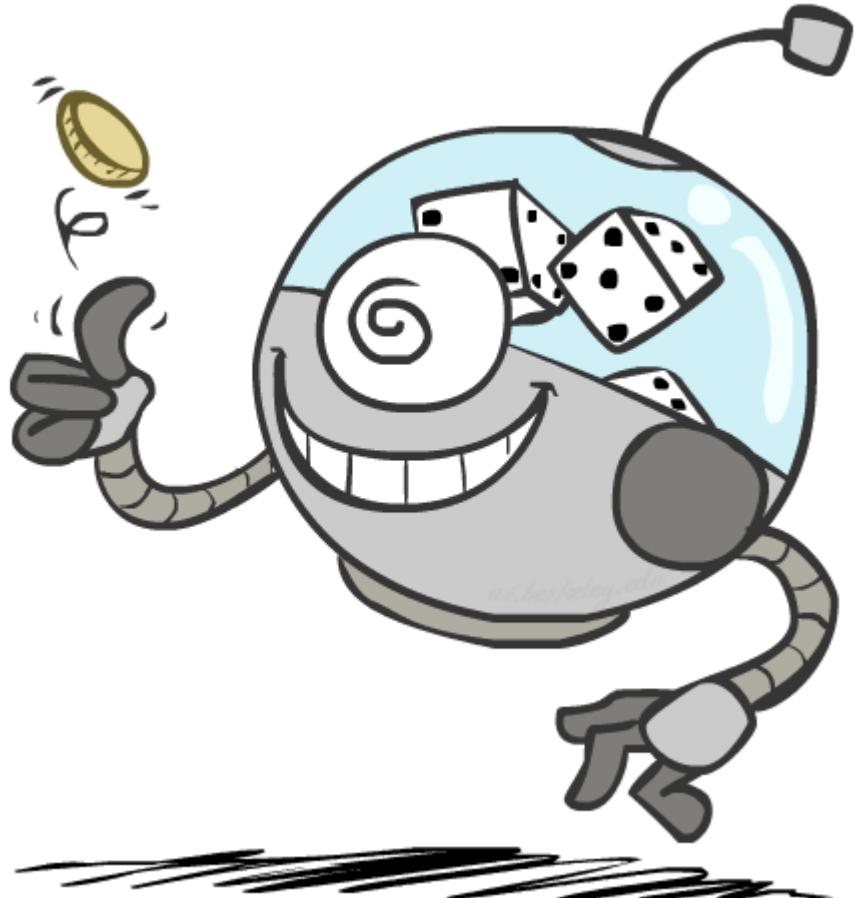


Instructor: Ngoc-Hoang LUONG

# Today

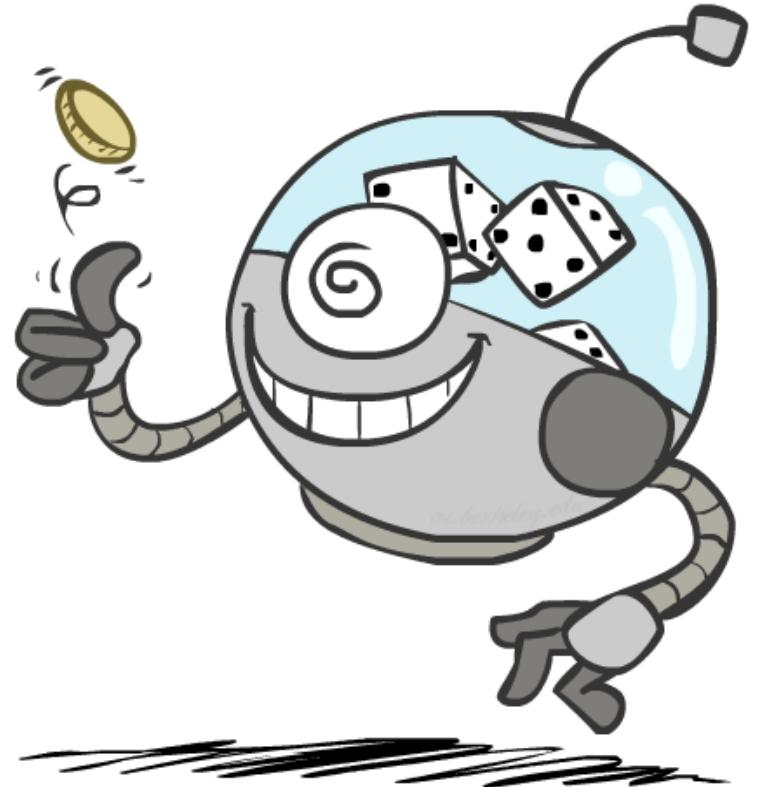
---

- Probability
  - Random Variables
  - Joint and Marginal Distributions
  - Conditional Distribution
  - Product Rule, Chain Rule, Bayes' Rule
  - Inference
  - Independence



# Random Variables

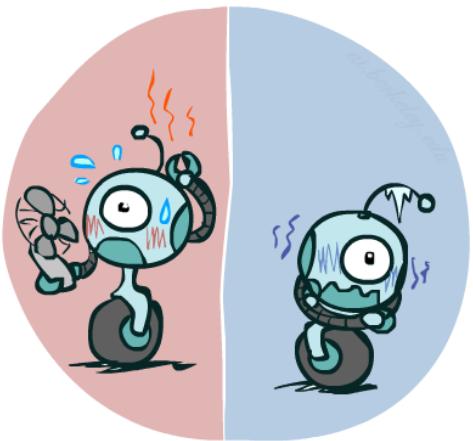
- A random variable is some aspect of the world about which we (may) have uncertainty
  - R = Is it raining?
  - T = Is it hot or cold?
  - D = How long will it take to drive to work?
  - L = Where is the ghost?
- We denote random variables with capital letters
- Random variables have domains
  - R in {true, false}
  - T in {hot, cold}
  - D in  $[0, \infty)$
  - L in possible locations, maybe  $\{(0,0), (0,1), \dots\}$



# Probability Distributions

- Associate a probability with each value

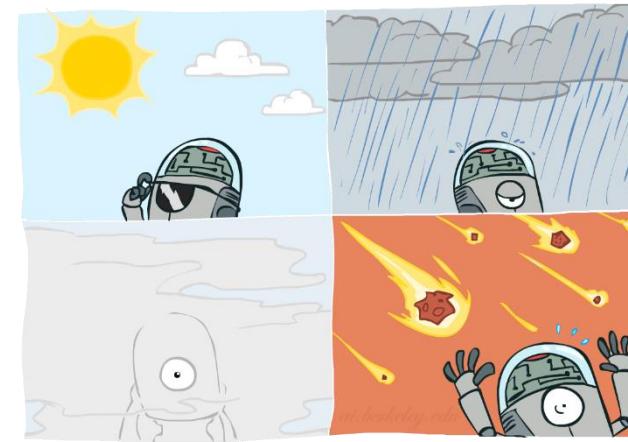
- Temperature:



$P(T)$

T	P
hot	0.5
cold	0.5

- Weather:



$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

# Probability Distributions

- Unobserved random variables have distributions

T	P
hot	0.5
cold	0.5

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

- A distribution is a TABLE of probabilities of values
- A probability (lower case value) is a single number

$$P(W = \text{rain}) = 0.1$$

- Must have:  $\forall x \ P(X = x) \geq 0$  and  $\sum_x P(X = x) = 1$

Shorthand notation:

$$P(\text{hot}) = P(T = \text{hot}),$$

$$P(\text{cold}) = P(T = \text{cold}),$$

$$P(\text{rain}) = P(W = \text{rain}),$$

...

OK if all domain entries are unique

# Joint Distributions

- A *joint distribution* over a set of random variables:  $X_1, X_2, \dots, X_n$  specifies a real number for each assignment (or *outcome*):

$$P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$$

$$P(x_1, x_2, \dots, x_n)$$

- Must obey:  $P(x_1, x_2, \dots, x_n) \geq 0$

$$\sum_{(x_1, x_2, \dots, x_n)} P(x_1, x_2, \dots, x_n) = 1$$

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

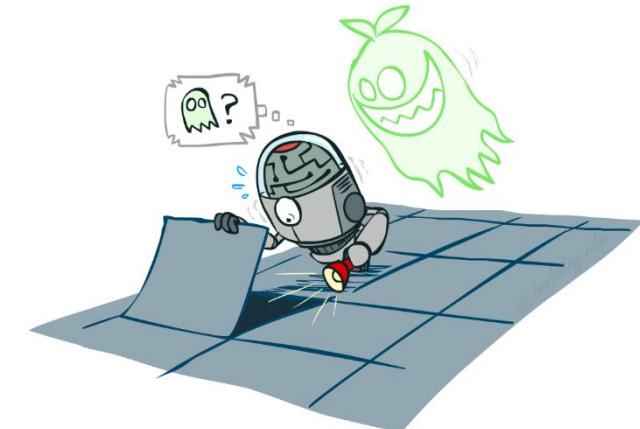
- Size of distribution if  $n$  variables with domain sizes  $d$ ?
  - For all but the smallest distributions, impractical to write out!

# Probabilistic Models

- A probabilistic model is a joint distribution over a set of random variables
- Probabilistic models:
  - (Random) variables with domains
  - Assignments are called *outcomes*
  - Joint distributions: say whether assignments (outcomes) are likely
  - *Normalized*: sum to 1.0
  - Ideally: only certain variables directly interact
- Constraint satisfaction problems:
  - Variables with domains
  - Constraints: state whether assignments are possible
  - Ideally: only certain variables directly interact

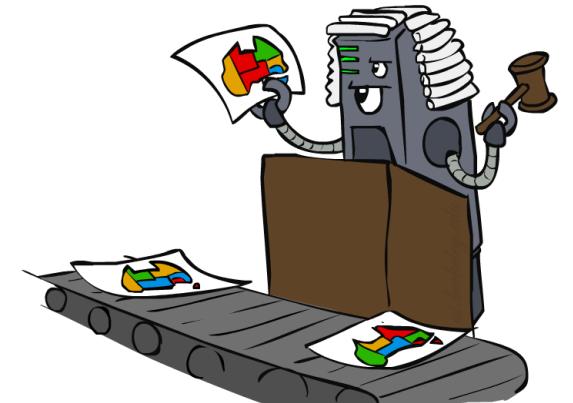
Distribution over T,W

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3



Constraint over T,W

T	W	P
hot	sun	T
hot	rain	F
cold	sun	F
cold	rain	T



# Events

- An *event* is a set  $E$  of outcomes

$$P(E) = \sum_{(x_1 \dots x_n) \in E} P(x_1 \dots x_n)$$

- From a joint distribution, we can calculate the probability of any event

- Probability that it's hot AND sunny?
- Probability that it's hot?
- Probability that it's hot OR sunny?

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

- Typically, the events we care about are *partial assignments*, like  $P(T=\text{hot})$

# Quiz: Events

- $P(+x, +y) ?$

$$P(X, Y)$$

- $P(+x) ?$

- $P(-y \text{ OR } +x) ?$

X	Y	P
+x	+y	0.2
+x	-y	0.3
-x	+y	0.4
-x	-y	0.1

# Marginal Distributions

- Marginal distributions are sub-tables which eliminate variables
- Marginalization (summing out): Combine collapsed rows by adding

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3



$$P(t) = \sum_s P(t, s)$$

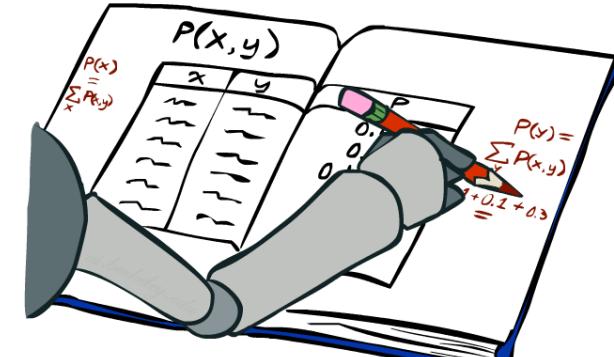


$$P(s) = \sum_t P(t, s)$$

$$P(X_1 = x_1) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2)$$

$P(T)$	
T	P
hot	0.5
cold	0.5

$P(W)$	
W	P
sun	0.6
rain	0.4



# Quiz: Marginal Distributions

$P(X, Y)$

X	Y	P
+x	+y	0.2
+x	-y	0.3
-x	+y	0.4
-x	-y	0.1

$$P(x) = \sum_y P(x, y)$$

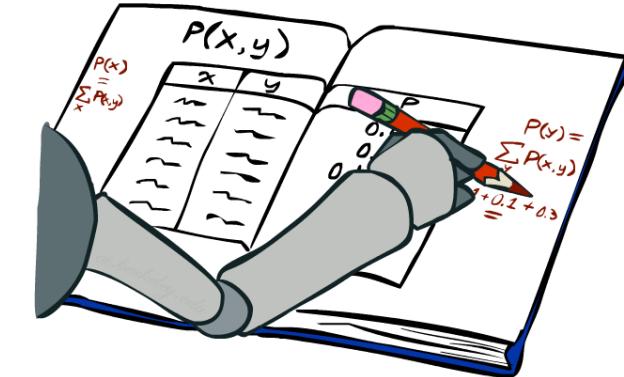
$$P(y) = \sum_x P(x, y)$$

$P(X)$

X	P
+x	
-x	

$P(Y)$

Y	P
+y	
-y	



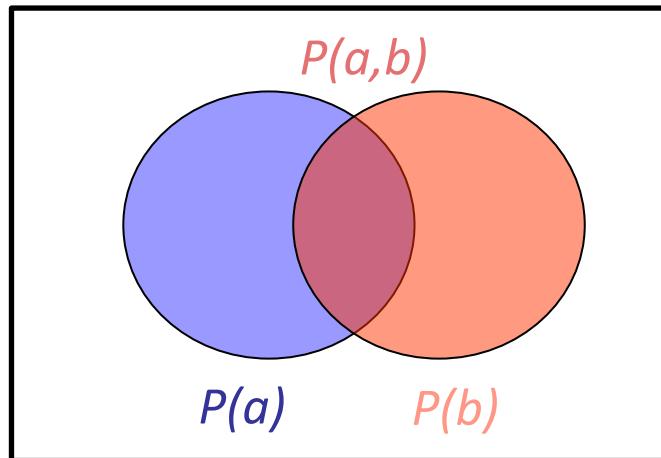
# Conditional Probabilities

- A simple relation between joint and conditional probabilities
  - In fact, this is taken as the *definition* of a conditional probability

$$P(a|b) = \frac{P(a,b)}{P(b)}$$

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3



$$P(W = s | T = c) = \frac{P(W = s, T = c)}{P(T = c)} = \frac{0.2}{0.5} = 0.4$$

$$\begin{aligned} &= P(W = s, T = c) + P(W = r, T = c) \\ &= 0.2 + 0.3 = 0.5 \end{aligned}$$

# Quiz: Conditional Probabilities

- $P(+x \mid +y) ?$

$P(X, Y)$

X	Y	P
+x	+y	0.2
+x	-y	0.3
-x	+y	0.4
-x	-y	0.1

- $P(-x \mid +y) ?$

- $P(-y \mid +x) ?$

# Conditional Distributions

- Conditional distributions are probability distributions over some variables given fixed values of others

Conditional Distributions

$$P(W|T)$$

$P(W T = hot)$	
W	P
sun	0.8
rain	0.2
$P(W T = cold)$	
W	P
sun	0.4
rain	0.6

Joint Distribution

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

# Normalization Trick

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$\begin{aligned} P(W = s|T = c) &= \frac{P(W = s, T = c)}{P(T = c)} \\ &= \frac{P(W = s, T = c)}{P(W = s, T = c) + P(W = r, T = c)} \\ &= \frac{0.2}{0.2 + 0.3} = 0.4 \end{aligned}$$



$P(W|T = c)$

W	P
sun	0.4
rain	0.6

$$\begin{aligned} P(W = r|T = c) &= \frac{P(W = r, T = c)}{P(T = c)} \\ &= \frac{P(W = r, T = c)}{P(W = s, T = c) + P(W = r, T = c)} \\ &= \frac{0.3}{0.2 + 0.3} = 0.6 \end{aligned}$$

# Normalization Trick

$$\begin{aligned} P(W = s|T = c) &= \frac{P(W = s, T = c)}{P(T = c)} \\ &= \frac{P(W = s, T = c)}{P(W = s, T = c) + P(W = r, T = c)} \\ &= \frac{0.2}{0.2 + 0.3} = 0.4 \end{aligned}$$

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

**SELECT** the joint probabilities matching the evidence  
→

T	W	P
cold	sun	0.2
cold	rain	0.3

$P(c, W)$

**NORMALIZE** the selection  
(make it sum to one)  
→

W	P
sun	0.4
rain	0.6

$P(W|T = c)$

$$\begin{aligned} P(W = r|T = c) &= \frac{P(W = r, T = c)}{P(T = c)} \\ &= \frac{P(W = r, T = c)}{P(W = s, T = c) + P(W = r, T = c)} \\ &= \frac{0.3}{0.2 + 0.3} = 0.6 \end{aligned}$$

# Normalization Trick

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

**SELECT** the joint probabilities matching the evidence  
→

$P(c, W)$

T	W	P
cold	sun	0.2
cold	rain	0.3

**NORMALIZE** the selection  
(make it sum to one)  
→

$P(W|T = c)$

W	P
sun	0.4
rain	0.6

- Why does this work? Sum of selection is  $P(\text{evidence})!$  ( $P(T=c)$ , here)

$$P(x_1|x_2) = \frac{P(x_1, x_2)}{P(x_2)} = \frac{P(x_1, x_2)}{\sum_{x_1} P(x_1, x_2)}$$

# Quiz: Normalization Trick

- $P(X | Y=-y) ?$

$P(X, Y)$

X	Y	P
+x	+y	0.2
+x	-y	0.3
-x	+y	0.4
-x	-y	0.1

**SELECT** the joint  
probabilities  
matching the  
evidence



**NORMALIZE** the  
selection  
(make it sum to one)



# To Normalize

- (Dictionary) To bring or restore to a **normal condition**

All entries sum to ONE

- Procedure:

- Step 1: Compute  $Z = \text{sum over all entries}$
- Step 2: Divide every entry by  $Z$

- Example 1

W	P
sun	0.2
rain	0.3

Normalize  $\rightarrow$   $Z = 0.5$

W	P
sun	0.4
rain	0.6

- Example 2

T	W	P
hot	sun	20
hot	rain	5
cold	sun	10
cold	rain	15

Normalize  $\rightarrow$   $Z = 50$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

# Probabilistic Inference

- Probabilistic inference: compute a desired probability from other known probabilities (e.g. conditional from joint)
- We generally compute conditional probabilities
  - $P(\text{on time} \mid \text{no reported accidents}) = 0.90$
  - These represent the agent's *beliefs* given the evidence
- Probabilities change with new evidence:
  - $P(\text{on time} \mid \text{no accidents, 5 a.m.}) = 0.95$
  - $P(\text{on time} \mid \text{no accidents, 5 a.m., raining}) = 0.80$
  - Observing new evidence causes *beliefs to be updated*



# Inference by Enumeration

- General case:

- Evidence variables:  $E_1 \dots E_k = e_1 \dots e_k$
- Query\* variable:  $Q$
- Hidden variables:  $H_1 \dots H_r$

$$\left. \begin{array}{l} E_1 \dots E_k = e_1 \dots e_k \\ Q \\ H_1 \dots H_r \end{array} \right\} X_1, X_2, \dots, X_n$$

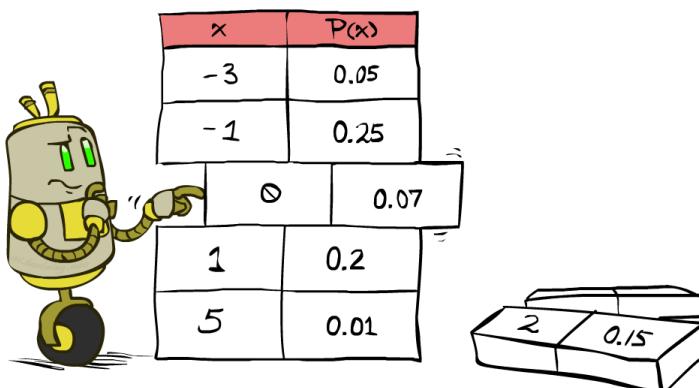
*All variables*

- We want:

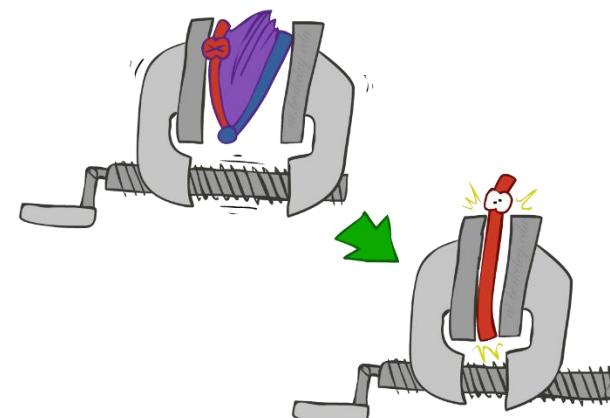
$$P(Q|e_1 \dots e_k)$$

\* Works fine with multiple query variables, too

- Step 1: Select the entries consistent with the evidence



- Step 2: Sum out H to get joint of Query and evidence



- Step 3: Normalize

$$\times \frac{1}{Z}$$

$$Z = \sum_q P(Q, e_1 \dots e_k)$$

$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots, X_n}$$

$$P(Q|e_1 \dots e_k) = \frac{1}{Z} P(Q, e_1 \dots e_k)$$

# Inference by Enumeration

- $P(W)?$
- $P(W | \text{winter})?$
- $P(W | \text{winter, hot})?$

S	T	W	P
summer	hot	sun	0.30
summer	hot	rain	0.05
summer	cold	sun	0.10
summer	cold	rain	0.05
winter	hot	sun	0.10
winter	hot	rain	0.05
winter	cold	sun	0.15
winter	cold	rain	0.20

# Inference by Enumeration

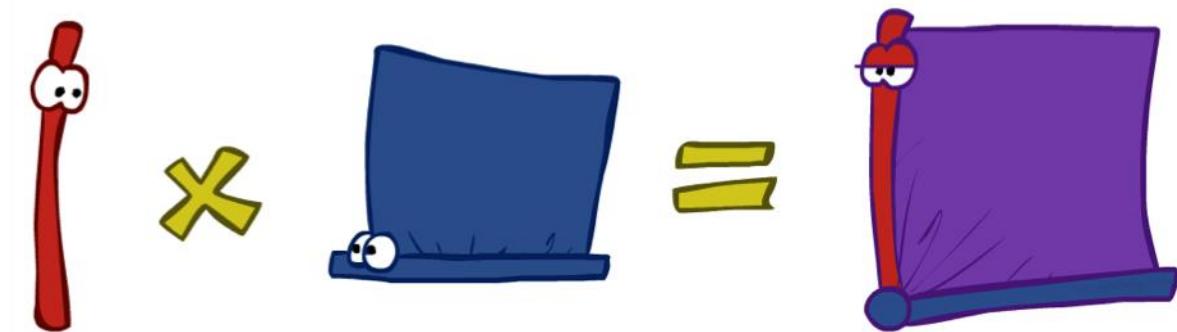
---

- Obvious problems:
  - Worst-case time complexity  $O(d^n)$
  - Space complexity  $O(d^n)$  to store the joint distribution

# The Product Rule

- Sometimes have conditional distributions but want the joint

$$P(y)P(x|y) = P(x, y) \quad \longleftrightarrow \quad P(x|y) = \frac{P(x, y)}{P(y)}$$



# The Product Rule

$$P(y)P(x|y) = P(x, y)$$

- Example:

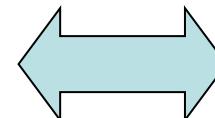
R	P
sun	0.8
rain	0.2

$$P(D|W)$$

D	W	P
wet	sun	0.1
dry	sun	0.9
wet	rain	0.7
dry	rain	0.3

$$P(D, W)$$

D	W	P
wet	sun	
dry	sun	
wet	rain	
dry	rain	



# The Chain Rule

---

- More generally, can always write any joint distribution as an incremental product of conditional distributions

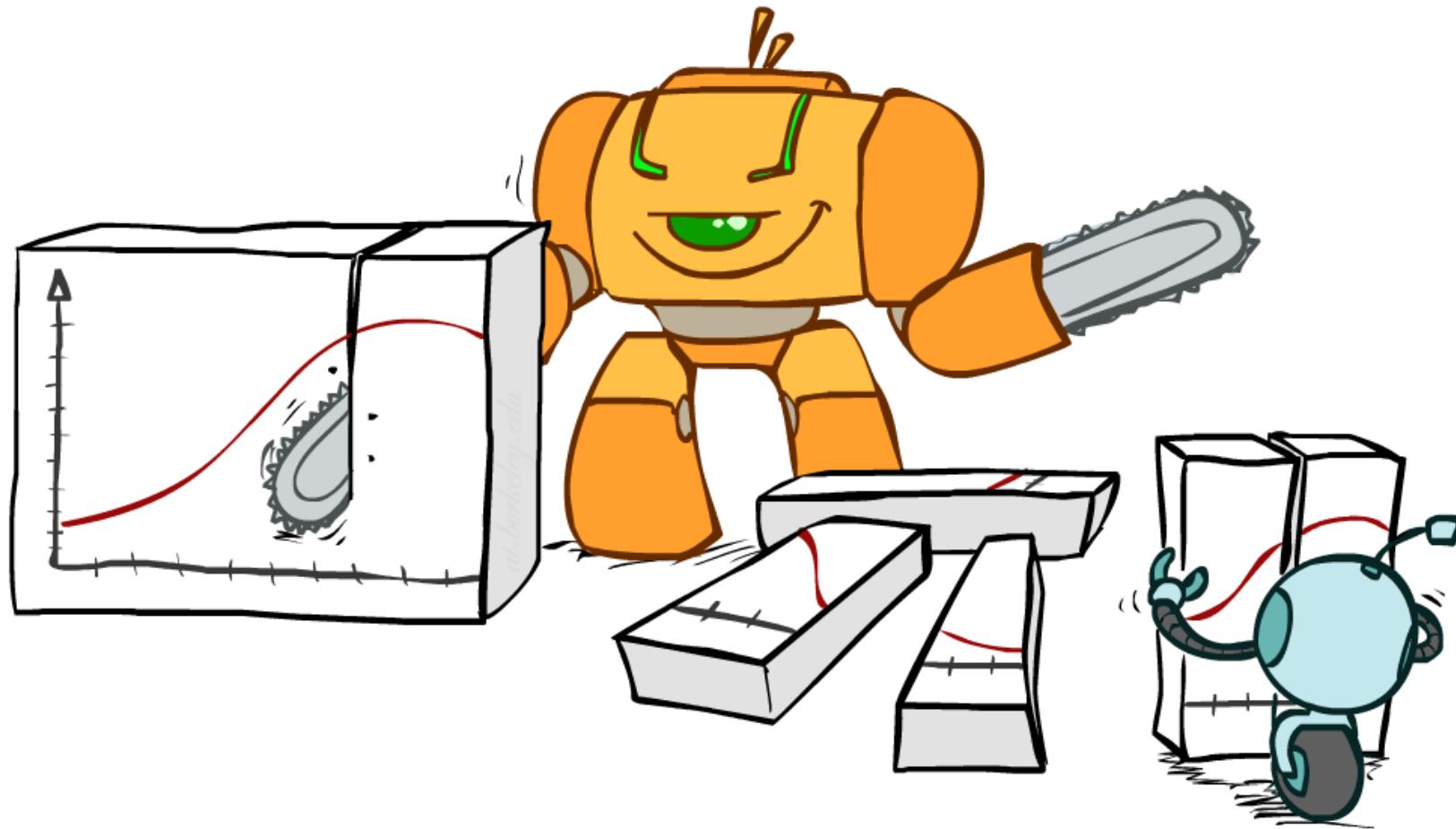
$$P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$$

$$P(x_1, x_2, \dots, x_n) = \prod_i P(x_i|x_1 \dots x_{i-1})$$

- Why is this always true?

# Bayes Rule

---



# Bayes' Rule

- Two ways to factor a joint distribution over two variables:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

That's my rule!

- Dividing, we get:

$$P(x|y) = \frac{P(y|x)}{P(y)}P(x)$$

- Why is this at all helpful?

- Lets us build one conditional from its reverse
- Often one conditional is tricky but the other one is simple
- Foundation of many systems we'll see later (e.g. ASR, MT)

- In the running for most important AI equation!



# Inference with Bayes' Rule

- Example: Diagnostic probability from causal probability:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})P(\text{cause})}{P(\text{effect})}$$

- Example:

- M: meningitis, S: stiff neck

$$\left. \begin{array}{l} P(+m) = 0.0001 \\ P(+s|m) = 0.8 \\ P(+s|-m) = 0.01 \end{array} \right\} \text{Example givens}$$

$$P(+m|s) = \frac{P(+s|m)P(+m)}{P(+s)} = \frac{P(+s|m)P(+m)}{P(+s|m)P(+m) + P(+s|-m)P(-m)} = \frac{0.8 \times 0.0001}{0.8 \times 0.0001 + 0.01 \times 0.999}$$

- Note: posterior probability of meningitis still very small
  - Note: you should still get stiff necks checked out! Why?

# Inference with Bayes' Rule

---

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})P(\text{cause})}{P(\text{effect})}$$

Example:

- I am 90% confident that I'm a good singer.  $P(\text{good singer}) = 0.9$
- If I'm a good singer, then 99% of people will like my singing.  $P(\text{like} | \text{good singer}) = 0.99$
- If I'm a bad singer, then 10% of people will like my singing.  $P(\text{like} | \text{bad singer}) = 0.10$
- I sing in my living room and my roommate covers his ears.
- I need to update my beliefs to account for what I've learned.
- I need to calculate:  $P(\text{good singer} | \text{roommate doesn't like my singing}) = ?$

# Quiz: Bayes' Rule

- Given:

$$P(W)$$

R	P
sun	0.8
rain	0.2

$$P(D|W)$$

D	W	P
wet	sun	0.1
dry	sun	0.9
wet	rain	0.7
dry	rain	0.3

- What is  $P(W | \text{dry})$  ?

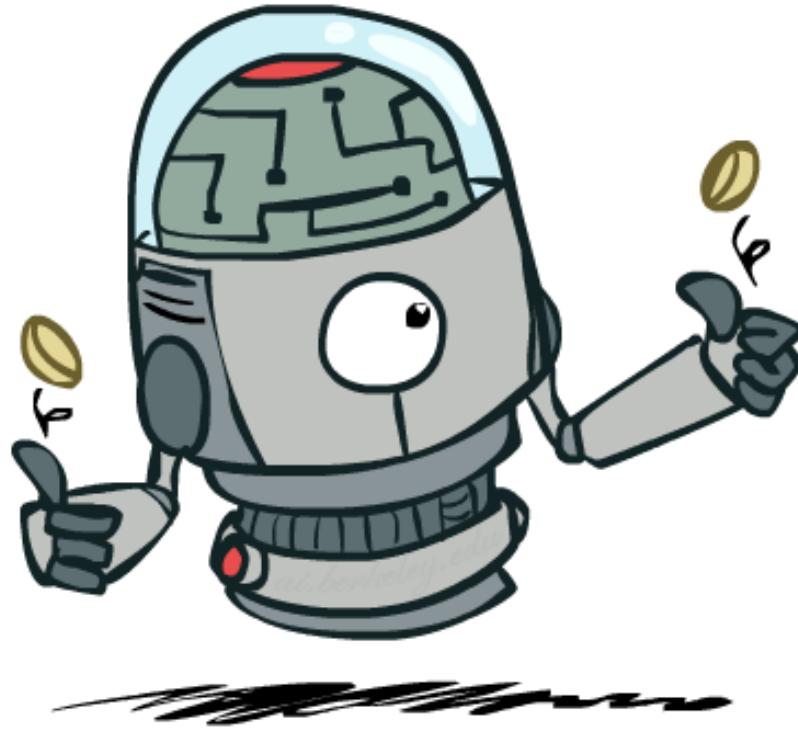
# Probabilistic Models

- Models describe how (a portion of) the world works
- **Models are always simplifications**
  - May not account for every variable
  - May not account for all interactions between variables
  - “All models are wrong; but some are useful.”
    - George E. P. Box
- What do we do with probabilistic models?
  - We (or our agents) need to reason about unknown variables, given evidence
  - Example: explanation (diagnostic reasoning)
  - Example: prediction (causal reasoning)
  - Example: value of information



# Independence

---



# Independence

- Two variables are *independent* if:

$$\forall x, y : P(x, y) = P(x)P(y)$$

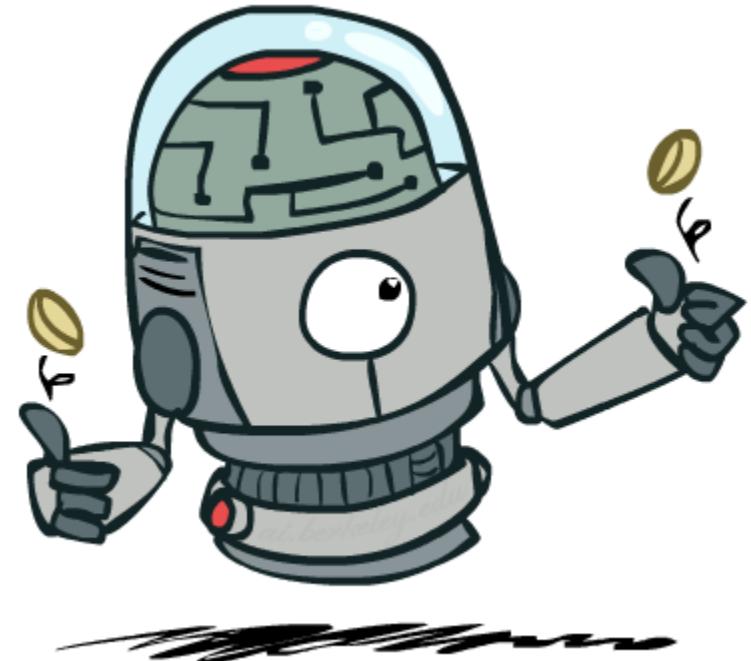
- This says that their joint distribution *factors* into a product two simpler distributions
- Another form:

$$\forall x, y : P(x|y) = P(x)$$

- We write:  $X \perp\!\!\!\perp Y$

- Independence is a simplifying *modeling assumption*

- *Empirical* joint distributions: at best “close” to independent
- What could we assume for {Weather, Traffic, Cavity, Toothache}?



# Example: Independence?

$P_1(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$P(T)$

T	P
hot	0.5
cold	0.5

$P_2(T, W)$

T	W	P
hot	sun	0.3
hot	rain	0.2
cold	sun	0.3
cold	rain	0.2

$P(W)$

W	P
sun	0.6
rain	0.4

# Example: Independence

- N fair, independent coin flips:

$$P(X_1)$$

H	0.5
T	0.5

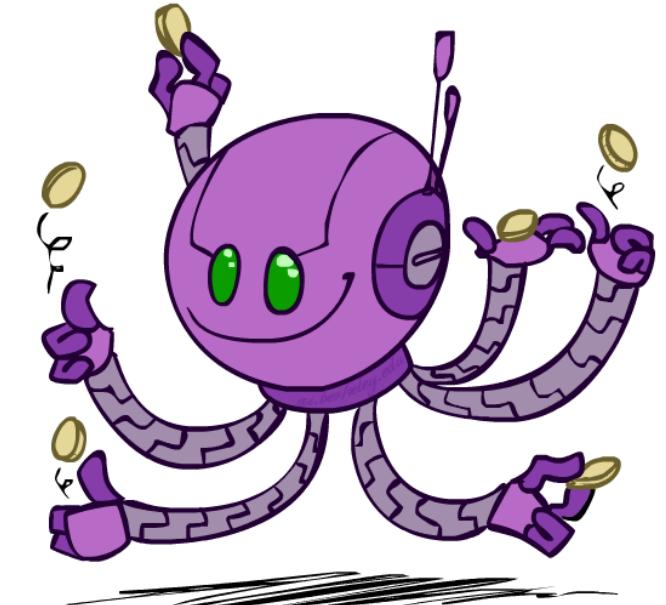
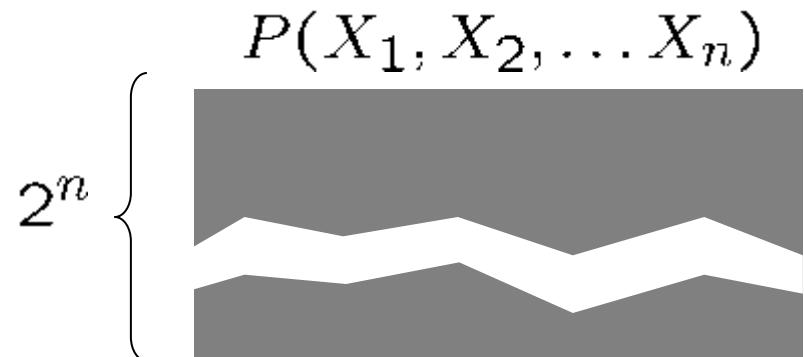
$$P(X_2)$$

H	0.5
T	0.5

...

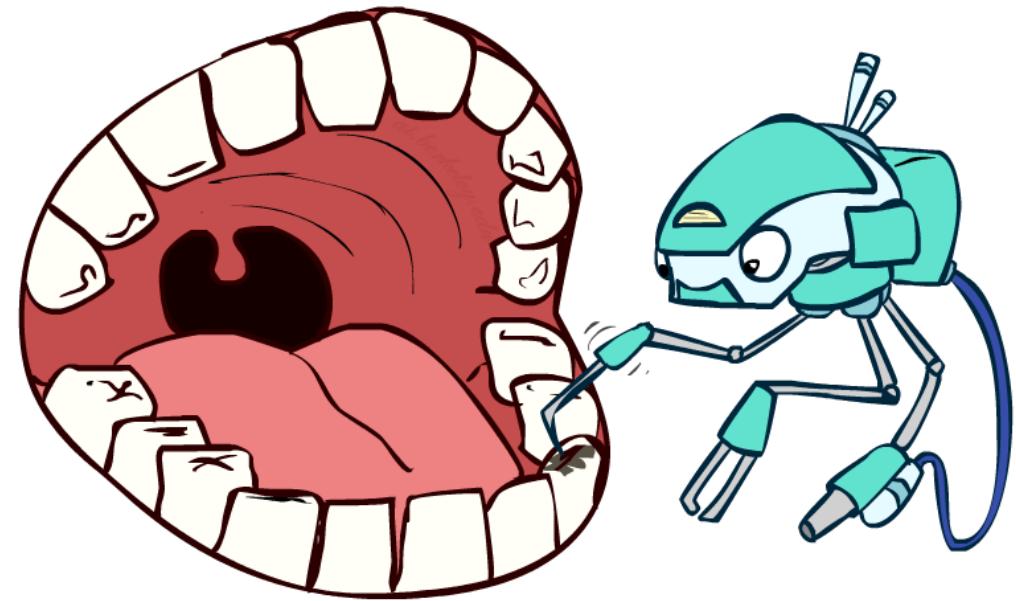
$$P(X_n)$$

H	0.5
T	0.5



# Conditional Independence

- $P(\text{Toothache}, \text{Cavity}, \text{Catch})$
- If I have a cavity, the probability that the probe catches it doesn't depend on whether I have a toothache:
  - $P(+\text{catch} | +\text{toothache}, +\text{cavity}) = P(+\text{catch} | +\text{cavity})$
- The same independence holds if I don't have a cavity:
  - $P(+\text{catch} | +\text{toothache}, -\text{cavity}) = P(+\text{catch} | -\text{cavity})$
- Catch is *conditionally independent* of Toothache given Cavity:
  - $P(\text{Catch} | \text{Toothache}, \text{Cavity}) = P(\text{Catch} | \text{Cavity})$
- Equivalent statements:
  - $P(\text{Toothache} | \text{Catch}, \text{Cavity}) = P(\text{Toothache} | \text{Cavity})$
  - $P(\text{Toothache}, \text{Catch} | \text{Cavity}) = P(\text{Toothache} | \text{Cavity}) P(\text{Catch} | \text{Cavity})$
  - One can be derived from the other easily



# Conditional Independence

---

- Unconditional (absolute) independence very rare (why?)
- *Conditional independence* is our most basic and robust form of knowledge about uncertain environments.
- $X$  is conditionally independent of  $Y$  given  $Z$   $X \perp\!\!\!\perp Y | Z$

if and only if:

$$\forall x, y, z : P(x, y|z) = P(x|z)P(y|z)$$

or, equivalently, if and only if

$$\forall x, y, z : P(x|z, y) = P(x|z)$$

# Conditional Independence

---

- What about this domain:

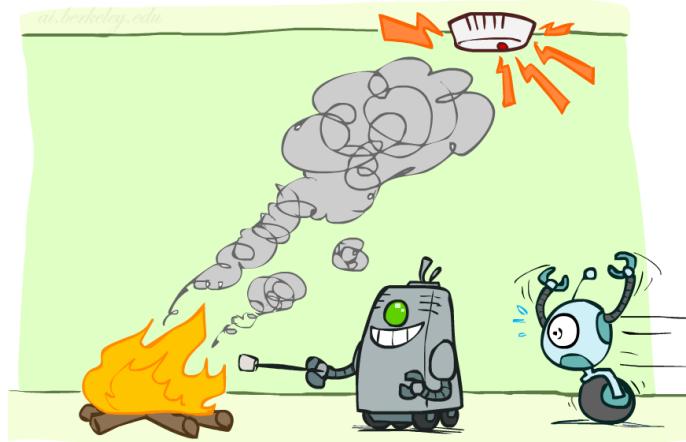
- Traffic
- Umbrella
- Raining



# Conditional Independence

- What about this domain:

- Fire
- Smoke
- Alarm



# Conditional Independence and the Chain Rule

- Chain rule:

$$P(X_1, X_2, \dots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \dots$$

- Trivial decomposition:

$$P(\text{Traffic, Rain, Umbrella}) =$$

$$P(\text{Rain})P(\text{Traffic}|\text{Rain})P(\text{Umbrella}|\text{Rain, Traffic})$$

- With assumption of conditional independence:

$$P(\text{Traffic, Rain, Umbrella}) =$$

$$P(\text{Rain})P(\text{Traffic}|\text{Rain})P(\text{Umbrella}|\text{Rain})$$



- Bayes' nets / graphical models help us express conditional independence assumptions

# Probability Theory

## Review

Faculty of Computer Science  
University of Information Technology (UIT)  
Vietnam National University - Ho Chi Minh City (VNU-HCM)

Maths for Computer Science, Fall 2022

# References

The contents of this document are taken mainly from the follow sources:

- John Tsitsiklis. Massachusetts Institute of Technology. Introduction to Probability.<sup>1</sup>
- Marek Rutkowski. University of Sydney. Probability Review.<sup>2</sup>
- <https://www.probabilitycourse.com/>

---

<sup>1</sup>[https://ocw.mit.edu/resources/  
res-6-012-introduction-to-probability-spring-2018/index.htm](https://ocw.mit.edu/resources/res-6-012-introduction-to-probability-spring-2018/index.htm)

<sup>2</sup>[http://www.maths.usyd.edu.au/u/UG/SM/MATH3075/r/Slides\\_1\\_Probability.pdf](http://www.maths.usyd.edu.au/u/UG/SM/MATH3075/r/Slides_1_Probability.pdf)

# Table of Contents

- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

# Table of Contents

- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

# Sample Space

- List (set) of all possible states of the world,  $\Omega$ . The states are called **samples** or **elementary events**.
- List (set) of possible **outcomes**,  $\Omega$ .
- List must be:
  - Mutually exclusive
  - Collectively exhaustive
  - At the “right” granularity
- The sample space  $\Omega$  is either **countable** or **uncountable**.

# Probability

A discrete sample space  $\Omega = (\omega_k)_{k \in I}$ , where the set  $I$  is countable.

## Definition (Probability)

A map  $P : \Omega \mapsto [0, 1]$  is called a **probability** on a discrete sample space  $\Omega$  if the following conditions are satisfied:

- $P(\omega_k) \geq 0$  for all  $k \in I$
- $\sum_{k \in I} P(\omega_k) = 1$

# Probability Measure

- Let  $\mathcal{F} = 2^\Omega$  be the set of all subsets of the sample space  $\Omega$ .
- $\mathcal{F}$  contains the **empty set**  $\emptyset$  and  $\Omega$ .
- Any set  $A \in \mathcal{F}$  is called an **event** (or a **random event**).
- The set  $\mathcal{F}$  is called the **event space**.
- Probability is assigned to **events**.

## Definition (Probability Measure)

A map  $P : \mathcal{F} \mapsto [0, 1]$  is called a **probability measure** on  $(\Omega, \mathcal{F})$  if

- For any sequence  $A_i \in \mathcal{F}, i = 1, 2, \dots$  of events such that  $A_i \cap A_j = \emptyset$  for all  $i \neq j$  we have

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$$

- $P(\Omega) = 1$

# Probability Measure

- A probability  $P : \Omega \mapsto [0, 1]$  on a discrete sample space  $\Omega$  uniquely specifies probability of all events  $A_k = \{\omega_k\}$ .
- $P(\{\omega_k\}) = P(\omega_k) = p_k$ .

## Theorem

Let  $P : \Omega \mapsto [0, 1]$  be a probability on a discrete sample space  $\Omega$ . Then the unique probability measure on  $(\Omega, \mathcal{F})$  generated by  $P$  satisfies for all  $A \in \mathcal{F}$

$$P(A) = \sum_{\omega_k \in A} P(\omega_k)$$

# Some properties of probability

- If  $A \subset B$ , then  $P(A) \leq P(B)$ .
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A \cup B) \leq P(A) + P(B)$
- $P(A \cup B \cup C) = P(A) + P(A^c \cap B) + P(A^c \cap B^c \cap C)$

# Table of Contents

- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

# Random Variables

- A random variable associates a value (a number) to every possible outcome.
- It can take discrete or continuous values.

## Notation

Random variable  $X$

Numerical value  $x$

- Different random variables can be defined on the same sample space.
- A function of one or several random variables is also a r.v.

# Probability Mass Function (pmf)

Probability mass function (pmf) of a discrete random variable  $X$ .

- It is the “probability law” or “probability distribution” of  $X$ .
- If we fix some  $x$ , then “ $X = x$ ” is an event.

