

机器学习与人工智能

Machine Learning and

Artificial Intelligence

Lecture 1 Introduction/Overview

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

Me...

Emails me: start with “[MLAI]” in the subject title



- Email: yingjiezhang@gsm.pku.edu.cn
- Office hour: Wed 10-11am; or by appointment
- Website: sites.google.com/view/yingjiezhang/home
- My Research:
 - Topics: Mobile and Sensor Technologies, Big Data and Smart City, User-generated Content, Sharing Economy, and Social Media.
 - Methodologies: Econometrics, Machine Learning, Text Mining, and Field Experiment.

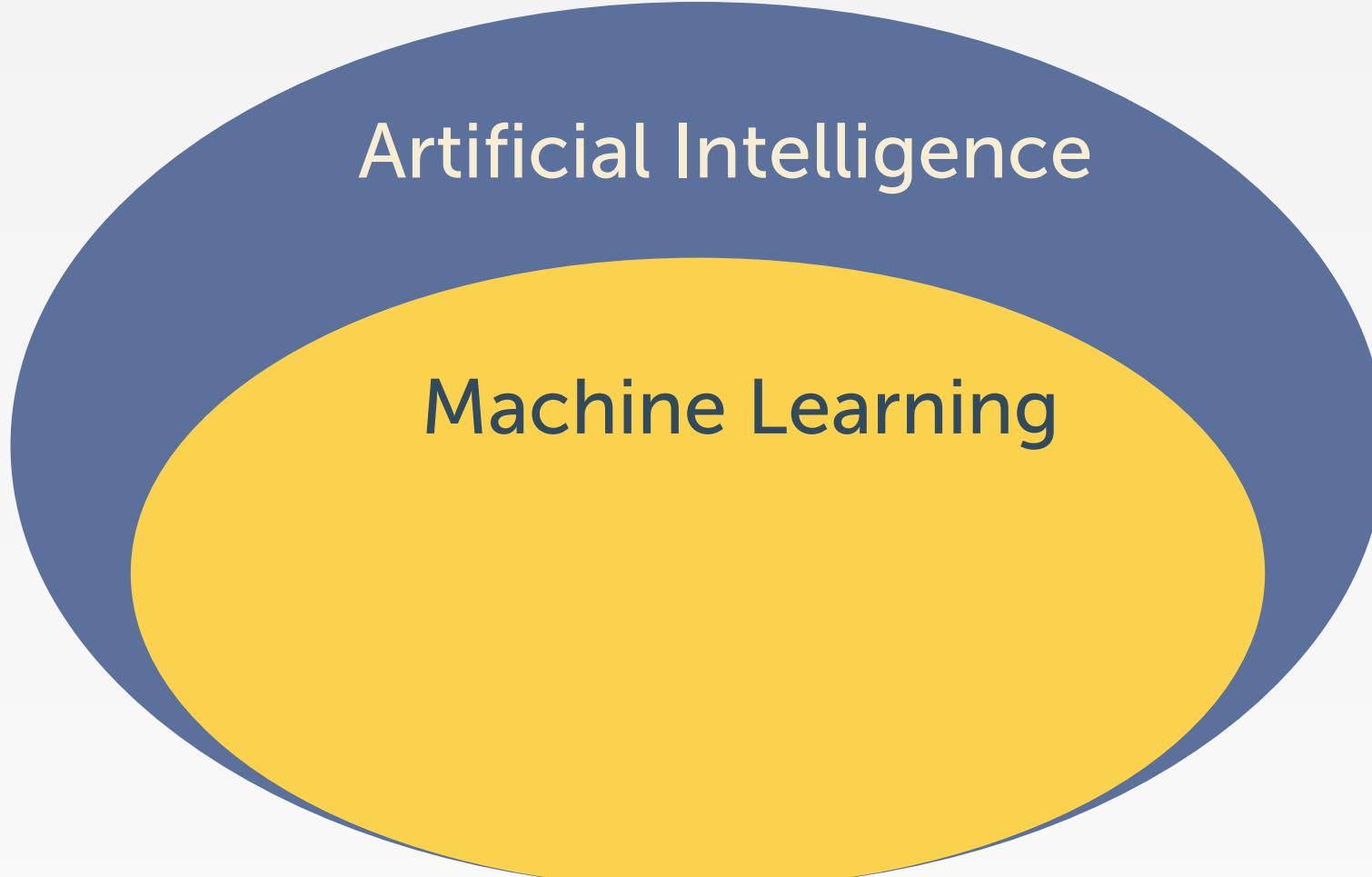
Teaching Assistant

Guangxin Yang (杨广鑫)

- 2nd PhD Student in Marketing
- Email: [ygx@stu.pku.edu.cn.](mailto:ygx@stu.pku.edu.cn)
- TA session: Saturday 10:00-11:00; or by appointment
- A novice at ML with you guys. Debug Python code together!

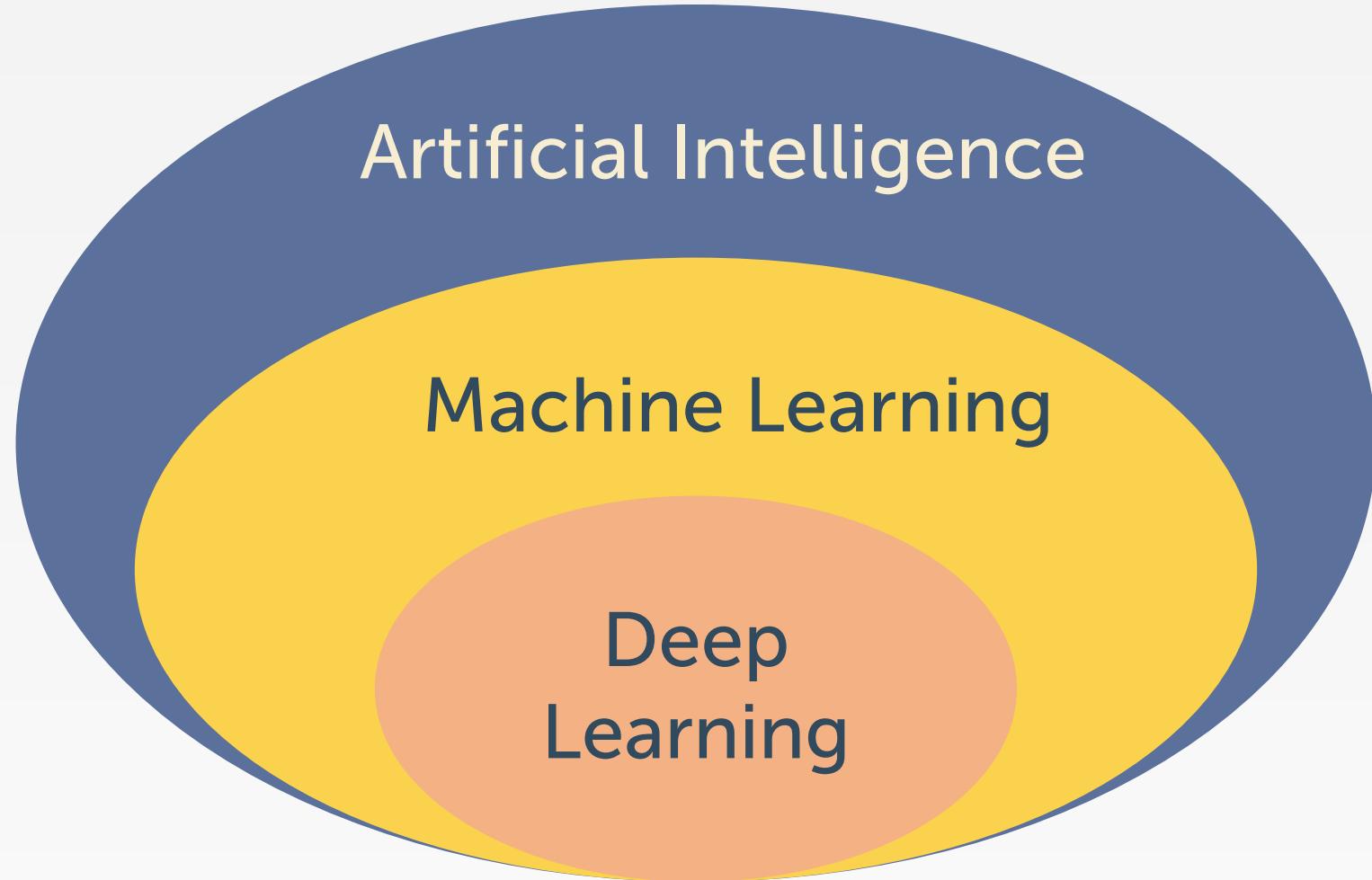
What is MACHINE LEARNING

Artificial Intelligence



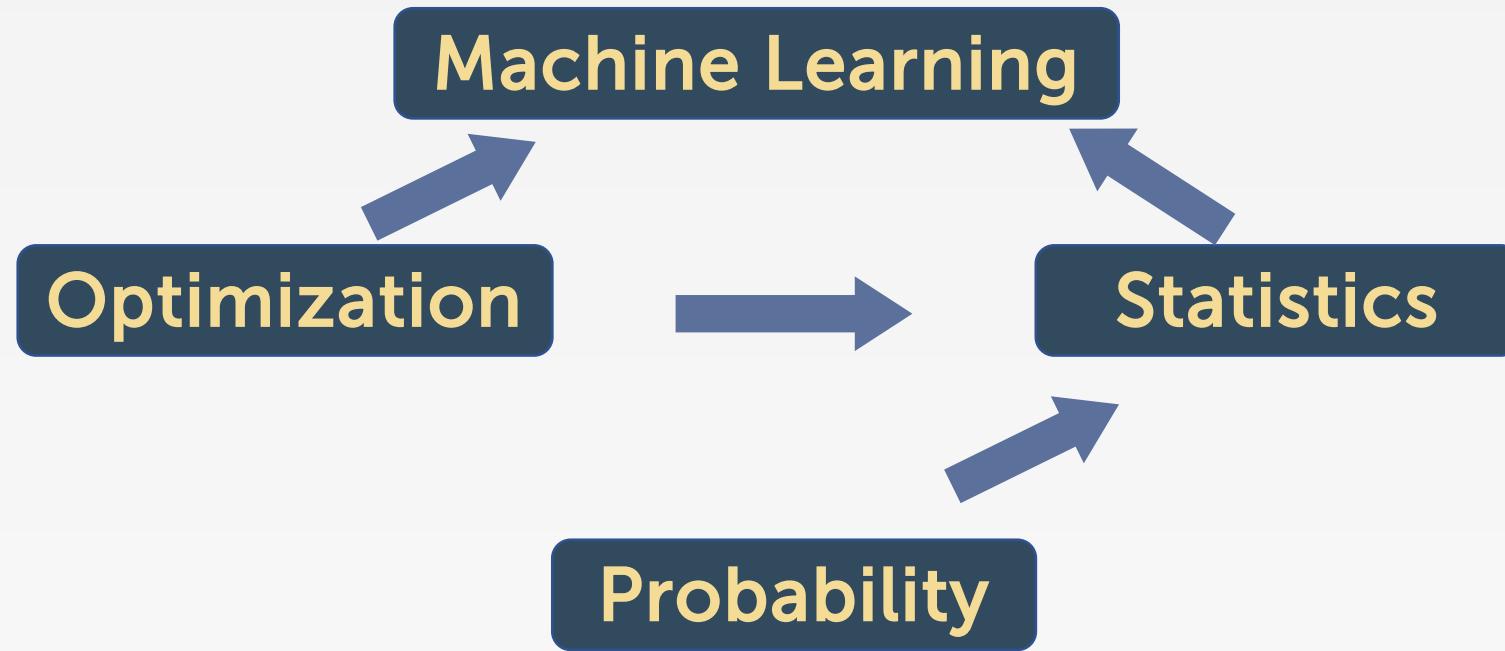
Artificial Intelligence

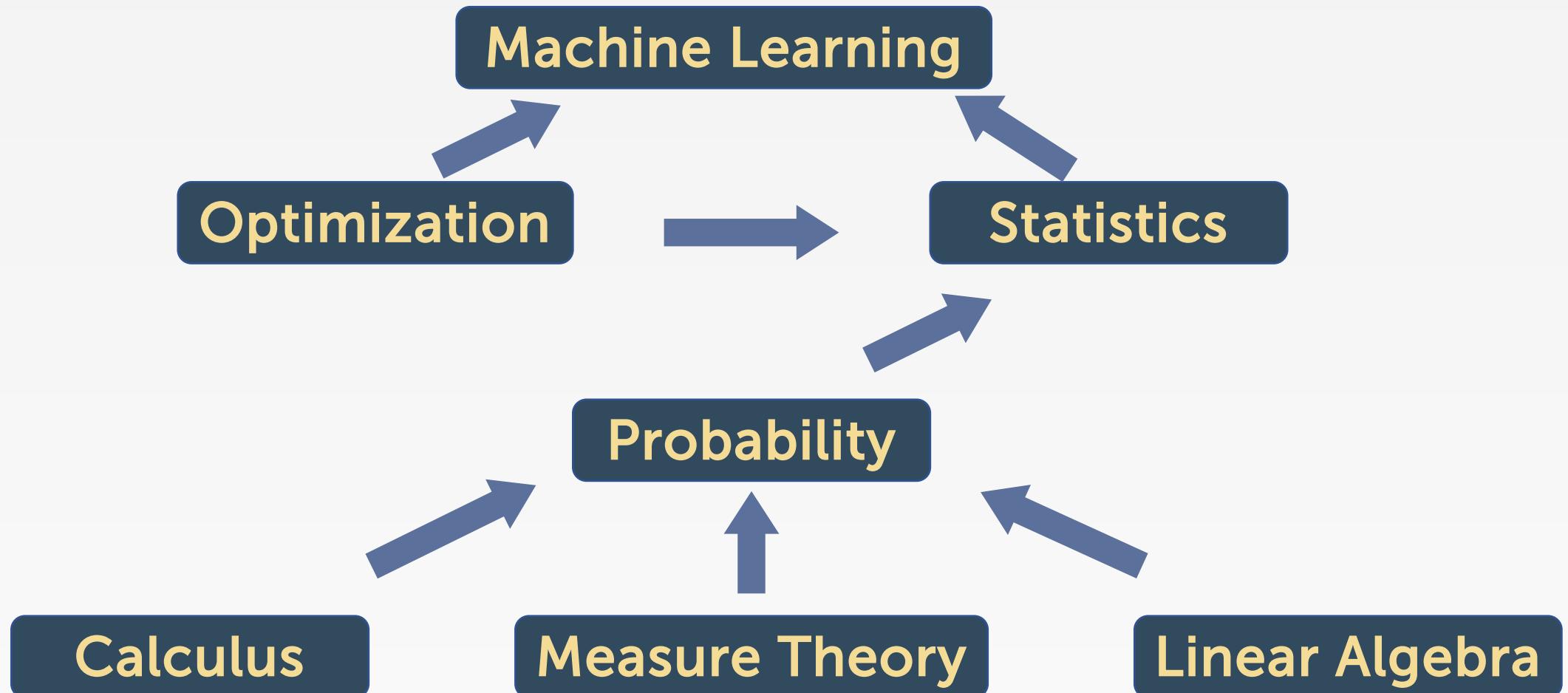
Machine Learning



Machine Learning







Computer Science

Machine Learning

Optimization

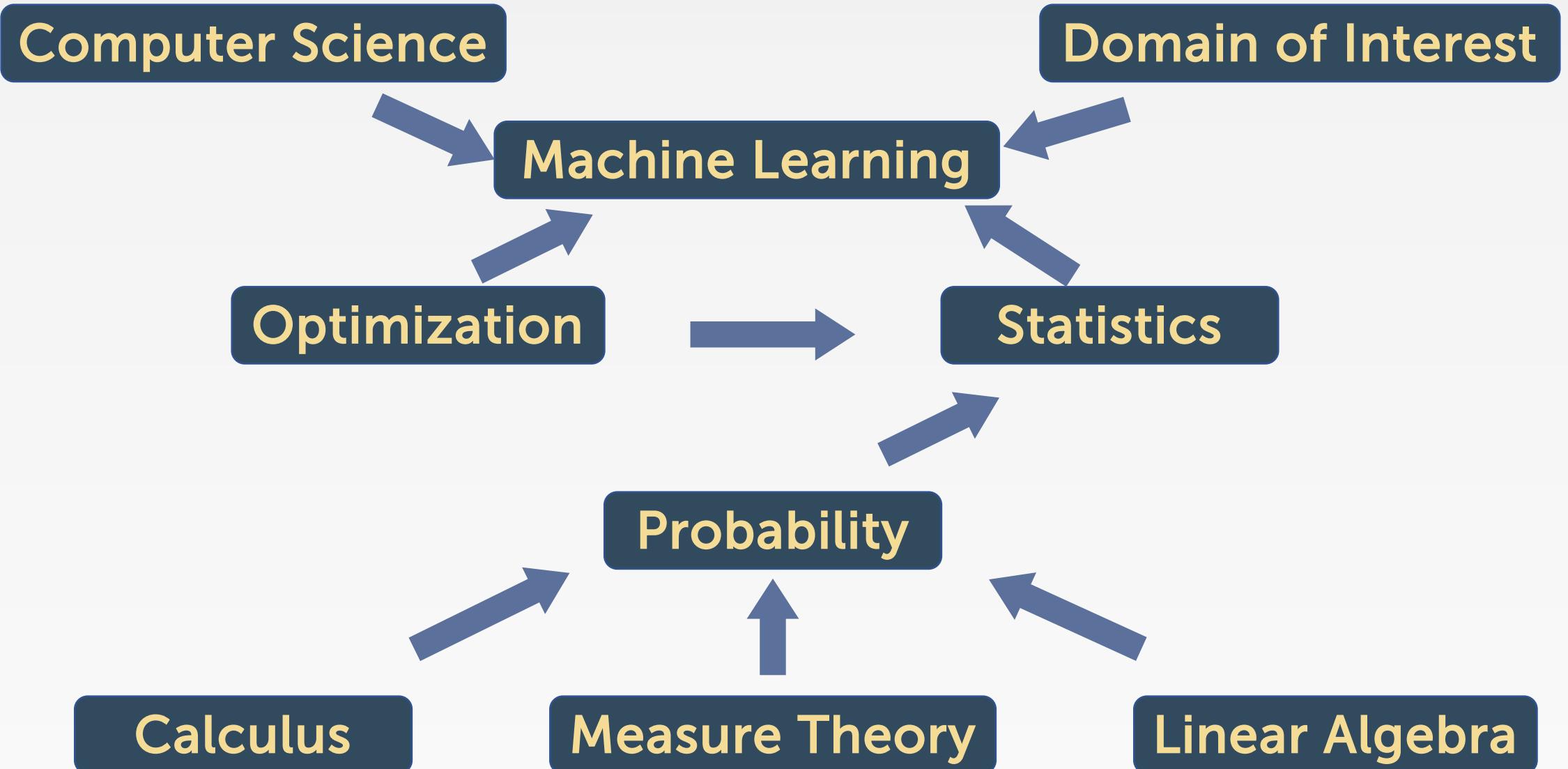
Statistics

Probability

Calculus

Measure Theory

Linear Algebra



ML is everywhere

Learning to drive an autonomous vehicle (robotics)