## Definition

$$p_X(x) = P(X = x) = P(\{\omega \in \Omega \text{ s.t. } X(\omega) = x\})$$

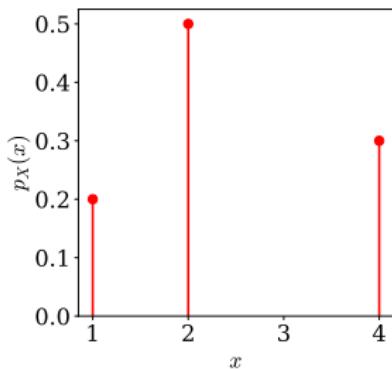
## Properties

- $p_X(x) \geq 0$
- $\sum_x p_X(x) = 1$

# Expectation

- Example: Play a game 1000 times. Random gain at each game is described by

$$X = \begin{cases} 1, & \text{with probability } 2/10 \sim 200 \\ 2, & \text{with probability } 5/10 \sim 500 \\ 4, & \text{with probability } 3/10 \sim 300 \end{cases}$$



- “Average” gain:

$$\frac{1 \cdot 200 + 2 \cdot 500 + 4 \cdot 300}{1000} = 2.4$$

- Definition:  $E[X] = \sum_x x p_X(x)$

# Expectation

- $E[X] = \sum_x x p_X(x)$
- $E(\cdot)$  is called the expectation operator.
- Average in a large number of independence experiments.
- Expectation of a r.v. can be seen as the weighted average.
- It is impossible to know the exact event to happen in the future and thus expectation is useful in making decisions when the probabilities of future outcomes are known.
- Any random variable defined on a finite set  $\Omega$  admits the expectation.
- When the set  $\Omega$  is countable but infinite, we need  $\sum_x |x| p_X(x) < \infty$  so that  $E[X]$  is well-defined.

# Expectation

## Definition

The **expectation (expected value or mean value)** of a random variable  $X$  on a discrete sample space  $\Omega$  is given by

$$E_P(X) = \mu := \sum_{k \in I} X(\omega_k)P(\omega_k) = \sum_{k \in I} x_k p_k$$

where  $P$  is a probability measure on  $\Omega$ .

## Definition

The **expectation (expected value or mean value)** of a discrete random variable  $X$  with range  $R_X = \{x_1, x_2, x_3, \dots\}$  (finite or countably infinite) is defined as

$$E(X) = \mu := \sum_{x_k \in R_X} x_k P(X = x_k) = \sum_{x_k \in R_X} x_k P_X(x_k)$$

# Elementary Properties of Expectation

## Definition

$$E[X] = \sum_x x p_X(x)$$

- If  $X \geq 0$  then  $E[X] \geq 0$ .  
For all outcomes  $w : X(w) \geq 0$ .

# Elementary Properties of Expectation

## Definition

$$E[X] = \sum_x xp_X(x)$$

- If  $X \geq 0$  then  $E[X] \geq 0$ .  
For all outcomes  $w : X(w) \geq 0$ .
- If  $a \leq X \leq b$  then  $a \leq E[X] \leq b$ .  
For all outcomes  $w : a \leq X(w) \leq b$ .  
$$E[X] = \sum_x xp_X(x) \geq \sum_x ap_X(x) = a \sum_x p_X(x) = a \cdot 1 = a$$

# Elementary Properties of Expectation

## Definition

$$E[X] = \sum_x xp_X(x)$$

- If  $X \geq 0$  then  $E[X] \geq 0$ .  
For all outcomes  $w : X(w) \geq 0$ .
- If  $a \leq X \leq b$  then  $a \leq E[X] \leq b$ .  
For all outcomes  $w : a \leq X(w) \leq b$ .  
$$E[X] = \sum_x xp_X(x) \geq \sum_x ap_X(x) = a \sum_x p_X(x) = a \cdot 1 = a$$
- If  $c$  is a constant,  $E[c] = c$   
$$E[c] = c \cdot p(c) = c$$

## Expected value rule, to compute $E[g(X)]$

- If  $X$  is a r.v. and  $Y = g(X)$ , then  $Y$  itself is a r.v.
- Average over  $y$ :

$$E[Y] = \sum_y y p_Y(y)$$

- Average over  $x$ :

### Theorem (Law of the unconscious statistician (LOTUS))

$$E[Y] = E[g(X)] = \sum_x g(x) p_X(x)$$

- $\sum_y \sum_{x:g(x)=y} g(x) p_X(x) = \sum_y \sum_{x:g(x)=y} y p_X(x) = \sum_y y p_Y(y) = E[Y].$
- $E[X^2] = \sum_x x^2 p_X(x).$
- **Caution:** In general,  $E[g(X)] \neq g(E[X]).$

# Linearity of Expectation

## Theorem

$$E[aX + b] = aE[X] + b$$

Example:  $X = \text{salary}$   $E[X] = \text{average salary}$ .

$Y = \text{new salary} = 2X + 100$   $E[Y] = E[2X + 100] = 2E[X] + 100$ .

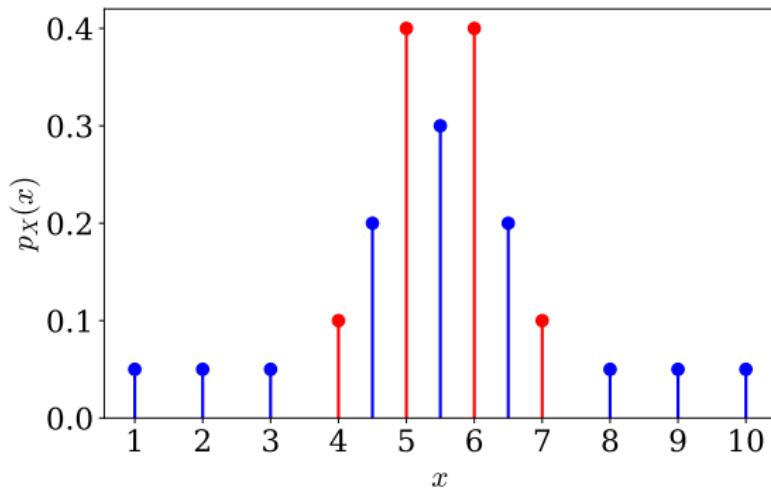
## Proof

Based on the expected value rule:  $g(x) = ax + b$ ;  $Y = g(X)$

$$\begin{aligned} E[Y] &= \sum_x g(x)p_X(x) \\ &= \sum_x (ax + b)p_X(x) \\ &= a \sum_x xp_X(x) + b \sum_x p_X(x) \\ &= aE[X] + b \end{aligned}$$

# Variance

- Variance is a measure of the spread of a random variable about its mean and also a measure of uncertainty.



- R.v.  $X$  with  $\mu = E[X]$ . Average distance from the mean?

$$E[X - \mu] = E[X] - \mu = \mu - \mu = 0$$

# Variance

- Variance is a measure of the spread of a random variable about its mean and also a measure of uncertainty.
- R.v.  $X$  with  $\mu = E[X]$ . Average distance from the mean?

$$E[X - \mu] = E[X] - \mu = \mu - \mu = 0$$

- Average of the squared distance from the mean.

## Definition (Variance)

The variance of a random variable  $X$  on a discrete sample space  $\Omega$  is defined as

$$Var(X) = \sigma^2 = E_P[(X - \mu)^2],$$

where  $P$  is a probability measure on  $\Omega$ .

# Variance

- $\text{Var}(X) = \sigma^2 = \text{E}[(X - \mu)^2]$
- $g(x) = (x - \mu)^2$
- To calculate, use the expected value rule,  $\text{E}[g(X)] = \sum_x g(x)p_X(x)$

$$\text{Var}(X) = \text{E}[g(X)] = \sum_x (x - \mu)^2 p_X(x)$$

- Variance is non-negative:  $\text{Var}(X) = \sigma^2 \geq 0$ .
- $\text{Var}(X) = 0$  iff  $X$  is deterministic.

## Definition (Standard Deviation)

The **standard deviation** of a random variable  $X$  is defined as

$$SD(X) = \sigma_X = \sqrt{\text{Var}(X)}$$

# Properties of the variance

## Theorem

For a random variable  $X$  and real numbers  $a$  and  $b$ ,

$$\text{Var}(aX + b) = a^2 \text{Var}(X)$$

- Notation  $\mu = E[X]$
- Let  $Y = X + b$ ,  $\gamma = E[Y] = \mu + b$ .

$$\text{Var}(Y) = E[(Y - \gamma)^2] = E[(X + b - (\mu + b))^2] = E[(X - \mu)^2] = \text{Var}(X)$$

- Let  $Y = aX$ ,  $\gamma = E[Y] = a\mu$

$$\begin{aligned}\text{Var}(Y) &= E[(aX - a\mu)^2] = E[a^2(X - \mu)^2] \\ &= a^2 E[(X - \mu)^2] = a^2 \text{Var}(X)\end{aligned}$$

# Properties of the variance

## Computational formula for the variance

$$Var(X) = E(X^2) - [E(X)]^2$$

$$\begin{aligned}Var(X) &= E[(X - \mu)^2] \\&= E[X^2 - 2\mu X + \mu^2] \\&= E[X^2] - 2\mu E[X] + \mu^2 \\&= E[X^2] - (E[X])^2\end{aligned}$$

# Independence and Expectation

- In general:  $E[g(X, Y)] \neq g(E[X], E[Y])$
- Exceptions:

$$E[aX + b] = aE[X] + b \quad E[X + Y + Z] = E[X] + E[Y] + E[Z]$$

## Theorem

If  $X, Y$  are independent:  $E[X, Y] = E[X]E[Y]$ ,

$g(X)$  and  $h(Y)$  are also independent:  $E[g(X), h(Y)] = E[g(X)]E[h(Y)]$

# Independence and Variances

- Always true:  $\text{Var}(aX) = a^2\text{Var}(X)$        $\text{Var}(X + a) = \text{Var}(X)$
- In general:  $\text{Var}(X + Y) \neq \text{Var}(X) + \text{Var}(Y)$
- However

## Theorem

If  $X, Y$  are independent:  $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y)$

## Proof.

Assume  $E[X] = E[Y] = 0$        $E[XY] = E[X]E[Y] = 0$ .

$$\begin{aligned}\text{Var}(X + Y) &= E[(X + Y)^2] = E[X^2 + 2XY + Y^2] \\ &= E[X^2] + 2E[XY] + E[Y^2] = \text{Var}(X) + \text{Var}(Y)\end{aligned}$$



# Table of Contents

- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

# Bernoulli Random Variables

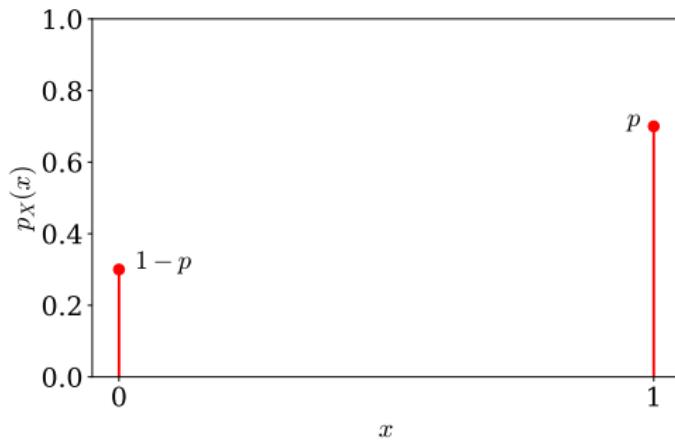
- A Bernoulli r.v.  $X$  takes two possible values, usually 0 and 1, modeling random experiments that have two possible outcomes (e.g., “success” and “failure”).
  - e.g., tossing a coin. The outcome is either Head or Tail.
  - e.g., taking an exam. The result is either Pass or Fail.
  - e.g., classifying images. An image is either Cat or Non-cat.

# Bernoulli Random Variables

## Definition

A random variable  $X$  is a Bernoulli random variable with parameter  $p \in [0, 1]$ , written as  $X \sim \text{Bernoulli}(p)$  if its PMF is given by

$$P_X(x) = \begin{cases} p, & \text{for } x = 1 \\ 1 - p, & \text{for } x = 0. \end{cases}$$



# Bernoulli & Indicator Random Variables

- A Bernoulli r.v.  $X$  with parameter  $p \in [0, 1]$  can also be described as

$$X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

- A Bernoulli r.v. is associated with a certain event  $A$ . If event  $A$  occurs, then  $X = 1$ ; otherwise,  $X = 0$ .
- Bernoulli r.v. is also called the indicator random variable of an event.

## Definition

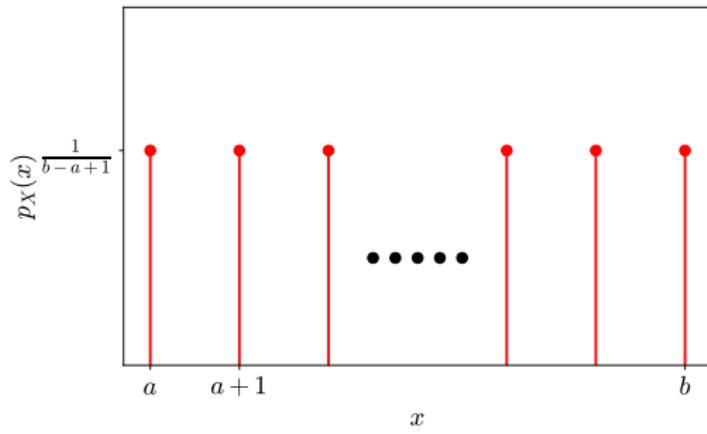
The indicator random variable of an event  $A$  is defined by

$$I_A = \begin{cases} 1 & \text{if the event } A \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

The indicator r.v. for an event  $A$  has Bernoulli distribution with parameter  $p = P(I_A = 1) = P_{I_A}(1) = P(A)$ . We can write  $I_A \sim \text{Bernoulli}(P(A))$ .

# Discrete Uniform Random Variables

- Parameters: integer  $a, b$ ;  $a \leq b$
- Experiment: Pick one of  $a, a + 1, \dots, b$  at random; all equally likely.
- Sample space:  $\{a, a + 1, \dots, b\}$
- Random variable  $X$ :  $X(\omega) = \omega$
- $b - a + 1$  possible values,  $P_X(x) = 1/(b - a + 1)$  for each value.
- Model of: complete ignorance.



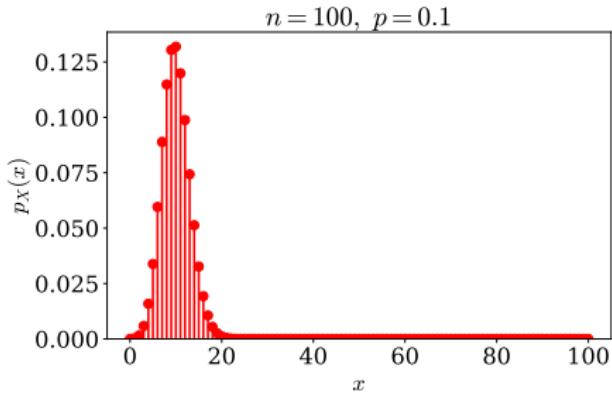
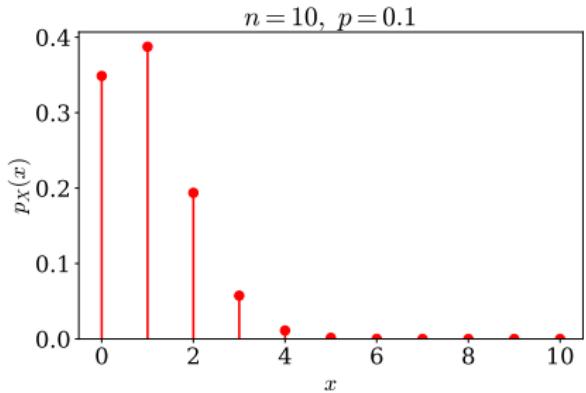
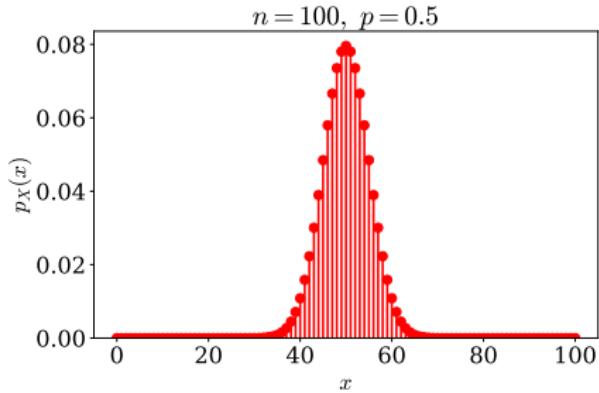
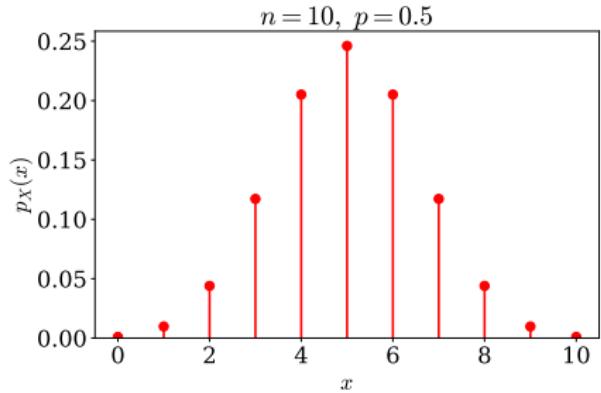
# Binomial Random Variables

- Parameters: Probability  $p \in [0, 1]$ , positive integer  $n$ .
- Experiment: e.g.,  $n$  independent tosses of a coin with  $P(\text{Head}) = p$
- Sample space: Set of sequences of  $H$  and  $T$  of length  $n$
- Random variable  $X$  : number of Heads observed.
- Model of: number of successes in a given number of independent trials.

## Examples

$$\begin{aligned}P_X(2) &= P(X = 2) \\&= P(\text{HHT}) + P(\text{HTH}) + P(\text{THH}) \\&= 3p^2(1 - p) \\&= \binom{3}{2}p^2(1 - p)\end{aligned}$$

# Binomial Random Variables



# Binomial Random Variables

- Let  $\Omega = \{0, 1, 2, \dots, n\}$  be the sample space and let  $X$  be the number of successes in  $n$  independent trials where  $p$  is the probability of success in a single Bernoulli trial.
- The probability measure  $P$  is called the binomial distribution if

$$P_X(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad \text{for } k = 0, 1, \dots, n$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

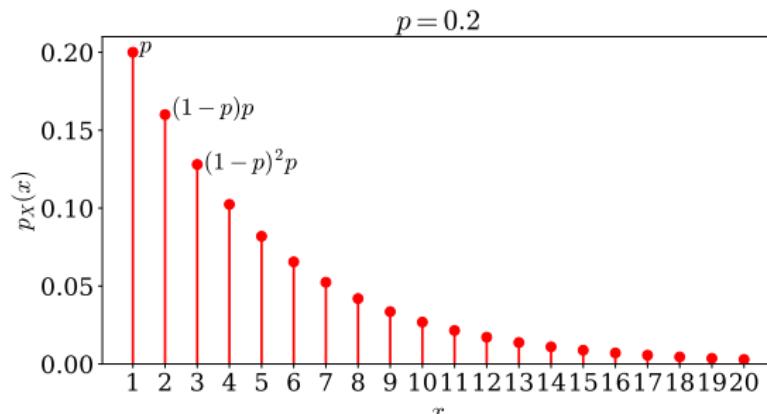
- Then

$$E[X] = np \quad \text{and} \quad Var(X) = np(1-p)$$

# Geometric Random Variables

- Parameters: Probability  $p \in (0, 1]$ .
- Experiment: infinitely many independent tosses of a coin;  
 $P(\text{Head}) = p$ .
- Sample space: Set of infinite sequences of H and T.
- Random variable  $X$  : number of tosses until the first Head.
- Model of: waiting times, number of trials until a success.

$$P_X(k) = P(X = k) = P(\underbrace{T \dots T}_{k-1} H) = (1 - p)^{k-1} p$$



# Geometric Random Variables

- Let  $\Omega = \{1, 2, 3, \dots\}$  be the sample space and  $X$  be the number of independent trials to achieve the first success.
- Let  $p$  stand for the probability of a success in a single trial.
- The probability measure  $P$  is called the geometric distribution if

$$P_X(k) = (1 - p)^{k-1} p \quad \text{for } k = 1, 2, 3 \dots$$

- Then

$$E[X] = \frac{1}{p} \quad \text{and} \quad Var(X) = \frac{1-p}{p^2}$$

- $P(\text{no Heads}) \leq P(\underbrace{T \dots T}_k) = (1 - p)^k$ . As  $k \rightarrow \infty$ ,  $(1 - p)^k \rightarrow 0$

# Table of Contents

- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

# Continuous Random Variables

## Definition

A random variable  $X$  on the sample space  $\Omega$  is said to have a continuous distribution if there exists a real-valued function  $f$  such that

$$f(x) \geq 0,$$

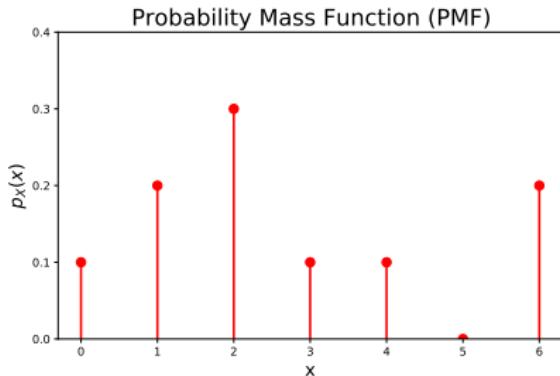
$$\int_{-\infty}^{\infty} f(x) \, dx = 1,$$

and for all real numbers  $a < b$ :

$$P(a \leq X \leq b) = \int_a^b f(x) \, dx.$$

Then  $f : \mathbb{R} \mapsto \mathbb{R}_+$  is called the **probability density function (PDF)** of a **continuous random variable  $X$** .

# Probability Density Function (PDF)

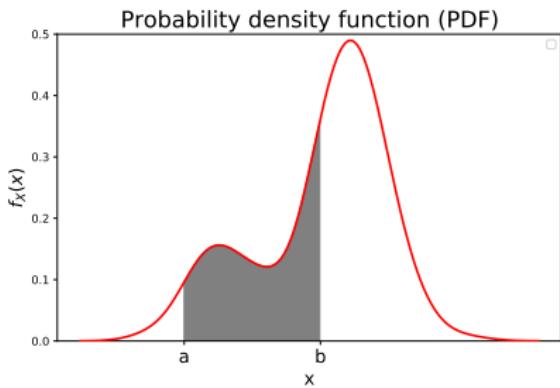
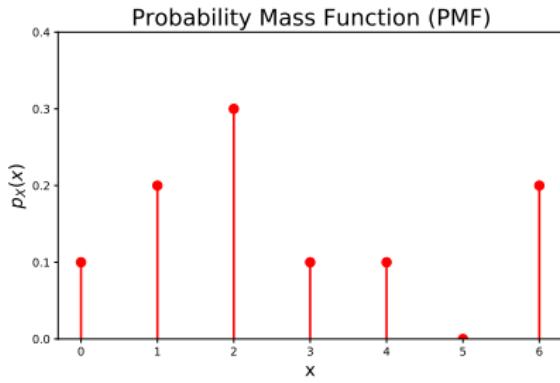


$$P(a \leq X \leq b) = \sum_{x: a \leq x \leq b} p_X(x)$$

$$p_X(x) \geq 0$$

$$\sum_x p_X(x) = 1$$

# Probability Density Function (PDF)



$$P(a \leq X \leq b) = \sum_{x: a \leq x \leq b} p_X(x)$$

$$p_X(x) \geq 0$$

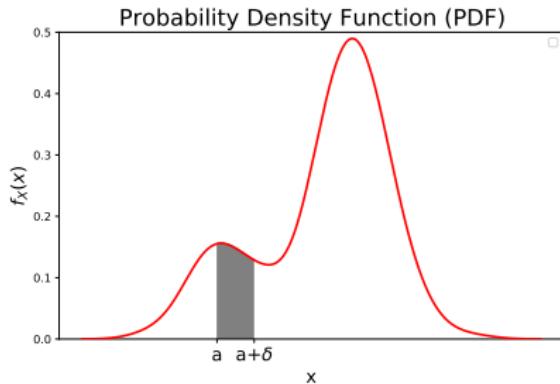
$$\sum_x p_X(x) = 1$$

$$P(a \leq X \leq b) = \int_a^b f_X(x) \, dx$$

$$f_X(x) \geq 0$$

$$\int_{-\infty}^{\infty} f_X(x) \, dx = 1$$

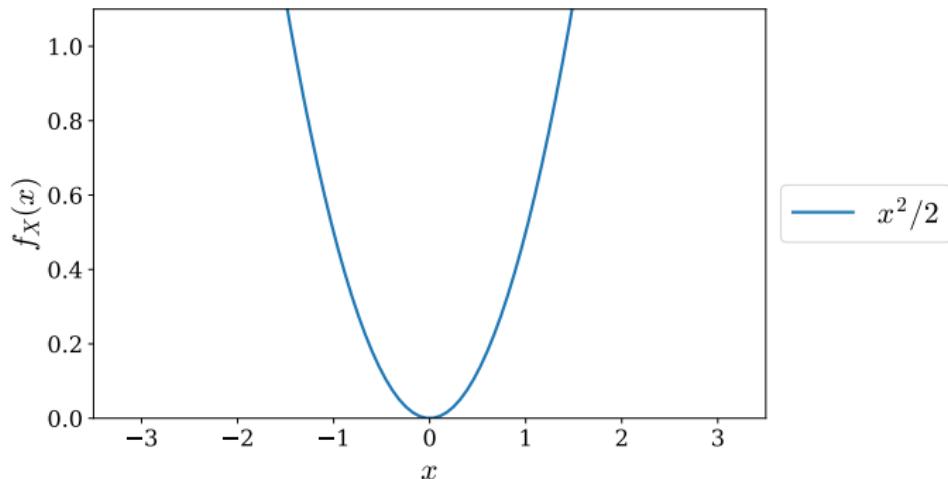
# Probability Density Function (PDF)



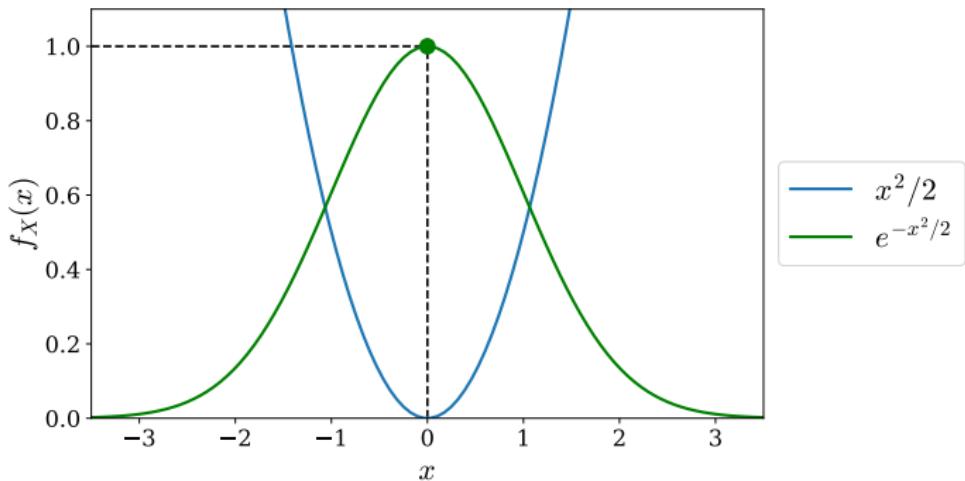
$$P(a \leq X \leq b) = \int_a^b f_X(x) \, dx$$

- $\delta > 0$ , small
- $P(a \leq X \leq a + \delta) \approx f_X(a).\delta$
- $P(X = a) = 0$
- Just like, a single point has zero length.
- But, a set of lots of points has a positive length.

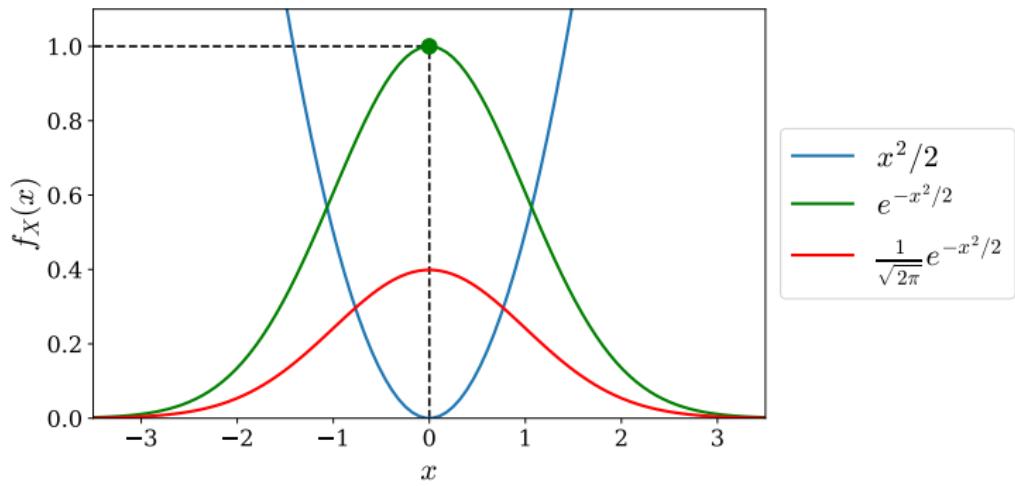
# Standard Normal (Gaussian) Random Variable $N(0, 1)$



# Standard Normal (Gaussian) Random Variable $N(0, 1)$



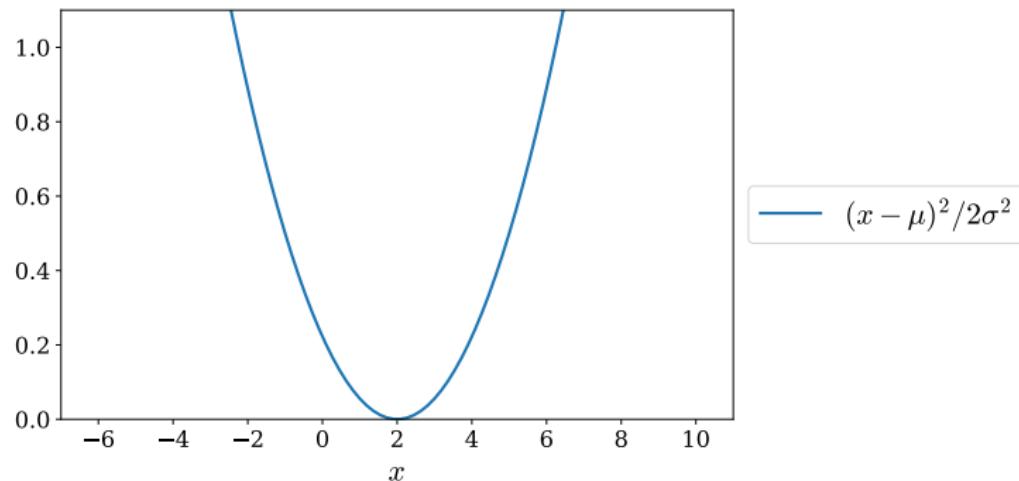
# Standard Normal (Gaussian) Random Variable $N(0, 1)$



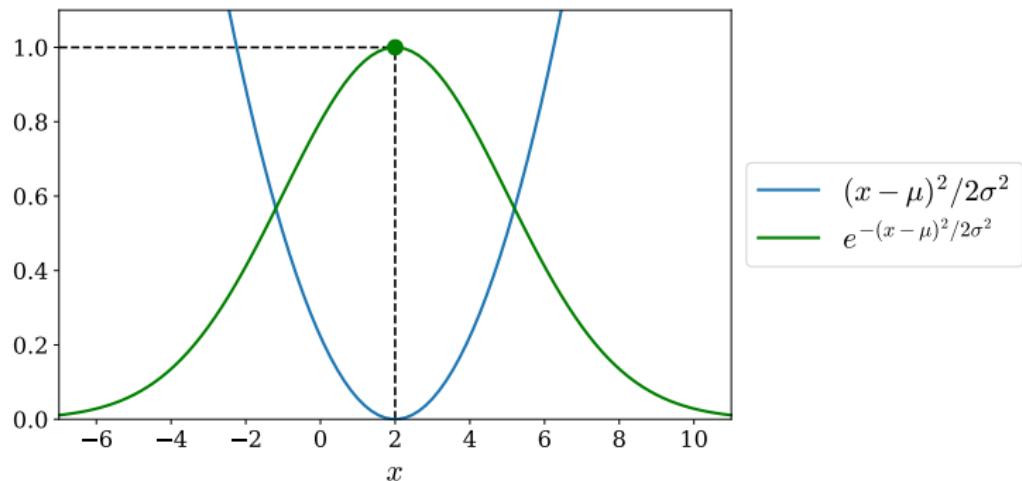
$$\int_{-\infty}^{\infty} e^{-x^2/2} dx = \sqrt{2\pi}$$

$$f_X(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$$

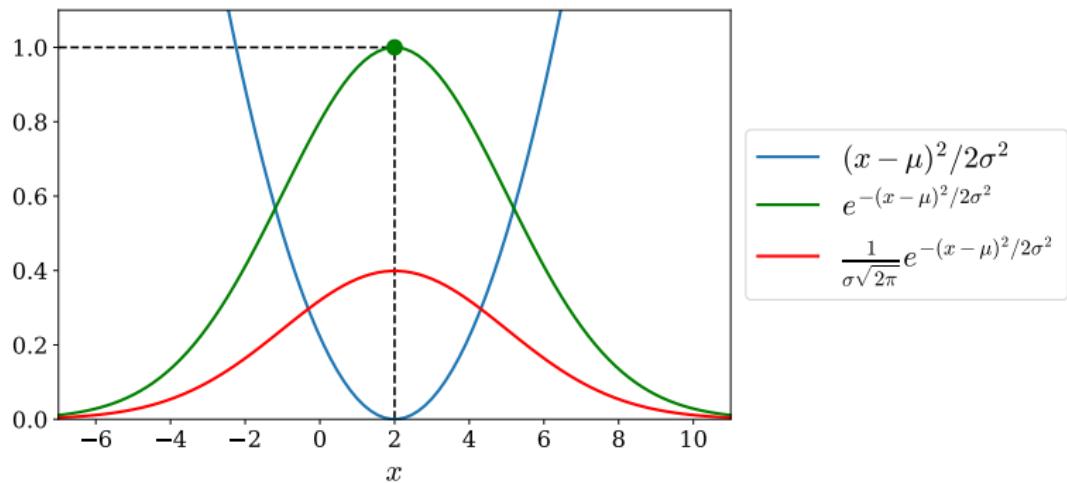
# General Normal (Gaussian) Random Variable $N(\mu, \sigma^2)$



# General Normal (Gaussian) Random Variable $N(\mu, \sigma^2)$



# General Normal (Gaussian) Random Variable $N(\mu, \sigma^2)$

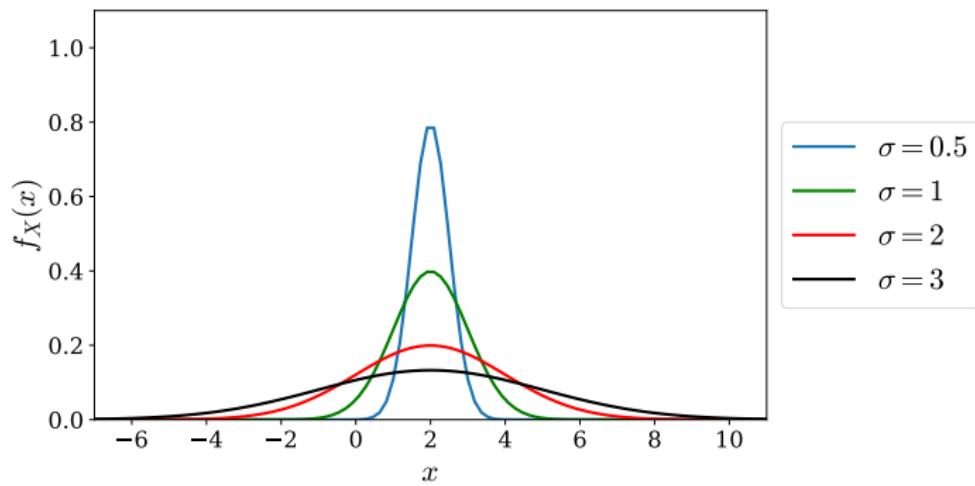


$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-(x-\mu)^2/2\sigma^2}$$

$$\mathbb{E}[X] = \mu$$

$$\text{Var}(X) = \sigma^2$$

# General Normal (Gaussian) Random Variable $N(\mu, \sigma^2)$



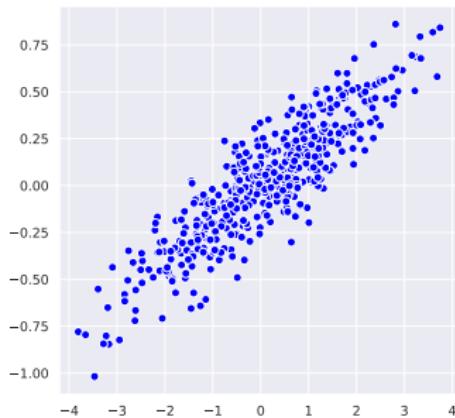
- Smaller  $\sigma$ , narrower PDF.
- Let  $Y = aX + b \quad N \sim N(\mu, \sigma^2)$
- Then,  $E[Y] = aE[X] + b \quad \text{Var}(Y) = a^2\sigma^2$  (always true)
- But also,  $Y \sim N(a\mu + b, a^2\sigma^2)$

# Table of Contents

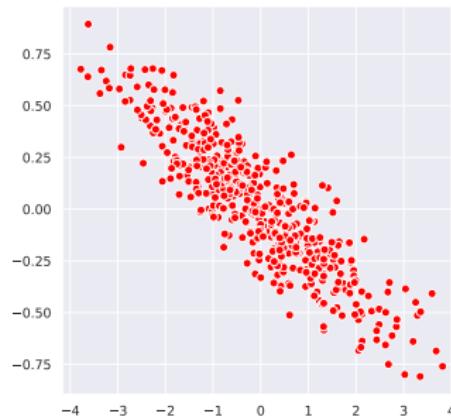
- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

# Covariance

- Consider zero-mean, discrete random variables  $X$  and  $Y$
- If they are independent,  $E[XY] = E[X]E[Y]=0$ .
- But if their joint PDF is as follows.



$$E[XY] > 0$$



$$E[XY] < 0$$

# Covariance

- Definition for general case:

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

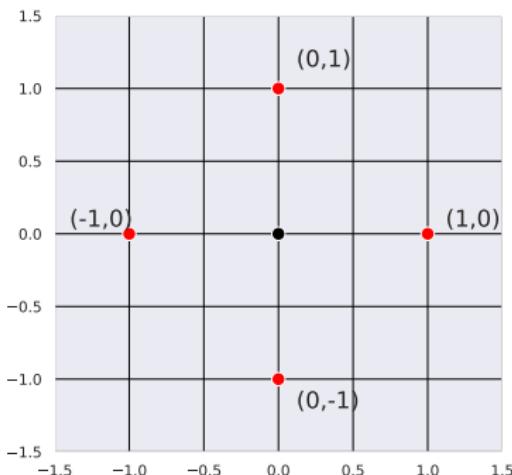
- The deviation of  $X$  from its mean value  $X - E[X]$ .
- The deviation of  $Y$  from its mean value  $Y - E[Y]$ .
- Whether these two deviations tend to have the same sign or not.
- In general, the covariance tells us whether two random variables tend to move together, both being high or both being low, on average.
- Whether they move in same direction or not.
- If the covariance is positive, it indicates that whenever the quantity  $X - E[X]$  is positive ( $X$  is above its mean), the deviation of  $Y$  from its mean will also tend to be positive  $Y - E[Y]$ .

# Covariance

- If  $X$  and  $Y$  are independent, then

$$\begin{aligned}\text{Cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\ &= E[(X - E[X])]E[(Y - E[Y])] \\ &= 0\end{aligned}$$

- However, the inverse is not true.



- $E[X] = 0, E[Y] = 0$
- $XY = 0$
- $\text{Cov}(X, Y) = 0$
- $X = 1 \implies Y = 0$ . Knowing the value of  $X$  tells a lot about  $Y$ . Therefore,  $X$  and  $Y$  are dependent.

# Covariance properties



$$\begin{aligned}\text{Cov}(X, X) &= E[(X - E[X])^2] = \text{Var}(X) \\ &= E[X^2] - (E[X])^2\end{aligned}$$

## Covariance properties



$$\begin{aligned}\text{Cov}(X, X) &= E[(X - E[X])^2] = \text{Var}(X) \\ &= E[X^2] - (E[X])^2\end{aligned}$$



$$\begin{aligned}\text{Cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\ &= E[XY - XE[Y] - E[X]Y + E[X]E[Y]] \\ &= E[XY] - E[XE[Y]] - E[E[X]Y] + E[E[X]E[Y]] \\ &= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] \\ &= E[XY] - E[X]E[Y]\end{aligned}$$

# Covariance properties



$$\begin{aligned}\text{Cov}(X, X) &= E[(X - E[X])^2] = \text{Var}(X) \\ &= E[X^2] - (E[X])^2\end{aligned}$$



$$\begin{aligned}\text{Cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\ &= E[XY - XE[Y] - E[X]Y + E[X]E[Y]] \\ &= E[XY] - E[XE[Y]] - E[E[X]Y] + E[E[X]E[Y]] \\ &= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] \\ &= E[XY] - E[X]E[Y]\end{aligned}$$

- Assume 0 means

$$\text{Cov}(aX + b, Y) = E[(aX + b)Y] = aE[XY] + bE[Y] = a\text{Cov}(X, Y)$$

# Joint distribution for multiple random variables

- The **covariance** between two random variables  $X$  and  $Y$  measures the degree to which  $X$  and  $Y$  are related

$$\text{Cov}[X, Y] \triangleq E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

- If  $\mathbf{x}$  is a  $D$ -dimensional random vector, its **covariance matrix** is defined to be the following symmetric, positive semi definite matrix

$$\text{Cov}[\mathbf{x}] \triangleq E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T] \triangleq \Sigma$$

$$= \begin{bmatrix} V[X_1] & \text{Cov}[X_1, X_2] & \dots & \text{Cov}[X_1, X_D] \\ \text{Cov}[X_2, X_1] & V[X_2] & \dots & \text{Cov}[X_2, X_D] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[X_D, X_1] & \text{Cov}[X_D, X_2] & \dots & V[X_D] \end{bmatrix}$$

from which we get

$$E[\mathbf{x}\mathbf{x}^T] = \Sigma + \mu\mu^T$$

# Multivariate Gaussian (normal) distribution

- The **multivariate normal (MVN)** density is defined

$$\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right]$$

where  $\boldsymbol{\mu} = E[\mathbf{y}] \in \mathbb{R}^D$ ,  $\boldsymbol{\Sigma} = \text{Cov}[\mathbf{y}]$  is the  $D \times D$  **covariance matrix**

$$\text{Cov}[\mathbf{y}] \triangleq E[(\mathbf{y} - E[\mathbf{y}])(\mathbf{y} - E[\mathbf{y}])^\top] \triangleq \boldsymbol{\Sigma}$$

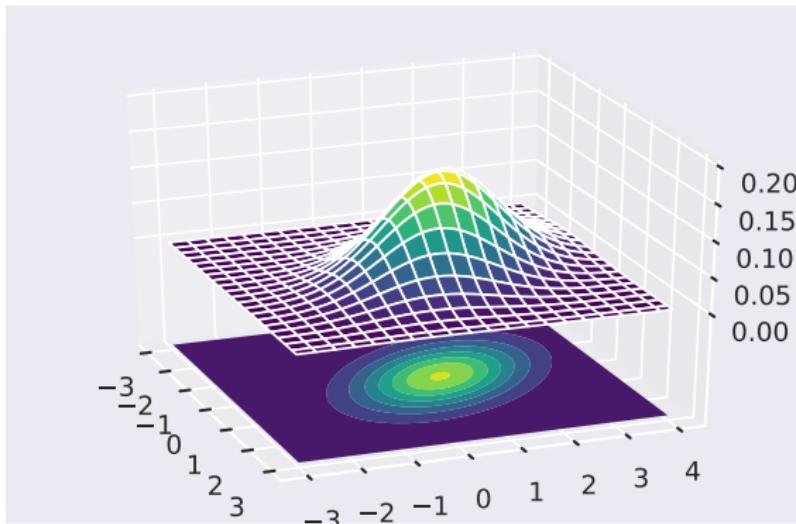
$$= \begin{bmatrix} V[Y_1] & \text{Cov}[Y_1, Y_2] & \dots & \text{Cov}[Y_1, Y_D] \\ \text{Cov}[Y_2, Y_1] & V[Y_2] & \dots & \text{Cov}[Y_2, Y_D] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[Y_D, Y_1] & \text{Cov}[Y_D, Y_2] & \dots & V[X_D] \end{bmatrix}$$

- A **full covariance matrix** had  $D(D + 1)/2$  parameters.

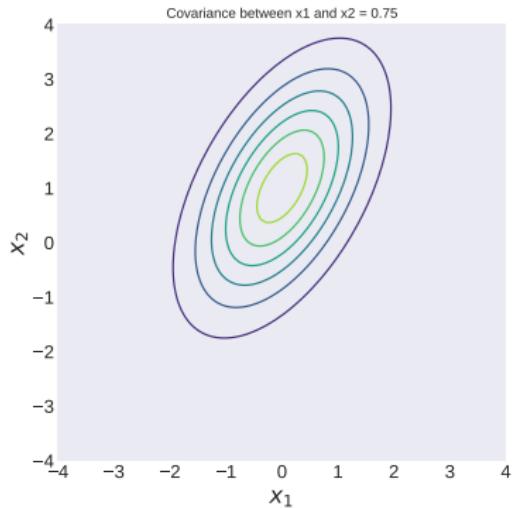
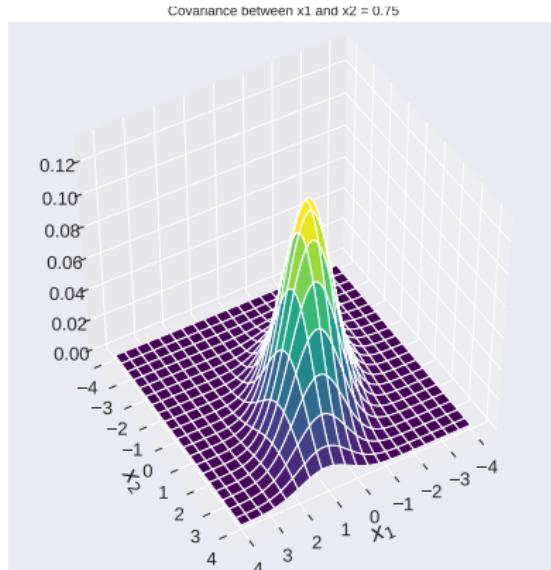
# Multivariate Gaussian (normal) distribution

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right]$$

where  $\boldsymbol{\mu} = E[\mathbf{y}] \in \mathbb{R}^D$ ,  $\boldsymbol{\Sigma} = \text{Cov}[\mathbf{y}]$  is the  $D \times D$  **covariance matrix**.



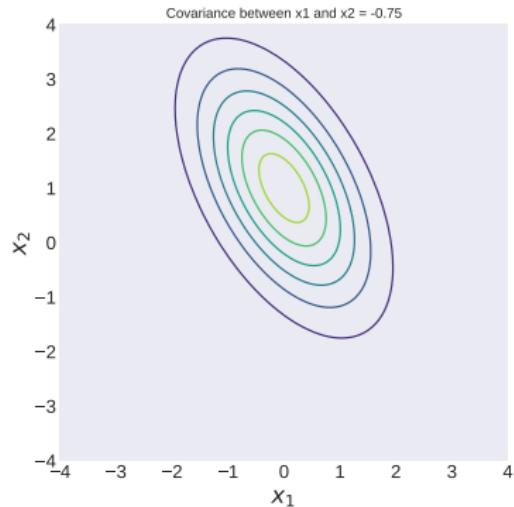
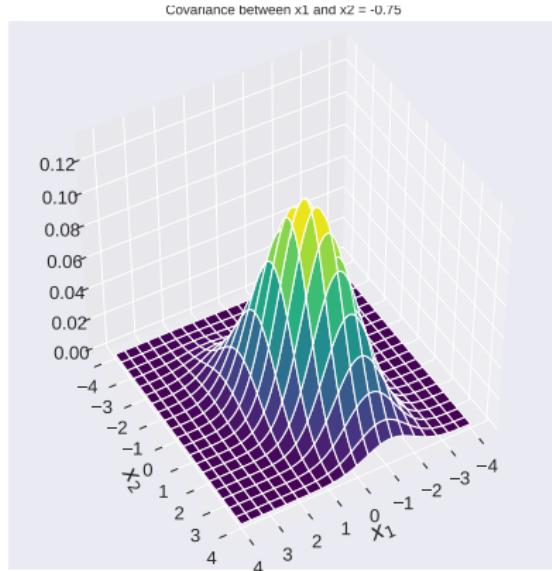
# Multivariate Gaussian - Full Covariance Matrix



$$\mu = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 2 \end{bmatrix}$$

A **full covariance matrix** had  $D(D + 1)/2$  parameters.

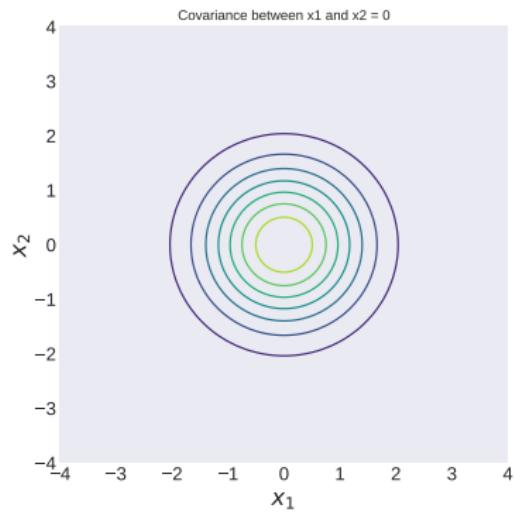
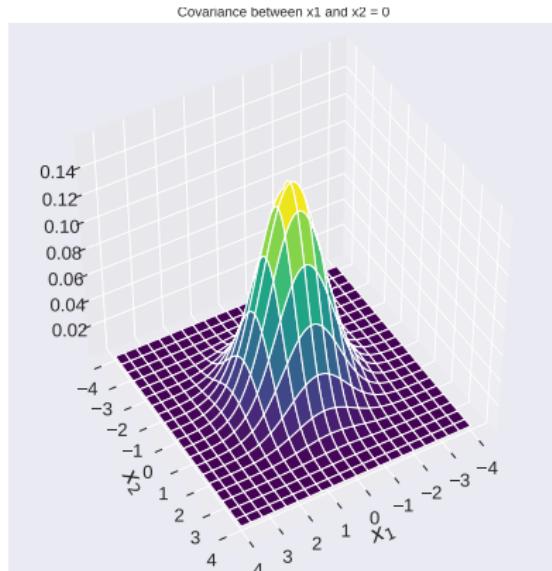
# Multivariate Gaussian - Full Covariance Matrix



$$\mu = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.75 \\ -0.75 & 2 \end{bmatrix}$$

A **full covariance matrix** had  $D(D + 1)/2$  parameters.

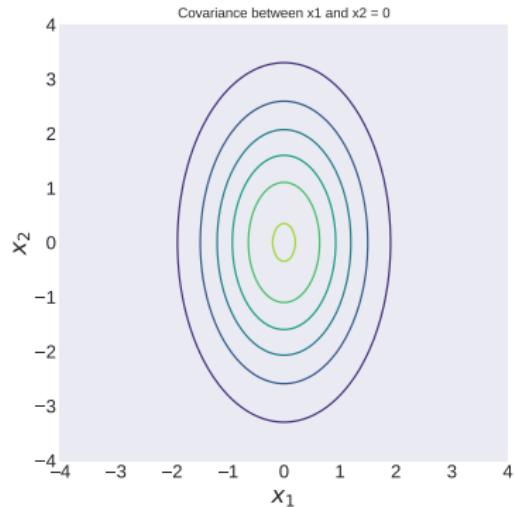
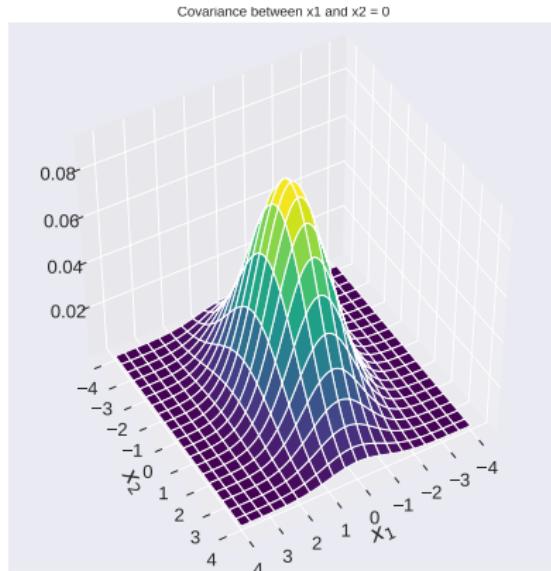
# Standard Multivariate Gaussian - Spherical Cov. Matrix



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

A spherical (isotropic) covariance matrix  $\Sigma = \sigma^2 I_D$  has one free parameter  $\sigma^2$ .

# Uncorrelated Multivariate Gaussian - Diagonal Cov. Matrix



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

A diagonal covariance matrix has  $D$  parameters.

## Bivariate Gaussian (normal) distribution

- In 2D, the MVN is known as the **bivariate Gaussian** distribution. Its PDF can be represented as  $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , where  $\mathbf{y} \in \mathbb{R}^2$ ,  $\boldsymbol{\mu} \in \mathbb{R}^2$  and

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

where  $\rho$  is the **correlation coefficient**

$$\text{corr}[Y_1, Y_2] \triangleq \frac{\text{Cov}[Y_1, Y_2]}{\sqrt{V[Y_1]V[Y_2]}} = \frac{\sigma_{12}^2}{\sigma_1\sigma_2}$$

# Correlation coefficient

- Dimensionless version of covariance:

$$\rho(X, Y) = \text{corr}[X, Y] \triangleq E\left[\frac{(X - E[X])}{\sigma_X} \frac{(Y - E[Y])}{\sigma_Y}\right] = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

# Correlation coefficient

- Dimensionless version of covariance:

$$\rho(X, Y) = \text{corr}[X, Y] \triangleq E\left[\frac{(X - E[X])}{\sigma_X} \frac{(Y - E[Y])}{\sigma_Y}\right] = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

- $1 \leq \rho \leq 1$ : measure of the degree of “association” between  $X$  and  $Y$ .

# Correlation coefficient

- Dimensionless version of covariance:

$$\rho(X, Y) = \text{corr}[X, Y] \triangleq E\left[\frac{(X - E[X])}{\sigma_X} \frac{(Y - E[Y])}{\sigma_Y}\right] = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

- $-1 \leq \rho \leq 1$ : measure of the degree of “association” between  $X$  and  $Y$ .
- Independent  $\implies \rho = 0$ : “uncorrelated” (the converse is not true).

# Correlation coefficient

- Dimensionless version of covariance:

$$\rho(X, Y) = \text{corr}[X, Y] \triangleq E\left[\frac{(X - E[X])}{\sigma_X} \frac{(Y - E[Y])}{\sigma_Y}\right] = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

- $1 \leq \rho \leq 1$ : measure of the degree of “association” between  $X$  and  $Y$ .
- Independent  $\implies \rho = 0$ : “uncorrelated” (the converse is not true).
- $\rho(X, X) = \frac{\text{Var}(X)}{\sigma_X^2} = 1$ .

# Correlation coefficient

- Dimensionless version of covariance:

$$\rho(X, Y) = \text{corr}[X, Y] \triangleq E\left[\frac{(X - E[X])}{\sigma_X} \frac{(Y - E[Y])}{\sigma_Y}\right] = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

- $1 \leq \rho \leq 1$ : measure of the degree of “association” between  $X$  and  $Y$ .
- Independent  $\implies \rho = 0$ : “uncorrelated” (the converse is not true).
- $\rho(X, X) = \frac{\text{Var}(X)}{\sigma_X^2} = 1$ .
- $|\rho| = 1 \iff (X - E[X]) = c(Y - E[Y])$ : linearly related, deterministic relation between the two rv.

# Correlation coefficient

- Dimensionless version of covariance:

$$\rho(X, Y) = \text{corr}[X, Y] \triangleq E\left[\frac{(X - E[X])}{\sigma_X} \frac{(Y - E[Y])}{\sigma_Y}\right] = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

- $1 \leq \rho \leq 1$ : measure of the degree of “association” between  $X$  and  $Y$ .
- Independent  $\implies \rho = 0$ : “uncorrelated” (the converse is not true).
- $\rho(X, X) = \frac{\text{Var}(X)}{\sigma_X^2} = 1$ .
- $|\rho| = 1 \iff (X - E[X]) = c(Y - E[Y])$ : linearly related, deterministic relation between the two rv.
- We have  $\text{Cov}(aX + b, Y) = a\text{Cov}(X, Y)$ . Therefore,

$$\rho(aX + b, Y) = \frac{a\text{Cov}(X, Y)}{|a|\sigma_X \sigma_Y} = \text{sign}(a) \cdot \rho(X, Y)$$

$\implies$  Changing the unit of  $X$  does not change correlation coefficient between  $X$  and  $Y$ .

# Table of Contents

- 1 Probability models and axioms
- 2 Discrete Random Variables
- 3 Examples of Discrete Probability Distributions
- 4 Continuous Random Variables
- 5 Covariance
- 6 Intro to Maximum Likelihood Estimation (MLE)

## Example

- A bag contains 3 balls, each ball is either red or blue.
- The number of blue balls  $\theta$  can be 0, 1, 2, 3.

## Example

- A bag contains 3 balls, each ball is either red or blue.
- The number of blue balls  $\theta$  can be 0, 1, 2, 3.
- Choose 4 balls randomly **with replacement**.

## Example

- A bag contains 3 balls, each ball is either red or blue.
- The number of blue balls  $\theta$  can be 0, 1, 2, 3.
- Choose 4 balls randomly **with replacement**.
- Random variables  $X_1, X_2, X_3, X_4$  are defined as

$$X_i = \begin{cases} 1, & \text{if the } i\text{-th chosen ball is blue} \\ 0, & \text{if the } i\text{-th chosen ball is red} \end{cases}$$

## Example

- A bag contains 3 balls, each ball is either red or blue.
- The number of blue balls  $\theta$  can be 0, 1, 2, 3.
- Choose 4 balls randomly **with replacement**.
- Random variables  $X_1, X_2, X_3, X_4$  are defined as

$$X_i = \begin{cases} 1, & \text{if the } i\text{-th chosen ball is blue} \\ 0, & \text{if the } i\text{-th chosen ball is red} \end{cases}$$

- After doing the experiment, the following values for  $X_i$ 's are observed:  $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$ .

## Example

- A bag contains 3 balls, each ball is either red or blue.
- The number of blue balls  $\theta$  can be 0, 1, 2, 3.
- Choose 4 balls randomly **with replacement**.
- Random variables  $X_1, X_2, X_3, X_4$  are defined as

$$X_i = \begin{cases} 1, & \text{if the } i\text{-th chosen ball is blue} \\ 0, & \text{if the } i\text{-th chosen ball is red} \end{cases}$$

- After doing the experiment, the following values for  $X_i$ 's are observed:  $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$ .
- Note that  $X_i$ 's are i.i.d. (independent and identically distributed) and  $X_i \sim \text{Bernoulli}(\frac{\theta}{3})$ . For which value of  $\theta$  is the probability of the observed sample is the largest?

## Example

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3}, & \text{for } x = 1 \\ 1 - \frac{\theta}{3}, & \text{for } x = 0 \end{cases}$$

## Example

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3}, & \text{for } x = 1 \\ 1 - \frac{\theta}{3}, & \text{for } x = 0 \end{cases}$$

$X_i$ 's are independent, the joint PMF of  $X_1, X_2, X_3, X_4$  can be written

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

$$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) = \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} = \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right)$$

## Example

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3}, & \text{for } x = 1 \\ 1 - \frac{\theta}{3}, & \text{for } x = 0 \end{cases}$$

$X_i$ 's are independent, the joint PMF of  $X_1, X_2, X_3, X_4$  can be written

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

$$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) = \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} = \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right)$$

$\theta$	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

## Example

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3}, & \text{for } x = 1 \\ 1 - \frac{\theta}{3}, & \text{for } x = 0 \end{cases}$$

$X_i$ 's are independent, the joint PMF of  $X_1, X_2, X_3, X_4$  can be written

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

$$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) = \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} = \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right)$$

$\theta$	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

The observed data is most likely to occur for  $\theta = 2$ .

We may choose  $\hat{\theta} = 2$  as our estimate of  $\theta$ .

# Maximum Likelihood Estimation (MLE)

## Definition

Let  $X_1, X_2, \dots, X_n$  be a random sample from a distribution with a parameter  $\theta$ .

Given that we have observed  $X_1 = x_1, X_2 = x_2, \dots, X_n = x_n$ , a maximum likelihood estimate of  $\theta$ , denoted as  $\hat{\theta}_{ML}$ , is a value of  $\theta$  that maximizes the likelihood function

$$L(x_1, x_2, \dots, x_n; \theta)$$

A maximum likelihood estimator (MLE) of the parameter  $\theta$ , denoted as  $\hat{\Theta}_{ML}$ , is a random variable  $\hat{\Theta}_{ML} = \hat{\Theta}(X_1, X_2, \dots, X_n)$  whose values  $X_1 = x_1, X_2 = x_2, \dots, X_n = x_n$  is given by  $\hat{\theta}_{ML}$ .

# Maximum Likelihood Estimation (MLE) Regularizations

Faculty of Computer Science  
University of Information Technology (UIT)  
Vietnam National University - Ho Chi Minh City (VNU-HCM)

Math for Computer Science, Fall 2022

# References

The contents of this document are taken mainly from the follow sources:

- Kevin P. Murphy. Probabilistic Machine Learning: An Introduction.<sup>1</sup>

---

<sup>1</sup><https://probml.github.io/pml-book/book1.html>

# Table of Contents

- 1 Introduction
- 2 Maximum Likelihood Estimation
- 3 Examples
- 4 MLE for Linear Regression
- 5 Regularization

# Table of Contents

1 Introduction

2 Maximum Likelihood Estimation

3 Examples

4 MLE for Linear Regression

5 Regularization

# Introduction

- The process of estimating  $\theta$  from  $\mathcal{D}$  is called **model fitting**, or **training**, is at the heart of machine learning.
- There are many methods for estimating  $\theta$ , and they involve an optimization problem of the form

$$\hat{\theta} = \operatorname{argmin}_{\theta} \mathcal{L}(\theta)$$

where  $\mathcal{L}(\theta)$  is some kind of loss function or objective function.

- The process of quantifying uncertainty about an unknown quantity estimated from a finite sample of data is called **inference**.
- In deep learning, the term “inference” refers to “prediction”, namely computing

$$p(y|x, \hat{\theta})$$

# Table of Contents

- 1 Introduction
- 2 Maximum Likelihood Estimation
- 3 Examples
- 4 MLE for Linear Regression
- 5 Regularization

# Maximum Likelihood Estimation

- The most common approach to parameter estimation is to pick the parameters that assign the highest probability to the training data. This is called **maximum likelihood estimation** or **MLE**.

$$\hat{\theta}_{\text{mle}} = \underset{\theta}{\operatorname{argmax}} p(\mathcal{D}|\theta)$$

- We usually assume the training examples are “independent and identically distributed”, and are sampled from the same distribution (i.e., the **iid** assumption). The conditional likelihood becomes

$$p(\mathcal{D}|\theta) = p(y_1, y_2, \dots, y_N | x_1, x_2, \dots, x_N, \theta) = \prod_{n=1}^N p(y_n | x_n, \theta)$$

- We usually work with the **log likelihood**, which decomposes into a sum of terms, one per example.

$$\text{LL}(\theta) = \log p(\mathcal{D}|\theta) = \log \prod_{n=1}^N p(y_n | x_n, \theta) = \sum_{n=1}^N \log p(y_n | x_n, \theta)$$

# Maximum Likelihood Estimation

- The MLE is given by

$$\hat{\theta}_{\text{mle}} = \operatorname{argmax}_{\theta} \sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \theta)$$

- Because most optimization algorithms are designed to *minimize* cost functions, we redefine the objective function to be the conditional **negative log likelihood** or **NLL**:

$$\text{NLL}(\theta) = -\log p(\mathcal{D} | \theta) = -\sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \theta)$$

- Minimizing this will give the MLE.

$$\hat{\theta}_{\text{mle}} = \operatorname{argmin}_{\theta} -\sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \theta)$$

# Table of Contents

- 1 Introduction
- 2 Maximum Likelihood Estimation
- 3 Examples
- 4 MLE for Linear Regression
- 5 Regularization

# Bernoulli Random Variables

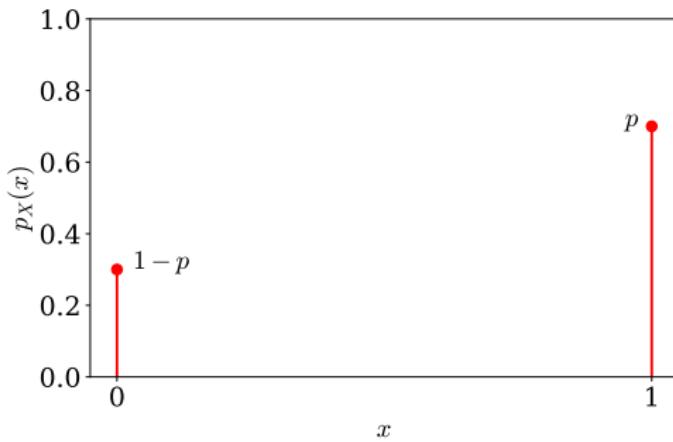
- A Bernoulli r.v.  $X$  takes two possible values, usually 0 and 1, modeling random experiments that have two possible outcomes (e.g., “success” and “failure”).
  - e.g., tossing a coin. The outcome is either Head or Tail.
  - e.g., taking an exam. The result is either Pass or Fail.
  - e.g., classifying images. An image is either Cat or Non-cat.

# Bernoulli Random Variables

## Definition

A random variable  $X$  is a Bernoulli random variable with parameter  $p \in [0, 1]$ , written as  $X \sim \text{Bernoulli}(p)$  if its PMF is given by

$$P_X(x) = \begin{cases} p, & \text{for } x = 1 \\ 1 - p, & \text{for } x = 0. \end{cases}$$



## Example

- A bag contains 3 balls, each ball is either red or blue.
- The number of blue balls  $\theta$  can be 0, 1, 2, 3.
- Choose 4 balls randomly **with replacement**.
- Random variables  $X_1, X_2, X_3, X_4$  are defined as

$$X_i = \begin{cases} 1, & \text{if the } i\text{-th chosen ball is blue} \\ 0, & \text{if the } i\text{-th chosen ball is red} \end{cases}$$

- After doing the experiment, the following values for  $X_i$ 's are observed:  $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$ .
- Note that  $X_i$ 's are i.i.d. (independent and identically distributed) and  $X_i \sim \text{Bernoulli}(\frac{\theta}{3})$ . For which value of  $\theta$  is the probability of the observed sample is the largest?

## Example

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3}, & \text{for } x = 1 \\ 1 - \frac{\theta}{3}, & \text{for } x = 0 \end{cases}$$

$X_i$ 's are independent, the joint PMF of  $X_1, X_2, X_3, X_4$  can be written

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

$$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) = \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} = \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right)$$

$\theta$	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

The observed data is most likely to occur for  $\theta = 2$ .

We may choose  $\hat{\theta} = 2$  as our estimate of  $\theta$ .

# MLE for the Bernoulli distribution

- Suppose  $Y$  is a random variable representing a coin toss.
- The event  $Y = 1$  corresponds to heads,  $Y = 0$  corresponds to tails.
- The probability distribution for this rv is the Bernoulli. The NLL for the Bernoulli distribution is

$$\begin{aligned}\text{NLL}(\theta) &= -\log \prod_{n=1}^N p(y_n|\theta) = -\log \prod_{n=1}^N \theta^{\mathbb{I}(y_n=1)}(1-\theta)^{\mathbb{I}(y_n=0)} \\ &= -\sum_{n=1}^N \mathbb{I}(y_n = 1) \log \theta + \mathbb{I}(y_n = 0) \log(1-\theta) \\ &= -[N_1 \log \theta + N_0 \log(1-\theta)]\end{aligned}$$

where  $N_1 = \sum_{n=1}^N \mathbb{I}(y_n = 1)$  is the number of heads, and  $N_0 = \sum_{n=1}^N \mathbb{I}(y_n = 0)$  is the number of tails.

- $N = N_0 + N_1$  is the **sample size**.

# MLE for the Bernoulli distribution

$$\text{NLL}(\theta) = -[N_1 \log \theta + N_0 \log(1 - \theta)]$$

- The derivative of the NLL is

$$\frac{d}{d\theta} \text{NLL}(\theta) = \frac{-N_1}{\theta} + \frac{N_0}{1 - \theta}$$

- The MLE can be found by solving  $\frac{d}{d\theta} \text{NLL}(\theta) = 0$ .
- The MLE is given by

$$\hat{\theta}_{\text{mle}} = \frac{N_1}{N_0 + N_1}$$

which is the **empirical** fraction of heads.

## MLE for the categorical distribution

- Suppose we roll a  $K$ -sided dice  $N$  times.
- Let  $Y_n \in \{1, \dots, K\}$  be the  $n$ -th outcome, where  $Y_n \sim \text{Cat}(\boldsymbol{\theta})$ .
- We want to estimate  $\boldsymbol{\theta}$  from the dataset  $\mathcal{D}\{y_n : n = 1 : N\}$ .
- The NLL is given by

$$\text{NLL}(\boldsymbol{\theta}) = - \sum_k N_k \log \theta_k$$

where  $N_k$  is the number of times the event  $Y = k$  is observed.

- To compute the MLE, we have to minimize the NLL subject to the **constraint** that

$$\sum_{k=1}^K \theta_k = 1$$

# MLE for the categorical distribution

- We use the method of Lagrange multipliers. The Lagrangian is as

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = -\sum_k N_k \log \theta_k - \lambda \left( 1 - \sum_k \theta_k \right)$$

- Taking derivatives with respect to  $\lambda$  yields the original constraint

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \sum_k \theta_k = 0$$

- Taking derivatives with respect to  $\theta_k$  yields

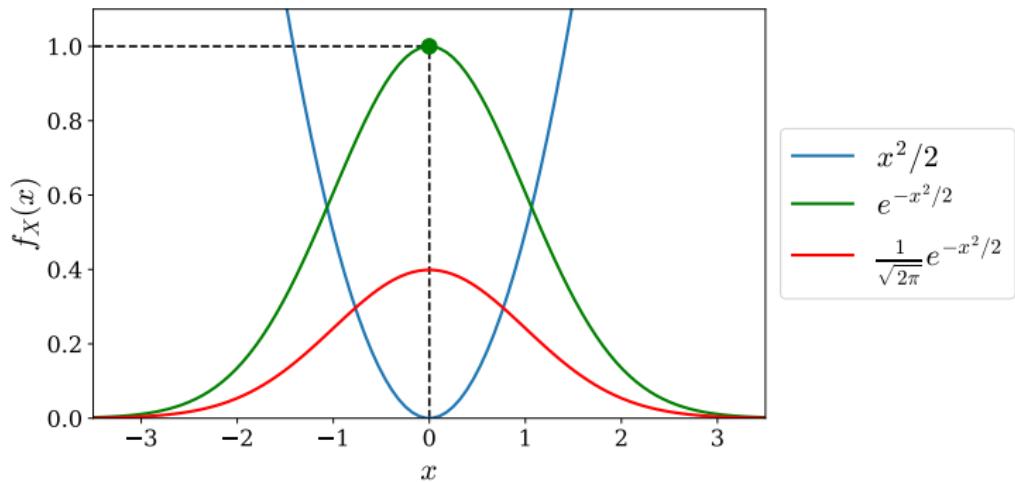
$$\frac{\partial \mathcal{L}}{\partial \theta_k} = -\frac{N_k}{\theta_k} + \lambda = 0 \longrightarrow N_k = \lambda \theta_k$$

- We can solve for  $\lambda$  using the sum-to-one constraint

$$\sum_k N_k = N = \lambda \sum_k \theta_k = \lambda$$

- Thus the MLE is given by  $\hat{\theta}_k = \frac{N_k}{\lambda} = \frac{N_k}{N}$ , the **empirical** fraction of times event  $k$  occurs.

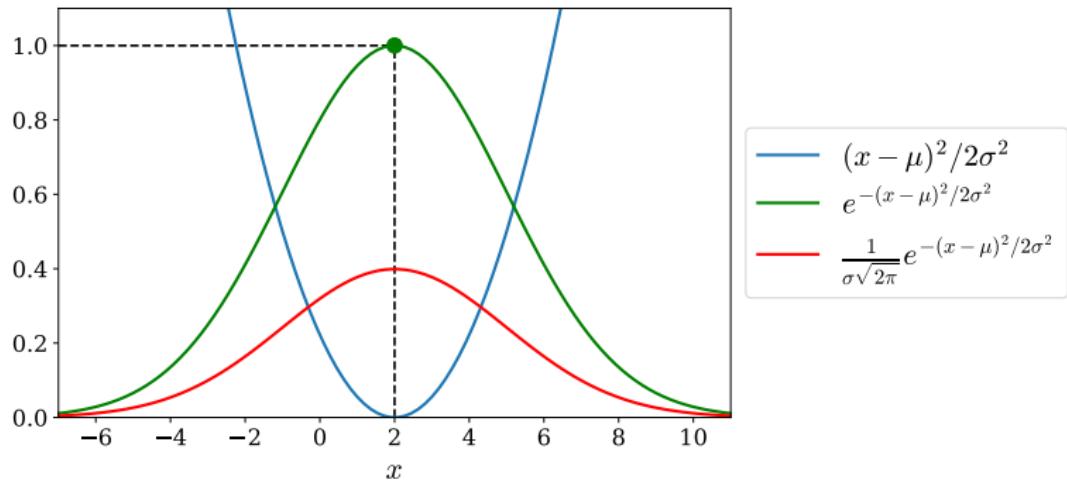
# Standard Normal (Gaussian) Random Variable $N(0, 1)$



$$\int_{-\infty}^{\infty} e^{-x^2/2} dx = \sqrt{2\pi}$$

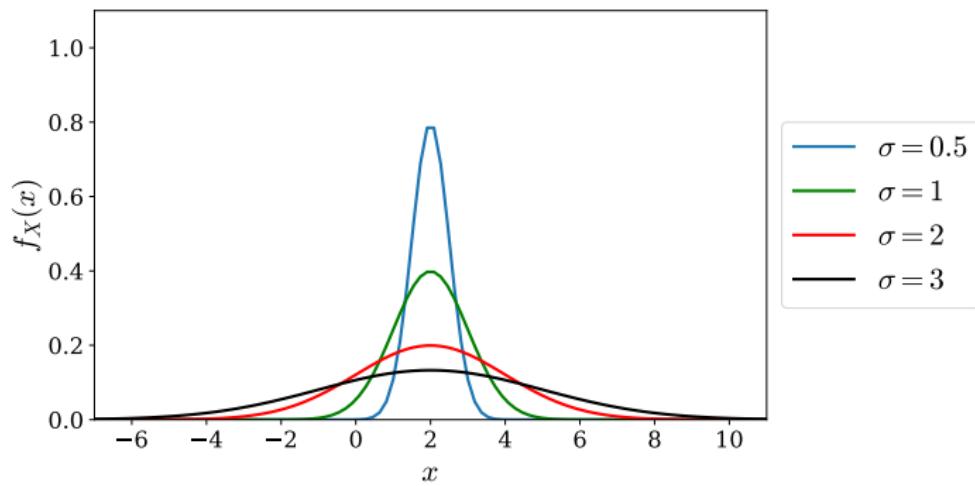
$$f_X(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$$

# General Normal (Gaussian) Random Variable $N(\mu, \sigma^2)$



$$\begin{aligned}f_X(x) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \\&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right) \\E[X] &= \mu \quad \text{Var}(X) = \sigma^2\end{aligned}$$

# General Normal (Gaussian) Random Variable $N(\mu, \sigma^2)$



- Smaller  $\sigma$ , narrower PDF.
- Let  $Y = aX + b$        $N \sim N(\mu, \sigma^2)$
- Then,  $E[Y] = aE[X] + b$        $\text{Var}(Y) = a^2\sigma^2$  (always true)
- But also,  $Y \sim N(a\mu + b, a^2\sigma^2)$

## Example

- We have  $N = 3$  data points  $y_1 = 1$ ,  $y_2 = 0.5$ ,  $y_3 = 1.5$  which are independent and Gaussian with **unknown** mean  $\mu$  and variance 1:

$$y_i \sim \mathcal{N}(\mu, 1)$$

- Likelihood  $P(y_1 y_2 y_3 | \mu) = P(y_1 | \mu)P(y_2 | \mu)P(y_3 | \mu)$ .
- Consider two guesses  $\mu = 1.0$  and  $\mu = 2.5$ . Which has higher likelihood?
- Finding the  $\mu$  that maximizes the likelihood is equivalent to moving the Gaussian until the product  $P(y_1 | \mu)P(y_2 | \mu)P(y_3 | \mu)$  is maximized.

# MLE for the univariate Gaussian

- $Y \sim \mathcal{N}(\mu, \sigma^2)$  and  $\mathcal{D} = \{y_n : n = 1 : N\}$  be an iid sample of size  $N$ .

$$p(y|\theta) = \mathcal{N}(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mu)^2\right)$$

- We can estimate the parameters  $\theta = (\mu, \sigma^2)$  using MLE.
- We derive the NLL, which is given by

$$\begin{aligned}\text{NLL}(\mu, \sigma^2) &= -\sum_{n=1}^N \log \left[ \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y_n - \mu)^2\right) \right] \\ &= \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mu)^2 + \frac{N}{2} \log(2\pi\sigma^2)\end{aligned}$$

- The minimum of this function must satisfy the following conditions

$$\frac{\partial}{\partial \mu} \text{NLL}(\mu, \sigma^2) = 0, \quad \frac{\partial}{\partial \sigma^2} \text{NLL}(\mu, \sigma^2) = 0$$

# MLE for the univariate Gaussian

- The solution is given by

$$\hat{\mu}_{\text{mle}} = \frac{1}{N} \sum_{n=1}^N y_n = \bar{y}$$

$$\hat{\sigma}_{\text{mle}}^2 = \frac{1}{N} \sum_{n=1}^N (y_n - \hat{\mu}_{\text{mle}})^2 = \frac{1}{N} \left[ \sum_{n=1}^N y_n^2 + \hat{\mu}_{\text{mle}}^2 - 2y_n \hat{\mu}_{\text{mle}} \right] = s^2 - \bar{y}^2$$

$$s^2 \triangleq \frac{1}{N} \sum_{n=1}^N y_n^2$$

- The quantities  $\bar{y}$  and  $s^2$  are called the **sufficient statistics** of the data because they are sufficient to compute the MLE.
- Sometimes, we might see the estimate for the variance as

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{n=1}^N (y_n - \hat{\mu}_{\text{mle}})^2$$

which is not the MLE, but is a different kind of estimate.

# Table of Contents

- 1 Introduction
- 2 Maximum Likelihood Estimation
- 3 Examples
- 4 MLE for Linear Regression
- 5 Regularization

# MLE for linear regression

- We can make the parameters of the Gaussian to be functions of some input variables

$$p(y|\mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}(y|f_{\mu}(\mathbf{x}; \boldsymbol{\theta}), f_{\sigma}(\mathbf{x}; \boldsymbol{\theta})^2)$$

$f_{\mu}(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}$  predicts mean, and  $f_{\sigma}(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}_+$  predicts variance.

- It is common to assume that the variance is *fixed*, and is *independent* of the input. This is called **homoscedastic regression**.
- Furthermore, it is common to assume the mean is a linear function of the input. The resulting model is called **linear regression**.

$$p(y|\mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T \mathbf{x} + b, \sigma^2)$$

where  $\boldsymbol{\theta} = (\mathbf{w}, b, \sigma)$ .

# MLE for linear regression

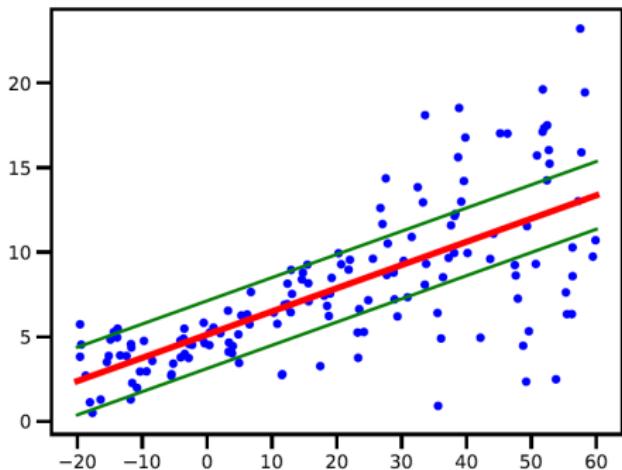


Figure: Linear regression using Gaussian output with mean  $\mu(x) = b + wx$  and fixed variance  $\sigma^2$ .

- The figure plots the 95% predictive interval  $[\mu(x) - 2\sigma, \mu(x) + 2\sigma]$ .
- This is the uncertainty in the predicted *observation*  $y$  given  $x$ , and capture the variability in the blue dots.

# MLE for linear regression

- Linear regression model

$$p(y|\mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T \mathbf{x}, \sigma^2)$$

where  $\boldsymbol{\theta} = (\mathbf{w}, \sigma^2)$ , and  $\mathbf{w} = (b, w_1, w_2, \dots, w_D)$ .

- Assume that  $\sigma^2$  is fixed, we estimate the weights  $\mathbf{w}$ . The NLL is

$$\text{NLL}(\mathbf{w}) = - \sum_{n=1}^N \log \left[ \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp \left( - \frac{1}{2\sigma^2} (y_n - \mathbf{w}^T \mathbf{x}_n)^2 \right) \right]$$

- Dropping the *irrelevant additive constants* gives the simplified objective, known as the **residual sum of squares** or **RSS**:

$$\text{RSS}(\mathbf{w}) = \sum_{n=1}^N (y_n - \mathbf{w}^T \mathbf{x}_n)^2 = \sum_{n=1}^N r_n^2$$

where  $r_n$  is the  $n$ -th **residual error**.

# MLE for linear regression

- **Residual sum of squares or RSS:**

$$\text{RSS}(\mathbf{w}) = \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

- **Mean squared error or MSE:**

$$\text{MSE}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

- **Root mean squared error or RMSE:**

$$\text{RMSE}(\mathbf{w}) = \sqrt{\text{MSE}(\mathbf{w})} = \sqrt{\frac{1}{N} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2}$$

- We can compute the MLE by minimizing the NLL, RSS, MSE, or RMSE. All give the same results.

# MLE for linear regression

- The RSS can be written in matrix notation as follows

$$\text{RSS}(\mathbf{w}) = \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y})$$

- The gradient is given by

$$\nabla_{\mathbf{w}} \text{RSS}(\mathbf{w}) = \mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{X}^\top \mathbf{y}$$

- Setting the gradient to zero  $\nabla_{\mathbf{w}} \text{RSS}(\mathbf{w}) = \mathbf{0}$  and solving gives

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

- These are known as the **normal equations**.
- The MLE solution  $\hat{\mathbf{w}}_{\text{mle}}$  is called the **ordinary least squares (OLS)** solution:

$$\hat{\mathbf{w}}_{\text{mle}} = \underset{\mathbf{w}}{\operatorname{argmin}} \text{RSS}(\mathbf{w}) = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

# MLE for linear regression

$$\hat{\mathbf{w}}_{\text{mle}} = \underset{\mathbf{w}}{\operatorname{argmin}} \text{RSS}(\mathbf{w}) = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- The quantity  $\mathbf{X}^\dagger = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$  is the (left) pseudo-inverse of the (non-square) matrix  $\mathbf{X}$ .
- Is the solution  $\hat{\mathbf{w}}_{\text{mle}}$  unique?
- The gradient is  $\nabla_{\mathbf{w}} \text{RSS}(\mathbf{w}) = \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y}$ . Then, the **Hessian** is

$$\mathbf{H}(\mathbf{w}) = \frac{\partial^2}{\partial \mathbf{w}^2} \text{RSS}(\mathbf{w}) = \mathbf{X}^T \mathbf{X}$$

- If  $\mathbf{X}$  is **full rank** (i.e., the columns of  $\mathbf{X}$  are linearly independent), then  $\mathbf{H}$  is **positive definite**, since for any  $\mathbf{v}$ , we have

$$\mathbf{v}^T (\mathbf{X}^T \mathbf{X}) \mathbf{v} = (\mathbf{X} \mathbf{v})^T (\mathbf{X} \mathbf{v}) = \|\mathbf{X} \mathbf{v}\|^2 > 0$$

- In the full rank case, the  $\text{RSS}(\mathbf{w})$  has a **unique global minimum**.

# Table of Contents

- 1 Introduction
- 2 Maximum Likelihood Estimation
- 3 Examples
- 4 MLE for Linear Regression
- 5 Regularization

# Overfitting

- MLE will try to pick parameters that minimize loss on the training set, but this may not result in a model that has low loss on future data. This is called **overfitting**.
- Ex: We want to predict the probability of heads when tossing a coin.
- We toss it  $N = 3$  times and observe 3 heads. The MLE is

$$\hat{\theta}_{\text{mle}} = \frac{N_1}{N_0 + N_1} = \frac{3}{3 + 0} = 1$$

- If we use this  $\text{Ber}(y|\hat{\theta}_{\text{mle}})$  to make predictions, we will predict that all future coin tosses will also be heads!!!
- The model has enough parameters to perfectly fit the observed training data, so it can perfectly match the **empirical** distribution.
- In most cases, the **empirical** distribution is not the same as the **true** distribution. Putting all the probability mass on the observed set of  $N$  examples will not leave over any probability for novel data in the future. The model may not **generalize**.

# Example: MLE for Linear Regression

## Example 1:

- Training data:  $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ,  $y_1 = 1$     $\mathbf{x}_2 = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$ ,  $y_2 = 1$ .
- $\mathbf{X} = \begin{bmatrix} 1 & 0 \\ 1 & \epsilon \end{bmatrix}$ ,    $\mathbf{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $\hat{\mathbf{w}}_{\text{mle}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = ?$

## Example 2:

- Training data:  $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ,  $y_1 = 1 + \epsilon$     $\mathbf{x}_2 = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$ ,  $y_2 = 1$ .
- $\mathbf{X} = \begin{bmatrix} 1 & 0 \\ 1 & \epsilon \end{bmatrix}$ ,    $\mathbf{y} = \begin{bmatrix} 1 + \epsilon \\ 1 \end{bmatrix}$
- $\hat{\mathbf{w}}_{\text{mle}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = ?$

# Example: MLE for Linear Regression

Example 1:

- Training data:  $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ,  $y_1 = 1$     $\mathbf{x}_2 = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$ ,  $y_2 = 1$ .
- $\mathbf{X} = \begin{bmatrix} 1 & 0 \\ 1 & \epsilon \end{bmatrix}$ ,    $\mathbf{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $\hat{\mathbf{w}}_{\text{mle}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$
- $\mathbf{X}^T \mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & \epsilon \end{bmatrix} = \begin{bmatrix} 2 & \epsilon \\ \epsilon & \epsilon^2 \end{bmatrix}$
- $(\mathbf{X}^T \mathbf{X})^{-1} = \begin{bmatrix} 1 & -1/\epsilon \\ -1/\epsilon & 2/\epsilon^2 \end{bmatrix}$
- $\hat{\mathbf{w}}_{\text{mle}} = \begin{bmatrix} 1 & -1/\epsilon \\ -1/\epsilon & 2/\epsilon^2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$

## Example: MLE for Linear Regression

Example 2:

- Training data:  $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ,  $y_1 = 1 + \epsilon$     $\mathbf{x}_2 = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$ ,  $y_2 = 1$ .
- $\mathbf{X} = \begin{bmatrix} 1 & 0 \\ 1 & \epsilon \end{bmatrix}$ ,    $\mathbf{y} = \begin{bmatrix} 1 + \epsilon \\ 1 \end{bmatrix}$
- $\hat{\mathbf{w}}_{\text{mle}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$
- $\mathbf{X}^T \mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & \epsilon \end{bmatrix} = \begin{bmatrix} 2 & \epsilon \\ \epsilon & \epsilon^2 \end{bmatrix}$
- $(\mathbf{X}^T \mathbf{X})^{-1} = \begin{bmatrix} 1 & -1/\epsilon \\ -1/\epsilon & 2/\epsilon^2 \end{bmatrix}$
- $\hat{\mathbf{w}}_{\text{mle}} = \begin{bmatrix} 1 & -1/\epsilon \\ -1/\epsilon & 2/\epsilon^2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 + \epsilon \\ 1 \end{bmatrix} = \begin{bmatrix} 1 + \epsilon \\ -1 \end{bmatrix}$ .

# Regularization

- The main solution to overfitting is to use **regularization**.
- We add a penalty term to the NLL (or empirical risk):

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = \left[ \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n; \boldsymbol{\theta})) \right] + \lambda C(\boldsymbol{\theta})$$

where  $\lambda \geq 0$  is the **regularization parameter**, and  $C(\boldsymbol{\theta})$  is some form of **complexity penalty**.

- A common complexity penalty is to use  $C(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$ , where  $p(\boldsymbol{\theta})$  is the **prior** for  $\boldsymbol{\theta}$ .
- If  $\ell$  is the log loss, the regularized objective becomes

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = -\frac{1}{N} \sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) - \lambda \log p(\boldsymbol{\theta})$$

# Maximum a posteriori estimation (MAP)

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = -\frac{1}{N} \sum_{n=1}^N \log p(y_n | x_n, \boldsymbol{\theta}) - \lambda \log p(\boldsymbol{\theta})$$

- By setting  $\lambda = 1$  and rescaling  $p(\boldsymbol{\theta})$  appropriately, we can equivalently minimize the following

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = - \left[ \sum_{n=1}^N \log p(y_n | x_n, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \right] = -[\log p(\mathcal{D} | \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})]$$

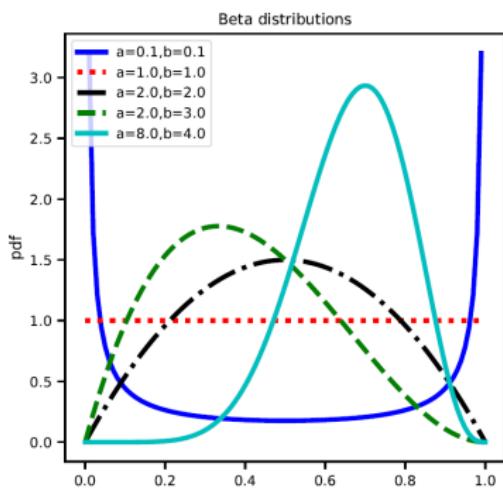
- Minimizing this is equivalent to maximizing the log posterior:

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{map}} &= \operatorname{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} | \mathcal{D}) = \operatorname{argmax}_{\boldsymbol{\theta}} \log \frac{p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \operatorname{argmax}_{\boldsymbol{\theta}} [\log p(\mathcal{D} | \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \text{const}]\end{aligned}$$

- This is **MAP estimation**, or **maximum a posteriori estimation**.

# MAP estimation for Bernoulli distribution

- Coin tossing. If we observe just one head, the MLE is  $\hat{\theta}_{\text{MLE}} = 1$ .
- To avoid this, we can add a penalty to  $\theta$  to discourage “extreme” values, such as  $\theta = 0$  or  $\theta = 1$ .
- We can use a beta distribution as our prior  $p(\theta) = \text{Beta}(\theta|a, b)$ , where  $a, b > 1$  encourages values of  $\theta$  near to  $a/(a + b)$ .



- If  $a = b = 1$ , we get uniform distribution
- If  $a$  and  $b$  are both less than 1, we get bimodal distribution.
- If  $a$  and  $b$  are both greater than 1, the distribution is unimodal.

$$\text{mean} = \frac{a}{a + b}$$

$$\text{var} = \frac{ab}{(a + b)^2(a + b + 1)}$$

## MAP estimate for Bernoulli distribution

- Using the beta distribution as our prior  $p(\theta) = \text{Beta}(\theta|a, b)$ , the log likelihood plus log prior becomes

$$\text{LL}(\theta) = \log p(\mathcal{D}|\theta) + \log p(\theta)$$

$$= [N_1 \log \theta + N_0 \log(1 - \theta)] + [(a - 1) \log \theta + (b - 1) \log(1 - \theta)]$$

- The MAP estimate is

$$\hat{\theta}_{\text{map}} = \frac{N_1 + a - 1}{N_1 + N_0 + a + b - 2}$$

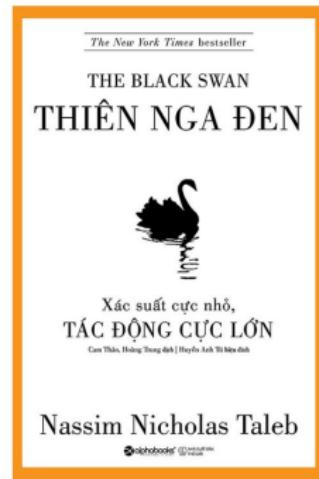
- If we set  $a = b = 2$ , that weakly favor a value of  $\theta$  near 0.5, the estimate becomes

$$\hat{\theta}_{\text{map}} = \frac{N_1 + 1}{N_1 + N_0 + 2}$$

- This is called **add-one smoothing** to avoid the **zero count problem**.