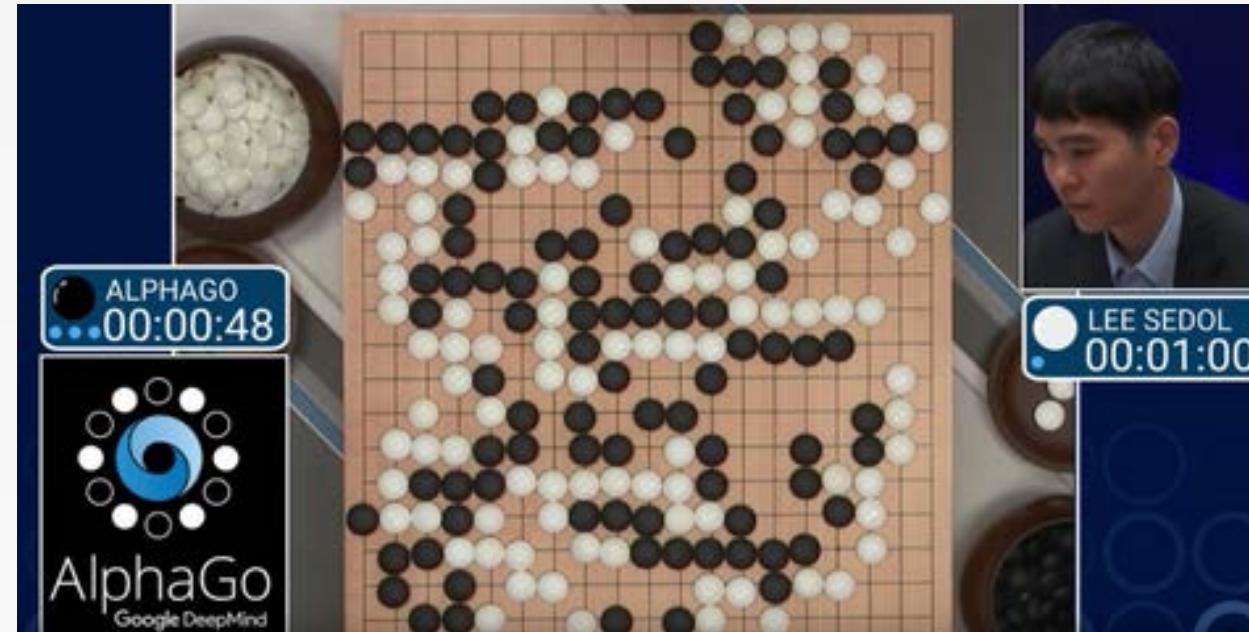
Tesla Self-Driving cars



Uber Self-Driving services

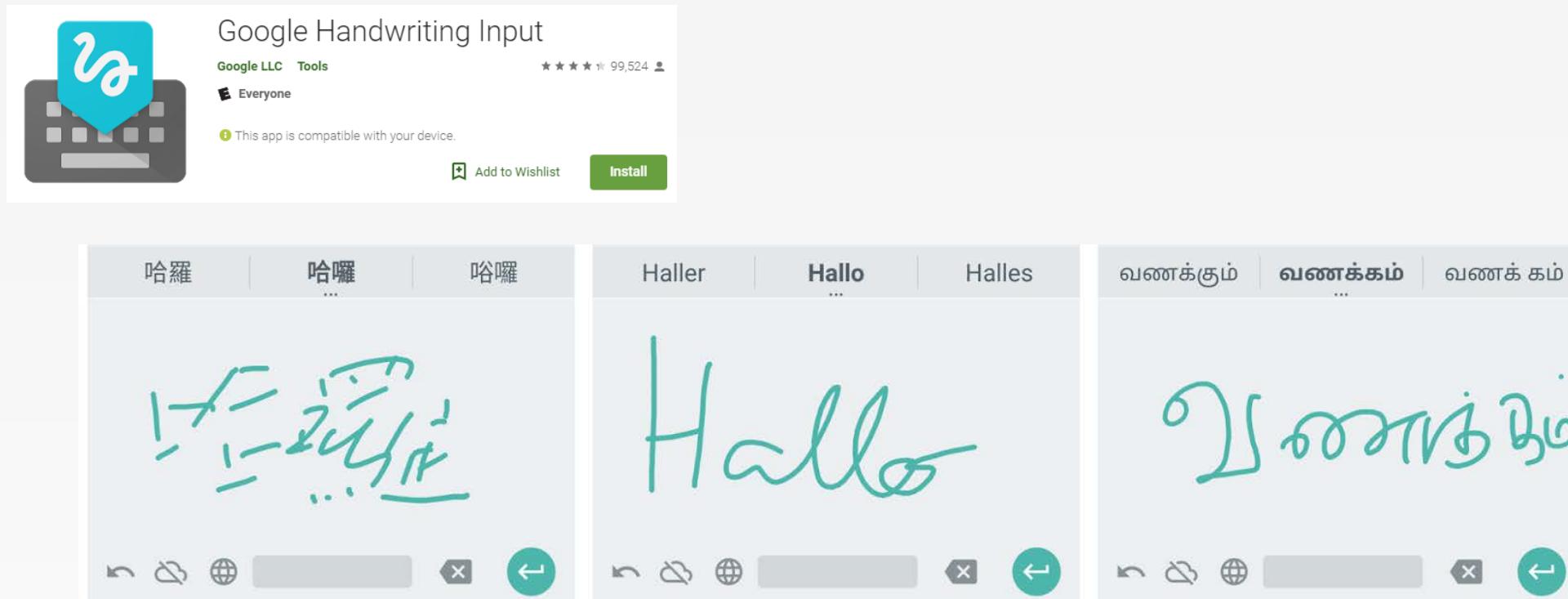
ML is everywhere

Learning to beat the masters at go games (Games/Reasoning)



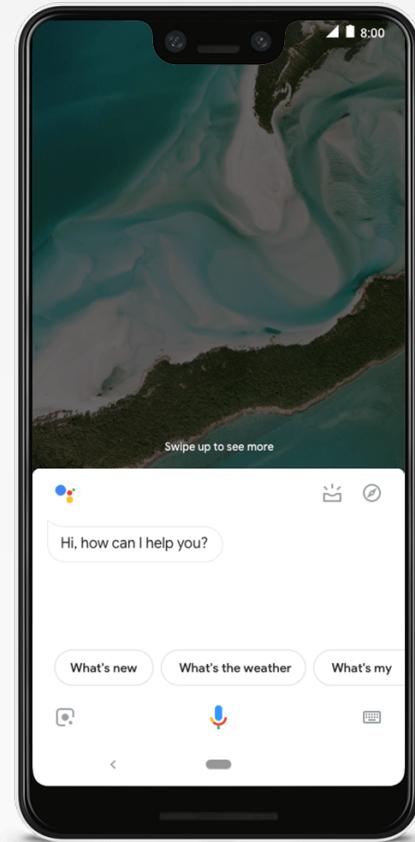
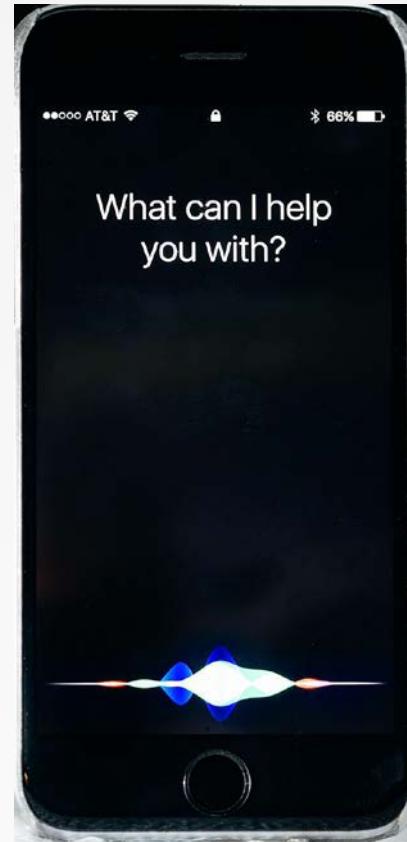
ML is everywhere

Learning to recognize handwriting (computer vision)



ML is everywhere

Learning to recognize spoken words (speech recognition)



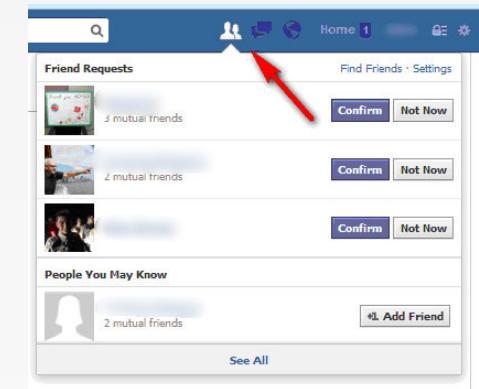
ML is everywhere

Learning to...