# Black swan paradox

- The zero-count problem, and overfitting, is analogous to the **black swan paradox**.
- It is used to illustrate the problem of **induction**: how to draw general conclusions about the future from specific observations from the past.
- The solution to the paradox is to admit that induction is *in general* impossible.
- The best we can do is to make plausible guesses by combining the **empirical data** with **prior knowledge**.



## Weight decay

- Polynomial regression with too much degree of freedom can result in overfitting. One solution is to reduce the degree of the polynomial.
- A more general solution is to **penalize** the **magnitude** of the weights (regression coefficients).
- We use a zero-mean Gaussian prior  $p(\mathbf{w})$ . The MAP estimate is

$$\hat{\mathbf{w}}_{\text{map}} = \underset{\mathbf{w}}{\operatorname{argmin}} \text{NLL}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

where  $\|\mathbf{w}\|_2^2 = \sum_{d=1}^D w_d^2$ . We penalize the magnitude of weight vectors  $\mathbf{w}$ , rather than the bias term  $b$ .

- The equation is called  $\ell_2$  **regularization** or **weight decay**.
- The larger the value of  $\lambda$ , the more the parameters are penalized for being large (i.e., deviating from the zero-mean prior), and thus the less flexible the model.

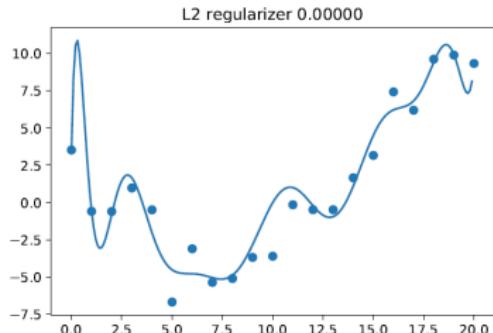
## Ridge regression

- In the case of linear regression, the weight decay penalization scheme is called **ridge regression**.
- Consider polynomial regression, where the predictor has the form

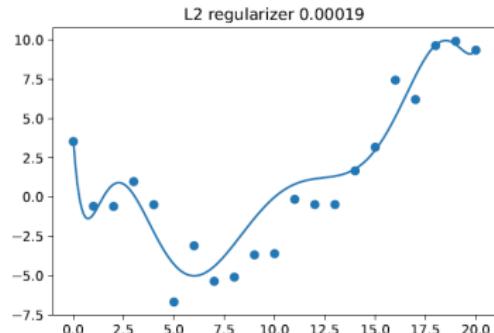
$$f(x; \boldsymbol{w}) = \sum_{d=0}^D w_d x^d = \boldsymbol{w}^\top [1, x, x^2, \dots, x^D]$$

- Suppose we use a high degree polynomial, say  $D = 14$ , even though we have a small dataset with just  $N = 21$  examples.
- MLE for the parameters will enable the model to fit the data very well, but the resulting function is very “wiggly”, thus resulting in overfitting.
- Increasing  $\lambda$  can reduce overfitting.

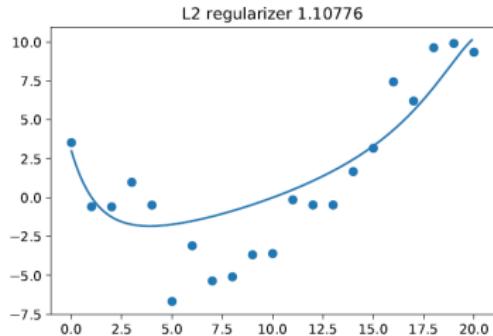
# Ridge regression



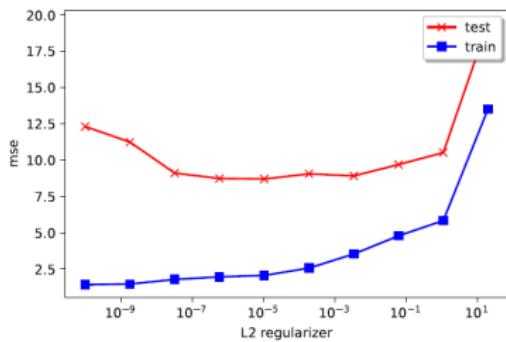
(a)



(b)



(c)



(d)

## Ridge regression

- MAP estimation with a zero-mean Gaussian prior

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | 0, \lambda^{-1} \mathbf{I}).$$

$$\begin{aligned}\hat{\mathbf{w}}_{\text{map}} &= \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{1}{2\tau^2} \mathbf{w}^\top \mathbf{w} \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2\end{aligned}$$

where  $\lambda = \frac{\sigma^2}{\tau^2}$  is proportional to the strength of the prior, and

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{d=1}^D |w_d|^2} = \sqrt{\mathbf{w}^\top \mathbf{w}}$$

is the  $\ell_2$  norm of the vector  $\mathbf{w}$ .

- We do not penalize the offset  $w_0$ , since that only affects the global mean of the output, and does not contribute to the overfitting.

# Ridge Regression

- The MAP estimate corresponds to minimizing the penalized objective:

$$J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2$$

where  $\lambda = \frac{\sigma^2}{\tau^2}$  is the strength of the regularizer.

- The derivative is given by

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2(\mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{X}^\top \mathbf{y} + \lambda\mathbf{w})$$

- Therefore,

$$\begin{aligned}\hat{\mathbf{w}}_{\text{map}} &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \mathbf{X}^\top \mathbf{y} \\ &= (\sum_n \mathbf{x}_n \mathbf{x}_n^\top + \lambda \mathbf{I}_D)^{-1} (\sum_n y_n \mathbf{x}_n)\end{aligned}$$

## Example: MAP for Linear Regression

- Maximum likelihood estimation. Let  $\epsilon = 0.1$
- Ex. 1:  $\hat{\mathbf{w}}_{\text{mle}} = \begin{bmatrix} 1 & -1/\epsilon \\ -1/\epsilon & 2/\epsilon^2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ .
- Ex. 2:  $\hat{\mathbf{w}}_{\text{mle}} = \begin{bmatrix} 1 & -1/\epsilon \\ -1/\epsilon & 2/\epsilon^2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 + \epsilon \\ 1 \end{bmatrix} = \begin{bmatrix} 1 + \epsilon \\ -1 \end{bmatrix} = \begin{bmatrix} 1.1 \\ -1 \end{bmatrix}$ .
- Maximum a posteriori estimation. Let  $\lambda = 0.05$
- $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_D) = \begin{bmatrix} 2 + \lambda & \epsilon \\ \epsilon & \epsilon^2 + \lambda \end{bmatrix} = \begin{bmatrix} 2.05 & 0.1 \\ 0.1 & 0.06 \end{bmatrix}$
- $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_D)^{-1} = \begin{bmatrix} 0.531 & -0.885 \\ -0.885 & 18.1416 \end{bmatrix}$
- Ex. 1:  $\hat{\mathbf{w}}_{\text{map}} = \begin{bmatrix} 0.531 & -0.885 \\ -0.885 & 18.1416 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.9735 \\ 0.0442 \end{bmatrix}$
- Ex. 2:  $\hat{\mathbf{w}}_{\text{map}} = \begin{bmatrix} 0.531 & -0.885 \\ -0.885 & 18.1416 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} 1 + \epsilon \\ 1 \end{bmatrix} = \begin{bmatrix} 1.0265 \\ -0.0442 \end{bmatrix}$

# Linear Algebra

## Review

Ngoc Hoang Luong

Faculty of Computer Science  
University of Information Technology (UIT)  
Vietnam National University - Ho Chi Minh City (VNU-HCM)

Maths for Computer Science, Fall 2022



# Machine Learning and Linear Algebra



1

<sup>1</sup><https://xkcd.com/1838/>

# References

The contents of this document are taken mainly from the following sources:

- ▶ Gilbert Strang. Linear Algebra and Learning from Data.  
<https://math.mit.edu/~gs/learningfromdata/>
- ▶ Gilbert Strang. Introduction to Linear Algebra.  
<http://math.mit.edu/~gs/linearalgebra/>
- ▶ Gilbert Strang. Linear Algebra for Everyone.  
<http://math.mit.edu/~gs/everyone/>

# Table of Contents

- ① Matrix-Vector Multiplication  $Ax$
- ② Matrix-Matrix Multiplication  $AB$
- ③ The Four Fundamental Subspaces of  $A$ :  $\mathbf{C}(A)$ ,  $\mathbf{C}(A^\top)$ ,  $\mathbf{N}(A)$ ,  $\mathbf{N}(A^\top)$
- ④ Elimination and  $A = LU$
- ⑤ Orthogonal Matrices, Subspaces, and Projections

# Table of Contents

- ① Matrix-Vector Multiplication  $A\mathbf{x}$
- ② Matrix-Matrix Multiplication  $AB$
- ③ The Four Fundamental Subspaces of  $A$ :  $\mathbf{C}(A)$ ,  $\mathbf{C}(A^\top)$ ,  $\mathbf{N}(A)$ ,  $\mathbf{N}(A^\top)$
- ④ Elimination and  $A = LU$
- ⑤ Orthogonal Matrices, Subspaces, and Projections

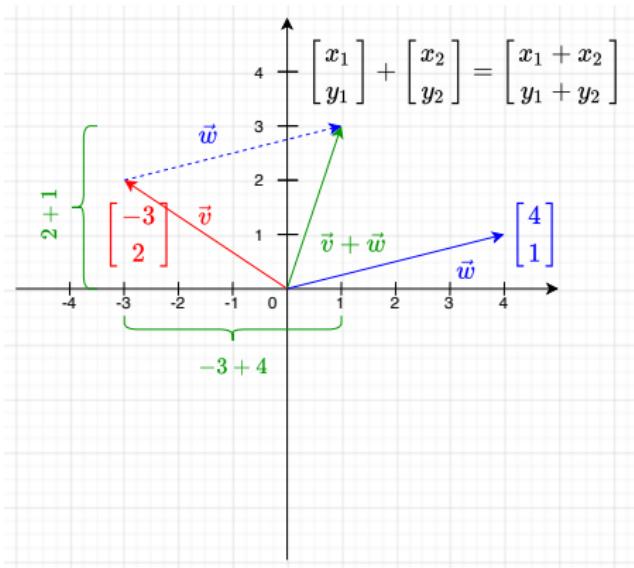
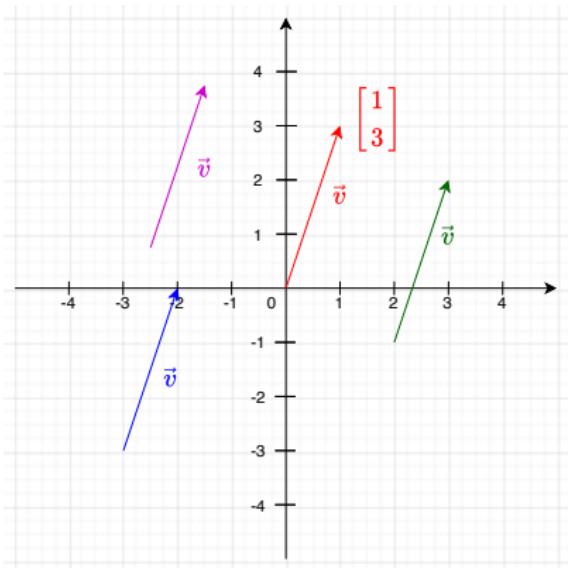


# Vectors

- ▶ Vectors are arrays of numerical values.
- ▶ Each numerical value is referred to as *coordinate*, *component*, *entry*, or *dimension*.
- ▶ The number of components is the vector *dimensionality*.
- ▶ e.g., a vector representation of a person: 25 years old (Age), making 30 dollars an hour (Salary), having 6 years of experience (Experience): [25, 30, 6].
- ▶ Vectors are special objects that can be added together and multiplied by scalars to produce another object of the same kind.

- ▶ **Geometric vectors** are often visualized as a quantity that has a **magnitude** as well as a **direction**.
- ▶ e.g., the velocity of a person moving at 1 meter/second in the eastern direction and 3 meters/second in the northern direction can be described as a directed line from the origin to  $(1, 3)$ .
- ▶ The **tail** of the vector is at the origin. The **head** is at  $(1, 3)$ .
- ▶ Geometric vectors can have arbitrary tails.
- ▶ Two geometric vectors can be added, such that  $\mathbf{x} + \mathbf{y} = \mathbf{z}$  is another geometric vector.
- ▶ Multiplication by a scalar  $\lambda \mathbf{x}, \lambda \in \mathbb{R}$ , is also a geometric vector.

# Vectors



# Vectors

- ▶ Polynomials are vectors. Adding two polynomials results in another polynomial. Multiplied by a scalar, the result is also a polynomial.
- ▶ Audio signals are also vectors. Addition of two audio signals and scalar multiplication result in new audio signals.
- ▶ Elements of  $\mathbb{R}^n$  (tuples of  $n$  real numbers) are vectors. For example,

$$\mathbf{a} = \begin{bmatrix} 6 \\ 14 \\ -3 \end{bmatrix} \in \mathbb{R}^3$$

is a triplet of numbers. Adding two vectors  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$  component-wise results in another vectors  $\mathbf{a} + \mathbf{b} = \mathbf{c} \in \mathbb{R}^n$ . Multiplying  $\mathbf{a} \in \mathbb{R}^n$  by  $\lambda \in \mathbb{R}$  results in a scaled vector  $\lambda\mathbf{a} \in \mathbb{R}^n$ .

# Basic Operations with Vectors

- ▶ Vector of the same dimensionality can be added or subtracted.
- ▶ Consider two  $d$ -dimensional vectors:

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 \\ \dots \\ x_d \end{bmatrix} + \begin{bmatrix} y_1 \\ \dots \\ y_d \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ \dots \\ x_d + y_d \end{bmatrix} \quad \mathbf{x} - \mathbf{y} = \begin{bmatrix} x_1 \\ \dots \\ x_d \end{bmatrix} - \begin{bmatrix} y_1 \\ \dots \\ y_d \end{bmatrix} = \begin{bmatrix} x_1 - y_1 \\ \dots \\ x_d - y_d \end{bmatrix}$$

- ▶ Vector addition is commutative:  $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$ .

# Basic Operations with Vectors

- ▶ A vector  $x \in \mathbb{R}^d$  can be scaled by a factor  $a \in \mathbb{R}$  as follows

$$v = ax = a \begin{bmatrix} x_1 \\ \dots \\ x_d \end{bmatrix} = \begin{bmatrix} ax_1 \\ \dots \\ ax_d \end{bmatrix}$$

- ▶ Scalar multiplication operation scales the “length” of the vector, but does not change the “direction” (i.e., relative values of different components)

# Basic Operations with Vectors

- ▶ The **dot product** between two vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$  is the sum of the element-wise multiplication of their individual components.

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^d x_i y_i$$

- ▶ The dot product is commutative:

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^d x_i y_i = \sum_{i=1}^d y_i x_i = \mathbf{y} \cdot \mathbf{x}$$

- ▶ The dot product is distributive:

$$\mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) = \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z}$$

# Basic Operations with Vectors

- The dot product of a vector with itself produces the squared Euclidean norm. The norm defines the vector length and is denoted by  $\|\cdot\|$ :

$$\|x\|^2 = \mathbf{x} \cdot \mathbf{x} = \sum_{i=1}^d x_i^2$$

- The Euclidean norm of  $x \in \mathbb{R}^d$  is defined as

$$\|x\|_2 = \sqrt{\sum_{i=1}^d x_i^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

and computes the Euclidean distance of  $x$  from the origin.

- The Euclidean norm is also known as the  $L_2$ -norm.

# Basic Operations with Vectors

- ▶ A generalization of the Euclidean norm is the  $L_p$ -norm, denoted by  $\|\cdot\|_p$ :

$$\|x\|_p = \left( \sum_{i=1}^d |x_i|^p \right)^{(1/p)}$$

where  $p$  is a positive value.

- ▶ When  $p = 1$ , we have the Manhattan norm, or the  $L_1$ -norm.

# Basic Operations with Vectors

- ▶ Vectors can be **normalized** to unit length by dividing them with their norm:

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|} = \frac{\mathbf{x}}{\sqrt{\mathbf{x} \cdot \mathbf{x}}}$$

- ▶ The resulting vector is a **unit vector**.
- ▶ The squared Euclidean distance  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$  can be shown to be the dot product of  $\mathbf{x} - \mathbf{y}$  with itself:

$$\|\mathbf{x} - \mathbf{y}\|^2 = (\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y}) = \sum_{i=1}^d (x_i - y_i)^2$$

# Basic Operations with Vectors

- ▶ **Cauchy-Schwarz Inequality:** the dot product between a pair of vectors is bounded above by the product of their lengths.

$$\left| \sum_{i=1}^d x_i y_i \right| = |\mathbf{x} \cdot \mathbf{y}| \leq \|\mathbf{x}\| \|\mathbf{y}\|$$

- ▶ **Triangle Inequality:** Consider the triangle formed by the origin,  $\mathbf{x}$ , and  $\mathbf{y}$ , the side length  $\|\mathbf{x} - \mathbf{y}\|$  is no greater than the sum  $\|\mathbf{x}\| + \|\mathbf{y}\|$  of the other two sides.

# Basic Operations with Vectors

- ▶ Consider the triangle created by the origin,  $\mathbf{x}$ , and  $\mathbf{y}$ . Find the angle  $\theta$  between  $\mathbf{x}$  and  $\mathbf{y}$ .
- ▶ The side lengths of this triangle are:  $a = \|\mathbf{x}\|$ ,  $b = \|\mathbf{y}\|$ , and  $c = \|\mathbf{x} - \mathbf{y}\|$ . Using the cosine law, we have:

$$\begin{aligned}\cos(\theta) &= \frac{a^2 + b^2 - c^2}{2ab} = \frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \|\mathbf{x} - \mathbf{y}\|^2}{2\|\mathbf{x}\|\|\mathbf{y}\|} \\ &= \frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - (\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})}{2\|\mathbf{x}\|\|\mathbf{y}\|} \\ &= \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\end{aligned}$$

- ▶ Two vectors are **orthogonal** if their dot product is 0.
- ▶ The vector  $\mathbf{0}$  is considered orthogonal to every vector.

## Definition

With  $m, n \in \mathbb{N}$ , a real-valued  $(m, n)$  matrix  $\mathbf{A}$  is an  $m \cdot n$ -tuple of elements  $a_{ij}, i = 1, \dots, m, j = 1, \dots, n$ , which is ordered according to a rectangular scheme consisting of  $m$  rows and  $n$  columns:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad a_{ij} \in \mathbb{R}$$

$\mathbb{R}^{m \times n}$  is the set of all real-valued  $(m, n)$ -matrices.

$\mathbf{A} \in \mathbb{R}^{m \times n}$  can also be represented as  $\mathbf{a} \in \mathbb{R}^{mn}$  by stacking all  $n$  columns of the matrix into a long vector.

- ▶ A matrix has the same number of rows as columns is a **square** matrix. Otherwise, it is a **rectangular** matrix.
- ▶ A matrix having more rows than columns is referred to as *tall*, while a matrix having more columns than rows is referred to as *wide* or *fat*.
- ▶ A scalar can be considered as a  $1 \times 1$  “matrix”.
- ▶ A  $d$ -dimensional vector can be considered a  $1 \times d$  matrix when it is treated as a **row vector**.
- ▶ A  $d$ -dimensional vector can be considered a  $d \times 1$  matrix when it is treated as a **column vector**.
- ▶ By defaults, vectors are assumed to be column vectors.

# Matrix-Vector Multiplication

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \boxed{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}$$

2

<sup>2</sup><https://xkcd.com/184/>

# Matrix-Vector Multiplication $Ax$

- ▶ Multiply  $A$  times  $x$  using rows of  $A$ .

$$\begin{bmatrix} 2 & 3 \\ 2 & 4 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2x_1 + 3x_2 \\ 2x_1 + 4x_2 \\ 3x_1 + 7x_2 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^* \mathbf{x} \\ \mathbf{a}_2^* \mathbf{x} \\ \mathbf{a}_3^* \mathbf{x} \end{bmatrix}$$

$Ax$  = dot products of rows of  $A$  with  $x$ .

- ▶ Multiply  $A$  times  $x$  using columns of  $A$ .

$$\begin{bmatrix} 2 & 3 \\ 2 & 4 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2$$

$Ax$  = combination of columns of  $\mathbf{a}_1$ ,  $\mathbf{a}_2$  (of  $A$ ) scaled by scalars  $x_1$ ,  $x_2$  respectively.

# Linear Combinations of Columns

$A\mathbf{x}$

$A\mathbf{x}$  is a linear combination of the columns of  $A$ .

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix}$$
$$A\mathbf{x} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_n \mathbf{a}_n$$

**Column space of  $A = \mathbf{C}(A) =$  all vectors  $A\mathbf{x}$**

**= all linear combinations of the columns**



UIT  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN

# Column Space of $A$

$$A\mathbf{x} = \begin{bmatrix} 2 & 3 \\ 2 & 4 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix}$$

- ▶ Each  $A\mathbf{x}$  is a vector in the  $\mathbb{R}^3$  space.
- ▶ All combinations  $A\mathbf{x} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2$  produce what part of  $\mathbb{R}^3$ ?
- ▶ Answer: a **plane**, containing:
  - the line of all vectors  $x_1\mathbf{a}_1$ ,
  - the line of all vectors  $x_2\mathbf{a}_2$ ,
  - the sum of any vector on one line + any vector on the other line, filling out an **infinite plane** containing the two lines, but not the whole  $\mathbb{R}^3$ .

## Definition

The combinations of the columns fill out the column space of  $A$ .

# Column Space of $A$

$$A\mathbf{x} = \begin{bmatrix} 2 & 3 \\ 2 & 4 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix}$$

- ▶  $\mathbf{C}(A)$  is plane.
- ▶ The plane includes  $(0, 0)$ , produced when  $x_1 = x_2 = 0$ .
- ▶ The plane includes  $(5, 6, 10) = \mathbf{a}_1 + \mathbf{a}_2$  and  $(-1, -2, -4) = \mathbf{a}_1 - \mathbf{a}_2$ .  
Every combination  $x_1\mathbf{a}_1 + x_2\mathbf{a}_2$  is in  $\mathbf{C}(A)$ .
- ▶ The probability the plane does not include a random point  $\text{rand}(3,1)$ ?  
Which points are in the plane?

$$A\mathbf{x} = \mathbf{b}$$

$\mathbf{b}$  is in  $\mathbf{C}(A)$  exactly when  $A\mathbf{x} = \mathbf{b}$  has a solution  $\mathbf{x}$ .

$\mathbf{x}$  shows how to express  $\mathbf{b}$  as a combination of the columns of  $A$ .

# Column Space of $A$

- $\mathbf{b} = (1, 1, 1)$  is not in  $\mathbf{C}(A)$  because

$$Ax = \begin{bmatrix} 2 & 3 \\ 2 & 4 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{is unsolvable.}$$

- What is the column space of  $A_2$ ?

$$\begin{bmatrix} 2 & 3 & 5 \\ 2 & 4 & 6 \\ 3 & 7 & 10 \end{bmatrix} \quad \begin{array}{l} \bullet \mathbf{a}_3 = \mathbf{a}_1 + \mathbf{a}_2, \text{ is already in } \mathbf{C}(A), \text{ the plane of } \mathbf{a}_1 \text{ and } \mathbf{a}_2. \\ \bullet \text{Including this } \mathbf{dependent} \text{ column does not go beyond } \mathbf{C}(A). \\ \bullet \mathbf{C}(A_2) = \mathbf{C}(A). \end{array}$$

- What is the column space of  $A_3$ ?

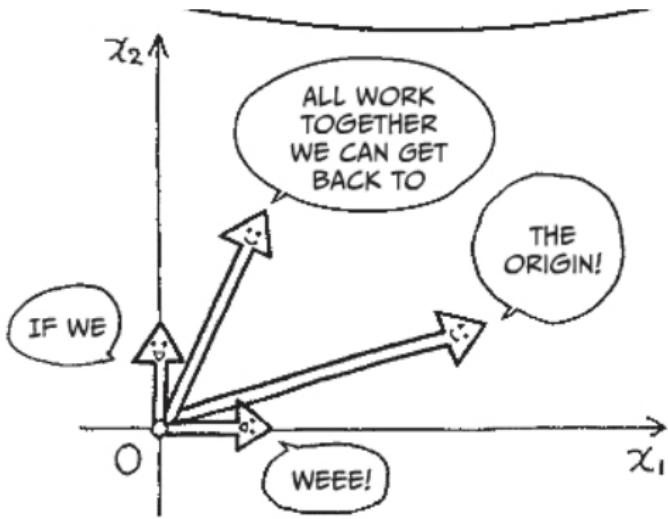
$$\begin{bmatrix} 2 & 3 & 1 \\ 2 & 4 & 1 \\ 3 & 7 & 1 \end{bmatrix} \quad \begin{array}{l} \bullet \mathbf{a}_3 = (1, 1, 1) \text{ is not in the plane } \mathbf{C}(A). \\ \bullet \text{Visualize the } xy\text{-plane and a third vector } (x_3, y_3, z_3) \text{ out of the plane (meaning that } z_3 \neq 0\text{).} \\ \bullet \mathbf{C}(A_3) = \mathbb{R}^3. \end{array}$$



# Column Spaces of $\mathbb{R}^3$

- ▶ Subspaces of  $\mathbb{R}^3$ :
  - The zero vector  $(0, 0, 0)$ .
  - A line of all vectors  $x_1\mathbf{a}_1$ .
  - A plane of all vectors  $x_1\mathbf{a}_1 + x_2\mathbf{a}_2$ .
  - The whole  $\mathbb{R}^3$  with all vectors  $x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + x_3\mathbf{a}_3$ .
- ▶ Vectors  $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$  need to be **independent**. The only combination that gives the zero vector is  $0\mathbf{a}_1 + 0\mathbf{a}_2 + 0\mathbf{a}_3$ .
- ▶ The zero vector is in every subspace.

# Linear Dependence



LINEAR DEPENDENCE

3

<sup>3</sup><https://mathsci2.appstate.edu/sjg/class/2240/hf14.html>

# Independent Columns, Basis, and Ranks of $A$

## Definition

A **basis** for a subspace is a full set of independent vectors: All vectors in the space are combinations of the basis vector.

Create a matrix  $C$  whose columns come directly from  $A$ :

- ▶ If column 1 of  $A$  is not all zero, put it into  $C$ .
- ▶ If column 2 of  $A$  is not a multiple of column 1, put it into  $C$ .
- ▶ If column 3 of  $A$  is not a combination of columns 1 and 2, put it into  $C$ . *Continue.*
- ▶ At the end,  $C$  will have  $r$  columns ( $r \leq n$ ). They are independent columns, and they are a “basis” for the column space  $\mathbf{C}(A)$ .



UIT

TRƯỜNG ĐẠI HỌC

CONG NGHỆ THÔNG TIN

# Independent Columns, Basis, and Ranks of $A$

If  $A = \begin{bmatrix} 1 & 3 & 8 \\ 1 & 2 & 6 \\ 0 & 1 & 2 \end{bmatrix}$  then  $C = \begin{bmatrix} 1 & 3 \\ 1 & 2 \\ 0 & 1 \end{bmatrix}$   $n = 3$  columns in  $A$   
 $r = 2$  columns in  $C$

If  $A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$  then  $C = A$   $n = 3$  columns in  $A$   
 $r = 3$  columns in  $C$

If  $A = \begin{bmatrix} 1 & 2 & 5 \\ 1 & 2 & 5 \\ 1 & 2 & 5 \end{bmatrix}$  then  $C = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$   $n = 3$  columns in  $A$   
 $r = 1$  columns in  $C$

- ▶ The number  $r$  counts independent columns.
- ▶ It is the “dimension” of the column space of  $A$  and  $C$  (same space).

## Definition

The **rank** of a matrix is the **dimension** of its column space.

## Rank Factorization $A = CR$

- ▶ The matrix  $C$  connects to  $A$  by a third matrix  $R$ :  $A = CR$ .
- ▶  $A \in \mathbb{R}^{m \times n}$ ,  $C \in \mathbb{R}^{m \times r}$ ,  $R \in \mathbb{R}^{r \times n}$

$$A = \begin{bmatrix} 1 & 3 & 8 \\ 1 & 2 & 6 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \end{bmatrix} = CR$$

- ▶  $C$  multiplies the first column of  $R$  produces column 1 of  $A$ .
- ▶  $C$  multiplies the second column of  $R$  produces column 2 of  $A$ .
- ▶  $C$  multiplies the third column of  $R$  produces column 3 of  $A$ .
- ▶ Combinations of the columns of  $C$  produce the columns of  $A$   
→ Put the right numbers in  $R$ .

### Definition

$R = \text{rref}(A)$  = row-reduced echelon form of  $A$ .

# Rank Factorization $A = CR$

$$A = \begin{bmatrix} 1 & 3 & 8 \\ 1 & 2 & 6 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \end{bmatrix} = CR$$

- ▶ The matrix  $R$  has  $r = 2$  rows  $r_1^*, r_2^*$ .
- ▶ Multiply row 1 of  $C$  with  $R$ , we get  $r_1^* + 3r_2^* \rightarrow$  row 1 of  $A$ .
- ▶ Multiply row 2 of  $C$  with  $R$ , we get  $r_1^* + 2r_2^* \rightarrow$  row 2 of  $A$ .
- ▶ Multiply row 3 of  $C$  with  $R$ , we get  $0r_1^* + 1r_2^* \rightarrow$  row 3 of  $A$ .
- ▶  $R$  has independent rows: No row is a combination of the other rows.  
Hint: Look at the zeros and ones in  $R$  - the identity matrix  $I$  in  $R$ .
- ▶ The rows of  $R$  are a **basis for the row space** of  $A$ .
- ▶ Notation: The row space of matrix  $A = \mathbf{C}(A^\top)$ .

# Rank Factorization $A = CR$

- ① The  $r$  columns of  $C$  are independent (by their construction).
- ② Every column of  $A$  is a combination of those  $r$  columns of  $C$  (because  $A = CR$ ).
- ③ The  $r$  rows of  $R$  are independent (they contain the matrix  $I_r$ ).
- ④ Every row of  $A$  is a combination of those  $r$  rows of  $R$  (because  $A = CR$ ).

## Key facts:

- ▶ The  $r$  columns of  $C$  is a **basis** for  $\mathbf{C}(A)$ : dimension  $r$ .
- ▶ The  $r$  rows of  $R$  is a **basis** for  $\mathbf{C}(A^\top)$ : dimension  $r$ .

## Notice

The number of independent columns = The number of independent rows.

The column space and row space of  $A$  both have dimension  $r$ .

The column rank of  $A$  = The row rank of  $A$ .

# Q&A

**Question:** If an  $n \times n$  matrix  $A$  has  $n$  independent columns, then  $C = ?$ ,  $R = ?$

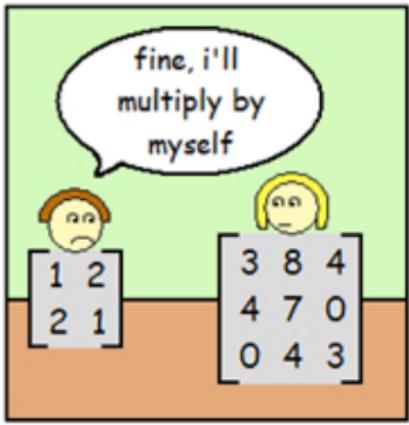
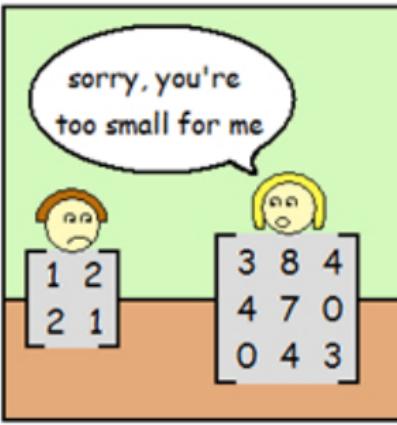
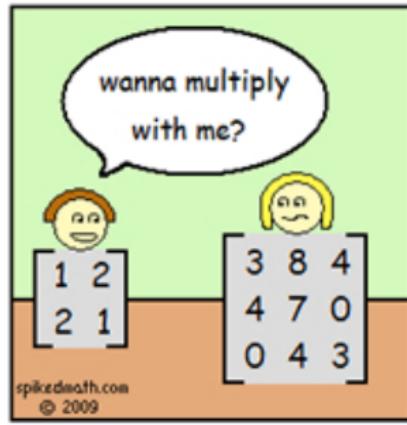
**Answer:**  $C = A$ ,  $R = I$ .

# Table of Contents

- 1 Matrix-Vector Multiplication  $Ax$
- 2 Matrix-Matrix Multiplication  $AB$
- 3 The Four Fundamental Subspaces of  $A$ :  $\mathbf{C}(A)$ ,  $\mathbf{C}(A^\top)$ ,  $\mathbf{N}(A)$ ,  $\mathbf{N}(A^\top)$
- 4 Elimination and  $A = LU$
- 5 Orthogonal Matrices, Subspaces, and Projections



# Matrix-Matrix Multiplication $AB$



<sup>4</sup><https://mathsci2.appstate.edu/sjg/class/2240/hf14.html>

# Compute $AB$ by Inner Products

- ▶ **Inner products** (rows times columns) produce each of the numbers in  $AB = C$ :

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ a_{21} & a_{22} & a_{23} \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & \cdot & b_{13} \\ \cdot & \cdot & b_{23} \\ \cdot & \cdot & b_{33} \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & c_{23} \\ \cdot & \cdot & \cdot \end{bmatrix}$$

- ▶  $c_{ij} = (\text{row } i \text{ of } A) \cdot (\text{column } j \text{ of } B)$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj} = \mathbf{a}_i^* \mathbf{b}_j$$

# Rank-1 Matrix

- ▶ **Outer products** (columns times rows) produce **rank one matrices**.

$$\mathbf{u}\mathbf{v}^\top = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 6 & 8 & 12 \\ 6 & 8 & 12 \\ 3 & 4 & 6 \end{bmatrix}$$

- ▶ An  $m \times 1$  matrix (a column  $\mathbf{u}$ ) times a  $1 \times p$  matrix (a row  $\mathbf{v}^\top$ ) gives an  $m \times p$  matrix.
- ▶ All columns of  $\mathbf{u}\mathbf{v}^\top$  are multiples of  $\mathbf{u}$ .
- ▶ All rows of  $\mathbf{u}\mathbf{v}^\top$  are multiples of  $\mathbf{v}^\top$ .
- ▶ The column space of  $\mathbf{u}\mathbf{v}^\top$  is the line through  $\mathbf{u}$ .
- ▶ The row space of  $\mathbf{u}\mathbf{v}^\top$  is the line through  $\mathbf{v}$ .
- ▶ All non-zero matrices  $\mathbf{u}\mathbf{v}^\top$  have rank one.

# $AB = \text{Sum of Rank-1 Matrices}$

- The product  $AB$  is the sum of columns  $\mathbf{a}_k$  times rows  $\mathbf{b}_k^*$ .

$$AB = \begin{bmatrix} | & & | \\ \mathbf{a}_1 & \dots & \mathbf{a}_n \\ | & & | \end{bmatrix} \begin{bmatrix} - & \mathbf{b}_1^* & - \\ \vdots & & \vdots \\ - & \mathbf{b}_n^* & - \end{bmatrix} = \mathbf{a}_1\mathbf{b}_1^* + \mathbf{a}_2\mathbf{b}_2^* + \cdots + \mathbf{a}_n\mathbf{b}_n^*$$

- Example:

$$\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 2 & 4 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 12 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 17 \end{bmatrix}$$

# Insight from Column times Row

- ▶ Looking for the important part of a matrix  $A$ .
- ▶ Factor  $A$  into  $CR$  and look at the pieces  $\mathbf{c}_k \mathbf{r}_k^*$  of  $A = CR$ .
- ▶ Factoring  $A$  into  $CR$  is the reverse of multiplying  $CR = A$ .
- ▶ The inside information about  $A$  is not visible until  $A$  is factored.

## Important Factorizations

- ①  $A = LU$ : elimination
- ②  $A = QR$ : orthogonalization
- ③  $S = Q\Lambda Q^\top$ : eigenvalues and orthonormal eigenvectors
- ④  $A = X\Lambda X^{-1}$ : diagonalization
- ⑤  $A = U\Sigma V^\top$ : Singular Value Decomposition (SVD)

# Inverse Matrices

- The square matrix  $A$  is invertible if there exists a matrix  $A^{-1}$  that

$$A^{-1}A = I \text{ and } AA^{-1} = I$$

- The matrix  $A$  cannot have two different inverses. Suppose  $BA = I$  and also  $AC = I$ . Then  $B = C$ .

$$B(AC) = (BA)C \text{ gives } BI = IC \text{ or } B = C.$$

- If  $A$  is invertible, the one and only solution to  $Ax = b$  is  $x = A^{-1}b$ .
- If  $Ax = \mathbf{0}$  for a nonzero vector  $x$ , then  $A$  has no inverse.
- If  $A$  and  $B$  are invertible then so is  $AB$ . The inverse of  $AB$  is

$$(AB)^{-1} = B^{-1}A^{-1}$$



# Table of Contents

- 1 Matrix-Vector Multiplication  $Ax$
- 2 Matrix-Matrix Multiplication  $AB$
- 3 The Four Fundamental Subspaces of  $A$ :  $\mathbf{C}(A)$ ,  $\mathbf{C}(A^\top)$ ,  $\mathbf{N}(A)$ ,  $\mathbf{N}(A^\top)$
- 4 Elimination and  $A = LU$
- 5 Orthogonal Matrices, Subspaces, and Projections

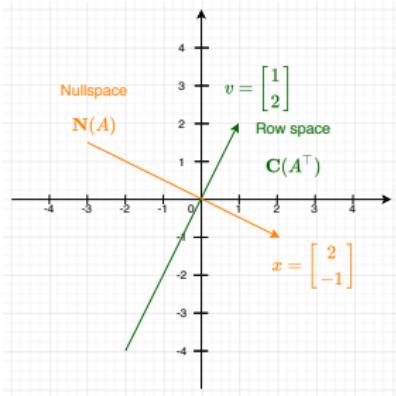


## Example 1

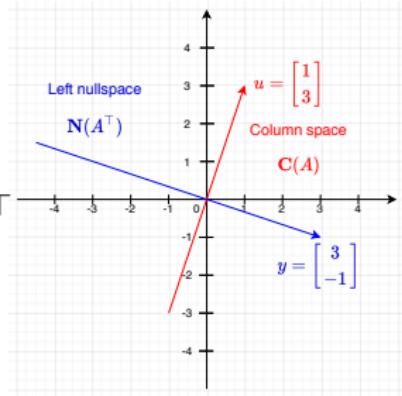
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} = \mathbf{u}\mathbf{v}^\top$$

- ▶ Column space  $\mathbf{C}(A)$  is the line through  $\mathbf{u} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ .
- ▶ Row space  $\mathbf{C}(A^\top)$  is the line through  $\mathbf{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ .
- ▶ Nullspace  $\mathbf{N}(A)$  is the line through  $\mathbf{x} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$ .  $A\mathbf{x} = \mathbf{0}$ .
- ▶ Left nullspace  $\mathbf{N}(A^\top)$  is the line through  $\mathbf{y} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$ .  $A^\top \mathbf{y} = \mathbf{0}$ .

# Example 1



$$A = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} = uv^\top$$



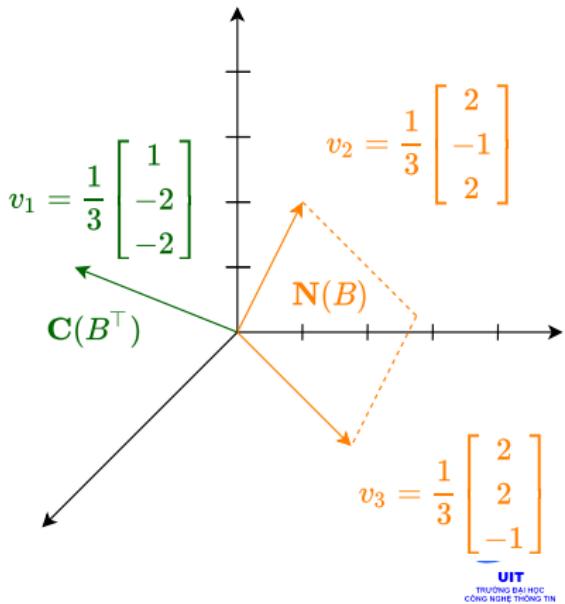
## Definition

The column space  $\mathbf{C}(A)$  contains all combinations of the columns of  $A$ .  
The row space  $\mathbf{C}(A^\top)$  contains all combinations of the columns of  $A^\top$ .  
The nullspace  $\mathbf{N}(A)$  contains all solutions  $x$  to  $Ax = \mathbf{0}$ .  
The left nullspace  $\mathbf{N}(A^\top)$  contains all solutions  $y$  to  $A^\top y = \mathbf{0}$ .

## Example 2

$$B = \begin{bmatrix} 1 & -2 & -2 \\ 3 & -6 & -6 \end{bmatrix}$$

- ▶ The row space  $\mathbf{C}(B^\top)$  is the infinite line through  $v_1 = \frac{1}{3}(1, -2, -2)$ .
- ▶  $Bx = \mathbf{0}$  has solutions  $x_1 = (2, 1, 0)$  and  $x_2 = (2, 0, 1)$ .
- ▶  $x_1$  and  $x_2$  are in the same plane with  $v_2 = \frac{1}{3}(2, -1, 2)$  and  $v_3 = \frac{1}{3}(2, 2, -1)$ .
- ▶ The nullspace  $\mathbf{N}(B)$  has an **orthonormal basis**  $v_2$  and  $v_3$ , is the infinite plane of  $v_2$  and  $v_3$ .
- ▶  $v_1, v_2, v_3$ : an orthonormal basis for  $\mathbb{R}^3$ .



# Subspaces of $A$

If  $Ax = \mathbf{0}$  then  $\begin{bmatrix} \text{row 1} \\ \vdots \\ \text{row } m \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

- ▶  $x$  is orthogonal to every row of  $A$ .
- ▶ Every  $x$  in the nullspace of  $A$  is orthogonal to the row space of  $A$ .
- ▶ Every  $y$  in the nullspace of  $A^\top$  is orthogonal to the column space of  $A$ .

$$\begin{array}{ccccc} \mathbf{N}(A) \perp \mathbf{C}(A^\top) & \mathbf{N}(A^\top) \perp \mathbf{C}(A) \\ \text{Dimensions} & n-r & r & m-r & r \end{array}$$

- ▶ Two orthogonal subspaces. The dimensions add to  $n$  and to  $m$ .

# Table of Contents

- 1 Matrix-Vector Multiplication  $Ax$
- 2 Matrix-Matrix Multiplication  $AB$
- 3 The Four Fundamental Subspaces of  $A$ :  $\mathbf{C}(A)$ ,  $\mathbf{C}(A^\top)$ ,  $\mathbf{N}(A)$ ,  $\mathbf{N}(A^\top)$
- 4 Elimination and  $A = LU$
- 5 Orthogonal Matrices, Subspaces, and Projections



# $Ax = b$ by Elimination

The usual order:

► Column 1.

- Row 1 is the first pivot row.
- Multiply row 1 by numbers  $l_{21}, l_{31}, \dots, l_{n1}$  and subtract from rows 2, 3, ..., n of A respectively.

$$\text{Multipliers } l_{21} = \frac{a_{21}}{a_{11}} \quad l_{31} = \frac{a_{31}}{a_{11}} \quad \dots \quad l_{n1} = \frac{a_{n1}}{a_{11}}$$

$$[A \mid b] = \left[ \begin{array}{cccc|c} 2 & 1 & -1 & 2 & 5 \\ 4 & 5 & -3 & 6 & 9 \\ -2 & 5 & -2 & 6 & 4 \\ 4 & 11 & -4 & 8 & 2 \end{array} \right] \rightarrow \left[ \begin{array}{cccc|c} 2 & 1 & -1 & 2 & 5 \\ 0 & 3 & -1 & 2 & -1 \\ 0 & 6 & -3 & 8 & 9 \\ 0 & 9 & -2 & 4 & -8 \end{array} \right]$$

# $Ax = b$ by Elimination

The usual order:

► Column 2.

- The **new** row 2 is the second pivot row.
- Multiply row 2 by numbers  $l_{32}, l_{42}, \dots, l_{n2}$  and subtract from rows 3, 4, ..., n of A respectively.

$$\text{Multipliers } l_{32} = \frac{a_{32}}{a_{22}} \quad l_{42} = \frac{a_{42}}{a_{22}} \quad \dots \quad l_{n2} = \frac{a_{n2}}{a_{22}}$$

$$\left[ \begin{array}{cccc|c} 2 & 1 & -1 & 2 & 5 \\ 0 & 3 & -1 & 2 & -1 \\ 0 & 6 & -3 & 8 & 9 \\ 0 & 9 & -2 & 4 & -8 \end{array} \right] \rightarrow \left[ \begin{array}{cccc|c} 2 & 1 & -1 & 2 & 5 \\ 0 & 3 & -1 & 2 & -1 \\ 0 & 0 & -1 & 4 & 11 \\ 0 & 0 & 1 & -2 & -5 \end{array} \right]$$

# $Ax = b$ by Elimination

The usual order:

► Column 3.

- The **new** row 3 is the third pivot row.
- Multiply row 3 by numbers  $l_{43}, l_{53}, \dots, l_{n3}$  and subtract from rows 4, 5, ..., n of A respectively.

$$\text{Multipliers } l_{43} = \frac{a_{43}}{a_{33}} \quad l_{53} = \frac{a_{53}}{a_{33}} \quad \dots \quad l_{n3} = \frac{a_{n3}}{a_{33}}$$

$$\left[ \begin{array}{cccc|c} 2 & 1 & -1 & 2 & 5 \\ 0 & 3 & -1 & 2 & -1 \\ 0 & 0 & -1 & 4 & 11 \\ 0 & 0 & 1 & -2 & -5 \end{array} \right] \rightarrow \left[ \begin{array}{cccc|c} 2 & 1 & -1 & 2 & 5 \\ 0 & 3 & -1 & 2 & -1 \\ 0 & 0 & -1 & 4 & 11 \\ 0 & 0 & 0 & 2 & 6 \end{array} \right] = [U \mid c]$$

- Columns 3 to n: Eliminating on A until obtaining the **upper triangular**  $U$ :  $n$  pivots on its **diagonal**.

# $Ax = b$ by Elimination

$$2x_1 + x_2 - x_3 + 2x_4 = 5$$

$$3x_2 - x_3 + 2x_4 = -1$$

$$- x_3 + 4x_4 = 11$$

$$2x_4 = 6$$

By back substitution, we get

$$x_4 = 3, \quad x_3 = 1, \quad x_2 = -2, \quad x_1 = 1$$

# Lower Triangular $L$ and Upper Triangular $U$

- ▶ Elimination on  $Ax = b$  produces the upper triangular matrix

$$U = \begin{bmatrix} 2 & 1 & -1 & 2 \\ 0 & 3 & -1 & 2 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

- ▶ and the lower triangular matrix

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -1 & 2 & 1 & 0 \\ 2 & 3 & -1 & 1 \end{bmatrix}$$

- ▶ Elimination factors  $A$  into a lower triangular  $L$  times an upper triangular  $U$ .

$$A = LU$$

# The Factorization $A = LU$

$$\begin{aligned} A &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} \text{pivot row 1} \\ \text{pivot row 2} \\ \text{pivot row 3} \\ \text{pivot row 4} \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 & 2 \\ 4 & 5 & -3 & 6 \\ -2 & 5 & -2 & 6 \\ 4 & 11 & -4 & 8 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ l_{21} \\ l_{31} \\ l_{41} \end{bmatrix} [\text{pivot row 1}] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 2 \\ -1 \\ 2 \end{bmatrix} [2 \ 1 \ -1 \ 2] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \\ &= \begin{bmatrix} 2 & 1 & -1 & 2 \\ 4 & 2 & -2 & 4 \\ -2 & -1 & 1 & -2 \\ 4 & 2 & -2 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & -1 & 2 \\ 0 & 6 & -3 & 8 \\ 0 & 9 & -2 & 4 \end{bmatrix} \end{aligned}$$

$$l_{ij} = \frac{a_{ij}}{a_{jj}}$$

The first step reduces the  $4 \times 4$  problem to a  $3 \times 3$  problem by removing  $l_1 u_1^*$ .



UIT  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN

# The Factorization $A = LU$

$$\begin{aligned} A &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} \text{pivot row 1} \\ \text{pivot row 2} \\ \text{pivot row 3} \\ \text{pivot row 4} \end{bmatrix} = l_1 u_1^* + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & -1 & 2 \\ 0 & 6 & -3 & 8 \\ 0 & 9 & -2 & 4 \end{bmatrix} \\ &= l_1 u_1^* + \begin{bmatrix} 0 \\ 1 \\ l_{32} \\ l_{42} \end{bmatrix} [\text{pivot row 2}] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} \quad l_{ij} = \frac{a_{ij}}{a_{jj}} \\ &= l_1 u_1^* + \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} [0 \ 3 \ -1 \ 2] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} \\ &= l_1 u_1^* + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & -1 & 2 \\ 0 & 6 & -2 & 4 \\ 0 & 9 & -3 & 6 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 1 & -2 \end{bmatrix} \end{aligned}$$

The second step reduces the  $3 \times 3$  problem to a  $2 \times 2$  problem by removing  $l_{22} u_2^*$



# The Factorization $A = LU$

$$\begin{aligned} A &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} \text{pivot row 1} \\ \text{pivot row 2} \\ \text{pivot row 3} \\ \text{pivot row 4} \end{bmatrix} = l_1 u_1^* + l_2 u_2^* + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 1 & -2 \end{bmatrix} \\ &= l_1 u_1^* + l_2 u_2^* + \begin{bmatrix} 0 \\ 0 \\ 1 \\ l_{43} \end{bmatrix} [\text{pivot row 3}] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x \end{bmatrix} \quad l_{ij} = \frac{a_{ij}}{a_{jj}} \\ &= l_1 u_1^* + l_2 u_2^* + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} [0 \ 0 \ -1 \ 4] + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x \end{bmatrix} \\ &= l_1 u_1^* + l_2 u_2^* + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & 1 & -4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \\ &= l_1 u_1^* + l_2 u_2^* + l_3 u_3^* + l_4 u_4^* \end{aligned}$$

The third step reduces the  $2 \times 2$  problem to a single number by removing  $l_3 u_3^*$ .

# Elimination and $A = LU$

- ▶ Start from  $[A \quad \mathbf{b}] = [LU \quad \mathbf{b}]$ .
- ▶ Elimination produces  $[U \quad L^{-1}\mathbf{b}] = [U \quad \mathbf{c}]$ .
- ▶ Elimination on  $A\mathbf{x} = \mathbf{b}$  produces the equation  $U\mathbf{x} = \mathbf{c}$  that are ready for back substitution.
- ▶  $A = LU = \sum l_i u_i^*$  = **sum of rank one matrices**.

# Table of Contents

- 1 Matrix-Vector Multiplication  $Ax$
- 2 Matrix-Matrix Multiplication  $AB$
- 3 The Four Fundamental Subspaces of  $A$ :  $\mathbf{C}(A)$ ,  $\mathbf{C}(A^\top)$ ,  $\mathbf{N}(A)$ ,  $\mathbf{N}(A^\top)$
- 4 Elimination and  $A = LU$
- 5 Orthogonal Matrices, Subspaces, and Projections



# Orthogonality

- ▶ Orthogonal  $\sim$  perpendicular.
- ▶ Orthogonal vectors  $x$  and  $y$ :

$$x^\top y = x_1y_1 + x_2y_2 + \cdots + x_ny_n = 0$$

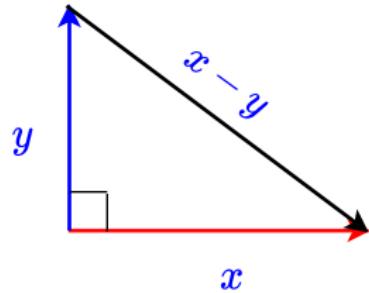
Law of Cosines:  $\theta$  is the angle between  $x$  and  $y$ :

$$\|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2\|x\|\|y\| \cos \theta$$

Orthogonal vectors have  $\cos \theta = 0$ .

Pythagoras Law:

$$\begin{aligned}\|x - y\|^2 &= \|x\|^2 + \|y\|^2 \\ (x - y)^\top (x - y) &= x^\top x + y^\top y \\ x^\top x + y^\top y - x^\top y - y^\top x &= x^\top x + y^\top y \\ x^\top y &= 0\end{aligned}$$



# Orthogonal Basis

- ▶ Orthogonal basis for a subspace: Every pair of basis vectors has  $\mathbf{v}_i^\top \mathbf{v}_j = 0$
- ▶ Orthonormal basis: Orthogonal basis of unit vectors: Every  $\mathbf{v}_i^\top \mathbf{v}_i = 1$  (length 1).
- ▶ From orthogonal to orthonormal, divide every basis vector  $\mathbf{v}_i$  by its length  $\|\mathbf{v}_i\|$ .
- ▶ The standard basis is orthogonal (and orthonormal) in  $\mathbb{R}^n$ :

Standard basis  $i, j, k$  in  $\mathbb{R}^3$      $i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$      $j = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$      $k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

- ▶ Every subspace of  $\mathbb{R}^n$  has an orthogonal basis.

# Orthogonal Subspaces

- Subspace **S** is orthogonal to subspace **T**: Every vector in **S** is orthogonal to every vector in **T**.



# Orthogonal Subspaces

- The row space  $\mathbf{C}(A^\top)$  is orthogonal to the nullspace  $\mathbf{N}(A)$ .

$$Ax = \begin{bmatrix} \text{row 1} \\ \vdots \\ \text{row } m \end{bmatrix} \begin{bmatrix} \mathbf{x} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

- The column space  $\mathbf{C}(A)$  is orthogonal to the left nullspace  $\mathbf{N}(A^\top)$ .

$$A^\top \mathbf{y} = \begin{bmatrix} (\text{column 1})^\top \\ \vdots \\ (\text{column } m)^\top \end{bmatrix} \begin{bmatrix} \mathbf{y} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

# Orthogonal Subspaces

- ▶ Every vector  $\mathbf{v}$  in  $\mathbb{R}^n$  has a row space component  $\mathbf{v}_{row}$  and a nullspace component  $\mathbf{v}_{null}$ :  $\mathbf{v} = \mathbf{v}_{row} + \mathbf{v}_{null}$

$$A\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- ▶ The row space  $\mathbf{C}(A^\top)$  is the plane of all vectors  $\beta_1 \mathbf{a}_1^* + \beta_2 \mathbf{a}_2^*$ .
- ▶ The nullspace  $\mathbf{N}(A)$  is the line through  $\mathbf{u} = (0, 0, 1)$ : all vectors  $\beta_3 \mathbf{u}$

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \in \mathbb{R}^3 \quad \mathbf{v} = \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}_{\mathbf{v}_{row}} = \underbrace{\beta_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{v}_{row}} + \underbrace{\beta_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}_{\mathbf{v}_{row}} + \underbrace{\beta_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{v}_{null}}$$

- ▶ Dimensions:  $\dim \mathbf{C}(A^\top) + \dim \mathbf{N}(A) = r + (n - r) = n$ .
- ▶ A row space basis ( $r$  vectors) and a nullspace basis ( $n - r$  vectors) produces a basis for the whole  $\mathbb{R}^n$  ( $n$  vectors).

# The Big Picture

## Fundamental Theorem in Linear Algebra

The row space and nullspace of  $A$  are orthogonal complements in  $\mathbb{R}^n$ .

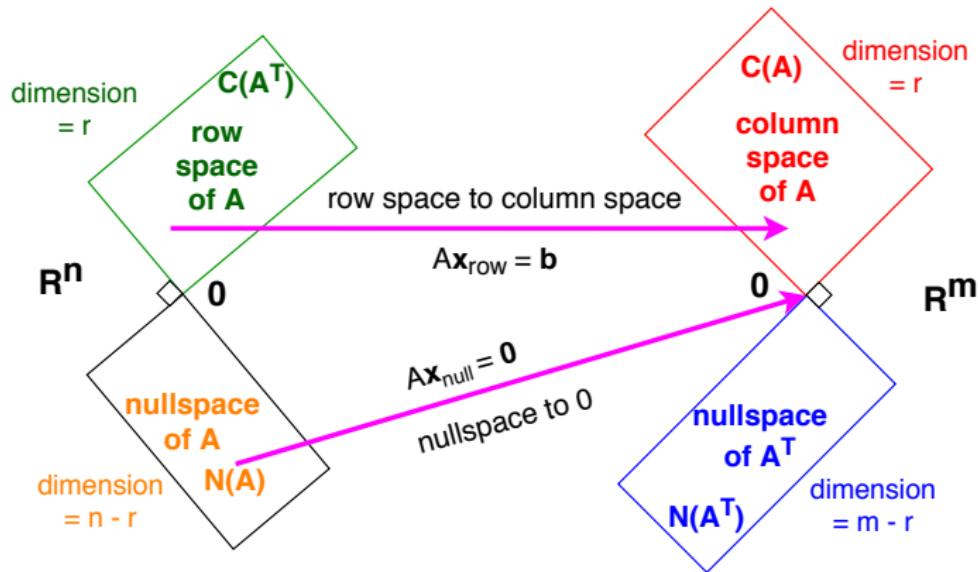


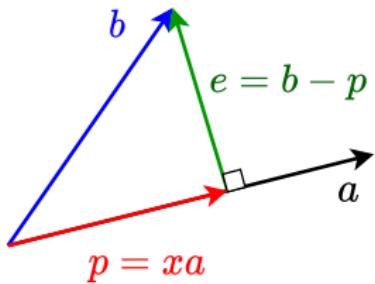
Figure: Two pairs of orthogonal subspaces.

# Projection onto a Line

►  $e = b - p$

►  $p = x\mathbf{a}$

► Because  $e$  is orthogonal to  $\mathbf{a}$ :



$$\mathbf{a}^\top e = 0$$

$$\mathbf{a}^\top (\mathbf{b} - \mathbf{p}) = 0$$

$$\mathbf{a}^\top (\mathbf{b} - x\mathbf{a}) = 0$$

$$x\mathbf{a}^\top \mathbf{a} = \mathbf{a}^\top \mathbf{b}$$

$$x = \frac{\mathbf{a}^\top \mathbf{b}}{\mathbf{a}^\top \mathbf{a}}$$

► Therefore,  $p = ax = a \frac{\mathbf{a}^\top \mathbf{b}}{\mathbf{a}^\top \mathbf{a}}$

► There is a **projection matrix**  $P$  that  $p = Pb$ .

$$P = \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}}$$

# Projection onto a Line

$$P = \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}}$$

- ▶ Column space of  $A$ : matrix-vector multiplication  $Ax \in \mathbf{C}(A)$ .
- ▶  $\mathbf{p} = Pb$ . What is the column space  $\mathbf{C}(P)$ ?
- ▶  $\mathbf{C}(P)$  is the line through  $\mathbf{a}$ .
- ▶ Is  $P$  symmetric?

$$P^\top = \left( \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}} \right)^\top = \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}} = P. \quad \text{Yes.}$$

- ▶ What if we project  $\mathbf{b}$  twice?

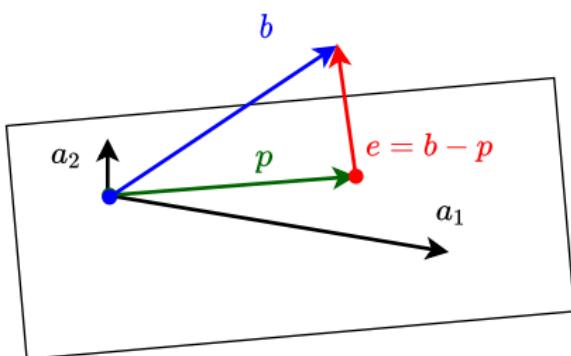
$$P^2 = \left( \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}} \right) \left( \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}} \right) = \frac{\mathbf{a}\mathbf{a}^\top}{\mathbf{a}^\top \mathbf{a}} = P$$

# Projection onto a Subspace

- ▶ Why bother with projection?
- ▶ Because  $Ax = b$  may have no solution ( $m \gg n$ ).  $b$  might not in the column space  $\mathbf{C}(A)$ .
- ▶ Solve  $A\hat{x} = p$  instead, where  $p$  is the projection of  $b$  onto the column space  $\mathbf{C}(A)$ .

# Projection onto a Subspace

- ▶ Choose two independent vectors  $\mathbf{a}_1, \mathbf{a}_2$  in the plane to form a basis.



$$A = \begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix}$$

- ▶ Plane of  $\mathbf{a}_1, \mathbf{a}_2$  = Column space of  $A$ .
- ▶  $\mathbf{p}$  is a linear combination of  $\mathbf{a}_1, \mathbf{a}_2$ .

$$\begin{aligned} \mathbf{p} &= \hat{x}_1 \mathbf{a}_1 + \hat{x}_2 \mathbf{a}_2 \\ &= A\hat{\mathbf{x}} \end{aligned}$$

- ▶ Find  $\hat{\mathbf{x}}$ .

# Projection onto a Subspace

- $p = A\hat{x}$ . Find  $\hat{x}$ .
- $e = b - p$  is perpendicular to the plane.

$$\begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \end{bmatrix} \begin{bmatrix} | \\ \mathbf{e} \\ | \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$A^\top \mathbf{e} = \mathbf{0}$$

$$A^\top(b - A\hat{x}) = \mathbf{0}$$

$$A^\top A\hat{x} = A^\top \mathbf{b}$$

$$\hat{x} = (A^\top A)^{-1} A^\top \mathbf{b}$$

- We have  $p = A\hat{x} = A(A^\top A)^{-1} A^\top \mathbf{b}$ .
- The projection matrix  $P$ :

$$P = A(A^\top A)^{-1} A^\top$$

# Projection onto a Subspace

$$P = A(A^\top A)^{-1}A^\top$$

- Is  $P$  symmetric?

$$\begin{aligned} P^\top &= (A(A^\top A)^{-1}A^\top)^\top = A((A^\top A)^{-1})^\top A^\top \\ &= A((A^\top A)^\top)^{-1}A^\top \\ &= A(A^\top A)^{-1}A^\top = P \end{aligned}$$

Yes.

- Is  $P^2 = P$ ?

$$\begin{aligned} P^2 &= A(A^\top A)^{-1}A^\top A(A^\top A)^{-1}A^\top \\ &= A(A^\top A)^{-1}(A^\top A)(A^\top A)^{-1}A^\top \\ &= A(A^\top A)^{-1}A^\top = P \end{aligned}$$

Yes.

# $Q$ with Orthonormal Columns

$$Q_1 = \frac{1}{3} \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} \quad Q_1^\top Q_1 = [1]$$

$$Q_2 = \frac{1}{3} \begin{bmatrix} 2 & 2 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \quad Q_2^\top Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Q_3 = \frac{1}{3} \begin{bmatrix} 2 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 2 & 2 \end{bmatrix} \quad Q_3^\top Q_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- ▶ Columns of  $Q$ 's are orthonormal.
- ▶ Each one of those matrices has  $Q^\top Q = I$ .
- ▶  $Q^\top$  is a **left inverse** of  $Q$ .
- ▶  $Q_3 Q_3^\top = I$ .  $Q_3^\top$  is also a **right inverse**.

# Orthogonal Projection

- ▶ All the matrices  $P = QQ^\top$  have  $P^T = P$ .

$$P^\top = (QQ^\top)^\top = QQ^\top = P$$

- ▶ All the matrices  $P = QQ^\top$  have  $P^2 = P$ .

$$P^2 = (QQ^\top)(QQ^\top) = Q(Q^\top Q)Q^\top = QQ^\top = P$$

- ▶  $P$  is a **projection matrix**.

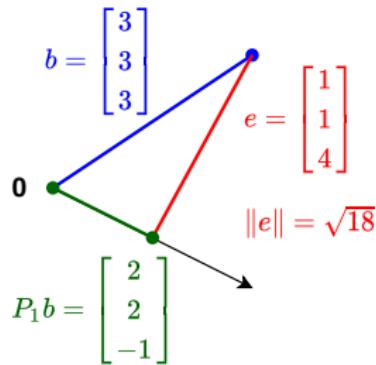
## Orthogonal Projection

If  $P^2 = P = P^\top$  then  $P\mathbf{b}$  is the orthogonal projection of  $\mathbf{b}$  onto the column space of  $P$ .

# Orthogonal Projection

- ▶ Project  $\mathbf{b} = (3, 3, 3)$  on the  $Q_1$  line.  $P_1 = Q_1 Q_1^\top$

$$P_1 \mathbf{b} = \frac{1}{9} \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} [2 \quad 2 \quad -1] \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} 9 = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$

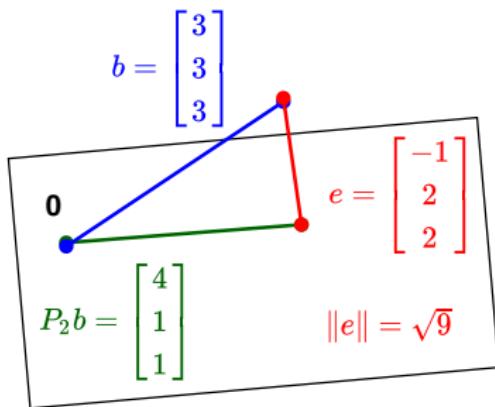


- ▶  $P_1$  splits  $\mathbf{b}$  in 2 perpendicular parts: projection  $P_1 \mathbf{b}$  and error  $\mathbf{e} = \mathbf{b} - P_1 \mathbf{b}$

# Orthogonal Projection

- ▶ Project  $\mathbf{b} = (3, 3, 3)$  on the  $Q_2$  plane.  $P_2 = Q_2 Q_2^\top$

$$P_2 \mathbf{b} = \frac{1}{9} \begin{bmatrix} 2 & 2 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 2 & -1 \\ 2 & -1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 2 & 2 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 9 \\ 9 \\ 9 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix}$$



- ▶  $P_2$  projects  $\mathbf{b}$  on the column space of  $Q_2$ .
- ▶ The error vector  $\mathbf{b} - P_2 \mathbf{b}$  is shorter than  $\mathbf{b} - P_1 \mathbf{b}$ .

# Orthogonal Projection

$$Q_3 = \frac{1}{3} \begin{bmatrix} 2 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 2 & 2 \end{bmatrix}$$

- ▶ What is  $P_3\mathbf{b} = Q_3Q_3^\top\mathbf{b}$  ?
- ▶ Project  $\mathbf{b}$  onto the whole space  $\mathbb{R}^3$ .
- ▶  $P_3 = Q_3Q_3^\top = I$ . Thus,  $P_3\mathbf{b} = \mathbf{b}$ . Vector  $\mathbf{b}$  is in  $\mathbb{R}^3$  already.
- ▶ The error  $e$  is **zero!!!**

# Orthogonalization

- ▶ Determine if a list of vectors  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$  is linearly independent.

## Gram-Smidt algorithm

given vectors  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$

for  $i = 1, \dots, k$

- ① Orthogonalization.  $\tilde{\mathbf{q}}_i = \mathbf{a}_i - (\mathbf{q}_1^\top \mathbf{a}_i) \mathbf{q}_1 - \dots - (\mathbf{q}_{i-1}^\top \mathbf{a}_i) \mathbf{q}_{i-1}$
- ② Test for linear dependence. If  $\tilde{\mathbf{q}}_i = 0$ , quit.
- ③ Normalization.  $\mathbf{q}_i = \tilde{\mathbf{q}}_i / \|\tilde{\mathbf{q}}_i\|$

- ▶ If the vectors are **linearly independent**, the Gram-Smidt algorithm produces an **orthonormal** collection of vectors  $\mathbf{q}_1, \dots, \mathbf{q}_k$ .
- ▶ If the vectors  $\mathbf{a}_1, \dots, \mathbf{a}_{j-1}$  are linearly independent, but  $\mathbf{a}_1, \dots, \mathbf{a}_j$  are linearly dependent, the algorithm detects this and terminates.

# Orthogonalization: Example

$$\mathbf{a}_1 = (-1, 1, -1, 1), \quad \mathbf{a}_2 = (-1, 3, -1, 3), \quad \mathbf{a}_3 = (1, 3, 5, 7)$$

Applying the Gram-Smidt algorithm gives the following results.

- $i = 1$ :

$$\tilde{\mathbf{q}}_1 = \mathbf{a}_1$$

$$\mathbf{q}_1 = \frac{1}{\|\tilde{\mathbf{q}}_1\|} \tilde{\mathbf{q}}_1 = (-1/2, 1/2, -1/2, 1/2)$$

- $i = 2$ :

$$\begin{aligned}\tilde{\mathbf{q}}_2 &= \mathbf{a}_2 - (\mathbf{q}_1^T \mathbf{a}_2) \mathbf{q}_1 \\ &= (-1, 3, -1, 3) - 4(-1/2, 1/2, -1/2, 1/2) = (1, 1, 1, 1)\end{aligned}$$

$$\mathbf{q}_2 = \frac{1}{\|\tilde{\mathbf{q}}_2\|} \tilde{\mathbf{q}}_2 = (1/2, 1/2, 1/2, 1/2)$$

## Orthogonalization: Example

- $i = 3$ :

$$\begin{aligned}\tilde{\mathbf{q}}_3 &= \mathbf{a}_3 - (\mathbf{q}_1^\top \mathbf{a}_3) \mathbf{q}_1 - (\mathbf{q}_2^\top \mathbf{a}_3) \mathbf{q}_2 \\ &= \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \end{bmatrix} - 2 \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} - 8 \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix} \\ \mathbf{q}_3 &= \frac{1}{\|\tilde{\mathbf{q}}_3\|} \tilde{\mathbf{q}}_3 = (-1/2, -1/2, 1/2, 1/2)\end{aligned}$$

- The completion of the Gram-Smidt algorithm without early termination indicates that the vectors  $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$  are linearly independent.

# QR factorization: $A = QR$

$$A = QR$$

$$\begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \dots & \mathbf{q}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{nn} \end{bmatrix}$$

$$r_{kk} = \|\tilde{\mathbf{q}}_k\|$$

$$r_{k-1,k} = \mathbf{q}_{k-1}^\top \mathbf{a}_k$$

## QR factorization: $A = QR$

$$\begin{aligned}\hat{x} &= (A^T A)^{-1} A^T b \\&= ((QR)^T (QR))^{-1} (QR)^T b \\&= (R^T Q^T QR)^{-1} R^T Q^T b \\&= (R^T R)^{-1} R^T Q^T b \quad (\text{because } Q^T Q = I) \\&= R^{-1} R^{-T} R^T Q^T b \\&= R^{-1} Q^T b\end{aligned}$$

Solving for  $\hat{x}$  by solving  $R\hat{x} = Q^T b$  with back-substitution.

$$\frac{\partial L_i}{\partial \theta} = \frac{\partial y_i}{\partial \theta} \frac{\partial L_i}{\partial y_i}$$

$$\frac{\partial L_i}{\partial y_i} = \frac{\partial(t_i - y_i)^2}{\partial y_i} = -2(t_i - y_i) = 2(y_i - t_i)$$

$$\frac{\partial y_i}{\partial \theta} = \frac{\partial(x_i \cdot \theta)}{\partial \theta} = x_i$$

$$\frac{\partial L_i}{\partial \theta} = 2x_i(y_i - t_i)$$

## Performance Evaluation

Mean Absolute Error (MAE)

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_i |y^{(i)} - \hat{y}^{(i)}| \quad (6)$$

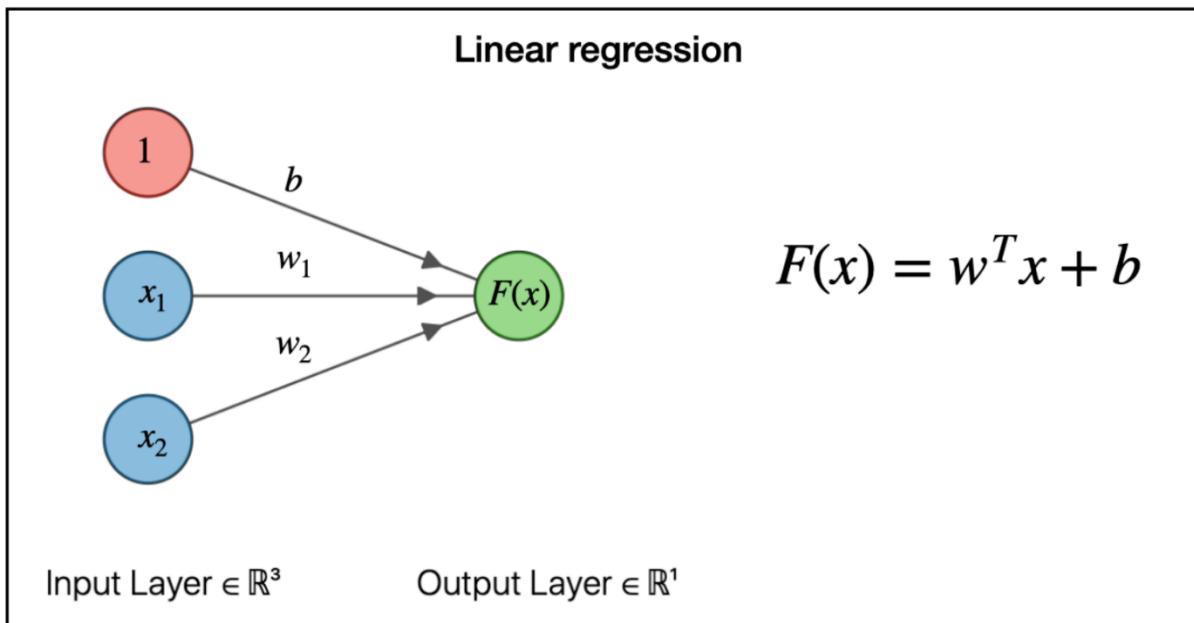
Mean Squared Error (MSE)

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_i (y^{(i)} - \hat{y}^{(i)})^2 \quad (7)$$

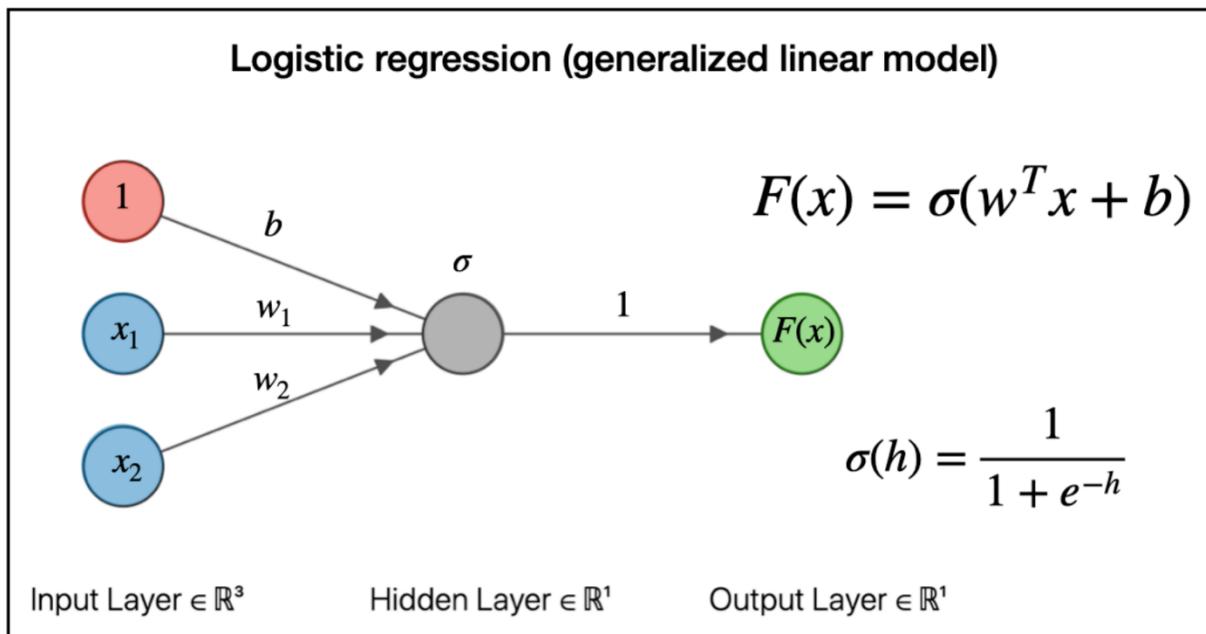
R-Squared (R2)

$$R^2(y, \hat{y}) = 1 - \frac{\sum_i (y^{(i)} - \hat{y}^{(i)})^2}{\sum_i (y^{(i)} - \bar{y})^2} \quad (8)$$

## LINEAR REGRESSION



## LOGISTIC REGRESSION



$$a = g(z) = \frac{1}{1 + e^{-z}}$$

# Maximum Likelihood Estimation (MLE) and Loss Function of Logistic Regression

$$a^y \cdot (1 - a)^{(1-y)}$$

ing:

$$\log(a^y \cdot (1 - a)^{(1-y)}) = \log(a^y) + \log((1 - a)^{(1-y)}) = y \log(a) + (1 - y) \log(1 - a)$$

$$L = -(y \log(a) + (1 - y) \log(1 - a))$$

## Neural network

### ^ Cross Entropy Loss Function

We want to find parameter values of our neural network that minimizes a **cost function**. We use the same cost function (or loss function) as in Logistic Regression or Softmax Regression.

For the neural network in this example, we have the cost function

$$J = -\sum_{i=1}^N [y^{(i)} \log(a^{[3](i)}) + (1 - y^{(i)}) \log(1 - a^{[3](i)})]$$

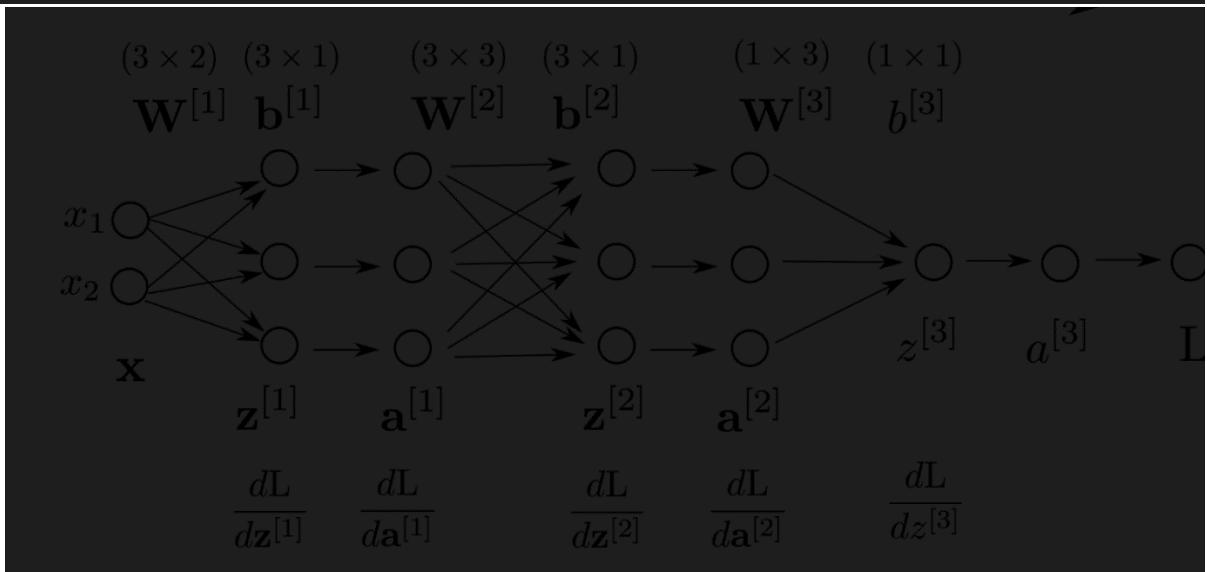
Because we have only two classes, this loss function is also called **Binary Cross Entropy Loss**.

If we have more than two classes, we use the general **Cross Entropy Loss** function.

$$J = -\sum_{i=1}^N \sum_{j=1}^C y_j^{(i)} \log(a_j^{[3]})^{y_j^{(i)}}$$

If we want to compute the loss function for a sample  $i$  in our dataset.

$$L = -(y^{(i)} \log(a^{[3](i)}) + (1 - y^{(i)}) \log(1 - a^{[3](i)}))$$



For the sake of simplicity, we will use only **one sample** here. So we can drop the notation  $(i)$ .

In Logistic Regression, we already computed

$$\frac{dL}{dz^{[3]}} = a^{[3]} - y$$

Next, we compute  $\frac{dz^{[3]}}{d\mathbf{W}^{[3]}}$  and  $\frac{dz^{[3]}}{db^{[3]}}$ .

$$z^{[3]} = \mathbf{W}^{[3]} \mathbf{a}^{[2]} + b^{[3]}$$

$$[z^{[3]}] = \begin{bmatrix} w_{1,1}^{[3]} & w_{1,2}^{[3]} & w_{1,3}^{[3]} \end{bmatrix} \begin{bmatrix} a_1^{[2]} \\ a_2^{[2]} \\ a_3^{[2]} \end{bmatrix} + [b^{[3]}] = w_{1,1}^{[3]} a_1^{[2]} + w_{1,2}^{[3]} a_2^{[2]} + w_{1,3}^{[3]} a_3^{[2]} + b^{[3]}$$

We can see that:

- If we change  $w_{1,1}^{[3]}$ ,  $z^{[3]}$  will change proportionally to  $a_1^{[2]}$ .
- If we change  $w_{1,2}^{[3]}$ ,  $z^{[3]}$  will change proportionally to  $a_2^{[2]}$ .
- If we change  $w_{1,3}^{[3]}$ ,  $z^{[3]}$  will change proportionally to  $a_3^{[2]}$ .
- Thus, if we change  $\mathbf{W}^{[3]}$ ,  $z^{[3]}$  will change proportionally to  $\mathbf{a}^{[2]}$ .

Therefore,

$$\frac{dz^{[3]}}{d\mathbf{W}^{[3]}} = \begin{bmatrix} \frac{dz^{[3]}}{dw_{1,1}^{[3]}} & \frac{dz^{[3]}}{dw_{1,2}^{[3]}} & \frac{dz^{[3]}}{dw_{1,3}^{[3]}} \end{bmatrix} = \begin{bmatrix} a_1^{[2]} & a_2^{[2]} & a_3^{[2]} \end{bmatrix} = (\mathbf{a}^{[2]})^T$$

Similarly, if we change  $b^{[3]}$  a small amount,  $z^{[3]}$  will also change the same amount in the same direction.

$$\frac{dz^{[3]}}{db^{[3]}} = 1.$$

Now, we can have

$$\frac{dL}{d\mathbf{W}^{[3]}} = \frac{dL}{dz^{[3]}} \frac{dz^{[3]}}{d\mathbf{W}^{[3]}} = (a^{[3]} - y)(\mathbf{a}^{[2]})^T$$

$$\frac{dL}{db^{[3]}} = \frac{dL}{dz^{[3]}} \frac{dz^{[3]}}{db^{[3]}} = (a^{[3]} - y)$$

Next, we need to calculate  $\frac{dL}{d\mathbf{W}^{[2]}}$ ,  $\frac{dL}{d\mathbf{b}^{[2]}}$ .

We need to compute  $\frac{dL}{d\mathbf{a}^{[2]}} = \frac{dL}{dz^{[3]}} \frac{dz^{[3]}}{d\mathbf{a}^{[2]}}$

$$z^{[3]} = \mathbf{W}^{[3]} \mathbf{a}^{[2]} + b^{[3]}$$

$$\begin{bmatrix} z^{[3]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[3]} & w_{1,2}^{[3]} & w_{1,3}^{[3]} \end{bmatrix} \begin{bmatrix} a_1^{[2]} \\ a_2^{[2]} \\ a_3^{[2]} \end{bmatrix} + \begin{bmatrix} b^{[3]} \end{bmatrix} = w_{1,1}^{[3]} a_1^{[2]} + w_{1,2}^{[3]} a_2^{[2]} + w_{1,3}^{[3]} a_3^{[2]} + b^{[3]}$$

$$\frac{dz^{[3]}}{d\mathbf{a}^{[2]}} = \begin{bmatrix} \frac{dz^{[3]}}{da_1^{[2]}} \\ \frac{dz^{[3]}}{da_2^{[2]}} \\ \frac{dz^{[3]}}{da_3^{[2]}} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[3]} \\ w_{1,2}^{[3]} \\ w_{1,3}^{[3]} \end{bmatrix} = (\mathbf{W}^{[3]})^T$$

$$\text{Therefore, } \frac{dL}{d\mathbf{a}^{[2]}} = \frac{dL}{dz^{[3]}} \frac{dz^{[3]}}{d\mathbf{a}^{[2]}} = (a^{[3]} - y)(\mathbf{W}^{[3]})^T$$

Next, we compute  $\frac{dL}{d\mathbf{z}^{[2]}}$ .

We have

$$\mathbf{a}^{[2]} = g(\mathbf{z}^{[2]}), \text{ where } g(z) = \frac{1}{1+e^{-z}} \text{ is the sigmoid function.}$$

That is,

$$a_1^{[2]} = g(z_1^{[2]}) = \frac{1}{1+e^{-z_1^{[2]}}}$$

$$a_2^{[2]} = g(z_2^{[2]}) = \frac{1}{1+e^{-z_2^{[2]}}}$$

$$a_3^{[2]} = g(z_3^{[2]}) = \frac{1}{1+e^{-z_3^{[2]}}}$$

In Logistic Regression, we already derived the derivative of the sigmoid function  $\frac{da}{dz} = a(1 - a)$ .

$$\begin{bmatrix} \frac{da_1^{[2]}}{dz_1^{[2]}} \\ \frac{da_2^{[2]}}{dz_2^{[2]}} \\ \frac{da_3^{[2]}}{dz_3^{[2]}} \end{bmatrix}$$

Therefore,

$$\frac{dL}{d\mathbf{z}^{[2]}} = \begin{bmatrix} \frac{dL}{dz_1^{[2]}} \\ \frac{dL}{dz_2^{[2]}} \\ \frac{dL}{dz_3^{[2]}} \end{bmatrix} = \begin{bmatrix} \frac{dL}{da_1^{[2]}} \frac{da_1^{[2]}}{dz_1^{[2]}} \\ \frac{dL}{da_2^{[2]}} \frac{da_2^{[2]}}{dz_2^{[2]}} \\ \frac{dL}{da_3^{[2]}} \frac{da_3^{[2]}}{dz_3^{[2]}} \end{bmatrix} = \frac{dL}{d\mathbf{a}^{[2]}} \circ (\mathbf{a}^{[2]} \circ (1 - \mathbf{a}^{[2]}))$$

Now, we can calculate  $\frac{dL}{d\mathbf{W}^{[2]}}$  and  $\frac{dL}{d\mathbf{b}^{[2]}}$ .

We have

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]} \mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$

$$\begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \\ z_3^{[2]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ b_3^{[2]} \end{bmatrix}$$

$$\begin{bmatrix} \frac{dL}{dz_1^{[2]}} \frac{dz_1^{[2]}}{dw_{1,1}^{[2]}} & \frac{dL}{dz_1^{[2]}} \frac{dz_1^{[2]}}{dw_{1,2}^{[2]}} & \frac{dL}{dz_1^{[2]}} \frac{dz_1^{[2]}}{dw_{1,3}^{[2]}} \\ \frac{dL}{dz_2^{[2]}} \frac{dz_2^{[2]}}{dw_{2,1}^{[2]}} & \frac{dL}{dz_2^{[2]}} \frac{dz_2^{[2]}}{dw_{2,2}^{[2]}} & \frac{dL}{dz_2^{[2]}} \frac{dz_2^{[2]}}{dw_{2,3}^{[2]}} \\ \frac{dL}{dz_3^{[2]}} \frac{dz_3^{[2]}}{dw_{3,1}^{[2]}} & \frac{dL}{dz_3^{[2]}} \frac{dz_3^{[2]}}{dw_{3,2}^{[2]}} & \frac{dL}{dz_3^{[2]}} \frac{dz_3^{[2]}}{dw_{3,3}^{[2]}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{dL}{dz_1^{[2]}} \\ \frac{dL}{dz_2^{[2]}} \\ \frac{dL}{dz_3^{[2]}} \end{bmatrix}$$

Next, we compute  $\frac{dL}{d\mathbf{a}^{[1]}} = \frac{dL}{d\mathbf{z}^{[2]}} \frac{d\mathbf{z}^{[2]}}{d\mathbf{a}^{[1]}}$ .

Again, we have

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$

$$\begin{aligned} \begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \\ z_3^{[2]} \end{bmatrix} &= \begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ b_3^{[2]} \end{bmatrix} \\ \frac{d\mathbf{z}^{[2]}}{d\mathbf{a}^{[1]}} &= \begin{bmatrix} \frac{dz_1^{[2]}}{da_1^{[1]}} & \frac{dz_2^{[2]}}{da_1^{[1]}} & \frac{dz_3^{[2]}}{da_1^{[1]}} \\ \frac{dz_1^{[2]}}{da_2^{[1]}} & \frac{dz_2^{[2]}}{da_2^{[1]}} & \frac{dz_3^{[2]}}{da_2^{[1]}} \\ \frac{dz_1^{[2]}}{da_3^{[1]}} & \frac{dz_2^{[2]}}{da_3^{[1]}} & \frac{dz_3^{[2]}}{da_3^{[1]}} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} & w_{1,3}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} & w_{2,3}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} & w_{3,3}^{[2]} \end{bmatrix} = (\mathbf{W}^{[2]})^T \end{aligned}$$

We then have

$$\frac{dL}{d\mathbf{a}^{[1]}} = \frac{d\mathbf{z}^{[2]}}{d\mathbf{a}^{[1]}} \frac{dL}{d\mathbf{z}^{[2]}} = (\mathbf{W}^{[2]})^T \frac{dL}{d\mathbf{z}^{[2]}}$$

Similarly as above, we can compute  $\frac{dL}{d\mathbf{z}^{[1]}}$  as well.

Now, we proceed to compute  $\frac{dL}{d\mathbf{W}^{[1]}}$ .

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix}$$

Similarly as above, we have  $\frac{dL}{d\mathbf{W}^{[1]}} = \frac{dL}{d\mathbf{z}^{[1]}} (\mathbf{a}^{[0]})^T = \frac{dL}{d\mathbf{z}^{[1]}} \mathbf{x}^T$

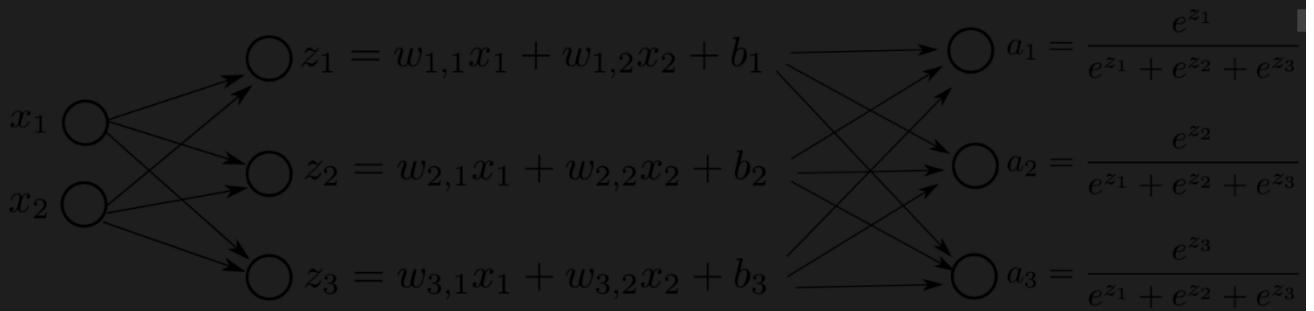
and

$$\frac{dL}{d\mathbf{b}^{[1]}} = \frac{dL}{d\mathbf{z}^{[1]}}.$$

Now, we already had all the gradients. We can do backpropagation.

## Softmax backpropagation

### Computation Graph



The parameters of our Softmax Regression model are:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

We need to **learn** these parameters of the Softmax Regression model. The process of finding out the appropriate values for these parameters from our dataset so that the predictions of the model is as close as possible to the targets is called **training**.

### Softmax

After getting  $\mathbf{z}$ , we apply the softmax function to compute  $\mathbf{a}$ .

$$\mathbf{a} = \text{softmax}(\mathbf{z})$$

$$a_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}$$

Here, because  $\mathbf{z}$  is of size  $(3 \times 1)$ ,  $\mathbf{a}$  is also of size  $(3 \times 1)$ , i.e.,  $(a_1, a_2, a_3)$ .

The softmax function above produces a **probability distribution**, i.e.,  $\sum a_i = 1$ .