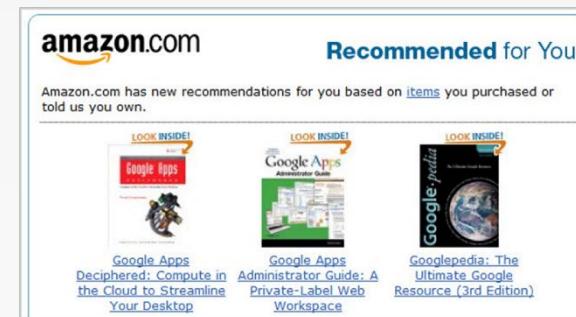
identify spam emails



suggest people you may know



recommend movies



...

Societal/Business Impacts of ML

- Search results are optimized for ad revenues

Baidu 百度

出国留学

百度一下

网页 资讯 视频 图片 知道 文库 贴吧 地图 采购 更多

百度为您找到相关结果约100,000,000个

▼ 搜索工具

出国留学网【专业的留学门户网站】

LIUXUE86

出国留学网从2005年创立至今,已经成为领先的中国教育门户网站。出国留学网下设留学、移民、签证、考试、范文、作文等频道,充分满足各类学生对不同类型资讯的需求。同时专注打造权威的工具箱及海内外...

出国留学网 百度快照

出国留学_新航道广州学校

随着人们生活质量的提高,很多学生有出国留学的想法,而美国一直都是最受中国留学生喜欢的国家,那么如何才能去美国留学呢?以下是去美国留学的两种方法:查看详情 30 21-08 香港留学|想要申请香港...

gz.xhd.cn/cglx/ 百度快照

第一留学网 - 出国留学中介机构_留学费用咨询 - 全球留学...

第一留学网专注出国留学服务,解答出国留学条件及留学途径留学费等知识,同时推荐正规合适的出国留学中介机构,并且为正在留学的用户提供高效便捷的全球留学院校排名信息查询。

www.hnrichfund.com/ 百度快照

北京留学_北京留学机构_专业出国留学中介-金吉列留学官网

金吉列北京留学服务中心,北京出国留学首选咨询服务结构,为您提供全方位北京留学,北京出国留学中介等留学权威资讯,想了解更多北京留学资讯,就到金吉列留学官网。

www.jjl.cn/ 百度快照

相关组织机构

展开 ▾

金吉列留学 中国500家最大私企	留学中介 留学一条龙服务	公费留学 要求学习结束后回国	中国留学服务中心 教育部直属事业单位
国际学校 供母语教育的学校	中外合作办学 高教未来发展大趋势	英国大学 拥有英国皇家特许状	多伦多大学 加拿大一所著名的大学
爱丁堡大学 英语国家中古老大学	墨尔本大学 世界顶尖的研究型大学	曼谷大学 泰国最大私立大学之一	平均学分绩点与质的计算



Societal/Business Impacts of ML

- Search results are optimized for ad revenues

Baidu 百度 出国留学 百度一下 搜索工具 网页 资讯 视频 图片 知道 文库 贴贴吧 地图 采购 更多

百度为您找到相关结果约100,000,000个

出国留学网【专业的留学门户网站】

LIUXUE 86 出国留学网从2005年创立至今,已经成为领先的中国教育门户网站。出国留学网下设留学、移民、签证、考试、范文、作文等频道,充分满足各类学生对不同类型资讯的需求。同时专注打造权威的工具箱及海内外...

出国留学_新航道广州学校

随着人们生活质量的提高,很多学生有出国留学的想法,而美国一直都是最受中国留学生喜欢的国家,那么如何才能去美国留学呢?以下是去美国留学的两种方法。查看详情 30 21-08 香港留学|想要申请香港...

gz.xhd.cn/cglx/ 百度快照

第一留学网 - 出国留学中介机构_留学费用咨询 - 全球留学...

第一留学网专注出国留学服务,解答出国留学条件及留学途径留学费等知识,同时推荐正规合适的出国留学中介机构,并且为正在留学的用户提供高效便捷的全球留学院校排名信息查询。

www.hnrichfund.com/ 百度快照

北京留学_北京留学机构_专业出国留学中介-金吉列留学官网

金吉列北京留学服务中心,北京出国留学首选咨询服务结构,为您提供全方位北京留学,北京出国留学中介等留学权威资讯,想了解更多北京留学资讯,就到金吉列留学官网。

www.jjl.cn/ 百度快照

相关组织机构 展开

金吉列留学 ZJ DIVERSIFICATION

魏则西事件五周年

时间: 2016年4月12日

2016年4月12日, 21岁的魏则西因滑膜肉瘤去世, 在其生前求医过程中, 通过百度搜索到武警北京总队第二医院, 被该医院宣传的“生物免疫疗法”、“斯坦福技术”所骗, 花费不菲却未收获任何效果, 贻误合理治疗时机。魏则西去世后, 莆田系医院虚假宣传、百度搜索竞价排名、部队医院对外承包混乱等问题引发社会强烈关注。

青年魏则西去世五周年

Societal/Business Impacts of ML

- Search results are optimized for ad revenues
- An autonomous vehicle is permitted to drive unassisted on the road

Societal/Business Impacts of ML

- Search results are optimized for ad revenues
- An autonomous vehicle is permitted to drive unassisted on the road



Societal/Business Impacts of ML

- Search results are optimized for ad revenues
- An autonomous vehicle is permitted to drive unassisted on the road



31岁企业家驾驶蔚来ES8车祸身亡，行驶数据浮出水面

2021年08月15日 19:22:36

来源：北京青年报

1595人参与 420评论



日前，一则蔚来ES8“自动驾驶”发生车祸死亡的事件引起网友关注。8月15日，疑似涉事车辆行驶数据曝光。

31岁创始人驾驶ES8车祸身亡

8月14日，认证名为“美一好”的个人公众号发布讣告称，2021年8月12日下午2时，上善若水投资管理公司创始人、意统天下餐饮管理公司创始人、美一好品牌管理公司创始人林文钦（昵称“萌剑客”），驾驶蔚来ES8汽车启用自动驾驶功能（NOP领航状态）后，在沈海高速涵江段发生交通事故，不幸逝世，终年31岁。

X

美一好 >

...

讣告 | 我们的“萌剑客”走了

2021年8月12日下午2时，上善若水投资管理公司创始人、意统天下餐饮管理公司创始人、美一好品牌管理公司创始人林文钦先生（昵称“萌剑客”），驾驶蔚来ES8汽车启用自动驾驶功能（NOP领航状态）后，在沈海高速涵江段发生交通事故，不幸逝世，终年31岁。

Societal/Business Impacts of ML

- Search results are optimized for ad revenues
- An autonomous vehicle is permitted to drive unassisted on the road
- A doctor is prompted by an intelligent system with a plausible diagnosis for her patient

Societal/Business Impacts of ML

- Search results are optimized for ad revenues
- An autonomous vehicle is permitted to drive unassisted on the road
- A doctor is prompted by an intelligent system with a plausible diagnosis for her patient



光

科普患教助手

优质医学科普信息

Guanghua School of Management

Societal/Business Impacts of ML

- Search results are optimized for ad revenues
- An autonomous vehicle is permitted to drive unassisted on the road
- A doctor is prompted by an intelligent system with a plausible diagnosis for her patient



**英国权威医学期刊 diss 医疗AI：在乳腺癌检测上取代放射科医生
是痴人说梦**

本文作者：我在思考中 2021-09-06 11:04

“ 导语：近日，《英国医学杂志》刊登了一篇研究工作。该团队工作对近年 AI 技术用于乳腺癌筛查的工作进行了检索，希望检验 AI 技术用于乳房 X 光摄像识别的准确度。