In classification, we want to compute the probability that a sample belongs to a class/category.

### Numerically stable version

We can compute  $\mathbf{a}$  as follows.

$$a_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}} = \frac{e^{z_i} e^K}{\sum_{j=1}^C e^{z_j} e^K} = \frac{e^{z_i + K}}{\sum_{j=1}^C e^{z_j + K}}$$

$$\text{where } K = -\max(z_1, z_2, \dots, z_C).$$

Similar to the case of **Logistic Regression**, we need to apply **Maximum Likelihood Estimation** (MLE) for **multiple examples**.

We need to find weights  $\mathbf{W}$  and bias  $\mathbf{b}$  to maximize the following.

$$\prod_{i=1}^N \prod_{j=1}^3 (a_j^{(i)})^{y_j^{(i)}} \text{ where } \mathbf{a} = \text{softmax}(\mathbf{W}\mathbf{x}^{(i)} + \mathbf{b})$$

$\mathbf{x}^{(i)}, \mathbf{y}^{(i)}$  is the  $i$ -th example in our dataset of  $N$  training examples.

Maximizing the above is equal to maximizing the following:

$$\log \left( \prod_{i=1}^N \prod_{j=1}^3 (a_j^{(i)})^{y_j^{(i)}} \right) = \sum_{i=1}^N \sum_{j=1}^3 \log(a_j^{(i)})^{y_j^{(i)}} = \sum_{i=1}^N \sum_{j=1}^3 y_j^{(i)} \log(a_j^{(i)})$$

Maximizing the above is similar to minimize the following:

$$J = - \sum_{i=1}^N \sum_{j=1}^3 y_j^{(i)} \log(a_j^{(i)})$$

which is the **loss function** (cost function/log loss function) for our Softmax Regression problem here.

This loss function can also be called **Cross Entropy Loss** function. In Logistic Regression (binary classification), this is **Binary Cross Entropy** loss.

In this example, we have only 3 classes **red, green, blue**. In the general case, where we need to make prediction for  $C$  classes, the loss function of Softmax Regression would be:  $J = - \sum_{i=1}^N \sum_{j=1}^C y_j^{(i)} \log(a_j^{(i)})$

## Numerical Stability of Loss Function

The loss function for one sample is

$$L = - \sum_{j=1}^C y_j \log(a_j)$$

Finally, we have

$$\frac{dL}{d\mathbf{W}} = \frac{dL}{d\mathbf{z}} \mathbf{x}^T = (\mathbf{a} - \mathbf{y}) \mathbf{x}^T$$

and

$$\frac{dL}{d\mathbf{b}} = \frac{dL}{d\mathbf{z}} = (\mathbf{a} - \mathbf{y})$$

# Optimization

Ngoc-Hoang Luong

University of Information Technology (UIT)  
Vietnam National University - Ho Chi Minh City (VNU-HCM)

Math for CS, Fall 2021

# References

The contents of this document are taken mainly from the follow sources:

- Kevin P. Murphy. Probabilistic Machine Learning: An Introduction.<sup>1</sup>

---

<sup>1</sup><https://probml.github.io/pml-book/book1.html>

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Introduction

- The core problem in ML is parameter estimation (model fitting).

# Introduction

- The core problem in ML is parameter estimation (model fitting).
- We need to solve an **optimization problem**: i.e., trying to find the values for a set of variables  $\theta \in \Theta$  that minimize a scalar-valued **loss function** or **cost function**:  $\mathcal{L} : \Theta \rightarrow \mathbb{R}$

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta) \quad (1)$$

# Introduction

- The core problem in ML is parameter estimation (model fitting).
- We need to solve an **optimization problem**: i.e., trying to find the values for a set of variables  $\theta \in \Theta$  that minimize a scalar-valued **loss function** or **cost function**:  $\mathcal{L} : \Theta \rightarrow \mathbb{R}$

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta) \quad (1)$$

- The **parameter space** is given by  $\Theta \in \mathbb{R}^D$ , where  $D$  is the number of variables being optimized.

# Introduction

- The core problem in ML is parameter estimation (model fitting).
- We need to solve an **optimization problem**: i.e., trying to find the values for a set of variables  $\theta \in \Theta$  that minimize a scalar-valued **loss function** or **cost function**:  $\mathcal{L} : \Theta \rightarrow \mathbb{R}$

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta) \quad (1)$$

- The **parameter space** is given by  $\Theta \in \mathbb{R}^D$ , where  $D$  is the number of variables being optimized.
- We focus on **continuous optimization**.

# Introduction

- The core problem in ML is parameter estimation (model fitting).
- We need to solve an **optimization problem**: i.e., trying to find the values for a set of variables  $\theta \in \Theta$  that minimize a scalar-valued **loss function** or **cost function**:  $\mathcal{L} : \Theta \rightarrow \mathbb{R}$

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta) \quad (1)$$

- The **parameter space** is given by  $\Theta \in \mathbb{R}^D$ , where  $D$  is the number of variables being optimized.
- We focus on **continuous optimization**.
- To maximize a **score function** or **reward function**  $R(\theta)$ , we can minimize  $\mathcal{L}(\theta) = -R(\theta)$ .

- The core problem in ML is parameter estimation (model fitting).
- We need to solve an **optimization problem**: i.e., trying to find the values for a set of variables  $\theta \in \Theta$  that minimize a scalar-valued **loss function** or **cost function**:  $\mathcal{L} : \Theta \rightarrow \mathbb{R}$

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta) \quad (1)$$

- The **parameter space** is given by  $\Theta \in \mathbb{R}^D$ , where  $D$  is the number of variables being optimized.
- We focus on **continuous optimization**.
- To maximize a **score function** or **reward function**  $R(\theta)$ , we can minimize  $\mathcal{L}(\theta) = -R(\theta)$ .
- The term **objective function** refers to a function we want to maximize or minimize.

# Introduction

- The core problem in ML is parameter estimation (model fitting).
- We need to solve an **optimization problem**: i.e., trying to find the values for a set of variables  $\theta \in \Theta$  that minimize a scalar-valued **loss function** or **cost function**:  $\mathcal{L} : \Theta \rightarrow \mathbb{R}$

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta) \quad (1)$$

- The **parameter space** is given by  $\Theta \in \mathbb{R}^D$ , where  $D$  is the number of variables being optimized.
- We focus on **continuous optimization**.
- To maximize a **score function** or **reward function**  $R(\theta)$ , we can minimize  $\mathcal{L}(\theta) = -R(\theta)$ .
- The term **objective function** refers to a function we want to maximize or minimize.
- An algorithm to find an optimum of an objective function is a **solver**.

# Local versus global optimization

- A point that satisfies Equation 1 is called a **global optimum**. Finding such a point is called **global optimization**.

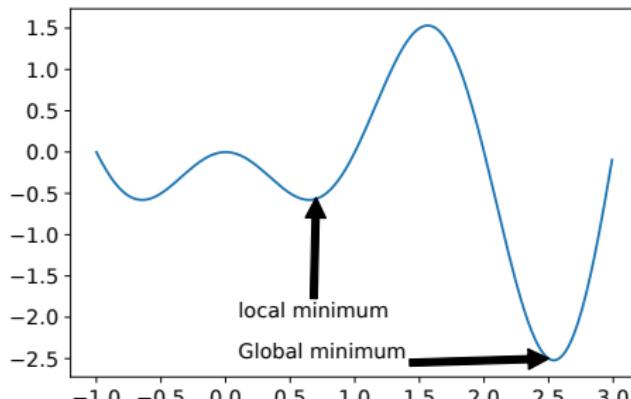
# Local versus global optimization

- A point that satisfies Equation 1 is called a **global optimum**. Finding such a point is called **global optimization**.
- In general, finding global optima is computationally **intractable**. We will try to find a **local optimum**.

# Local versus global optimization

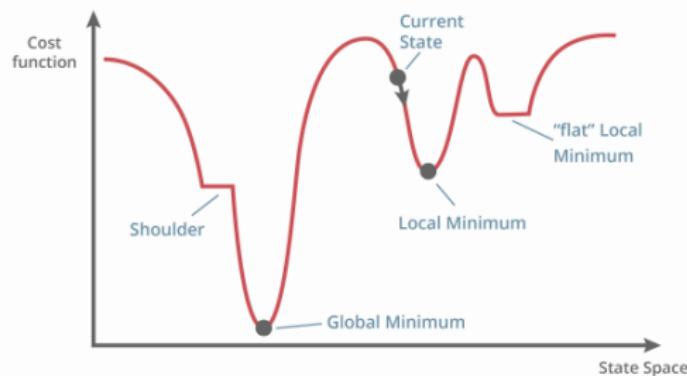
- A point that satisfies Equation 1 is called a **global optimum**. Finding such a point is called **global optimization**.
- In general, finding global optima is computationally **intractable**. We will try to find a **local optimum**.
- For continuous problem, a local optimum is a point  $\theta^*$  which has lower (or equal) cost than “nearby” points.

$$\exists \delta > 0, \forall \theta \in \Theta, \text{ s.t. } \|\theta - \theta^*\| < \delta, \mathcal{L}(\theta^*) \leq \mathcal{L}(\theta) \quad (2)$$



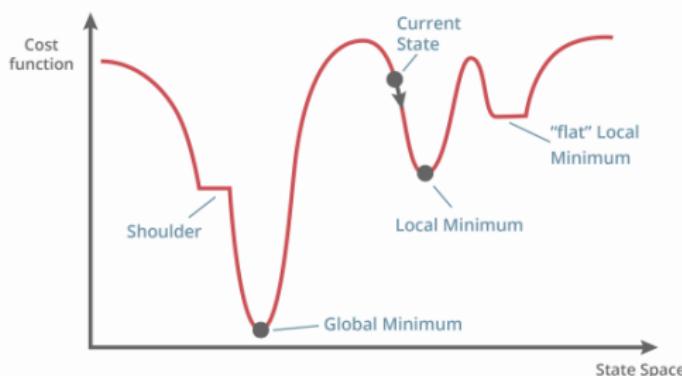
# Local versus global optimization

- A local minimum could be surrounded by other local minima with the same objective value; this is known as a **flat local minimum**.



# Local versus global optimization

- A local minimum could be surrounded by other local minima with the same objective value; this is known as a **flat local minimum**.

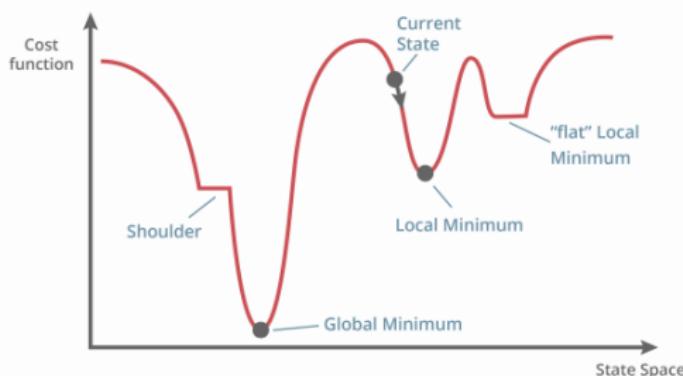


- A point is said to be a **strict local minimum** if its cost is strictly lower than those of neighboring points.

$$\exists \delta > 0, \forall \boldsymbol{\theta} \in \Theta, \boldsymbol{\theta} \neq \boldsymbol{\theta}^* : \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\| < \delta, \mathcal{L}(\boldsymbol{\theta}^*) < \mathcal{L}(\boldsymbol{\theta}) \quad (3)$$

# Local versus global optimization

- A local minimum could be surrounded by other local minima with the same objective value; this is known as a **flat local minimum**.



- A point is said to be a **strict local minimum** if its cost is strictly lower than those of neighboring points.

$$\exists \delta > 0, \forall \theta \in \Theta, \theta \neq \theta^* : \|\theta - \theta^*\| < \delta, \mathcal{L}(\theta^*) < \mathcal{L}(\theta) \quad (3)$$

- We can define a (strict) **local maximum** analogously.

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Derivatives

- The topic of **calculus** concerns computing “**rates of change**” of functions as we vary their inputs.

# Derivatives

- The topic of **calculus** concerns computing “**rates of change**” of functions as we vary their inputs.
- Consider a scalar-argument function  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Its **derivative** at a point  $a$  is the quantity

$$f'(x) \triangleq \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

assuming the limit exists.

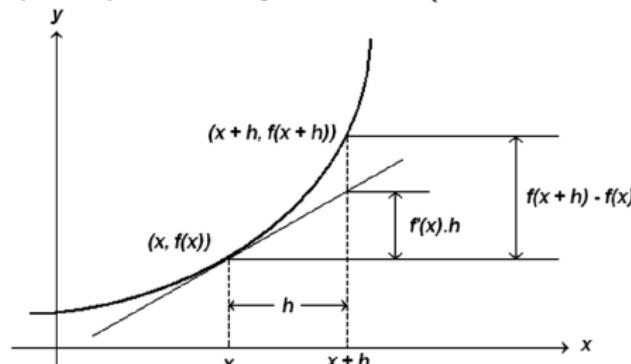
# Derivatives

- The topic of **calculus** concerns computing “**rates of change**” of functions as we vary their inputs.
- Consider a scalar-argument function  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Its **derivative** at a point  $a$  is the quantity

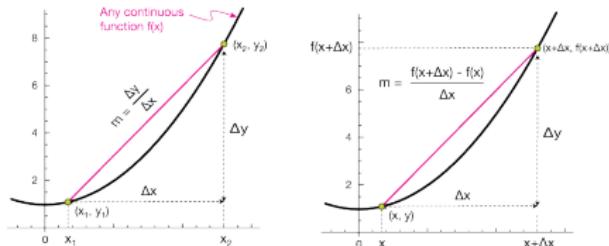
$$f'(x) \triangleq \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

assuming the limit exists.

- This measures how quickly the output changes when we move a small distance in the input space away from  $x$  (i.e., the “rate of change”).



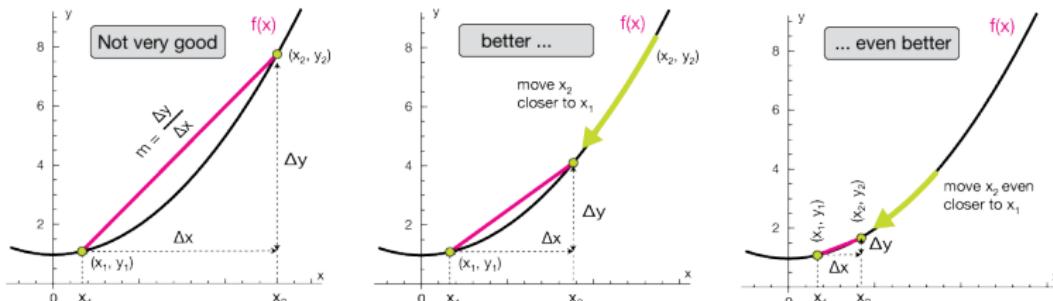
# Derivatives



- $f'(x)$  can be seen as the slope of the tangent line at  $f(x)$

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x$$

for small  $\Delta x$ .



# Derivatives

- We can compute a **finite difference** approximation to the derivative by using a finite step size  $h$

$$\begin{aligned}f'(x) &= \underbrace{\lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}}_{\text{forward difference}} \\&= \underbrace{\lim_{h \rightarrow 0} \frac{f(x + h/2) - f(x - h/2)}{h}}_{\text{central difference}} \\&= \underbrace{\lim_{h \rightarrow 0} \frac{f(x) - f(x - h)}{h}}_{\text{backward difference}}\end{aligned}$$

- The smaller the step size  $h$ , the better the estimate.

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$
- $f'(x)$  computes the derivative at  $x$  (assuming the derivative exists at that point).

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$
- $f'(x)$  computes the derivative at  $x$  (assuming the derivative exists at that point).
- The prime symbol  $f'$  to denote derivative is **Lagrange notation**.

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$
- $f'(x)$  computes the derivative at  $x$  (assuming the derivative exists at that point).
- The prime symbol  $f'$  to denote derivative is **Lagrange notation**.
- The second derivative function, which measures how quickly the gradient is changing, is denoted by  $f''$ .

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$
- $f'(x)$  computes the derivative at  $x$  (assuming the derivative exists at that point).
- The prime symbol  $f'$  to denote derivative is **Lagrange notation**.
- The second derivative function, which measures how quickly the gradient is changing, is denoted by  $f''$ .
- The  $n$ 'th derivative function is denote  $f^{(n)}$ .

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$
- $f'(x)$  computes the derivative at  $x$  (assuming the derivative exists at that point).
- The prime symbol  $f'$  to denote derivative is **Lagrange notation**.
- The second derivative function, which measures how quickly the gradient is changing, is denoted by  $f''$ .
- The  $n$ 'th derivative function is denote  $f^{(n)}$ .
- We can use **Leibniz notation**, if we denote the function by  $y = f(x)$ , and its derivative by  $\frac{dy}{dx}$  or  $\frac{d}{dx}f(x)$ .

# Derivatives

- We can think of **differentiation** as an operator that maps functions to functions,  $D(f) = f'$
- $f'(x)$  computes the derivative at  $x$  (assuming the derivative exists at that point).
- The prime symbol  $f'$  to denote derivative is **Lagrange notation**.
- The second derivative function, which measures how quickly the gradient is changing, is denoted by  $f''$ .
- The  $n$ 'th derivative function is denote  $f^{(n)}$ .
- We can use **Leibniz notation**, if we denote the function by  $y = f(x)$ , and its derivative by  $\frac{dy}{dx}$  or  $\frac{d}{dx}f(x)$ .
- To denote the evaluation of the derivative at a point  $a$ , we write

$$\left. \frac{df}{dx} \right|_{x=a}$$

# Gradients

- We extend the notion of derivatives to handle vector-argument functions,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , by defining the **partial derivative** of  $f$  with respect to  $x_i$  to be

$$\frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$$

where  $\mathbf{e}_i$  is the  $i$ 'th unit vector,  $\mathbf{e}_i = (0, \dots, 1, \dots, 0)$  with the  $i$ 'th element = 1 and all the other elements are 0.

# Gradients

- We extend the notion of derivatives to handle vector-argument functions,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , by defining the **partial derivative** of  $f$  with respect to  $x_i$  to be

$$\frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$$

where  $\mathbf{e}_i$  is the  $i$ 'th unit vector,  $\mathbf{e}_i = (0, \dots, 1, \dots, 0)$  with the  $i$ 'th element = 1 and all the other elements are 0.

- The **gradient** of  $f$  at a point  $\mathbf{x}$  is the vector of its partial derivatives

$$\mathbf{g} = \frac{\partial f}{\partial \mathbf{x}} = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} = \frac{\partial f}{\partial x_1} \mathbf{e}_1 + \dots + \frac{\partial f}{\partial x_n} \mathbf{e}_n$$

# Gradients

- We extend the notion of derivatives to handle vector-argument functions,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , by defining the **partial derivative** of  $f$  with respect to  $x_i$  to be

$$\frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$$

where  $\mathbf{e}_i$  is the  $i$ 'th unit vector,  $\mathbf{e}_i = (0, \dots, 1, \dots, 0)$  with the  $i$ 'th element = 1 and all the other elements are 0.

- The **gradient** of  $f$  at a point  $\mathbf{x}$  is the vector of its partial derivatives

$$\mathbf{g} = \frac{\partial f}{\partial \mathbf{x}} = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} = \frac{\partial f}{\partial x_1} \mathbf{e}_1 + \dots + \frac{\partial f}{\partial x_n} \mathbf{e}_n$$

- To emphasize the point at which the gradient is evaluated, we write

$$\mathbf{g}(\mathbf{x}^*) \triangleq \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*}$$

# Gradients

- Example:

$$f(x_1, x_2) = x_1^2 + x_1x_2 + 3x_2^2$$

$$\nabla f(x_1, x_2) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{pmatrix}$$

- The nabla operator  $\nabla$  maps a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  to another function  $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ .
- Since  $\mathbf{g}()$  is a vector-valued function, it is known as a vector field.

# Directional derivative

- The **directional derivative** measures how much the function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  changes along a direction  $\mathbf{v}$  in space.

$$D_{\mathbf{v}}f(\mathbf{x}) = \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h}$$

- We can approximate this numerically using 2 function calls to  $f$ , regardless of  $n$ .
- By contrast, a numerical approximation to the standard gradient vector takes  $n + 1$  calls (or  $2n$  if using central differences).
- The directional derivative along  $\mathbf{v}$  is the scalar product of the gradient  $\mathbf{g}$  and the vector  $\mathbf{v}$ :

$$D_{\mathbf{v}}f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v}$$

## Directional derivative

**Example:** Let  $f(x, y) = x^2y$ . Find the derivative of  $f$  in the direction  $(1,2)$  at the point  $(3,2)$ .

- The gradient  $\nabla f(x, y)$  is:

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2xy \\ x^2 \end{pmatrix}$$

$$\nabla f(3, 2) = \begin{pmatrix} 12 \\ 9 \end{pmatrix} = 12 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 9 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 12e_1 + 9e_2$$

- Let  $\mathbf{u} = u_1\mathbf{e}_1 + u_2\mathbf{e}_2$  be a unit vector. The derivative of  $f$  in the direction of  $\mathbf{u}$  at  $(3,2)$  is:

$$\begin{aligned} D_{\mathbf{u}}f(3, 2) &= \nabla f(3, 2) \cdot \mathbf{u} \\ &= (12\mathbf{e}_1 + 9\mathbf{e}_2) \cdot (u_1\mathbf{e}_1 + u_2\mathbf{e}_2) \\ &= 12u_1 + 9u_2 \end{aligned}$$

# Directional derivative

Example (cont.)

- The unit vector in the direction of vector  $(1,2)$  is:

$$\mathbf{u} = \frac{(1, 2)}{\|(1, 2)\|} = \frac{(1, 2)}{\sqrt{1^2 + 2^2}} = \frac{(1, 2)}{\sqrt{5}} = (1/\sqrt{5}, 2/\sqrt{5})$$

- The directional derivative at  $(3,2)$  in the direction of  $(1,2)$  is:

$$\begin{aligned} D_{\mathbf{u}} f(3, 2) &= 12u_1 + 9u_2 \\ &= \frac{12}{\sqrt{5}} + \frac{18}{\sqrt{5}} = \frac{30}{\sqrt{5}} \end{aligned}$$

- We normalize vector  $(1,2)$  so that the directional derivative is independent of its magnitude and depending only on its direction.

# Directional derivative

**Example 2:** Let  $f(x, y) = x^2y$ . Find the derivative of  $f$  in the direction of  $(2,1)$  at the point  $(3,2)$ .

# Directional derivative

**Example 2:** Let  $f(x, y) = x^2y$ . Find the derivative of  $f$  in the direction of  $(2,1)$  at the point  $(3,2)$ .

- The unit vector in the direction of  $(2,1)$  is:

$$\mathbf{u} = \frac{(2, 1)}{\sqrt{5}} = (2/\sqrt{5}, 1/\sqrt{5})$$

- The directional derivative of  $f$  at  $(3,2)$  in the direction of  $(2,1)$  is:

$$\begin{aligned} D_{\mathbf{u}}f(3, 2) &= 12u_1 + 9u_2 \\ &= \frac{24}{\sqrt{5}} + \frac{9}{\sqrt{5}} = \frac{33}{\sqrt{5}} \end{aligned}$$

# Directional derivative

## Questions:

- At a point  $a$ , in which direction  $u$  is the directional derivative  $D_u f(a)$  maximal?
- What is the directional derivative in that direction  $D_u f(a) = ?$

# Directional derivative

## Questions:

- At a point  $\mathbf{a}$ , in which direction  $\mathbf{u}$  is the directional derivative  $D_{\mathbf{u}}f(\mathbf{a})$  maximal?
- What is the directional derivative in that direction  $D_{\mathbf{u}}f(\mathbf{a}) = ?$

The relationship between the **gradient** and the **directional derivative**:

$$\begin{aligned} D_{\mathbf{u}}f(\mathbf{a}) &= \nabla f(\mathbf{a}) \cdot \mathbf{u} \\ &= \|\nabla f(\mathbf{a})\| \|\mathbf{u}\| \cos \theta \quad [\theta \text{ is the angle between } \mathbf{u} \text{ and the gradient.}] \\ &= \|\nabla f(\mathbf{a})\| \cos \theta \quad [\mathbf{u} \text{ is a unit vector.}] \end{aligned}$$

# Directional derivative

## Questions:

- At a point  $\mathbf{a}$ , in which direction  $\mathbf{u}$  is the directional derivative  $D_{\mathbf{u}}f(\mathbf{a})$  maximal?
- What is the directional derivative in that direction  $D_{\mathbf{u}}f(\mathbf{a}) = ?$

The relationship between the **gradient** and the **directional derivative**:

$$\begin{aligned} D_{\mathbf{u}}f(\mathbf{a}) &= \nabla f(\mathbf{a}) \cdot \mathbf{u} \\ &= \|\nabla f(\mathbf{a})\| \|\mathbf{u}\| \cos \theta \quad [\theta \text{ is the angle between } \mathbf{u} \text{ and the gradient.}] \\ &= \|\nabla f(\mathbf{a})\| \cos \theta \quad [\mathbf{u} \text{ is a unit vector.}] \end{aligned}$$

The maximal value of  $D_{\mathbf{u}}f(\mathbf{a})$  occurs when  $\mathbf{u}$  and  $\nabla f(\mathbf{a})$  point in the same direction (i.e.,  $\theta = 0$ ).

# Directional derivative

$$D_{\mathbf{u}} f(\mathbf{a}) = \nabla f(\mathbf{a}) \cdot \mathbf{u}$$

$= \|\nabla f(\mathbf{a})\| \|\mathbf{u}\| \cos \theta$  [ $\theta$  is the angle between  $\mathbf{u}$  and the gradient.]

$= \|\nabla f(\mathbf{a})\| \cos \theta$  [ $\mathbf{u}$  is a unit vector.]

- When  $\theta = 0$ , the directional derivative  $D_{\mathbf{u}} f(\mathbf{a}) = \|\nabla f(\mathbf{a})\|$ .
- When  $\theta = \pi$ , the directional derivative  $D_{\mathbf{u}} f(\mathbf{a}) = -\|\nabla f(\mathbf{a})\|$ .
- For what value of  $\theta$  is  $D_{\mathbf{u}} f(\mathbf{a}) = 0$ ?

# Jacobian

- Consider a function that maps a vector to another vector,  
 $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . The **Jacobian matrix** of this function is an  $m \times n$  matrix of partial derivatives:

$$J_f(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top} \triangleq \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(\mathbf{x})^\top \\ \vdots \\ \nabla f_m(\mathbf{x})^\top \end{pmatrix}$$

- We layout the results in the same orientation as the output  $f$ . This is called the numerator layout of the Jacobian formulation.

# Hessian

- For a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  that is twice differentiable, the **Hessian matrix** is the (symmetric)  $n \times n$  matrix of second partial derivatives

$$\mathbf{H}_f = \frac{\partial^2 f}{\partial \mathbf{x}^2} = \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

- The Hessian is the Jacobian of the gradient.

# Hessian

**Example:** Find the Hessian of  $f(x, y) = x^2y + y^2x$  at the point (1,1).

# Hessian

**Example:** Find the Hessian of  $f(x, y) = x^2y + y^2x$  at the point (1,1).

- First, compute the gradient (i.e., first-order partial derivatives):

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2xy + y^2 \\ x^2 + 2yx \end{pmatrix}$$

# Hessian

**Example:** Find the Hessian of  $f(x, y) = x^2y + y^2x$  at the point (1,1).

- First, compute the gradient (i.e., first-order partial derivatives):

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2xy + y^2 \\ x^2 + 2yx \end{pmatrix}$$

- Second, compute the Hessian (i.e., second-order partial derivatives):

$$\mathbf{H}_f(x, y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2y & 2x + 2y \\ 2x + 2y & 2x \end{pmatrix}$$

# Hessian

**Example:** Find the Hessian of  $f(x, y) = x^2y + y^2x$  at the point (1,1).

- First, compute the gradient (i.e., first-order partial derivatives):

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2xy + y^2 \\ x^2 + 2yx \end{pmatrix}$$

- Second, compute the Hessian (i.e., second-order partial derivatives):

$$\mathbf{H}_f(x, y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2y & 2x + 2y \\ 2x + 2y & 2x \end{pmatrix}$$

- Finally, evaluate the Hessian matrix at the point (1,1):

$$\mathbf{H}_f(1, 1) = \begin{pmatrix} 2 & 4 \\ 4 & 2 \end{pmatrix}$$

# Geometric meaning

- If we follow the direction  $\mathbf{d}$  from  $\mathbf{x}$ , we can define a uni-dimensional function  $g(\alpha)$ :

$$g(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$$

$$g'(\alpha) = \mathbf{d}^T \nabla f(\mathbf{x} + \alpha\mathbf{d})$$

$$g''(\alpha) = \mathbf{d}^T \nabla^2 f(\mathbf{x} + \alpha\mathbf{d}) \mathbf{d}$$

- Interpretation

$$g'(0) = \mathbf{d}^T \nabla f(\mathbf{x}) \quad [\text{directional derivative}]$$

$$g''(0) = \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} \quad [\text{directional curvature}]$$

- If  $g''(0)$  is non-negative with a certain  $\mathbf{d}$ :  $f$  is convex in direction  $\mathbf{d}$ .
- If  $g''(0)$  is non-negative for all  $\mathbf{d}$ :  $\nabla^2 f(\mathbf{x})$  is positive semidefinite  $\rightarrow f$  is convex at  $\mathbf{x}$ .

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $x^T A x \geq 0$  for all  $x$ ,

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $x^T A x \geq 0$  for all  $x$ ,
- positive definite ( $A \succ 0$ ) if  $x^T A x > 0$  for all  $x \neq 0$ ,

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $x^T A x \geq 0$  for all  $x$ ,
- positive definite ( $A \succ 0$ ) if  $x^T A x > 0$  for all  $x \neq 0$ ,
- negative semidefinite ( $A \preceq 0$ ) if  $x^T A x \leq 0$  for all  $x$ ,

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $x^T A x \geq 0$  for all  $x$ ,
- positive definite ( $A \succ 0$ ) if  $x^T A x > 0$  for all  $x \neq 0$ ,
- negative semidefinite ( $A \preceq 0$ ) if  $x^T A x \leq 0$  for all  $x$ ,
- negative definite ( $A \prec 0$ ) if  $x^T A x < 0$  for all  $x \neq 0$ ,

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $x^T A x \geq 0$  for all  $x$ ,
- positive definite ( $A \succ 0$ ) if  $x^T A x > 0$  for all  $x \neq 0$ ,
- negative semidefinite ( $A \preceq 0$ ) if  $x^T A x \leq 0$  for all  $x$ ,
- negative definite ( $A \prec 0$ ) if  $x^T A x < 0$  for all  $x \neq 0$ ,
- indefinite if none of the above apply.

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $x^T A x \geq 0$  for all  $x$ ,
- positive definite ( $A \succ 0$ ) if  $x^T A x > 0$  for all  $x \neq 0$ ,
- negative semidefinite ( $A \preceq 0$ ) if  $x^T A x \leq 0$  for all  $x$ ,
- negative definite ( $A \prec 0$ ) if  $x^T A x < 0$  for all  $x \neq 0$ ,
- indefinite if none of the above apply.

# Definitions

We say that a symmetric  $n \times n$  matrix  $A$  is:

- positive semidefinite ( $A \succeq 0$ ) if  $\mathbf{x}^\top A \mathbf{x} \geq 0$  for all  $\mathbf{x}$ ,
- positive definite ( $A \succ 0$ ) if  $\mathbf{x}^\top A \mathbf{x} > 0$  for all  $\mathbf{x} \neq 0$ ,
- negative semidefinite ( $A \preceq 0$ ) if  $\mathbf{x}^\top A \mathbf{x} \leq 0$  for all  $\mathbf{x}$ ,
- negative definite ( $A \prec 0$ ) if  $\mathbf{x}^\top A \mathbf{x} < 0$  for all  $\mathbf{x} \neq 0$ ,
- indefinite if none of the above apply.
- The expression  $\mathbf{x}^\top A \mathbf{x}$  is a function of  $\mathbf{x}$  called the quadratic form associated to  $A$ . (It's made up of terms like  $x_i^2$  and  $x_i x_j$ .)
- We make these definitions for a symmetric matrix  $A$ , i.e.,  $A^\top = A$ .
- Hessian matrices are symmetric.

# Diagonal matrices

For a diagonal matrix

$$D = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & d_n \end{bmatrix}$$

the quadratic form

$$\mathbf{x}^\top D \mathbf{x} = [x_1 \ x_2 \ \dots \ x_n] \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is just  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$ .

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = 0$ , so  $D \succ 0$ :  $D$  is positive definite.

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = 0$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = 0$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .
- $D$  is indefinite if the signs of  $d_1, \dots, d_n$  are mixed.

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = 0$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .
- $D$  is indefinite if the signs of  $d_1, \dots, d_n$  are mixed.

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = 0$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .
- $D$  is indefinite if the signs of  $d_1, \dots, d_n$  are mixed.

**Example:** Consider the function  $f(x, y) = x^2 + 2y^2$ .

- The gradient  $\nabla f(x, y) = (2x, 4y)$ .

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = 0$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .
- $D$  is indefinite if the signs of  $d_1, \dots, d_n$  are mixed.

**Example:** Consider the function  $f(x, y) = x^2 + 2y^2$ .

- The gradient  $\nabla f(x, y) = (2x, 4y)$ .
- The Hessian matrix of  $f$  is:

$$\mathbf{H}_f(x, y) = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $x$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $x = \mathbf{0}$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .
- $D$  is indefinite if the signs of  $d_1, \dots, d_n$  are mixed.

**Example:** Consider the function  $f(x, y) = x^2 + 2y^2$ .

- The gradient  $\nabla f(x, y) = (2x, 4y)$ .
- The Hessian matrix of  $f$  is:

$$\mathbf{H}_f(x, y) = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

- For an arbitrary  $x \in \mathbb{R}^2$ , we have

$$\mathbf{x}^\top \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \mathbf{x} = 2x_1^2 + 4x_2^2 > 0 \text{ for all } \mathbf{x} \neq \mathbf{0}.$$

# Diagonal matrices

- If  $d_1, \dots, d_n$  are all nonnegative, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  must be nonnegative for any  $\mathbf{x}$ , so  $D \succeq 0$ :  $D$  is positive semidefinite.
- If  $d_1, \dots, d_n$  are all positive, then  $d_1x_1^2 + d_2x_2^2 + \dots + d_nx_n^2$  can only be 0 if  $\mathbf{x} = \mathbf{0}$ , so  $D \succ 0$ :  $D$  is positive definite.
- If  $d_1, \dots, d_n \leq 0$ , then  $D \preceq 0$ , and if  $d_1, \dots, d_n < 0$ , then  $D \prec 0$ .
- $D$  is indefinite if the signs of  $d_1, \dots, d_n$  are mixed.

**Example:** Consider the function  $f(x, y) = x^2 + 2y^2$ .

- The gradient  $\nabla f(x, y) = (2x, 4y)$ .
- The Hessian matrix of  $f$  is:

$$\mathbf{H}_f(x, y) = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

- For an arbitrary  $\mathbf{x} \in \mathbb{R}^2$ , we have

$$\mathbf{x}^\top \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \mathbf{x} = 2x_1^2 + 4x_2^2 > 0 \text{ for all } \mathbf{x} \neq \mathbf{0}.$$

- So,  $\mathbf{H}_f(x, y) \succ 0$  for all  $(x, y) \in \mathbb{R}^2$ .  $\mathbf{H}_f(x, y)$  is positive definite.

## Positive definiteness and eigenvalues

- For an  $n \times n$  matrix  $A$ , if a nonzero vector  $x \in \mathbb{R}^n$  satisfies

$$Ax = \lambda x$$

for some scalar  $\lambda \in \mathbb{R}$ , we call  $\lambda$  an eigenvalue of  $A$  and  $x$  its associated eigenvector.

- If  $A$  is an  $n \times n$  symmetric matrix, then it can be factored as

$$A = Q^T \Lambda Q = Q^T \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & \lambda_n \end{bmatrix} Q$$

where  $\lambda_1, \dots, \lambda_n$  are the eigenvalues of  $A$  and the columns of  $Q$  are the corresponding eigenvectors.

# Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

## Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

## Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

# Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

We can classify the matrix  $A$  by looking at the eigenvalues of  $A$ .

- $A \succeq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$

# Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

We can classify the matrix  $A$  by looking at the eigenvalues of  $A$ .

- $A \succeq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$
- $A \succ 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n > 0$

# Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

We can classify the matrix  $A$  by looking at the eigenvalues of  $A$ .

- $A \succeq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$
- $A \succ 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n > 0$
- $A \preceq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \leq 0$

# Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

We can classify the matrix  $A$  by looking at the eigenvalues of  $A$ .

- $A \succeq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$
- $A \succ 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n > 0$
- $A \preceq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \leq 0$
- $A \prec 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n < 0$

# Positive definiteness and eigenvalues

- Apply to the quadratic form  $\mathbf{x}^T A \mathbf{x}$ , we get

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T Q^T \Lambda Q \mathbf{x} = (Q\mathbf{x})^T \Lambda (Q\mathbf{x})$$

- If we substitute  $\mathbf{y} = Q\mathbf{x}$  (converting to a different basis), the quadratic form becomes diagonal:

$$\mathbf{x}^T A \mathbf{x} = \mathbf{y}^T \Lambda \mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$$

We can classify the matrix  $A$  by looking at the eigenvalues of  $A$ .

- $A \succeq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$
- $A \succ 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n > 0$
- $A \preceq 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n \leq 0$
- $A \prec 0$  if  $\lambda_1, \lambda_2, \dots, \lambda_n < 0$
- $A$  is indefinite if it has both positive and negative eigenvalues.

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Optimality conditions for local vs global optima

- For continuous, twice differentiable functions, we can characterize the **points** which correspond to **local optima**.

# Optimality conditions for local vs global optima

- For continuous, twice differentiable functions, we can characterize the **points** which correspond to **local optima**.
- Let  $\mathbf{g}(\theta) = \nabla \mathcal{L}(\theta)$  be the **gradient** vector, and  $\mathbf{H}(\theta) = \nabla^2 \mathcal{L}(\theta)$  be the **Hessian** matrix.

# Optimality conditions for local vs global optima

- For continuous, twice differentiable functions, we can characterize the **points** which correspond to **local optima**.
- Let  $\mathbf{g}(\theta) = \nabla \mathcal{L}(\theta)$  be the **gradient** vector, and  $\mathbf{H}(\theta) = \nabla^2 \mathcal{L}(\theta)$  be the **Hessian** matrix.
- Consider a point  $\theta^* \in \mathbb{R}^D$ , and let  $\mathbf{g}^* = \mathbf{g}(\theta)|_{\theta^*}$  be the gradient at that point, and  $\mathbf{H}^* = \mathbf{H}(\theta)|_{\theta^*}$  be the corresponding Hessian.

# Optimality conditions for local vs global optima

- For continuous, twice differentiable functions, we can characterize the **points** which correspond to **local optima**.
- Let  $\mathbf{g}(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta})$  be the **gradient** vector, and  $\mathbf{H}(\boldsymbol{\theta}) = \nabla^2 \mathcal{L}(\boldsymbol{\theta})$  be the **Hessian** matrix.
- Consider a point  $\boldsymbol{\theta}^* \in \mathbb{R}^D$ , and let  $\mathbf{g}^* = \mathbf{g}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^*}$  be the gradient at that point, and  $\mathbf{H}^* = \mathbf{H}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^*}$  be the corresponding Hessian.
- **Necessary conditions:** If  $\boldsymbol{\theta}^*$  is a local minimum, then we must have  $\mathbf{g}^* = \mathbf{0}$  (i.e.,  $\boldsymbol{\theta}^*$  must be a **stationary point**), and  $\mathbf{H}^*$  must be positive semi-definite.

# Optimality conditions for local vs global optima

- For continuous, twice differentiable functions, we can characterize the **points** which correspond to **local optima**.
- Let  $\mathbf{g}(\theta) = \nabla \mathcal{L}(\theta)$  be the **gradient** vector, and  $\mathbf{H}(\theta) = \nabla^2 \mathcal{L}(\theta)$  be the **Hessian** matrix.
- Consider a point  $\theta^* \in \mathbb{R}^D$ , and let  $\mathbf{g}^* = \mathbf{g}(\theta)|_{\theta^*}$  be the gradient at that point, and  $\mathbf{H}^* = \mathbf{H}(\theta)|_{\theta^*}$  be the corresponding Hessian.
- **Necessary conditions:** If  $\theta^*$  is a local minimum, then we must have  $\mathbf{g}^* = \mathbf{0}$  (i.e.,  $\theta^*$  must be a **stationary point**), and  $\mathbf{H}^*$  must be positive semi-definite.
- **Sufficient conditions:** If  $\mathbf{g}^* = \mathbf{0}$  and  $\mathbf{H}^*$  is positive definite, then  $\theta^*$  is a local optimum.

# Optimality conditions for local vs global optima

① **Necessary conditions:** If  $\theta^*$  is a local minimum, then we must have  $g^* = 0$  (i.e.,  $\theta^*$  must be a **stationary point**), and  $H^*$  must be positive semi-definite.

- Suppose we were at a point  $\theta^*$  at which the gradient is non-zero.
- At such a point, we could decrease the function by following the negative gradient a small distance, so this would not be optimal.
- So the gradient must be zero.

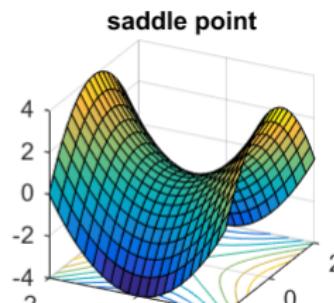
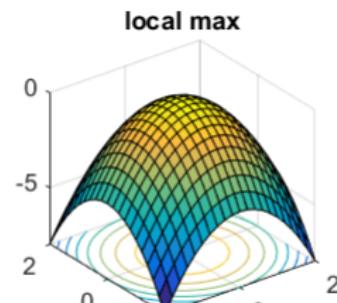
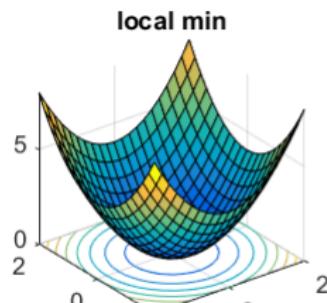
# Optimality conditions for local vs global optima

① **Necessary conditions:** If  $\theta^*$  is a local minimum, then we must have  $g^* = 0$  (i.e.,  $\theta^*$  must be a **stationary point**), and  $H^*$  must be positive semi-definite.

- Suppose we were at a point  $\theta^*$  at which the gradient is non-zero.
- At such a point, we could decrease the function by following the negative gradient a small distance, so this would not be optimal.
- So the gradient must be zero.

② **Sufficient conditions:** If  $g^* = 0$  and  $H^*$  is positive definite, then  $\theta^*$  is a local optimum.

- Why a zero gradient is not sufficient?
- The stationary point could be a local minimum, local maximum, or **saddle** point.



# Global optimizers

- We classify a stationary point of a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  as a **global minimizer** if the Hessian matrix of  $f$  is positive semidefinite **everywhere**,
- and as a **global maximizer** if the Hessian matrix is negative semidefinite everywhere.
- If the Hessian matrix is positive definite, or negative definite, the minimizer and maximizer (respectively) is strict.

## Example

Let  $f(x_1, x_2) = (x_1^2 + x_2^2 - 1)^2 + (x_2^2 - 1)^2$ .

- The gradient is  $\nabla f(\mathbf{x}) = 4 \begin{pmatrix} (x_1^2 + x_2^2 - 1)x_1 \\ (x_1^2 + x_2^2 - 1)x_2 + (x_2^2 - 1)x_2 \end{pmatrix}$

## Example

Let  $f(x_1, x_2) = (x_1^2 + x_2^2 - 1)^2 + (x_2^2 - 1)^2$ .

- The gradient is  $\nabla f(\mathbf{x}) = 4 \begin{pmatrix} (x_1^2 + x_2^2 - 1)x_1 \\ (x_1^2 + x_2^2 - 1)x_2 + (x_2^2 - 1)x_2 \end{pmatrix}$
- The stationary points are  $(0,0)$ ,  $(1,0)$ ,  $(-1,0)$ ,  $(0,1)$ ,  $(0,-1)$ .

## Example

Let  $f(x_1, x_2) = (x_1^2 + x_2^2 - 1)^2 + (x_2^2 - 1)^2$ .

- The gradient is  $\nabla f(\mathbf{x}) = 4 \begin{pmatrix} (x_1^2 + x_2^2 - 1)x_1 \\ (x_1^2 + x_2^2 - 1)x_2 + (x_2^2 - 1)x_2 \end{pmatrix}$
- The stationary points are  $(0,0)$ ,  $(1,0)$ ,  $(-1,0)$ ,  $(0,1)$ ,  $(0,-1)$ .
- The Hessian is  $\nabla^2 f(\mathbf{x}) = 4 \begin{pmatrix} 3x_1^2 + x_2^2 - 1 & 2x_1x_2 \\ 2x_1x_2 & x_1^2 + 6x_2^2 - 2 \end{pmatrix}$

## Example

Let  $f(x_1, x_2) = (x_1^2 + x_2^2 - 1)^2 + (x_2^2 - 1)^2$ .

- The gradient is  $\nabla f(\mathbf{x}) = 4 \begin{pmatrix} (x_1^2 + x_2^2 - 1)x_1 \\ (x_1^2 + x_2^2 - 1)x_2 + (x_2^2 - 1)x_2 \end{pmatrix}$
- The stationary points are  $(0,0)$ ,  $(1,0)$ ,  $(-1,0)$ ,  $(0,1)$ ,  $(0,-1)$ .
- The Hessian is  $\nabla^2 f(\mathbf{x}) = 4 \begin{pmatrix} 3x_1^2 + x_2^2 - 1 & 2x_1x_2 \\ 2x_1x_2 & x_1^2 + 6x_2^2 - 2 \end{pmatrix}$
- Since  $\nabla^2 f(0,0) = 4 \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix} \prec 0$ , it follows that  $(0,0)$  is a **strict local maximum** point.

## Example

Let  $f(x_1, x_2) = (x_1^2 + x_2^2 - 1)^2 + (x_2^2 - 1)^2$ .

- The gradient is  $\nabla f(\mathbf{x}) = 4 \begin{pmatrix} (x_1^2 + x_2^2 - 1)x_1 \\ (x_1^2 + x_2^2 - 1)x_2 + (x_2^2 - 1)x_2 \end{pmatrix}$
- The stationary points are  $(0,0)$ ,  $(1,0)$ ,  $(-1,0)$ ,  $(0,1)$ ,  $(0,-1)$ .
- The Hessian is  $\nabla^2 f(\mathbf{x}) = 4 \begin{pmatrix} 3x_1^2 + x_2^2 - 1 & 2x_1x_2 \\ 2x_1x_2 & x_1^2 + 6x_2^2 - 2 \end{pmatrix}$
- Since  $\nabla^2 f(0,0) = 4 \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix} \prec 0$ , it follows that  $(0,0)$  is a **strict local maximum** point.
- By the fact that  $f(x_1, 0) = (x_1^2 - 1)^2 + 1 \rightarrow \infty$  as  $x_1 \rightarrow \infty$ , the function is not bounded above, and thus  $(0,0)$  is not a global maximum point.

## Example

- $\nabla^2 f(1, 0) = \nabla^2 f(-1, 0) = 4 \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$ , which is an indefinite matrix.

## Example

- $\nabla^2 f(1, 0) = \nabla^2 f(-1, 0) = 4 \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$ , which is an indefinite matrix.
- Hence (1,0) and (-1,0) are **saddle points**.

## Example

- $\nabla^2 f(1, 0) = \nabla^2 f(-1, 0) = 4 \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$ , which is an indefinite matrix.
- Hence (1,0) and (-1,0) are **saddle points**.
- $\nabla^2 f(0, 1) = \nabla^2 f(0, -1) = 4 \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$ , which is positive semidefinite.

## Example

- $\nabla^2 f(1, 0) = \nabla^2 f(-1, 0) = 4 \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$ , which is an indefinite matrix.
- Hence  $(1, 0)$  and  $(-1, 0)$  are **saddle points**.
- $\nabla^2 f(0, 1) = \nabla^2 f(0, -1) = 4 \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$ , which is positive semidefinite.
- The fact that the Hessian matrices of  $f$  at  $(0, 1)$  and  $(0, -1)$  are positive semidefinite is **not enough** to conclude that these are local minimum points; they **might be** saddle points.

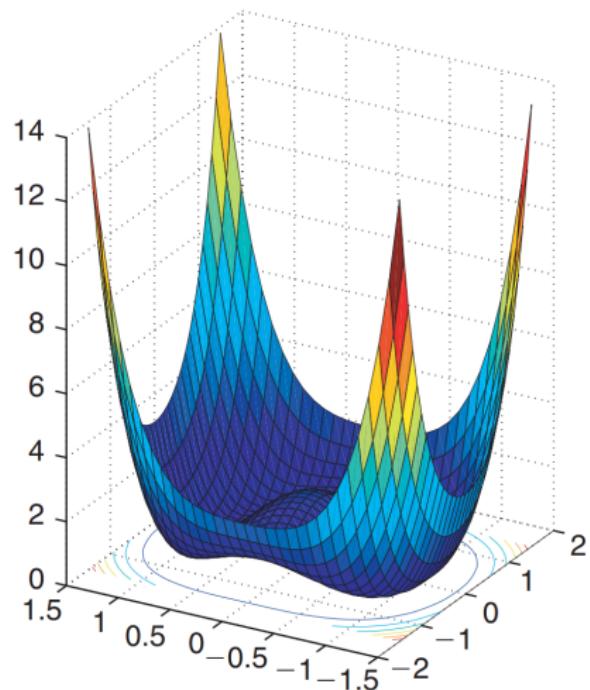
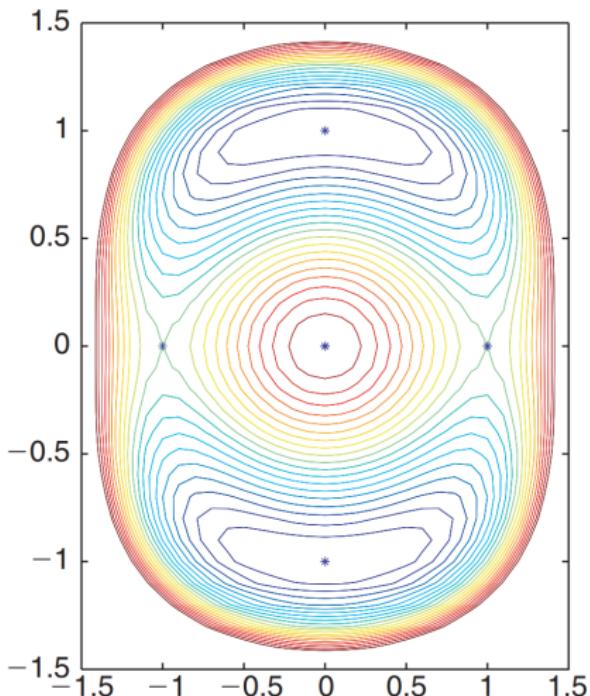
## Example

- $\nabla^2 f(1, 0) = \nabla^2 f(-1, 0) = 4 \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$ , which is an indefinite matrix.
- Hence  $(1, 0)$  and  $(-1, 0)$  are **saddle points**.
- $\nabla^2 f(0, 1) = \nabla^2 f(0, -1) = 4 \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$ , which is positive semidefinite.
- The fact that the Hessian matrices of  $f$  at  $(0, 1)$  and  $(0, -1)$  are positive semidefinite is **not enough** to conclude that these are local minimum points; they **might be** saddle points.
- However, in this case, since  $f(0, 1) = f(0, -1) = 0$  and the function is lower bounded by zero,  $(0, 1)$  and  $(0, -1)$  are **global minimum points**.

## Example

- $\nabla^2 f(1, 0) = \nabla^2 f(-1, 0) = 4 \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$ , which is an indefinite matrix.
- Hence  $(1, 0)$  and  $(-1, 0)$  are **saddle points**.
- $\nabla^2 f(0, 1) = \nabla^2 f(0, -1) = 4 \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$ , which is positive semidefinite.
- The fact that the Hessian matrices of  $f$  at  $(0, 1)$  and  $(0, -1)$  are positive semidefinite is **not enough** to conclude that these are local minimum points; they **might be** saddle points.
- However, in this case, since  $f(0, 1) = f(0, -1) = 0$  and the function is lower bounded by zero,  $(0, 1)$  and  $(0, -1)$  are **global minimum points**.
- Because there are two global minimum points, they are **nonstrict** global minima, but they are **strict** local minimum points, since each has a neighborhood in which it is the unique minimizer.

# Example



# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Constrained vs unconstrained optimization

- In **unconstrained optimization**, we find any value in the parameter space  $\Theta$  that minimizes the loss.

# Constrained vs unconstrained optimization

- In **unconstrained optimization**, we find any value in the parameter space  $\Theta$  that minimizes the loss.
- We can also have a set of **constraints**  $\mathcal{C}$  on the allowable values.

# Constrained vs unconstrained optimization

- In **unconstrained optimization**, we find any value in the parameter space  $\Theta$  that minimizes the loss.
- We can also have a set of **constraints**  $\mathcal{C}$  on the allowable values.
- We partition the set of constraints  $\mathcal{C}$  into:
  - **Inequality constraints:**  $g_j(\theta) \leq 0$  for  $j \in \mathcal{I}$ .
  - **Equality constraints:**  $h_k(\theta) = 0$  for  $k \in \mathcal{E}$ .

# Constrained vs unconstrained optimization

- In **unconstrained optimization**, we find any value in the parameter space  $\Theta$  that minimizes the loss.
- We can also have a set of **constraints**  $\mathcal{C}$  on the allowable values.
- We partition the set of constraints  $\mathcal{C}$  into:
  - **Inequality constraints:**  $g_j(\boldsymbol{\theta}) \leq 0$  for  $j \in \mathcal{I}$ .
  - **Equality constraints:**  $h_k(\boldsymbol{\theta}) = 0$  for  $k \in \mathcal{E}$ .
- The **feasible set** is the subset of the parameter space that satisfies the constraints:

$$\mathcal{C} = \{\boldsymbol{\theta} : g_j(\boldsymbol{\theta}) \leq 0 : j \in \mathcal{I}, h_k(\boldsymbol{\theta}) = 0 : k \in \mathcal{E}\} \subseteq \mathbb{R}^D$$

# Constrained vs unconstrained optimization

- In **unconstrained optimization**, we find any value in the parameter space  $\Theta$  that minimizes the loss.
- We can also have a set of **constraints**  $\mathcal{C}$  on the allowable values.
- We partition the set of constraints  $\mathcal{C}$  into:
  - **Inequality constraints:**  $g_j(\boldsymbol{\theta}) \leq 0$  for  $j \in \mathcal{I}$ .
  - **Equality constraints:**  $h_k(\boldsymbol{\theta}) = 0$  for  $k \in \mathcal{E}$ .
- The **feasible set** is the subset of the parameter space that satisfies the constraints:

$$\mathcal{C} = \{\boldsymbol{\theta} : g_j(\boldsymbol{\theta}) \leq 0 : j \in \mathcal{I}, h_k(\boldsymbol{\theta}) = 0 : k \in \mathcal{E}\} \subseteq \mathbb{R}^D$$

- Our **constrained optimization** problem is

$$\hat{\boldsymbol{\theta}}^* = \operatorname{argmin}_{\boldsymbol{\theta} \in \mathcal{C}} \mathcal{L}(\boldsymbol{\theta})$$

# Constrained vs unconstrained optimization

- In **unconstrained optimization**, we find any value in the parameter space  $\Theta$  that minimizes the loss.
- We can also have a set of **constraints**  $\mathcal{C}$  on the allowable values.
- We partition the set of constraints  $\mathcal{C}$  into:
  - **Inequality constraints:**  $g_j(\boldsymbol{\theta}) \leq 0$  for  $j \in \mathcal{I}$ .
  - **Equality constraints:**  $h_k(\boldsymbol{\theta}) = 0$  for  $k \in \mathcal{E}$ .
- The **feasible set** is the subset of the parameter space that satisfies the constraints:

$$\mathcal{C} = \{\boldsymbol{\theta} : g_j(\boldsymbol{\theta}) \leq 0 : j \in \mathcal{I}, h_k(\boldsymbol{\theta}) = 0 : k \in \mathcal{E}\} \subseteq \mathbb{R}^D$$

- Our **constrained optimization** problem is

$$\hat{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta} \in \mathcal{C}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta})$$

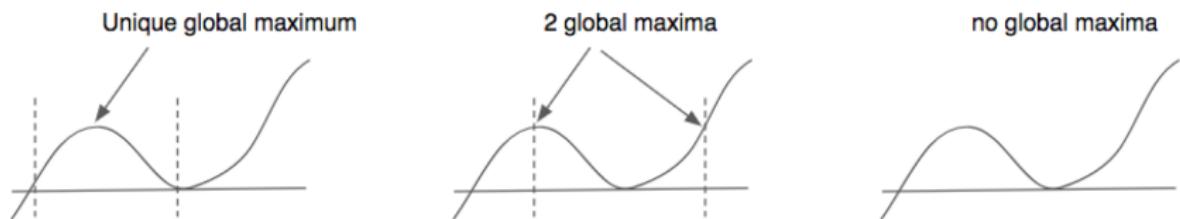
- If  $\mathcal{C} = \mathbb{R}^D$ , it is called **unconstrained optimization**.

# Constrained vs unconstrained optimization

- Constraints can change the number of optima of a function.

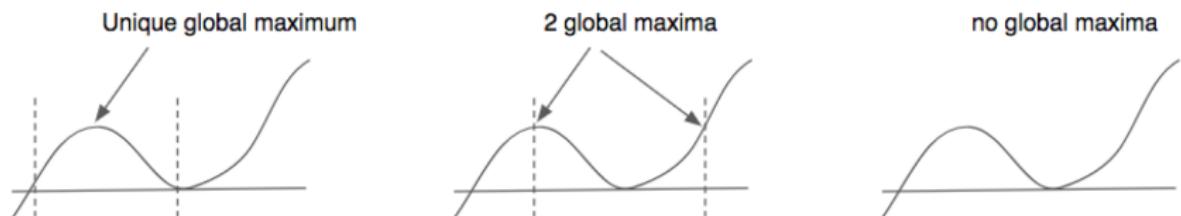
# Constrained vs unconstrained optimization

- Constraints can change the number of optima of a function.
- A function that was unbounded (no well-defined global maximum or minimum) can acquire multiple maxima or minima when we add constraints.



# Constrained vs unconstrained optimization

- Constraints can change the number of optima of a function.
- A function that was unbounded (no well-defined global maximum or minimum) can acquire multiple maxima or minima when we add constraints.



- The task of finding any point (regardless of its cost) in the feasible set is called **feasibility problem**.

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Convex sets

- In **convex optimization**, the objective is a convex function defined over a convex set.
- In such problems, every local minimum is also a global minimum.

# Convex sets

- In **convex optimization**, the objective is a convex function defined over a convex set.
- In such problems, every local minimum is also a global minimum.
- Many models are designed so that their training objectives are convex.

# Convex sets

- In **convex optimization**, the objective is a convex function defined over a convex set.
- In such problems, every local minimum is also a global minimum.
- Many models are designed so that their training objectives are convex.
- We say  $\mathcal{S}$  is a **convex set** if, for any  $\mathbf{x}, \mathbf{x}' \in \mathcal{S}$ , we have

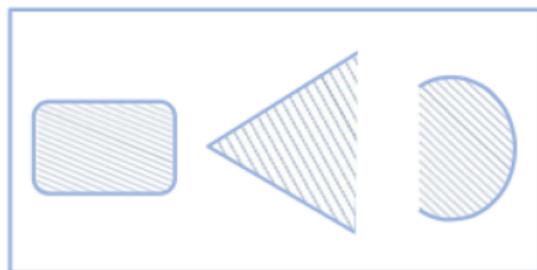
$$\lambda \mathbf{x} + (1 - \lambda) \mathbf{x}' \in \mathcal{S}, \forall \lambda \in [0, 1]$$

# Convex sets

- In **convex optimization**, the objective is a convex function defined over a convex set.
- In such problems, every local minimum is also a global minimum.
- Many models are designed so that their training objectives are convex.
- We say  $\mathcal{S}$  is a **convex set** if, for any  $x, x' \in \mathcal{S}$ , we have

$$\lambda x + (1 - \lambda)x' \in \mathcal{S}, \forall \lambda \in [0, 1]$$

- If we draw a line from  $x$  to  $x'$ , all points on the line lie inside the set.



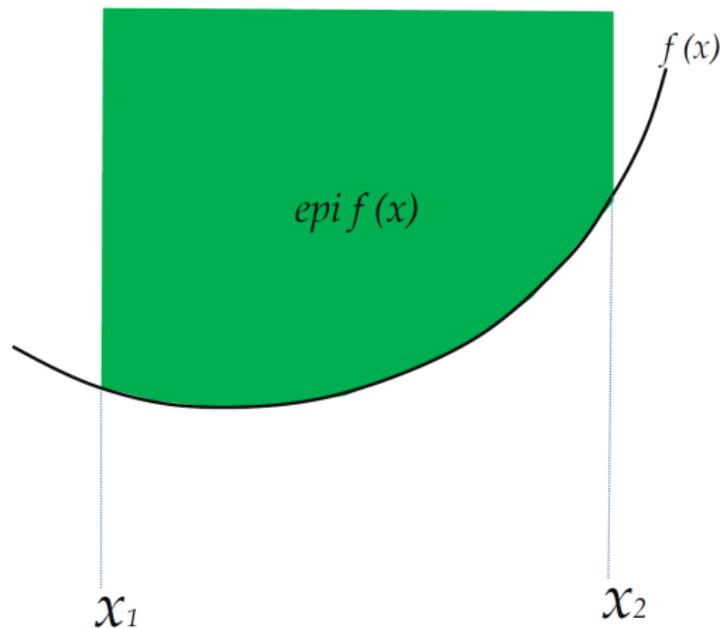
Convex



Not Convex

# Convex functions

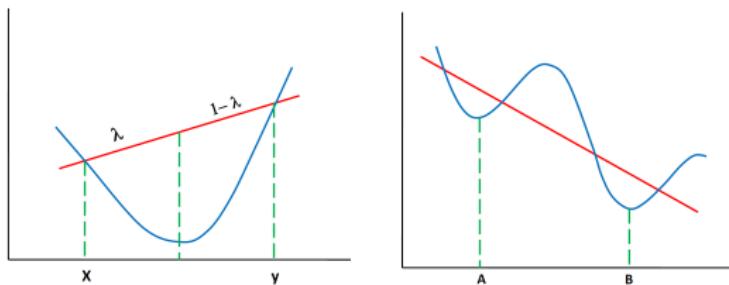
- $f$  is a **convex function** if its **epigraph** (the set of points above the function) defines a convex set.



# Convex functions

- $f(\mathbf{x})$  is called a convex function if it is defined on a convex set, and if, for any  $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ , and for any  $0 \leq \lambda \leq 1$ , we have:

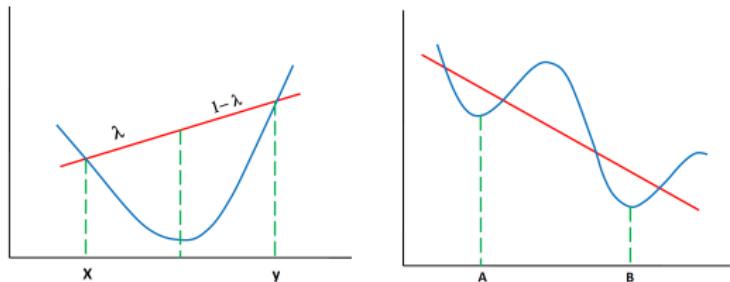
$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$



# Convex functions

- $f(\mathbf{x})$  is called a convex function if it is defined on a convex set, and if, for any  $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ , and for any  $0 \leq \lambda \leq 1$ , we have:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$

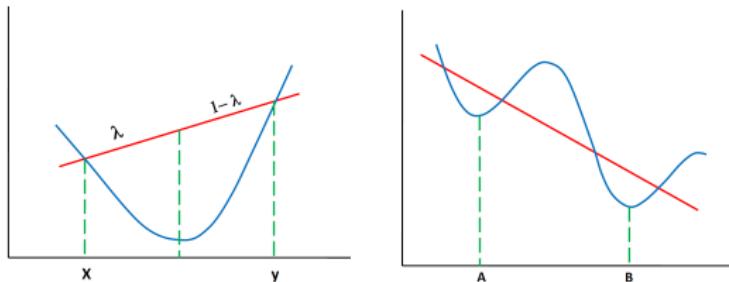


- A function is **strictly convex** if the inequality is strict.

# Convex functions

- $f(\mathbf{x})$  is called a convex function if it is defined on a convex set, and if, for any  $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ , and for any  $0 \leq \lambda \leq 1$ , we have:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$

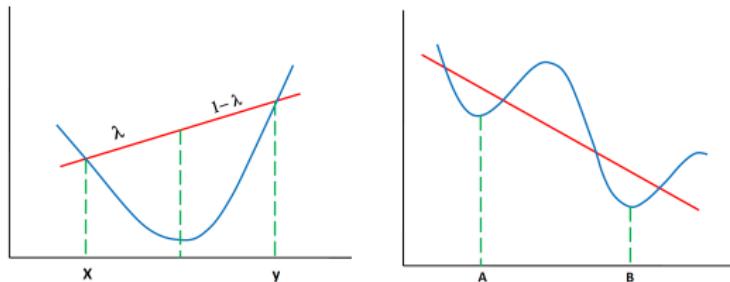


- A function is **strictly convex** if the inequality is strict.
- A function is **concave** if  $-f(x)$  is convex.

# Convex functions

- $f(\mathbf{x})$  is called a convex function if it is defined on a convex set, and if, for any  $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ , and for any  $0 \leq \lambda \leq 1$ , we have:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$

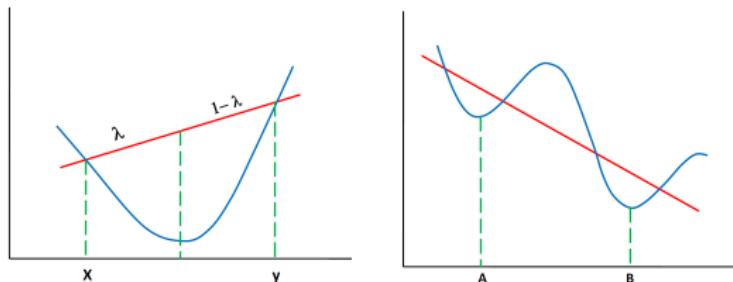


- A function is **strictly convex** if the inequality is strict.
- A function is **concave** if  $-f(\mathbf{x})$  is convex.
- A function can be neither convex nor concave.

# Convex functions

- $f(\mathbf{x})$  is called a convex function if it is defined on a convex set, and if, for any  $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ , and for any  $0 \leq \lambda \leq 1$ , we have:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$



- A function is **strictly convex** if the inequality is strict.
- A function is **concave** if  $-f(x)$  is convex.
- A function can be neither convex nor concave.
- Some examples of 1d convex functions:  $x^2$ ,  $e^{ax}$ ,  $-\log(x)$ ,  $x^a (a > 1, x > 0)$ ,  $|x|^a (a \geq 1)$ ,  $x \log x (x > 0)$ .

## Theorem

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice differentiable over its domain. Then  $f$  is convex iff  $\mathbf{H} = \nabla^2 f(\mathbf{x})$  is positive semi-definite for all  $\mathbf{x} \in \text{dom}(f)$ . Furthermore,  $f$  is strictly convex if  $\mathbf{H}$  is positive definite.

## Theorem

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice differentiable over its domain. Then  $f$  is convex iff  $\mathbf{H} = \nabla^2 f(\mathbf{x})$  is positive semi-definite for all  $\mathbf{x} \in \text{dom}(f)$ . Furthermore,  $f$  is strictly convex if  $\mathbf{H}$  is positive definite.

- For example, consider the quadratic form

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$$

## Theorem

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice differentiable over its domain. Then  $f$  is convex iff  $\mathbf{H} = \nabla^2 f(\mathbf{x})$  is positive semi-definite for all  $\mathbf{x} \in \text{dom}(f)$ . Furthermore,  $f$  is strictly convex if  $\mathbf{H}$  is positive definite.

- For example, consider the quadratic form

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$$

- This is convex if  $\mathbf{A}$  is positive semi-definite.

## Theorem

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice differentiable over its domain. Then  $f$  is convex iff  $\mathbf{H} = \nabla^2 f(\mathbf{x})$  is positive semi-definite for all  $\mathbf{x} \in \text{dom}(f)$ . Furthermore,  $f$  is strictly convex if  $\mathbf{H}$  is positive definite.

- For example, consider the quadratic form

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$$

- This is convex if  $\mathbf{A}$  is positive semi-definite.
- This is strictly convex if  $\mathbf{A}$  is positive definite.
- It is neither convex nor concave if

## Theorem

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice differentiable over its domain. Then  $f$  is convex iff  $\mathbf{H} = \nabla^2 f(\mathbf{x})$  is positive semi-definite for all  $\mathbf{x} \in \text{dom}(f)$ . Furthermore,  $f$  is strictly convex if  $\mathbf{H}$  is positive definite.

- For example, consider the quadratic form

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$$

- This is convex if  $\mathbf{A}$  is positive semi-definite.
- This is strictly convex if  $\mathbf{A}$  is positive definite.
- It is neither convex nor concave if  $\mathbf{A}$  has eigenvalues of mixed sign.

## Theorem

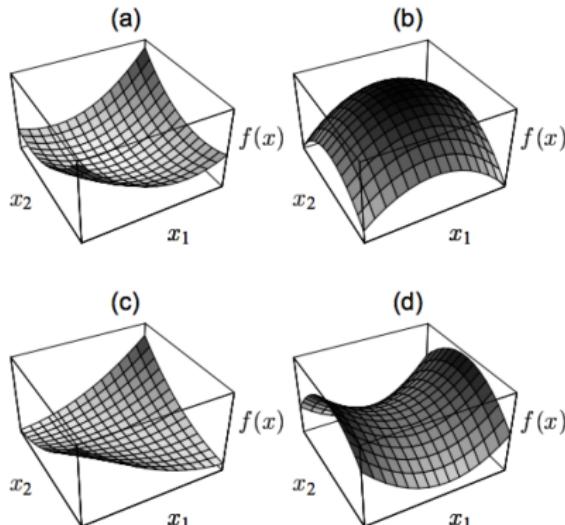
Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice differentiable over its domain. Then  $f$  is convex iff  $\mathbf{H} = \nabla^2 f(\mathbf{x})$  is positive semi-definite for all  $\mathbf{x} \in \text{dom}(f)$ . Furthermore,  $f$  is strictly convex if  $\mathbf{H}$  is positive definite.

- For example, consider the quadratic form

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$$

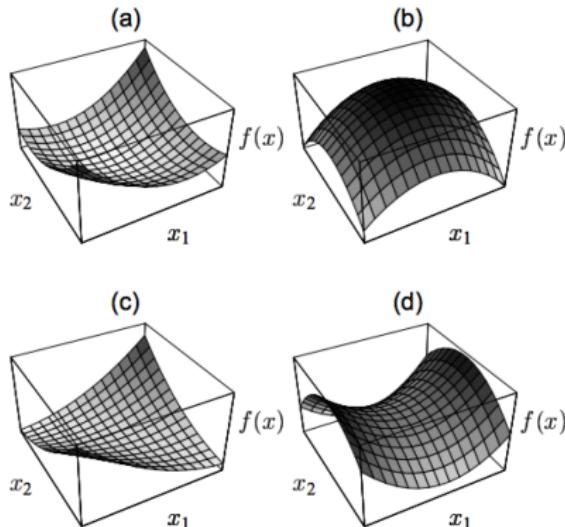
- This is convex if  $\mathbf{A}$  is positive semi-definite.
- This is strictly convex if  $\mathbf{A}$  is positive definite.
- It is neither convex nor concave if  $\mathbf{A}$  has eigenvalues of mixed sign.
- Intuitively, a convex function is shaped like a bowl.

# Convex functions



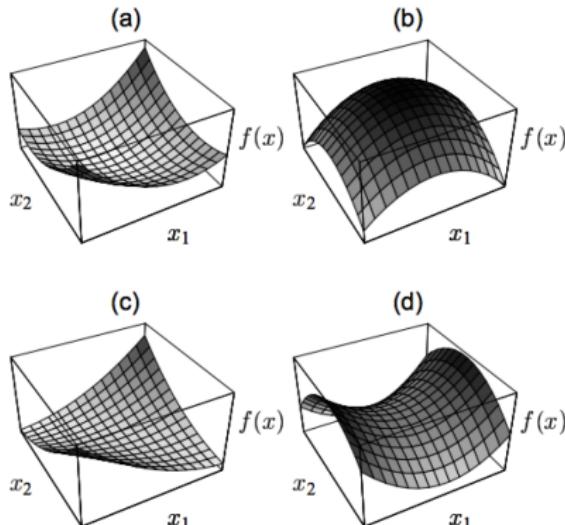
- The quadratic form  $f(x) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$  in  $2d$ .

# Convex functions



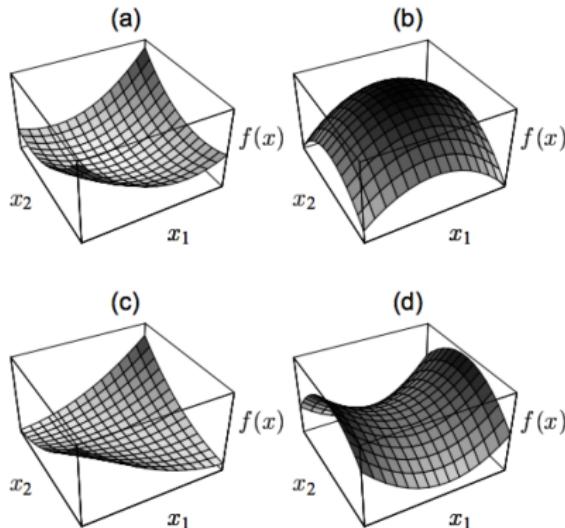
- The quadratic form  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x}$  in  $2d$ .
- (a)  $\mathbf{A}$  is positive definite, so  $f$  is strictly convex.

# Convex functions



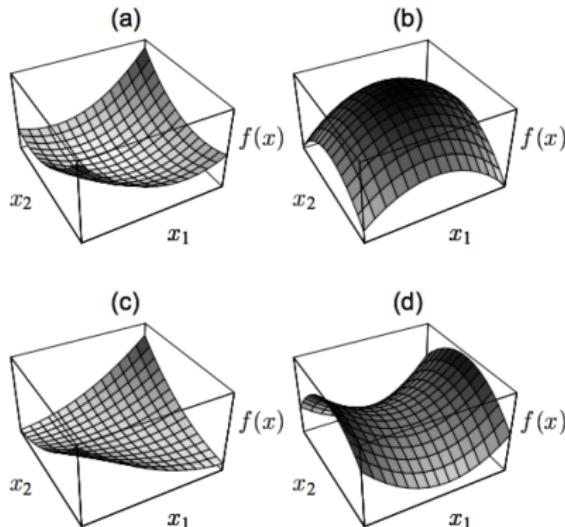
- The quadratic form  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$  in  $2d$ .
- (a)  $\mathbf{A}$  is positive definite, so  $f$  is strictly convex.
- (b)  $\mathbf{A}$  is negative definite, so  $f$  is strictly concave.

# Convex functions



- The quadratic form  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x}$  in  $2d$ .
- (a)  $\mathbf{A}$  is positive definite, so  $f$  is strictly convex.
- (b)  $\mathbf{A}$  is negative definite, so  $f$  is strictly concave.
- (c)  $\mathbf{A}$  is positive semi-definite, but singular, so  $f$  is convex.

# Convex functions

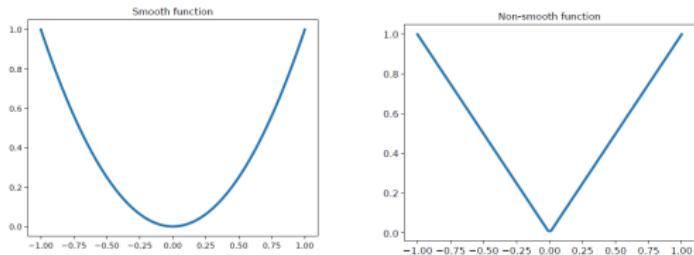


- The quadratic form  $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$  in  $2d$ .
- (a)  $\mathbf{A}$  is positive definite, so  $f$  is strictly convex.
- (b)  $\mathbf{A}$  is negative definite, so  $f$  is strictly concave.
- (c)  $\mathbf{A}$  is positive semi-definite, but singular, so  $f$  is convex.
- (d)  $\mathbf{A}$  is indefinite, so  $f$  is neither convex nor concave.

# Table of Contents

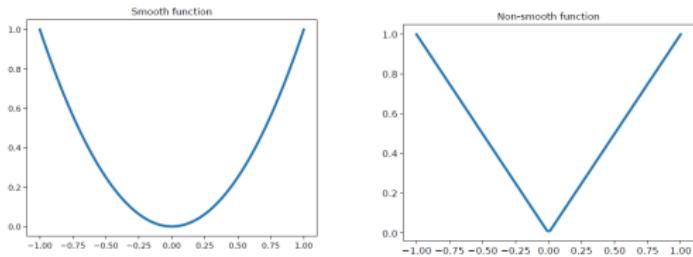
- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# Smooth vs nonsmooth optimization



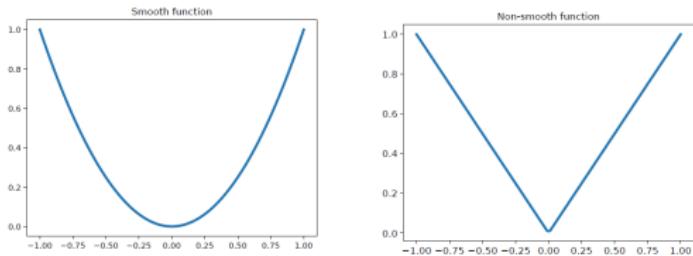
- In **smooth optimization**, the objective and constraints are continuously differentiable functions.

# Smooth vs nonsmooth optimization



- In **smooth optimization**, the objective and constraints are continuously differentiable functions.
- In **nonsmooth optimization**, there are some points where the gradient of the objective or the constraints is not well-defined.

# Smooth vs nonsmooth optimization

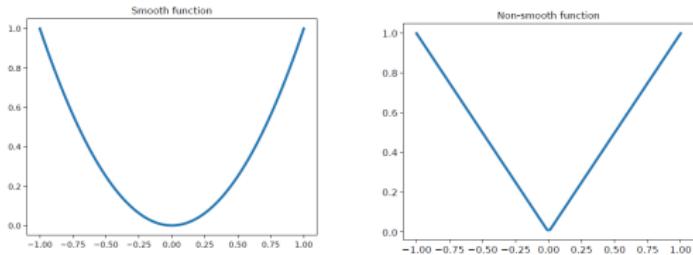


- In **smooth optimization**, the objective and constraints are continuously differentiable functions.
- In **nonsmooth optimization**, there are some points where the gradient of the objective or the constraints is not well-defined.
- In some problems, we partition the objective into a part that contains smooth terms, and a part that contains the nonsmooth terms:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_s(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$

where  $\mathcal{L}_s$  is smooth (differentiable), and  $\mathcal{L}_r$  is nonsmooth ("rough").

# Smooth vs nonsmooth optimization



- In **smooth optimization**, the objective and constraints are continuously differentiable functions.
- In **nonsmooth optimization**, there are some points where the gradient of the objective or the constraints is not well-defined.
- In some problems, we partition the objective into a part that contains smooth terms, and a part that contains the nonsmooth terms:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_s(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$

where  $\mathcal{L}_s$  is smooth (differentiable), and  $\mathcal{L}_r$  is nonsmooth ("rough").

- In ML,  $\mathcal{L}_s$  is the train loss, and  $\mathcal{L}_r$  is a regularizer, like  $\ell_1$  norm of  $\boldsymbol{\theta}$ .

# Smooth vs nonsmooth optimization

- For smooth functions, we can quantify the degree of smoothness using the **Lipschitz constant**.

# Smooth vs nonsmooth optimization

- For smooth functions, we can quantify the degree of smoothness using the **Lipschitz constant**.
- In the  $1d$  case, this is defined as any constant  $L \geq 0$  such that, for all real  $x_1$  and  $x_2$ , we have:

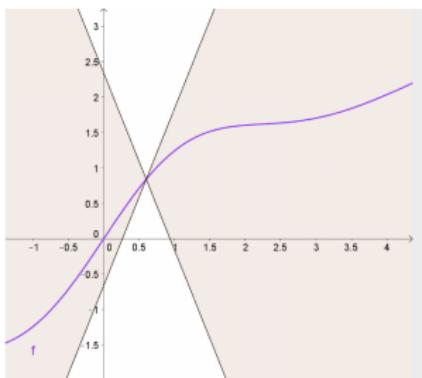
$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$$

# Smooth vs nonsmooth optimization

- For smooth functions, we can quantify the degree of smoothness using the **Lipschitz constant**.
- In the  $1d$  case, this is defined as any constant  $L \geq 0$  such that, for all real  $x_1$  and  $x_2$ , we have:

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$$

- Given a constant  $L$ , the function output cannot change by more than  $L$  if we change the function input by 1 unit.



# Smooth vs nonsmooth optimization

$$f(x) = \frac{x^2}{100} - 2x + 1$$

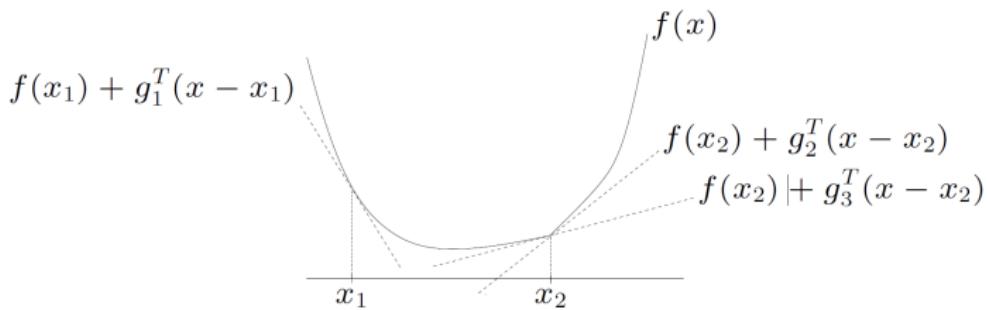
$$f(x) = \exp(\cos(e^{-x}))$$



# Subgradients

- We generalize the notion of a derivative to work with functions which have local discontinuities.
- For a convex function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , we say  $\mathbf{g} \in \mathbb{R}^n$  is a **subgradient** of  $f$  at  $\mathbf{x} \in \text{dom}(f)$  if for all vector  $\mathbf{z} \in \text{dom}(f)$ ,

$$f(\mathbf{z}) \geq f(\mathbf{x}) + \mathbf{g}^\top (\mathbf{z} - \mathbf{x})$$



- At  $x_1$ ,  $f$  is differentiable, and  $\mathbf{g}_1$  is the unique subgradient at  $x_1$
- At  $x_2$ ,  $f$  is not differentiable, and there are many subgradients at  $x_2$ .

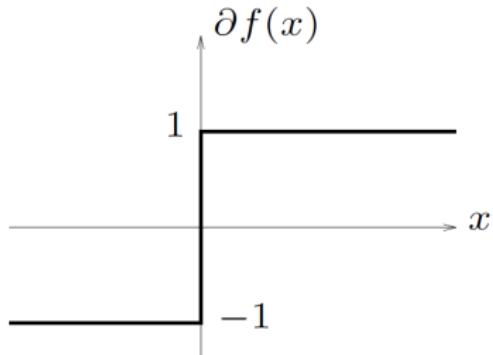
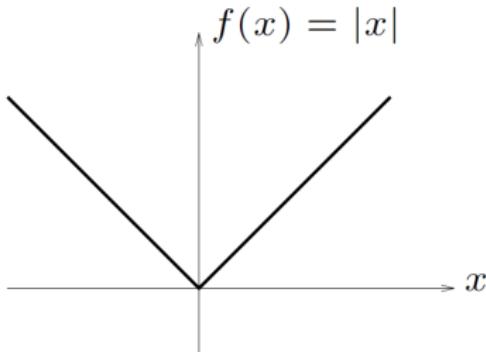
# Subgradients

- A function  $f$  is called subdifferentiable at  $x$ , if there is at least one subgradient at  $x$ .
- The set of such subgradients is called **subdifferential** of  $f$  at  $x$ , denoted as  $\partial f(x)$
- For example, consider  $f(x) = |x|$ . Its subdifferential is given by

$$\partial f(x) = \begin{cases} \{-1\}, & \text{if } x < 0 \\ [-1, 1], & \text{if } x = 0 \\ \{+1\}, & \text{if } x > 0 \end{cases}$$

where  $[-1, 1]$  here means any value between -1 and 1 (inclusive).

# Subgradients



$$\partial f(x) = \begin{cases} \{-1\}, & \text{if } x < 0 \\ [-1, 1], & \text{if } x = 0 \\ \{+1\}, & \text{if } x > 0 \end{cases}$$

where  $[-1, 1]$  here means any value between -1 and 1 (inclusive).

# Table of Contents

- 1 Introduction
- 2 Matrix calculus
- 3 Positive definite matrices
- 4 Optimality conditions
- 5 Constrained vs unconstrained optimization
- 6 Convex vs nonconvex optimization
- 7 Smooth vs nonsmooth optimization
- 8 First-order methods

# First-order methods

- We consider **iterative** optimization methods that leverage **first order** derivatives of the objective function.

# First-order methods

- We consider **iterative** optimization methods that leverage **first order** derivatives of the objective function.
- They compute which directions point “downhill”, but ignore curvature information.

# First-order methods

- We consider **iterative** optimization methods that leverage **first order** derivatives of the objective function.
- They compute which directions point “downhill”, but ignore curvature information.
- All these algorithms require the user specify a starting point  $\theta_0$ .

# First-order methods

- We consider **iterative** optimization methods that leverage **first order** derivatives of the objective function.
- They compute which directions point “downhill”, but ignore curvature information.
- All these algorithms require the user specify a starting point  $\theta_0$ .
- At each iteration  $t$ , an update is performed

$$\theta_{t+1} = \theta_t + \rho_t d_t$$

where  $\rho_t$  is the **step size** or **learning rate**, and  $d_t$  is a **descent direction**, e.g, the negative of the **gradient** given by  $g_t = \nabla_{\theta} \mathcal{L}(\theta)|_{\theta_t}$ .

- We consider **iterative** optimization methods that leverage **first order** derivatives of the objective function.
- They compute which directions point “downhill”, but ignore curvature information.
- All these algorithms require the user specify a starting point  $\theta_0$ .
- At each iteration  $t$ , an update is performed

$$\theta_{t+1} = \theta_t + \rho_t d_t$$

where  $\rho_t$  is the **step size** or **learning rate**, and  $d_t$  is a **descent direction**, e.g, the negative of the **gradient** given by  $g_t = \nabla_{\theta} \mathcal{L}(\theta)|_{\theta_t}$ .

- The update steps are continued until a **stationary point** is reached, where the gradient is zero.

## Descent direction

- A direction  $d$  is a **descent direction** if there is a small enough (but nonzero) amount  $\rho$  that we can move in direction  $d$  and be guaranteed to decrease the function value.

## Descent direction

- A direction  $d$  is a **descent direction** if there is a small enough (but nonzero) amount  $\rho$  that we can move in direction  $d$  and be guaranteed to decrease the function value.
- We require there exists an  $\rho_{max} > 0$  such that

$$\mathcal{L}(\theta + \rho d) < \mathcal{L}(\theta)$$

for all  $0 < \rho < \rho_{max}$ .

# Descent direction

- A direction  $d$  is a **descent direction** if there is a small enough (but nonzero) amount  $\rho$  that we can move in direction  $d$  and be guaranteed to decrease the function value.
- We require there exists an  $\rho_{max} > 0$  such that

$$\mathcal{L}(\boldsymbol{\theta} + \rho d) < \mathcal{L}(\boldsymbol{\theta})$$

for all  $0 < \rho < \rho_{max}$ .

- The gradient at the current iterate,

$$\mathbf{g}_t \triangleq \nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_t} = \nabla \mathcal{L}(\boldsymbol{\theta}_t) = \mathbf{g}(\boldsymbol{\theta}_t)$$

points in the direction of maximal increase in  $f$ , so the negative gradient is a descent direction.

# Descent direction

- Any direction  $d$  is also a descent direction if the angle  $\theta$  between  $d$  and  $-g_t$  is less than 90 degrees and satisfies

$$d^T g_t = \|d\| \|g_t\| \cos(\theta) < 0$$

# Descent direction

- Any direction  $d$  is also a descent direction if the angle  $\theta$  between  $d$  and  $-g_t$  is less than 90 degrees and satisfies

$$d^T g_t = \|d\| \|g_t\| \cos(\theta) < 0$$

- The best choice would be to pick  $d_t = -g_t$ .

## Descent direction

- Any direction  $d$  is also a descent direction if the angle  $\theta$  between  $d$  and  $-g_t$  is less than 90 degrees and satisfies

$$d^T g_t = \|d\| \|g_t\| \cos(\theta) < 0$$

- The best choice would be to pick  $d_t = -g_t$ .
- This is the direction of **steepest descent**.

## Step size (learning rate)

- The sequence of step sizes  $\{\rho_t\}$  is called the **learning rate schedule**.

## Step size (learning rate)

- The sequence of step sizes  $\{\rho_t\}$  is called the **learning rate schedule**.
- The simplest method is to use constant step size,  $\rho_t = \rho$ .

## Step size (learning rate)

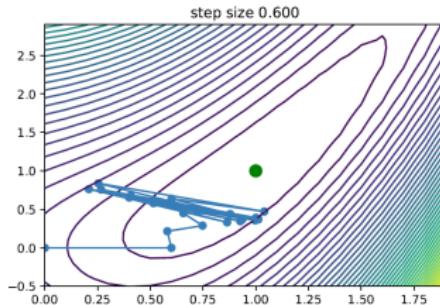
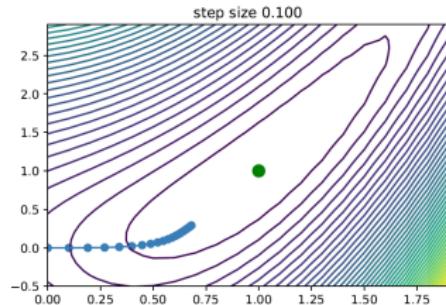
- The sequence of step sizes  $\{\rho_t\}$  is called the **learning rate schedule**.
- The simplest method is to use constant step size,  $\rho_t = \rho$ .
- However, if it is too large, the method may fail to converge. If it is too small, the function will converge but very slowly.

# Step size (learning rate)

- The sequence of step sizes  $\{\rho_t\}$  is called the **learning rate schedule**.
- The simplest method is to use constant step size,  $\rho_t = \rho$ .
- However, if it is too large, the method may fail to converge. If it is too small, the function will converge but very slowly.
- Example:

$$\mathcal{L}(\theta) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$$

- Pick our descent direction  $d_t = -g_t$ . Consider  $\rho_t = 0.1$  vs  $\rho_t = 0.6$ :



# Line search

- The **optimal step size** can be found by finding the value that maximally decreases the objective along the chosen direction by solving the  $1d$  minimization problem

$$\rho_t = \underset{\rho > 0}{\operatorname{argmin}} \phi_t(\rho) = \underset{\rho > 0}{\operatorname{argmin}} \mathcal{L}(\theta_t + \rho d_t)$$

- This is **line search**: we are searching along the line defined by  $d_t$ .
- $\phi_t(\rho) = \mathcal{L}(\theta_t + \rho d_t)$  is a convex function of an affine function of  $\rho$ , for fixed  $\theta_t$  and  $d$ .
- If the loss is convex, this subproblem is also convex.

# Line search

- Example, consider the quadratic loss

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \boldsymbol{\theta}^\top \mathbf{A} \boldsymbol{\theta} + \mathbf{b}^\top \boldsymbol{\theta} + c$$

- Compute the derivatives of  $\phi(\rho) = \mathcal{L}(\boldsymbol{\theta} + \rho \mathbf{d})$  gives

$$\begin{aligned}\frac{d\phi(\rho)}{d\rho} &= \frac{d}{d\rho} \left[ \frac{1}{2} (\boldsymbol{\theta} + \rho \mathbf{d})^\top \mathbf{A} (\boldsymbol{\theta} + \rho \mathbf{d}) + \mathbf{b}^\top (\boldsymbol{\theta} + \rho \mathbf{d}) + c \right] \\ &= \mathbf{d}^\top \mathbf{A} (\boldsymbol{\theta} + \rho \mathbf{d}) + \mathbf{d}^\top \mathbf{b} \\ &= \mathbf{d}^\top (\mathbf{A} \boldsymbol{\theta} + \mathbf{b}) + \rho \mathbf{d}^\top \mathbf{A} \mathbf{d}\end{aligned}$$

- Solving for  $\frac{d\phi(\rho)}{d\rho} = 0$  gives

$$\rho = -\frac{\mathbf{d}^\top (\mathbf{A} \boldsymbol{\theta} + \mathbf{b})}{\mathbf{d}^\top \mathbf{A} \mathbf{d}}$$

- This is **exact line search**. There are several methods, such as **Armijo backtracking method**, that try to ensure reduction in the objective function without spending too much time trying to solve this subproblem precisely.