DEFINING a ML Problem

ML brings together different areas

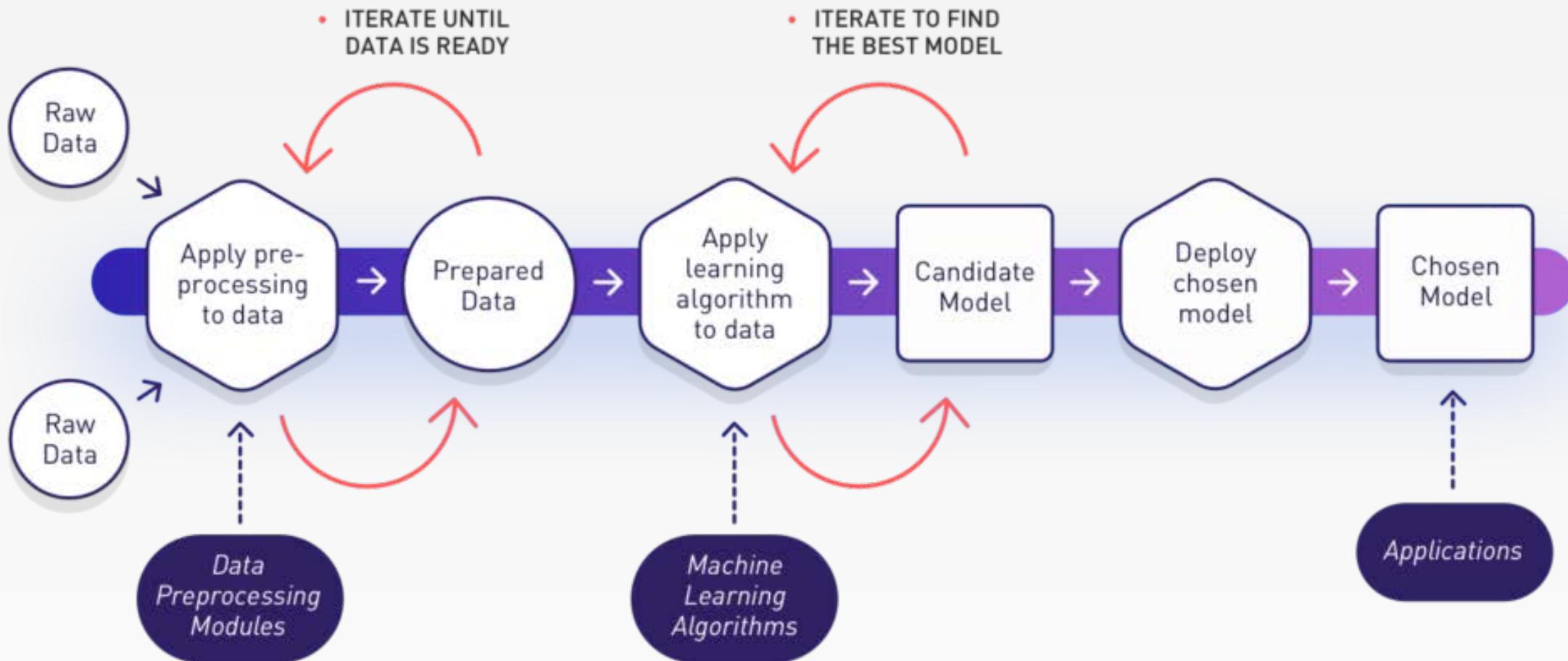
ML brings together different areas

- Statistical methods
 - Infer conclusions from data
 - Estimate reliability of predictions
- Computer science
 - Large-scale computing architectures
 - Algorithms for capturing, manipulating, indexing, combining, retrieving and performing predictions on data
 - Software pipelines that manage the complexity of multiple subtasks

ML brings together different areas

- Statistical methods
 - Infer conclusions from data
 - Estimate reliability of predictions
- Computer science
 - Large-scale computing architectures
 - Algorithms for capturing, manipulating, indexing, combining, retrieving and performing predictions on data
 - Software pipelines that manage the complexity of multiple subtasks
- Economics, biology, psychology
 - How can an individual or system efficiently improve their performance in a given environment?
 - What is learning and how can it be optimized?

Machine Learning Workflow



Define a Learning Problem

- ***Definition:*** A computer program learns if its performance at tasks in T , as measured by P , improves with experience E .

Define a Learning Problem

- ***Definition:*** A computer program learns if its performance at tasks in T , as measured by P , improves with experience E .
- **Three components**
 - Task, T
 - Performance measure, P
 - Training, E

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

Example I : Handwriting recognition

- T:
- P:
- E:

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

Example I : Handwriting recognition

- T: recognizing and classifying handwritten words with images
- P: percent of words correctly classified
- E: a database of handwritten words with given classifications

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

Example II : Self-driving

- T:
- P:
- E:

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

Example II : Self-driving

- T: driving on public four-lane highways using vision sensors
- P: average distance traveled before an error
- E: a sequence of images and steering commands recorded while observing a human driver

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

Exercise: Siri response to voice commands

- T: ?
- P: ?
- E: ?

Solution #1 Expert Systems

- Over 20 years ago, we had *rule-based systems*:
 1. Put a bunch of linguists in a room
 2. Have them think about the structure of their native language and write down the rules they devise

Solution #1 Expert Systems

- Over 20 years ago, we had *rule-based systems*:
 1. Put a bunch of linguists in a room
 2. Have them think about the structure of their native language and write down the rules they devise

Give me directions to Starbucks

If: "give me directions to X"

Then: directions(here, nearest(X))

Solution #1 Expert Systems

- Over 20 years ago, we had *rule-based systems*:
 1. Put a bunch of linguists in a room
 2. Have them think about the structure of their native language and write down the rules they devise

Give me directions to Starbucks

If: "give me directions to X"

Then: directions(here, nearest(X))

How do I get to Starbucks?

If: "how do I get to X"

Then: directions(here, nearest(X))

Solution #1 Expert Systems

- Over 20 years ago, we had *rule-based systems*:
 1. Put a bunch of linguists in a room
 2. Have them think about the structure of their native language and write down the rules they devise

Give me directions to Starbucks

If: "give me directions to X"

Then: directions(here, nearest(X))

How do I get to Starbucks?

If: "how do I get to X"

Then: directions(here, nearest(X))

Where is the nearest Starbucks?

If: "where is the nearest X"

Then: directions(here, nearest(X))

Solution #2 Annotate Data and Learn

- Experts:
 - Very good at answering questions about specific cases
 - Not very good at telling HOW they do it
- 1990s: So why not just have them tell you what they do on SPECIFIC CASES and then let MACHINE LEARNING tell you how to come to the same decisions that they did

Solution #2 Annotate Data and Learn

- Collect raw sentences $\{x^{(1)}, \dots, x^{(n)}\}$
- Experts annotate their meaning $\{y^{(1)}, \dots, y^{(n)}\}$

Solution #2 Annotate Data and Learn

- Collect raw sentences $\{x^{(1)}, \dots, x^{(n)}\}$
- Experts annotate their meaning $\{y^{(1)}, \dots, y^{(n)}\}$

$x^{(1)}$: Give me directions to Starbucks

$y^{(1)}$: directions(here,
nearest(Starbucks))

$x^{(2)}$: Send a text to John that I'll be late

$y^{(2)}$: txtmsg(John, I'll be late)

$x^{(3)}$: Show me the closest Starbucks

$y^{(3)}$: map(nearest(Starbucks))

$x^{(4)}$: Set an alarm for seven in the morning

$y^{(4)}$: setalarm(7:00AM)

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

Exercise: Siri response to voice commands

- T:
- P:
- E:

Define a Learning Problem

- Three components

- Task T
- Performance measure P
- Training E

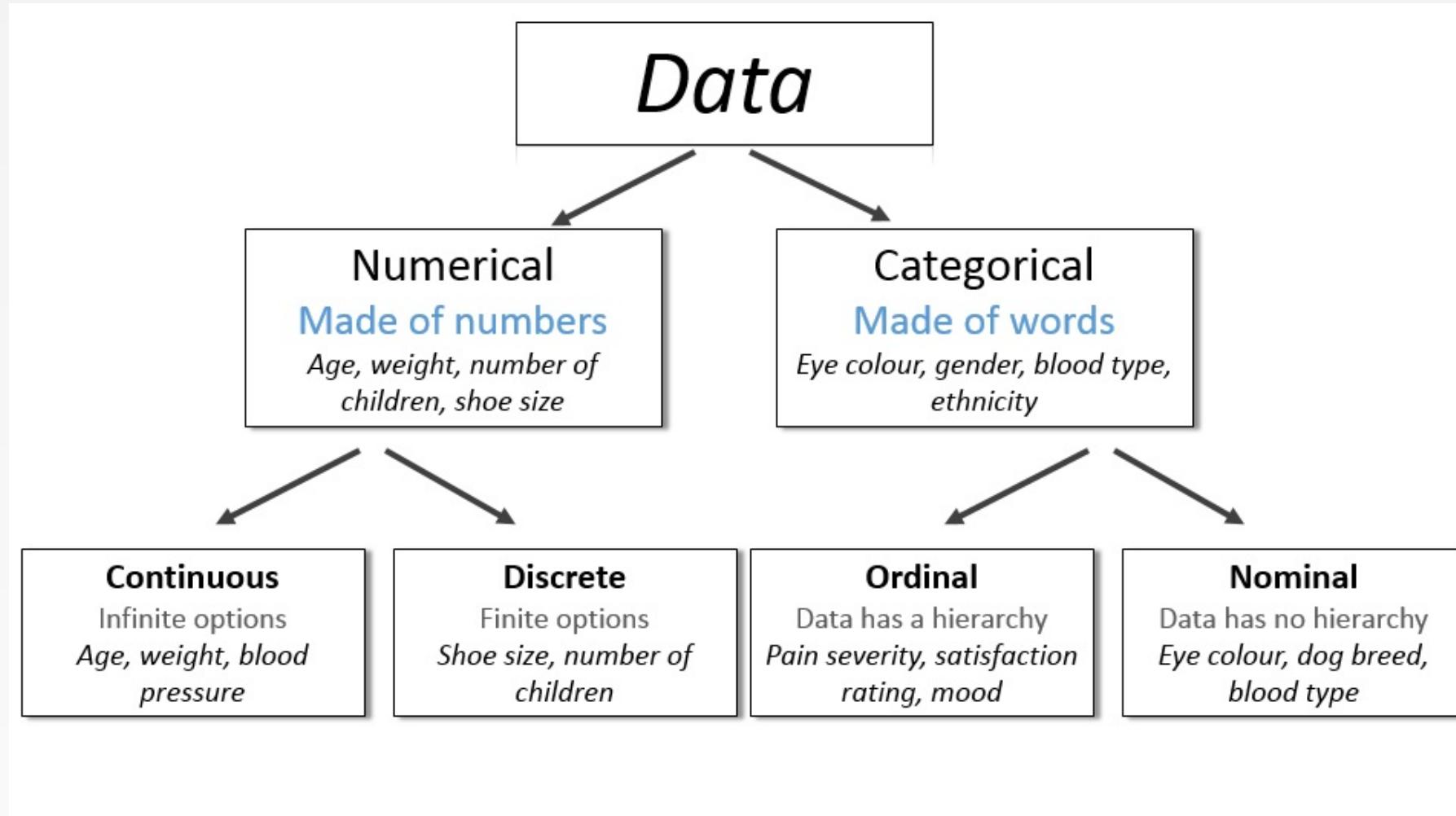
Exercise: Siri response to voice commands