# Regularization

## Gradient Descent with Momentum

## RMSprop

## Adam

13th December 2021

# Overview

- 1 Regularization
- 2 Mini-batch gradient descent
- 3 Gradient descent with momentum
- 4 RMSprop
- 5 Adam

# Regularization

What is regularization and why do we need it?

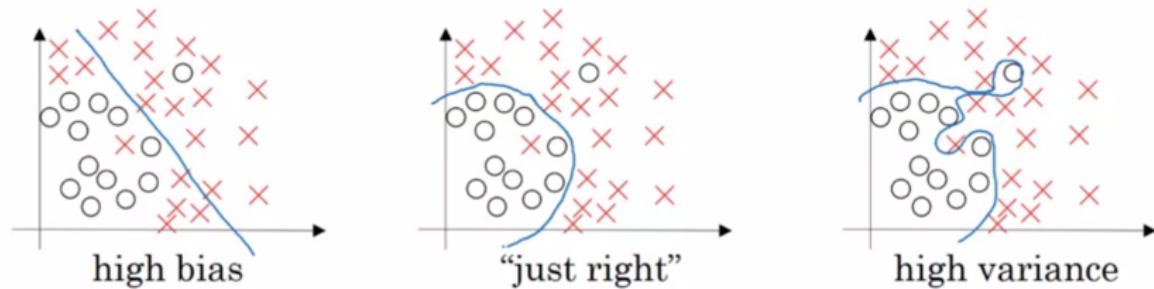


Figure: Bias-Variance <sup>1</sup>

---

<sup>1</sup><https://www.coursera.org/learn/deep-neural-network>

# Regularization - Logistic Regression

$$z^{(i)} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \dots + w_d x_d^{(i)}$$
$$a^{(i)} = \sigma(z^{(i)})$$

L2 regularization:

$$\min_{\mathbf{w}} L2(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L^{(i)}(a^{(i)}, y^{(i)}) + \frac{\lambda}{2N} \|\mathbf{w}\|_2^2$$

where  $\|\mathbf{w}\|_2^2 = \sum_{j=1}^d w_j^2$ .

L1 regularization:

$$\min_{\mathbf{w}} L1(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L^{(i)}(a^{(i)}, y^{(i)}) + \frac{\lambda}{2N} \|\mathbf{w}\|_1$$

where  $\|\mathbf{w}\|_1 = \sum_{j=1}^d |w_j|$ .

**Note:** We can omit regularization on  $w_0$ .

# Regularization - Neural Network

$$a^{(i)} = f^{[K]}(\mathbf{W}^{[K]}(\dots(f^{[2]}(\mathbf{W}^{[2]}(f^{[1]}(\mathbf{W}^{[1]}\mathbf{x}^{(i)}))))))$$

L2 regularization:

$$\min_{\mathbf{w}} L2(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L^{(i)}(a^{(i)}, y^{(i)}) + \frac{\lambda}{2N} \sum_{k=1}^K \|\mathbf{W}^{[k]}\|_F^2$$

where  $\|\mathbf{W}^{[k]}\|_F^2 = \sum_{i=1}^{n^{[k]}} \sum_{j=1}^{n^{[k-1]}} (w_{ij}^{[k]})^2$  is the Frobenius norm of the weight matrix in layer  $k$ .

L1 regularization:

$$\min_{\mathbf{w}} L1(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L^{(i)}(a^{(i)}, y^{(i)}) + \frac{\lambda}{2N} \sum_{k=1}^K \|\mathbf{W}^{[k]}\|_1$$

where  $\|\mathbf{W}^{[k]}\|_1 = \sum_{i=1}^{n^{[k]}} \sum_{j=1}^{n^{[k-1]}} |w_{ij}^{[k]}|$ .

**Note:** We can omit regularization on  $w_{i0}^{[k]}$ .

## Regularization - Neural Network

$\nabla L(\mathbf{W}^{[k]})$  is the gradient of the loss function without regularization obtained from backpropagation.

The gradient of the loss function with L2 regularization can be computed to update the weight matrix of layer  $k$  as:

$$\nabla L2(\mathbf{W}^{[k]}) \leftarrow \nabla L(\mathbf{W}^{[k]}) + \frac{\lambda}{N} \mathbf{W}^{[k]}$$

$$\mathbf{W}^{[k]} \leftarrow \mathbf{W}^{[k]} - \gamma \nabla L2(\mathbf{W}^{[k]})$$

$$\mathbf{W}^{[k]} \leftarrow \mathbf{W}^{[k]} - \gamma \left( \nabla L(\mathbf{W}^{[k]}) + \frac{\lambda}{N} \mathbf{W}^{[k]} \right)$$

$$\mathbf{W}^{[k]} \leftarrow \left( 1 - \frac{\gamma \lambda}{N} \right) \mathbf{W}^{[k]} - \gamma \nabla L(\mathbf{W}^{[k]})$$

# Gradient descent for $L$ -layer neural network

**Result:** weights  $\mathbf{W}^{[k]}$  for all layers.

**for**  $k \leftarrow 1$  **to**  $L$  **do**

    |  $\mathbf{W}^{[k]} \leftarrow \text{random}()$    *[Random initialization]*

**end**

**for**  $t \leftarrow 1$  **to**  $\text{max\_iterations}$  **do**

    | **for**  $k \leftarrow 1$  **to**  $L$  **do**

        |  $\mathbf{A}^{[k]} \leftarrow f(\mathbf{Z}^{[k]}) \leftarrow f(\mathbf{W}^{[k]} \cdot \mathbf{A}^{[k-1]})$    *[Forward propagation]*

    | **end**

    | **for**  $k \leftarrow L$  **to** 1 **do**

        |  $\nabla L(\mathbf{Z}^{[k]}) \leftarrow \nabla L(\mathbf{A}^{[k]}) \circ f'(\mathbf{Z}^{[k]})$

        |  $\nabla L(\mathbf{W}^{[k]}) \leftarrow \frac{1}{N} \nabla L(\mathbf{Z}^{[k]}) \cdot \mathbf{A}^{[k-1]T}$    *[Backpropagation]*

        |  $\nabla L(\mathbf{A}^{[k-1]}) \leftarrow \mathbf{W}^{[k]T} \cdot \nabla L(\mathbf{Z}^{[k]})$

    | **end**

    | **for**  $k \leftarrow 1$  **to**  $L$  **do**

        |  $\mathbf{W}^{[k]} \leftarrow \mathbf{W}^{[k]} - \gamma \nabla L(\mathbf{W}^{[k]})$    *[Update parameters]*

    | **end**

**end**

## Batch gradient descent

If the training set is too large, it takes a lot of time to process the whole training set for each step of gradient descent.

## Mini-batch gradient descent

If the training set is too large, it takes a lot of time to process the whole training set for each step of gradient descent.

→ Divide the training set into multiple mini-batches of **reasonable** sizes.

$$\mathbf{X} = [\mathbf{x}^{(1)} \quad \mathbf{x}^{(2)} \dots \mathbf{x}^{(s)} \quad | \quad \mathbf{x}^{(s+1)} \dots \mathbf{x}^{(2s)} \quad | \quad \mathbf{x}^{(2s+1)} \dots \quad | \quad \dots \mathbf{x}^{(N)}]$$

$$= [\mathbf{X}_{\{1\}} \quad | \quad \mathbf{X}_{\{2\}} \quad | \quad \dots \quad | \quad \mathbf{X}_{\{N/s\}}]$$

$$\mathbf{Y} = [y^1 \quad y^2 \dots y^s \quad | \quad y^{s+1} \dots y^{2s} \quad | \quad y^{2s+1} \dots \quad | \quad \dots y^N]$$

$$= [\mathbf{Y}_{\{1\}} \quad | \quad \mathbf{Y}_{\{2\}} \quad | \quad \dots \quad | \quad \mathbf{Y}_{\{N/s\}}]$$

At each iteration  $t$ , perform gradient descent using one mini-batch of training examples  $(\mathbf{X}_{\{t\}}; \mathbf{Y}_{\{t\}})$ .

## Mini-batch gradient descent

- Mini-batch size  $s = N \longrightarrow$  Normal gradient descent.
- Mini-batch size  $s = 1 \longrightarrow$  Stochastic gradient descent.
- Mini-batch sizes are normally set as powers of 2, e.g.,  
 $s = 64, 128, 256, 512$

# Problem with gradient descent

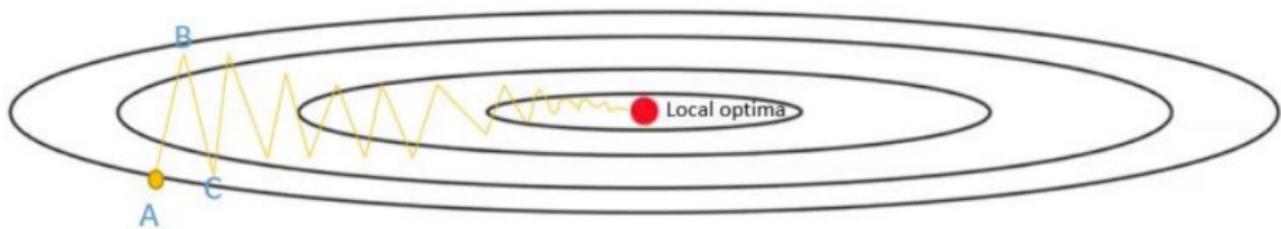


Figure: Gradient descent <sup>2</sup>

---

<sup>2</sup><https://engmrk.com/gradient-descent-with-momentum/>.

# Problem with gradient descent

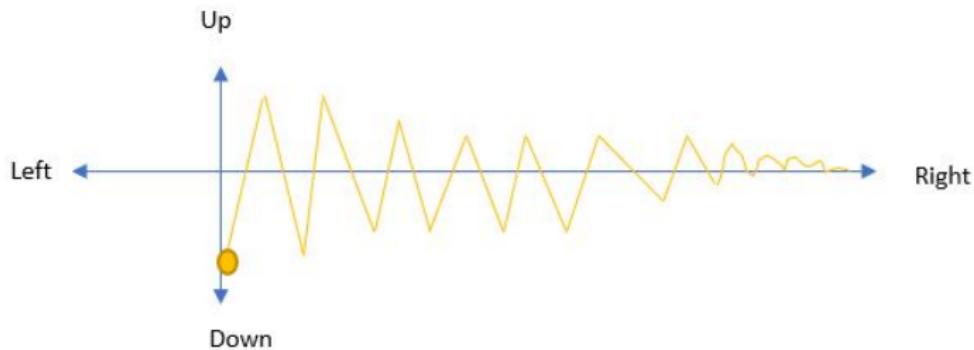


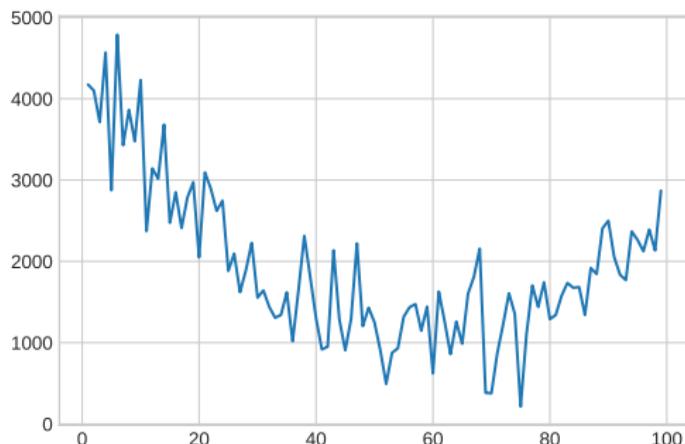
Figure: Gradient descent<sup>3</sup>

---

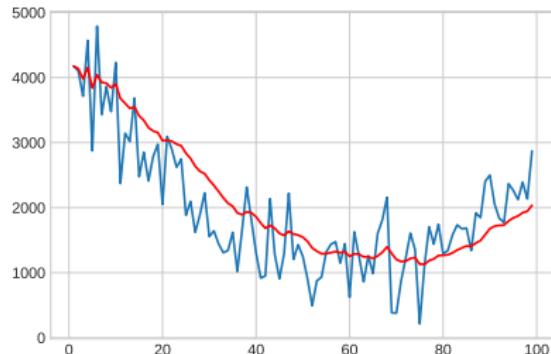
<sup>3</sup><https://engmrk.com/gradient-descent-with-momentum/>.

# Exponentially weighted moving average

Example: How to model/forecast the stock/share price of a certain company?



## Exponentially weighted moving average



$$m_{100} = \beta m_{99} + (1 - \beta) \text{price}_{100}$$

$$m_{99} = \beta m_{98} + (1 - \beta) \text{price}_{99}$$

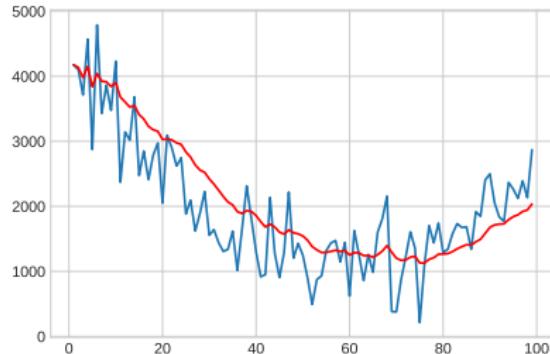
...

$$m_t = \beta m_{t-1} + (1 - \beta) \text{price}_t$$

...

$$m_1 = \beta m_0 + (1 - \beta) \text{price}_1$$

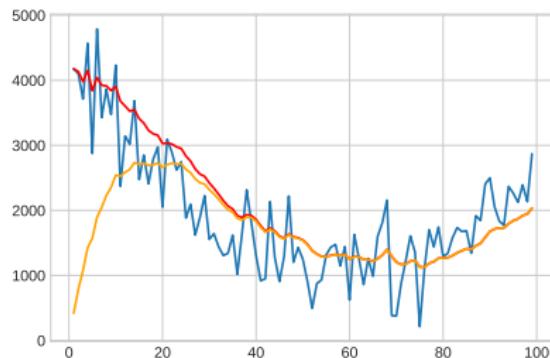
## Exponentially weighted moving average



$$\beta = 0.9$$

$$\begin{aligned}m_{100} &= 0.9m_{99} + (1 - 0.9)price_{100} \\&= 0.1price_{100} + 0.9m_{99} \\&= 0.1price_{100} + 0.9(0.1price_{99} + 0.9m_{98}) \\&= 0.1price_{100} + 0.9 \cdot 0.1price_{99} + (0.9)^2 m_{98} \\&= 0.1price_{100} + 0.09price_{99} + (0.9)^2(0.1price_{98} + 0.9m_{97}) \\&= 0.1price_{100} + 0.09price_{99} + 0.081price_{98} + (0.9)^3 m_{97} \\&\equiv \dots\end{aligned}$$

# Exponentially weighted moving average



Actually, without the below correction, we will get the orange line instead of the red line. Why?

$$m_t = \beta m_{t-1} + (1 - \beta) price_t$$

$$m_t^{corrected} = \frac{m_t}{1 - \beta^t}$$

# Gradient descent with momentum

Update weights at iteration  $t$ :

$$\begin{aligned}\mathbf{g}_{\{t\}} &\leftarrow \nabla L(\mathbf{W}_{\{t-1\}}) \\ \mathbf{m}_{\{t\}} &\leftarrow \beta_1 \mathbf{m}_{\{t-1\}} + (1 - \beta_1) \mathbf{g}_{\{t\}} \\ \mathbf{W}_{\{t\}} &\leftarrow \mathbf{W}_{\{t-1\}} - \gamma \mathbf{m}_{\{t\}}\end{aligned}$$

Initialization:  $\mathbf{m}_{\{0\}} = \mathbf{0}$

Hyperparameters:  $\beta_1 = 0.9$ ; learning rate  $\gamma$

# Gradient descent with momentum

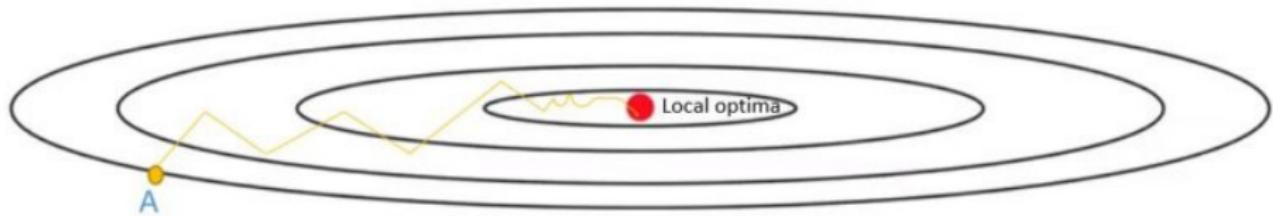


Figure: Gradient descent with momentum <sup>4</sup>

---

<sup>4</sup><https://engmrk.com/gradient-descent-with-momentum/>.

# Problem with gradient descent

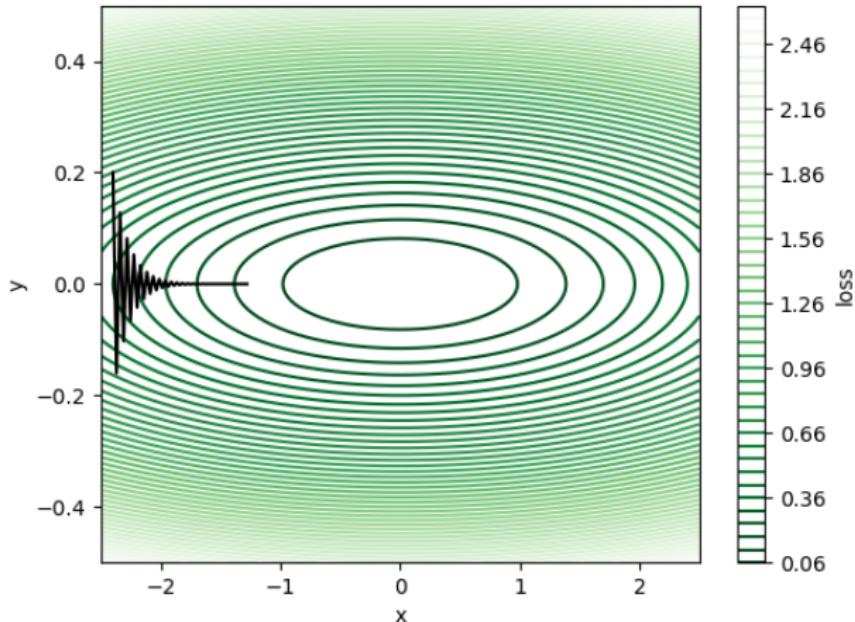


Figure: Gradient descent <sup>5</sup>

<sup>5</sup>[https://github.com/hengluchang/visualizing\\_momentum](https://github.com/hengluchang/visualizing_momentum).

# Gradient descent with momentum

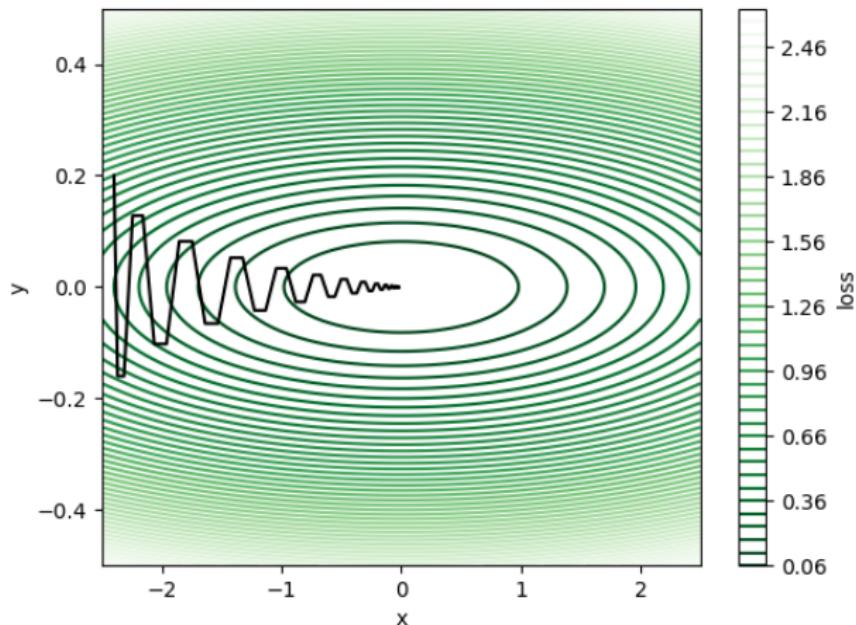


Figure: Gradient descent with momentum  $\beta_1 = 0.8$ <sup>6</sup>

<sup>6</sup>[https://github.com/hengluchang/visualizing\\_momentum](https://github.com/hengluchang/visualizing_momentum).

# RMSprop

Update weights at iteration  $t$ :

$$\begin{aligned}\mathbf{g}_{\{t\}} &\leftarrow \nabla L(\mathbf{W}_{\{t-1\}}) \\ \mathbf{s}_{\{t\}} &\leftarrow \beta_2 \mathbf{s}_{\{t-1\}} + (1 - \beta_2) \mathbf{g}_t^2 \\ \mathbf{W}_{\{t\}} &\leftarrow \mathbf{W}_{\{t-1\}} - \gamma \frac{\mathbf{g}_{\{t\}}}{\sqrt{\mathbf{s}_{\{t\}}} + \epsilon}\end{aligned}$$

Initialization:  $\mathbf{s}_{\{0\}} = \mathbf{0}$

Hyperparameters:  $\beta_2 = 0.9$ ;  $\epsilon = 10^{-8}$ ; learning rate  $\gamma$

# Adam - Adaptive moment estimation

Update weights at iteration  $t$ :

$$\begin{aligned}\mathbf{g}_{\{t\}} &\leftarrow \nabla L(\mathbf{W}_{\{t-1\}}) \\ \mathbf{m}_{\{t\}} &\leftarrow \beta_1 \mathbf{m}_{\{t-1\}} + (1 - \beta_1) \mathbf{g}_{\{t\}} \\ \mathbf{s}_{\{t\}} &\leftarrow \beta_2 \mathbf{s}_{\{t-1\}} + (1 - \beta_2) \mathbf{g}_t^2 \\ \mathbf{m}_{\{t\}}^{corrected} &\leftarrow \frac{\mathbf{m}_{\{t\}}}{1 - \beta_1^t} \\ \mathbf{s}_{\{t\}}^{corrected} &\leftarrow \frac{\mathbf{s}_{\{t\}}}{1 - \beta_2^t} \\ \mathbf{W}_{\{t\}} &\leftarrow \mathbf{W}_{\{t-1\}} - \gamma \frac{\mathbf{m}_{\{t\}}^{corrected}}{\sqrt{\mathbf{s}_{\{t\}}^{corrected}} + \epsilon}\end{aligned}$$

Initialization:  $\mathbf{m}_{\{0\}} = \mathbf{s}_{\{0\}} = \mathbf{0}$

Hyperparameters:  $\beta_1 = 0.9$ ;  $\beta_2 = 0.999$ ;  $\epsilon = 10^{-8}$ ; learning rate  $\gamma$

## References

- ① Visualizing Gradient Descent with Momentum in Python:  
[https://github.com/hengluchang/visualizing\\_momentum](https://github.com/hengluchang/visualizing_momentum).
- ② Coursera - Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization:  
<https://www.coursera.org/learn/deep-neural-network/>
- ③ Coursera - Neural Networks and Deep Learning:  
<https://www.coursera.org/learn/neural-networks-deep-learning/>

# Data Preprocessing

Quan Minh Phan & Ngoc Hoang Luong

University of Information Technology

Vietnam National University Ho Chi Minh City

November 20, 2022

# Overview

1 A roadmap for building machine learning system

2 Data Pre-processing

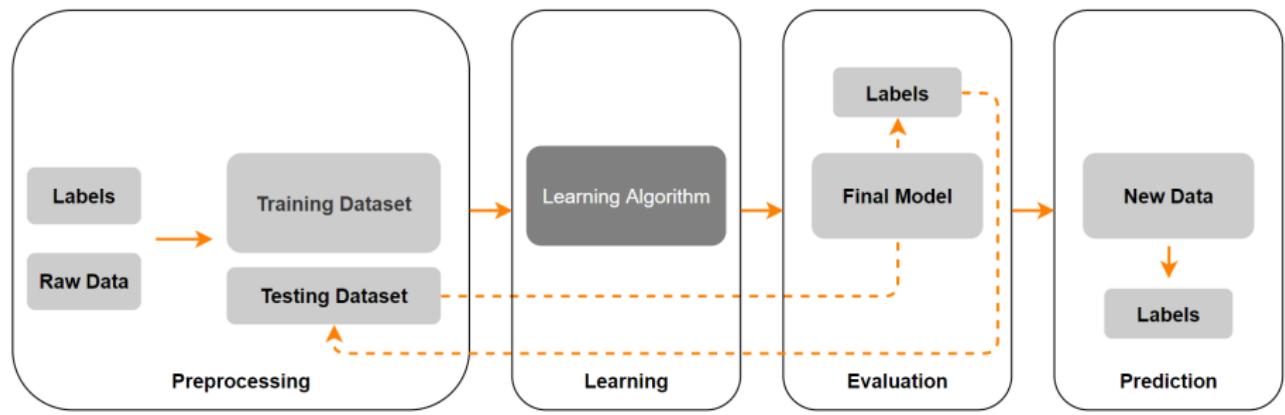
3 K-Nearest Neighbors

4 Model Evaluation

# Roadmap

5 major steps:

- Data Pre-processing
- Model Learning
- Model Evaluation
- Prediction
- Model Deployment



# Overview

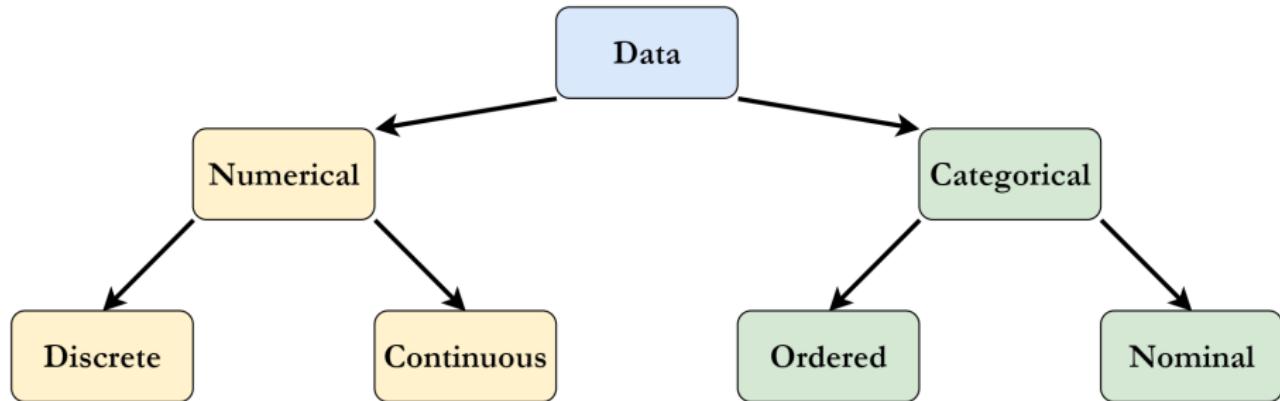
1 A roadmap for building machine learning system

2 Data Pre-processing

3 K-Nearest Neighbors

4 Model Evaluation

# Types of Data



**Numerical:** quantitative data

- Discrete: the number of students, the age of a person, ...
- Continuous: the height of a person, the score of a student, ....

**Categorical:** qualitative data

- Ordered: food ratings (excellent, good, bad), feelings (happy, not bad, bad), ...
- Nominal: the name of students, ...

# How to load data?

## Syntax (load)

```
pandas.read_csv(filepath)
```

## Examples

```
>> import pandas as pd  
>> data = pd.read_csv('/content/drive/MyDrive/Colab/mini_data.csv')
```

## Syntax (show)

```
pandas.DataFrame.head(n)
```

## Examples

```
>> data.head(n = 5)
```

# Data Representation

	ID	Age	Sex	Height	Grade	Good-looking
0	1	21.0	Male	171.0	Good	Yes
1	2	20.0	Male	170.0	Bad	No
2	3	19.0	Female	NaN	Bad	No
3	4	17.0	Male	165.0	Excellent	Yes
4	5	NaN	Female	166.0	Good	Yes

**Independent variables** should NOT contain

- Missing or NULL values
- Outliers
- Data on different scales
- Special characters
- ...

# Data Cleaning

- The processes of detecting and correcting (or removing) missing values or outliers.
- Ensuring data is correct, consistent and usable.

# Missing values

- In .csv files, missing values are usually represented as empty, 'NA', 'N/A', 'null', 'nan', 'NaN'.

	ID	Age	Sex	Height	Grade	Good-looking
0	1	21.0	Male	171.0	Good	Yes
1	2	20.0	Male	170.0	Bad	No
2	3	19.0	Female	NaN	Bad	No
3	4	17.0	Male	165.0	Excellent	Yes
4	5	NaN	Female	166.0	Good	Yes

# Missing values (cont.)

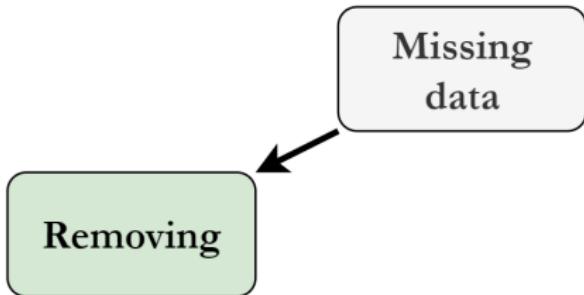
## Syntax (count 'NaN')

**pandas.DataFrame.isna().sum()**

## Examples

```
> countNULL = data.isna().sum()  
> null_columns = countNULL[countNULL > 0]  
> null_columns
```

# How to handle?



# Removing

## Syntax

**pandas.DataFrame.dropna(*inplace*)**

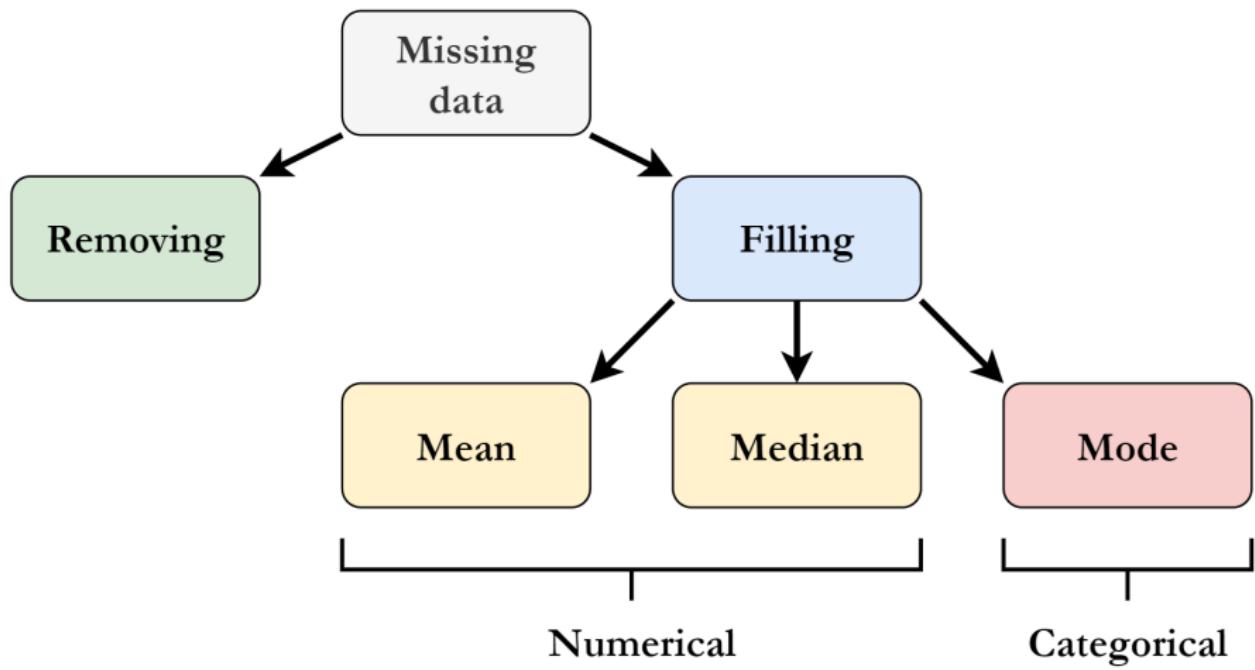
## Examples

> `data.dropna(inplace = True)`

or

> `data = data.dropna(inplace = False)`

## How to handle? (cont.)



# Filling

## Examples

Find the mean, median, and mode for the following list of values:

13, 18, 13, 14, 13, 16, 14, 21, 13

### Mean

- $\text{mean} = (13 + 18 + 13 + 14 + 13 + 16 + 14 + 21 + 13)/9 = 15$

### Median

- Sorting the list: 13, 13, 13, 13, 14, 14, 16, 18, 21
- $\text{median} = 14$

### Mode

- $\text{mode} = 13$

# Filling (cont.)

## Step 1: Calculating the filling values

### Syntax (calculate the mean)

**pandas.DataFrame.mean()**

### Examples

```
> mean_age = data['Age'].mean()  
> mean_age
```

### Syntax (calculate the median)

**pandas.DataFrame.median()**

### Examples

```
> median_height = data['Height'].median()  
> median_height
```

# Filling (cont.)

## Step 1: Calculating the filling values

Syntax (calculate the mode)

**pandas.DataFrame.mode()[0]**

### Examples

```
> mode_grade = data['Grade'].mode()[0]  
> mode_grade
```

# Filling (cont.)

## Step 2: Replacing 'NaN' by the filling values

### Syntax

```
pandas.DataFrame.fillna(value, inplace)
```

### Examples

```
> data['Age'].fillna(value = mean_age, inplace = True)  
> data['Height'].fillna(value = median_height, inplace = True)  
> data['Grade'].fillna(value = mode_grade, inplace = True)
```

# Outliers

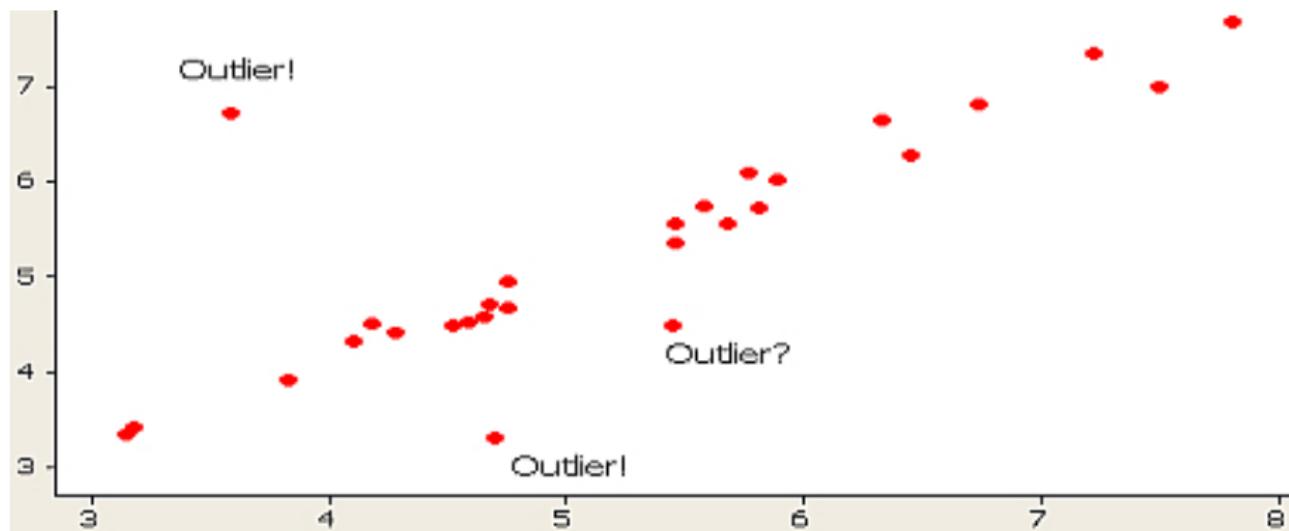


Figure: Examples of outliers

# Outliers

Syntax (plot the outliers)

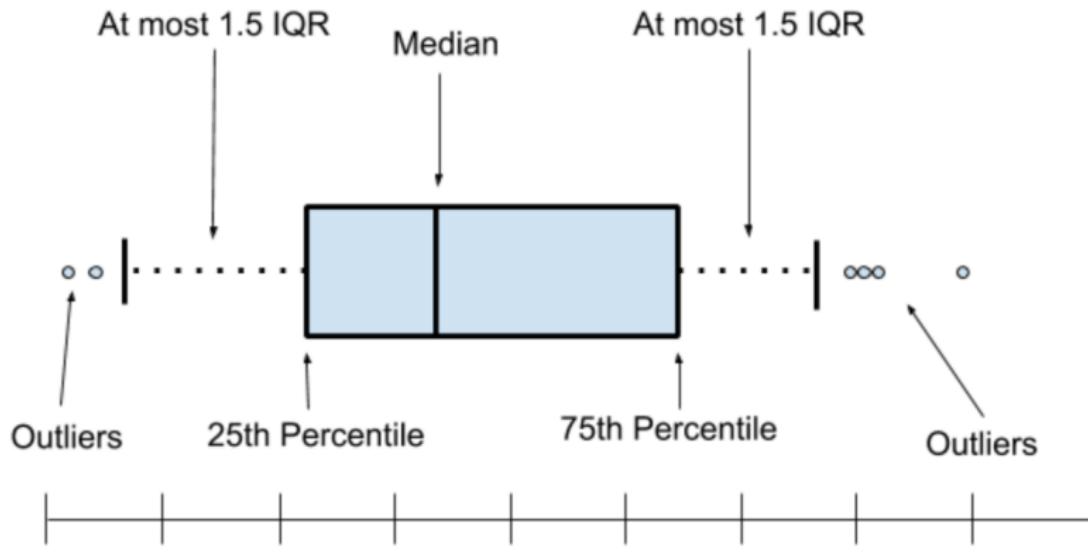
`seaborn.boxplot(data)`

Examples

```
>> import seaborn as sns
```

```
>> sns.boxplot(data['Height'])
```

# Outliers



X Axis

Shows data range and labels  
the values you are graphing.

# Outliers

## Examples

Find the outliers on 71, 70, 90, 70, 70, 60, 70, 72, 72, 320, 71, 69

# Outliers

## Examples

Find the outliers on 71, 70, 90, 70, 70, 60, 70, 72, 72, 320, 71, 69

## Solution

- Sort the data: 60, 69, 70, 70, 70, 70, 71, 71, 72, 72, 90, 320
- Calculate the median ( $Q_2$ )  $\rightarrow (70 + 71)/2 = 70.5$
- Calculate the lower quartile ( $Q_1$ )  $\rightarrow (70 + 70)/2 = 70.0$
- Calculate the upper quartile ( $Q_3$ )  $\rightarrow (72 + 72)/2 = 72$
- Calculate the interquartile range (IQR)  $\rightarrow Q_3 - Q_1 = 72 - 70 = 2$
- Find the upper and lower fences.  
Lower fence =  $Q_1 - 1.5 * \text{IQR} = 70 - 1.5 * 2 = 67$   
Upper fence =  $Q_3 + 1.5 * \text{IQR} = 71.5 + 1.5 * 2 = 74.5$
- The data points that are lower than the lower fence and greater than the upper fence are outliers  $\rightarrow$  outliers: 60; 90; 320.

# Outliers (cont.)

## Examples

```
>> Q1 = data['Height'].quantile(0.25)
    Q3 = data['Height'].quantile(0.75)
    IQR = Q3 - Q1

>> low_fence = Q1 - (1.5 * IQR)
    up_fence = Q3 + (1.5 * IQR)

>> data[((data['Height'] < low_fence)|(data['Height'] > up_fence))]

>> data = data[~((data['Height'] < low_fence)|(data['Height'] >
    up_fence))]
```

# Data Transformation

**Label Encoding:** replacing each value in a categorical column with numbers from 0 to  $N - 1$

## Syntax (initialize)

```
sklearn.preprocessing.LabelEncoder()
```

## Examples

```
>> from sklearn.preprocessing import LabelEncoder  
>> label_encoder = LabelEncoder()
```

# Label Encoding

## Syntax (fit & transform)

```
sklearn.preprocessing.LabelEncoder().fit_transform(X)
```

## Examples

```
>> data['Sex'] = label_encoder.fit_transform(data['Sex'])
```

# Data Transformation (cont.)

**One-hot Encoding:** dividing a categorical column into  $n$  number of columns with  $n$  is the total number of unique labels in that column.

## Syntax (initialize)

```
sklearn.preprocessing.OneHotEncoder(sparse)
```

## Examples

```
>> from sklearn.preprocessing import OneHotEncoder  
>> one_hot_encoder = OneHotEncoder(sparse = False)
```

# One-hot Encoding

## Syntax (fit & transform)

```
sklearn.preprocessing.OneHotEncoder().fit_transform(X)
```

## Examples

```
>> column = 'Grade'  
>> data_new_column = one_hot_encoder.fit_transform(data[[name_col]])  
>> new_column = pd.DataFrame(data=data_new,  
    columns=encoder.get_feature_names([column]))  
>> data = pd.concat([data.drop(columns=[column, 'Good-looking']),  
    new_column, data['Good-looking']], axis=1)
```

# Data Scaling

**Normalization:** involves to the rescaling of the features to a range of [0, 1]

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}$$

where:

- $x_{max}$ : the largest value of column  $x$
- $x_{min}$ : the smallest value of column  $x$

**Standardization:** centers the columns at the mean 0 with the standard deviation 1

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

where:

- $\mu_x$ : the mean of column  $x$
- $\sigma_x$ : the standard deviation of column  $x$

# Normalization

## Syntax

```
sklearn.preprocessing.MinMaxScaler()
```

## Examples

```
>> from sklearn.preprocessing import MinMaxScaler  
>> min_max_scaler = MinMaxScaler()  
>> data[['Age']] = min_max_scaler.fit_transform(data[['Age']])
```

# Standardization

## Syntax

```
sklearn.preprocessing.StandardScaler()
```

## Examples

```
>> from sklearn.preprocessing import StandardScaler  
>> std_scaler = StandardScaler()  
>> data[['Height']] = std_scaler.fit_transform(data[['Height']])
```

# Data Splitting

## Syntax

```
sklearn.model_selection.train_test_split(X, y, test_size, random_state)
```

## Examples

```
>> from sklearn.model_selection import train_test_split  
>> X = data.drop(columns = ['Good-looking', 'ID'])  
     y = data['Good-looking']  
>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

# Exercises

DataPreprocessing\_exercise.pdf

# Overview

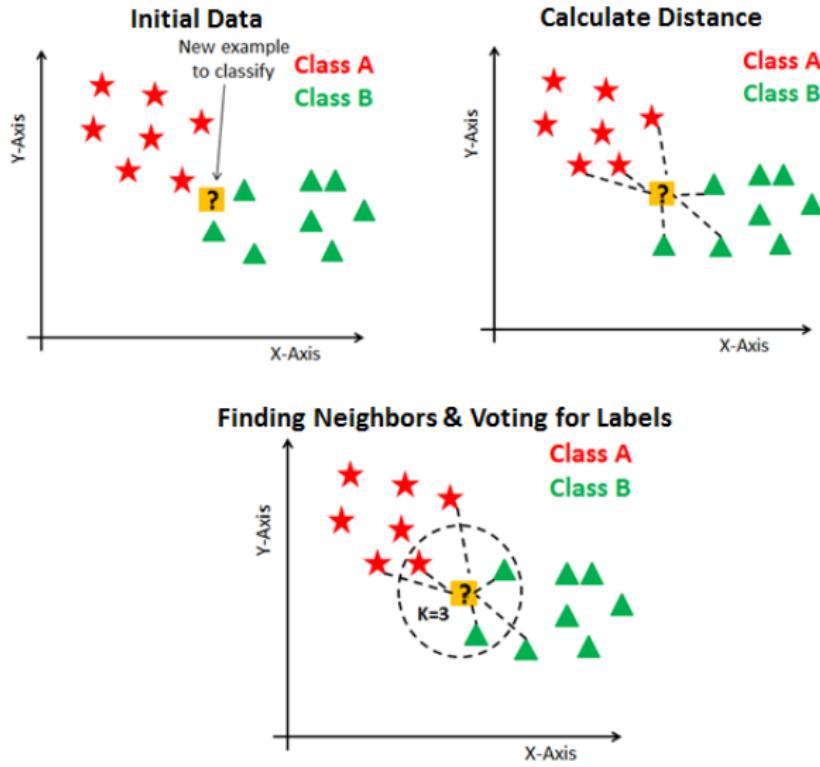
1 A roadmap for building machine learning system

2 Data Pre-processing

3 K-Nearest Neighbors

4 Model Evaluation

# Recall



# How to implement?

## Syntax (initialize)

```
sklearn.neighbors.KNeighborsClassifier(n_neighbors, p)
```

where:

- *n\_neighbors*: the number of neighbors ( $K$ )
- *p*: power parameter for the Minkowski metric.
  - ▶  $p = 1$ : Manhattan distance
  - ▶  $p = 2$ : Euclidean distance
  - ▶  $p > 2$ : Minkowski distance

## Examples

```
>> from sklearn.neighbors import KNeighborsClassifier  
>> clf = KNeighborsClassifier(n_neighbors = 3, p = 2)
```

# How to implement? (cont.)

## Syntax (fit)

```
sklearn.neighbors.KNeighborsClassifier().fit(X, y)
```

## Examples

```
>> clf.fit(X_train, y_train)
```

## Syntax (predict)

```
sklearn.neighbors.KNeighborsClassifier().predict(X)
```

## Examples

```
>> y_pred = clf.predict(X_test)
```

# Overview

- 1 A roadmap for building machine learning system
- 2 Data Pre-processing
- 3 K-Nearest Neighbors
- 4 Model Evaluation

# Performance Metrics

## Classification

- Accuracy
- Confusion matrix
- Precision and Recall
- F1 score

## Regression

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R-Squared

### Syntax (import)

```
from sklearn.metrics import ...
```

### Examples

```
>> from sklearn.metrics import accuracy_score  
>> accuracy = accuracy_score(y_test, y_pred)  
accuracy
```

# Exercise

KNN\_exercise.pdf