- T: predicting action from speech
- P: percent of correct actions taken in user pilot study
- E: examples of (speech, action) pairs

Problem Formulation

- Formulate a problem in more than one ways:
- Loan applications:
 - Credit score (regression)
 - Default probability (density estimation)
 - Loan decision (classification)

Data Types



SYLLABUS

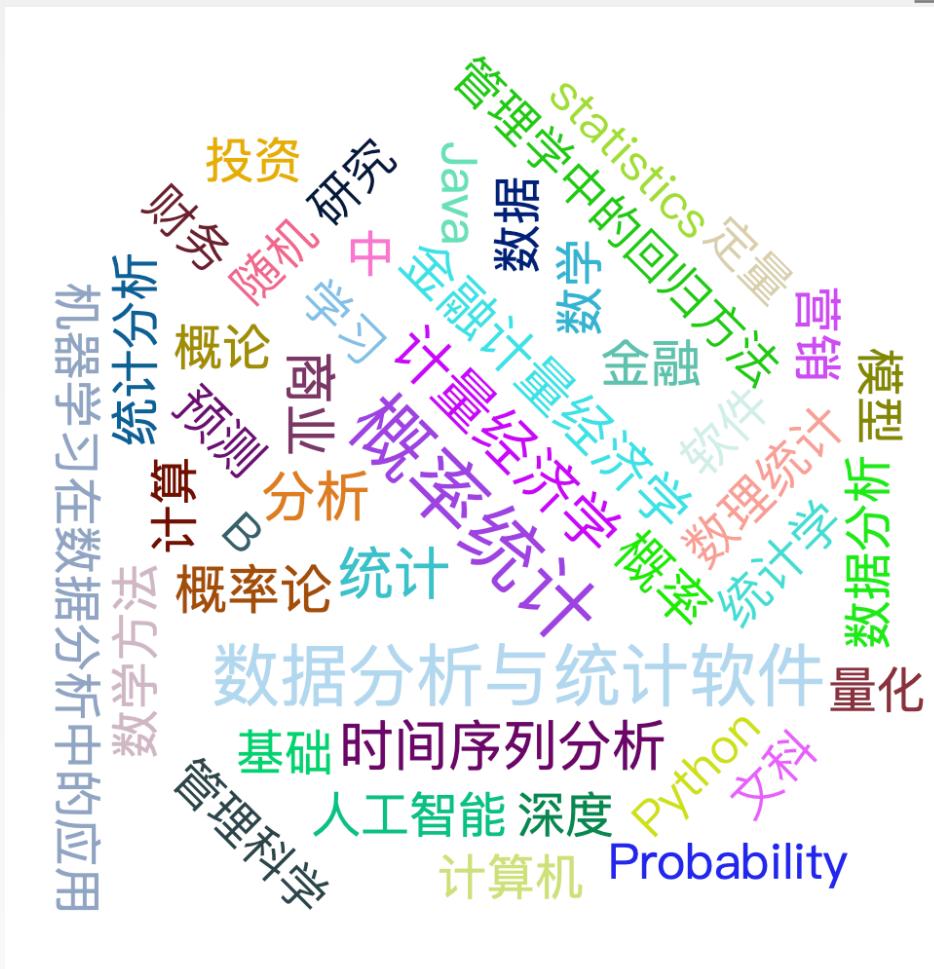
学业背景



专业



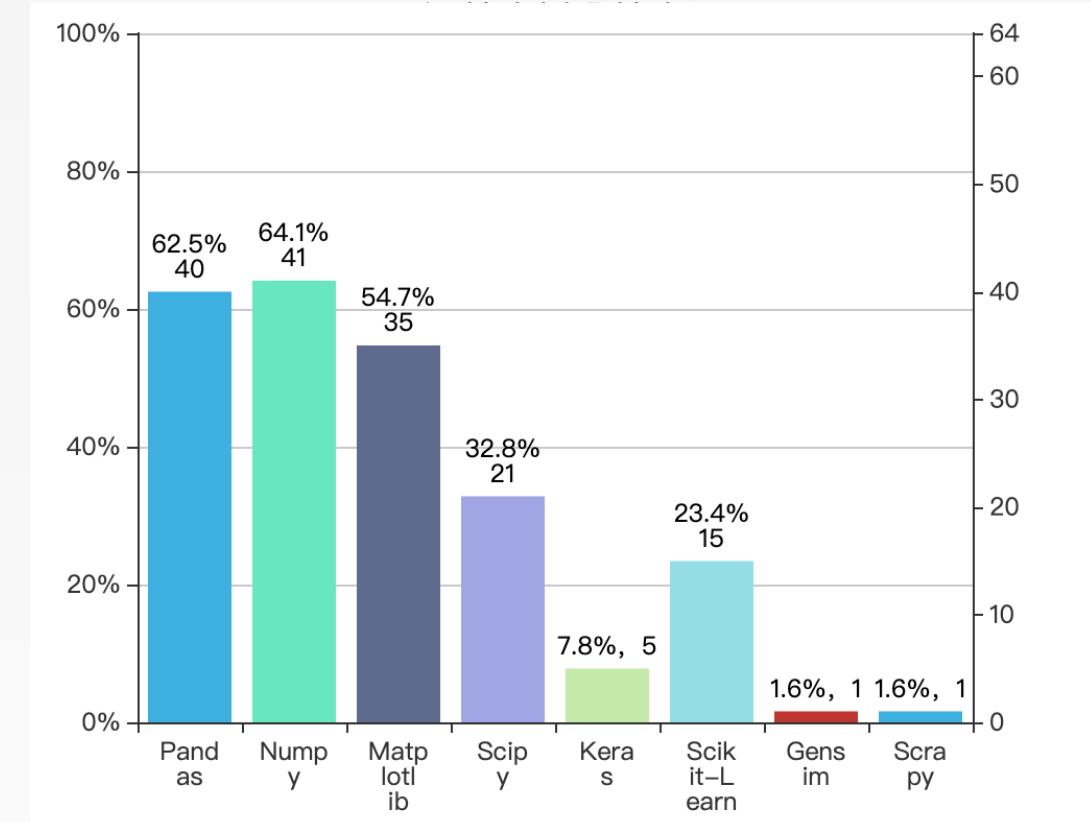
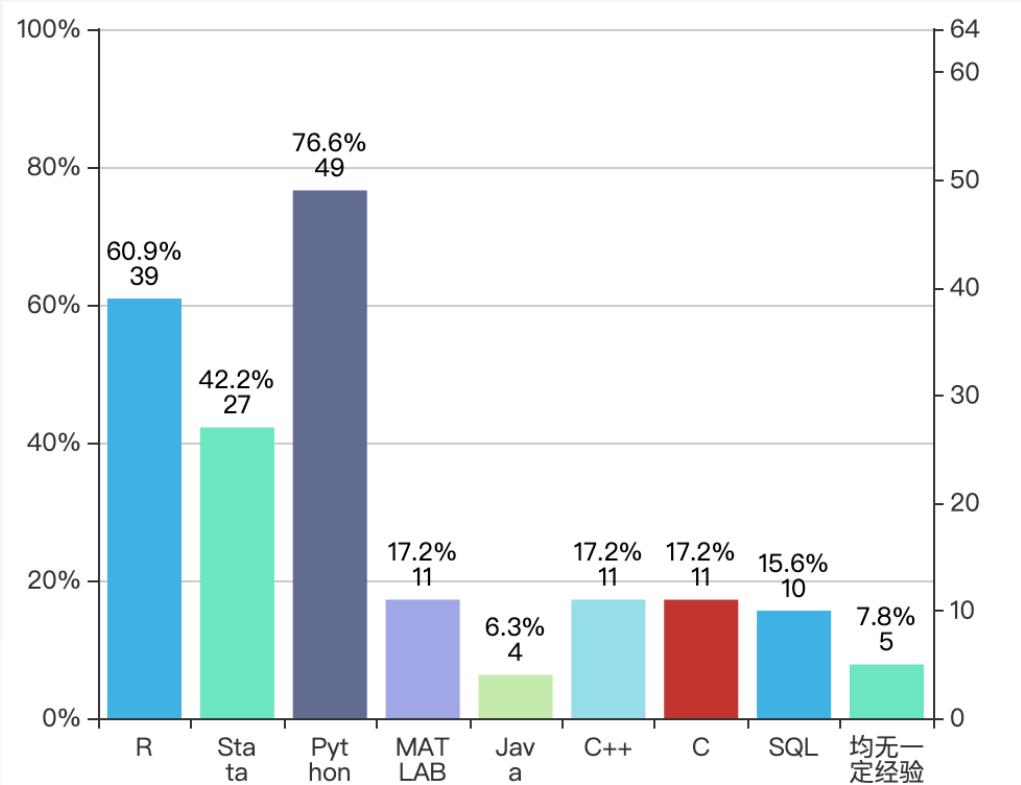
Minor



先修课程



编程背景



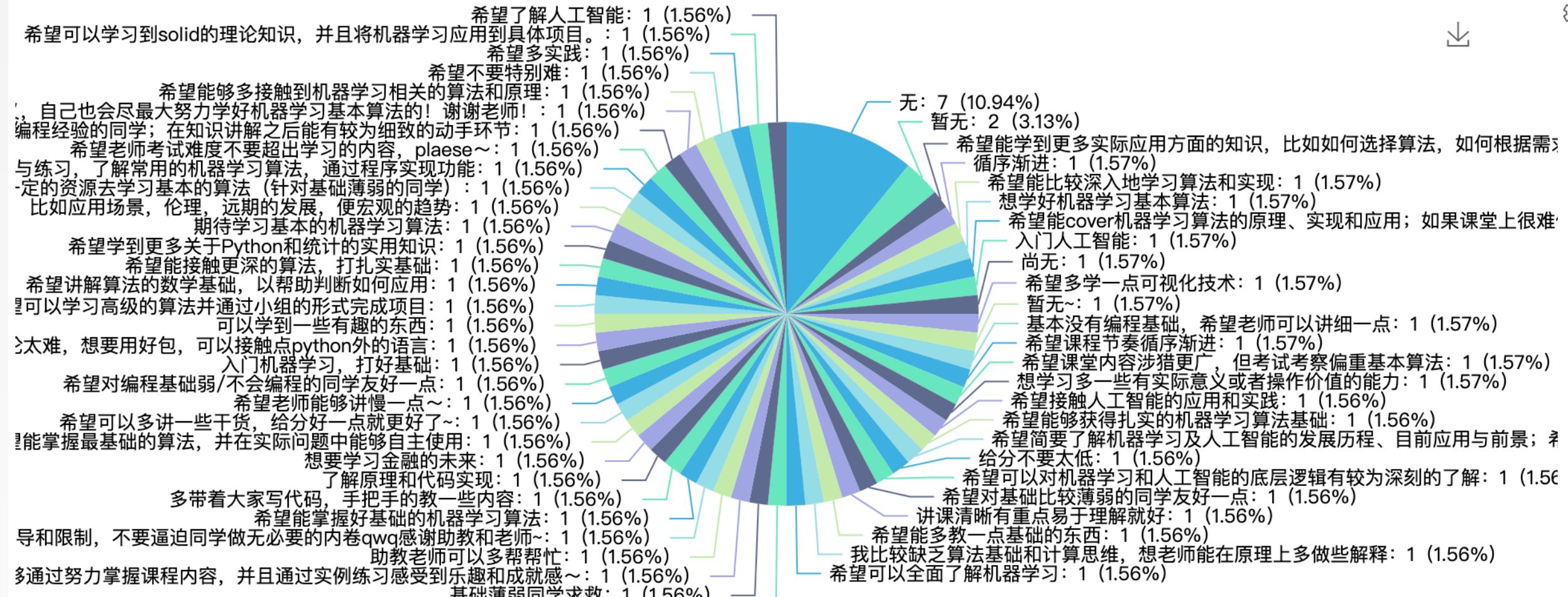
课程期待

42%

心有余而力不足，想学好机器学习基本算法

58%

期待，可以学习更加fancy的算法



Logistics

Logistics

- Homework:
 - 3 individual assignments
 - Late assignment is *NOT* accepted
 - Supervised Learning
 - Unsupervised Learning + Model Evaluation
 - Advanced Topics (e.g., Deep Learning, Reinforcement Learning)
 - **Re-grade:** You can appeal your grade with a one-page explanation. A re-grade may cause your grade to either go up or go down.

Logistics

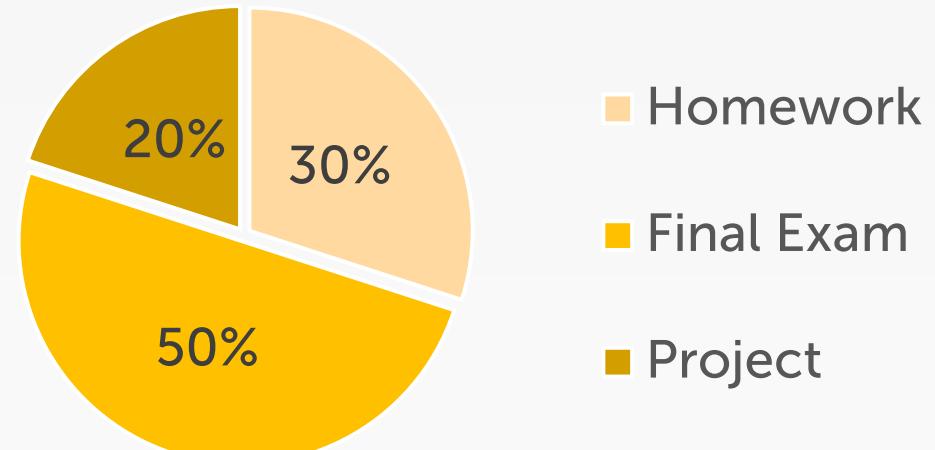
- Homework:
 - 3 individual assignments
 - Late assignment is *NOT* accepted
 - Supervised Learning
 - Unsupervised Learning + Model Evaluation
 - Advanced Topics (e.g., Deep Learning, Reinforcement Learning)
 - **Re-grade:** You can appeal your grade with a one-page explanation. A re-grade may cause your grade to either go up or go down.
- Group Project
 - At most 5 students per group (No Free-riding)
 - Proposal
 - In-class presentation
 - Final reports

Logistics

- Homework:
 - 3 individual assignments
 - Late assignment is *NOT* accepted
 - Supervised Learning
 - Unsupervised Learning + Model Evaluation
 - Advanced Topics (e.g., Deep Learning, Reinforcement Learning)
 - **Re-grade:** You can appeal your grade with a one-page explanation. A re-grade may cause your grade to either go up or go down.
- Group Project
 - At most 5 students per group (No Free-riding)
 - Proposal
 - In-class presentation
 - Final reports
- Final Exam
 - Week 12 (in-class)

Logistics

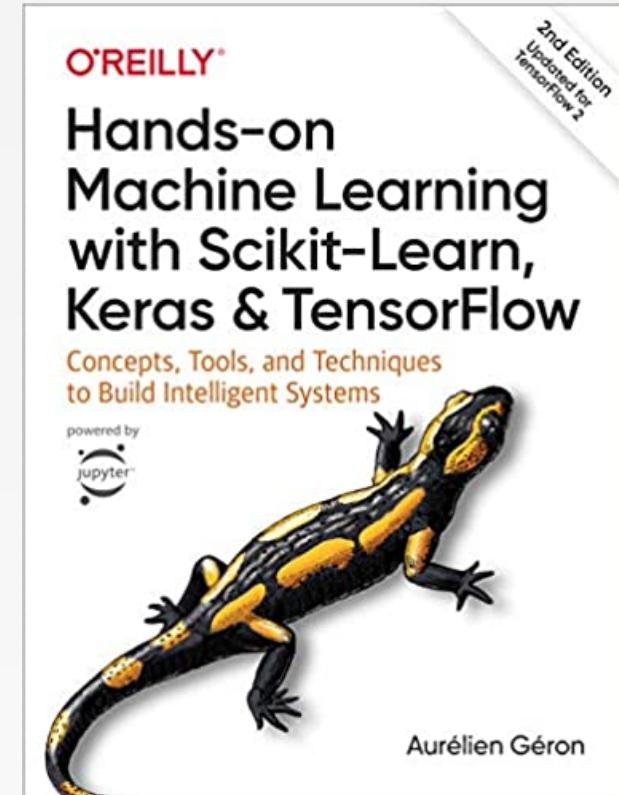
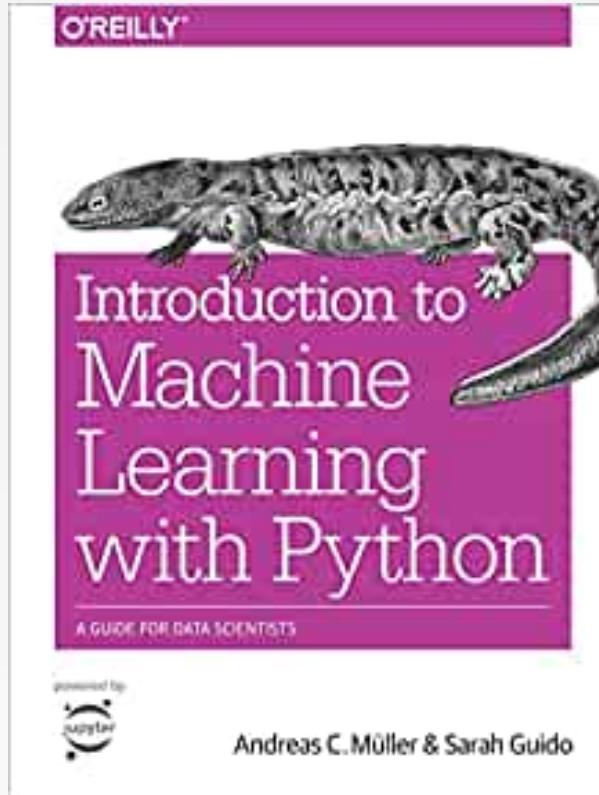
- Academic Honor Code: **No plagiarism!**
 - form study groups (with arbitrary number of people); discuss and work on homework problems in groups
 - write down the solutions independently
 - write down the names of people with whom you've discussed the homework
- Class participation



Topics

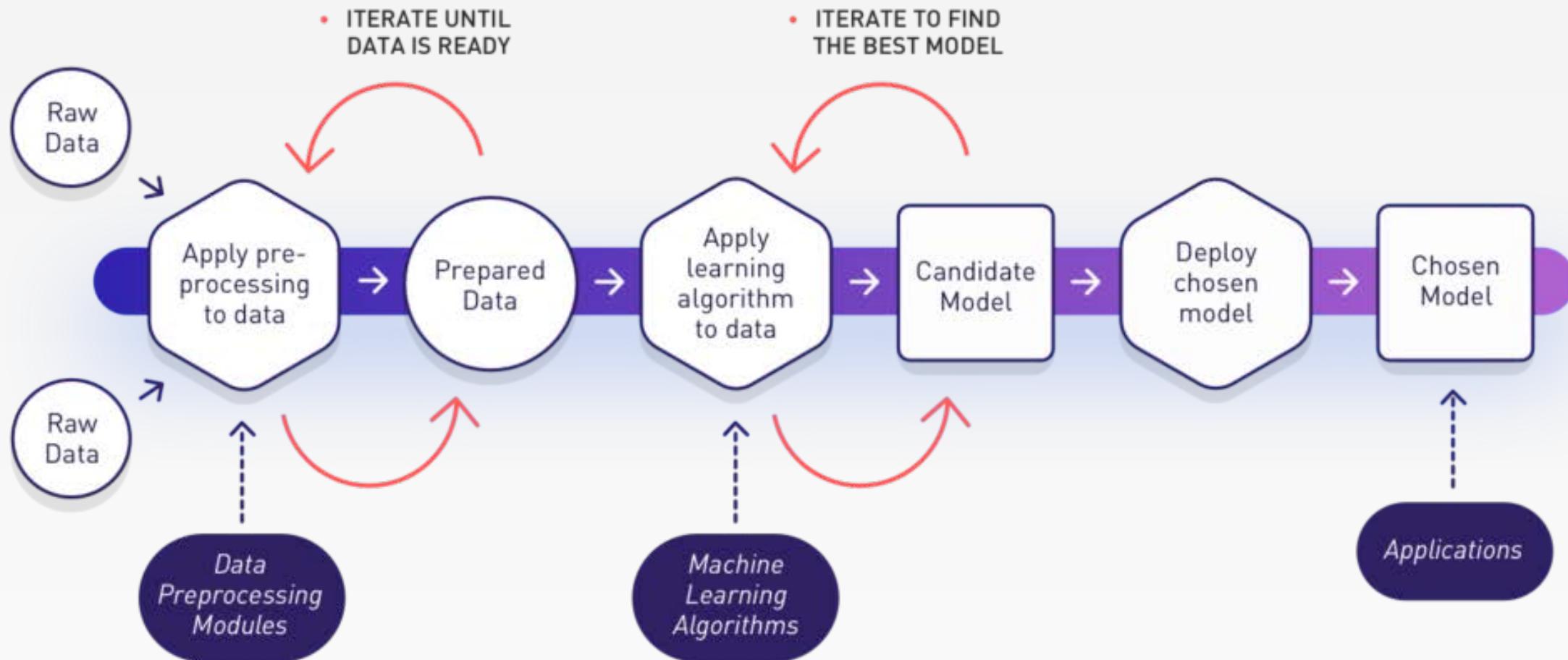
- Regression
- K-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- SVM
- Ensemble Models
- Cross Validation
- Overfitting / Underfitting
- Model Evaluation and Selection
- Unsupervised Learning
 - Clustering
 - PCA
- Reinforcement Learning
- Deep learning
 - Neural network
 - Backpropagation
 - CNNs, LSTM, etc.

Recommended Books



Overview of ML Models

Machine Learning Workflow



Applied ML

- Understand basic ML concepts and workflow (require basic statistics/probability background)
- Apply properly “black-box” ML components and features
- From theory to real-world practice

Types of ML Systems

Types of ML Systems

Criteria

Whether or not they are trained with human supervision

Types of ML Systems

Criteria

Whether or not they are trained with human supervision

Supervised Learning

Fraud detection
Prediction of stock markets

Types of ML Systems

Criteria

Whether or not they are trained with human supervision

Supervised Learning

Fraud detection
Prediction of stock markets

Unsupervised Learning

Customer segmentation
Recommendation

Types of ML Systems

Criteria

Whether or not they are trained with human supervision

Supervised Learning

Fraud detection
Prediction of stock markets

Unsupervised Learning

Customer segmentation
Recommendation

Semi-supervised Learning

Photo-hosting service
Speech analysis
Web-content classification

Types of ML Systems

Criteria

Whether or not they are trained with human supervision

Supervised Learning

Fraud detection
Prediction of stock markets

Unsupervised Learning

Customer segmentation
Recommendation

Semi-supervised Learning

Photo-hosting service
Speech analysis
Web-content classification

Reinforcement Learning

Robotics
Go games
Self-driving cars

Types of ML Systems

Criteria

Whether or not they are trained with human supervision



Supervised Learning

Fraud detection
Prediction of stock markets



Unsupervised Learning

Customer segmentation
Recommendation

Semi-supervised Learning

Photo-hosting service
Speech analysis
Web-content classification

Reinforcement Learning

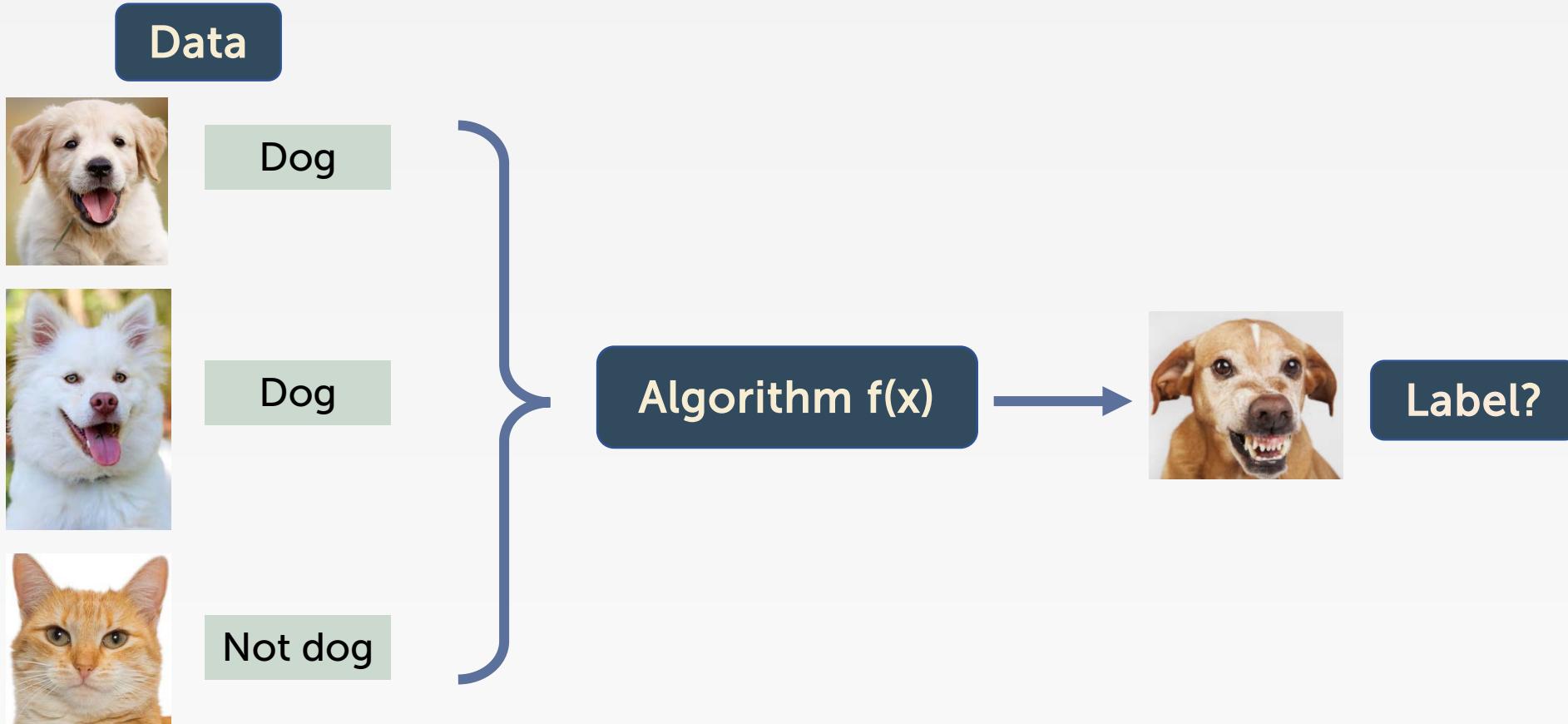
Robotics
Go games
Self-driving cars

Supervised Learning

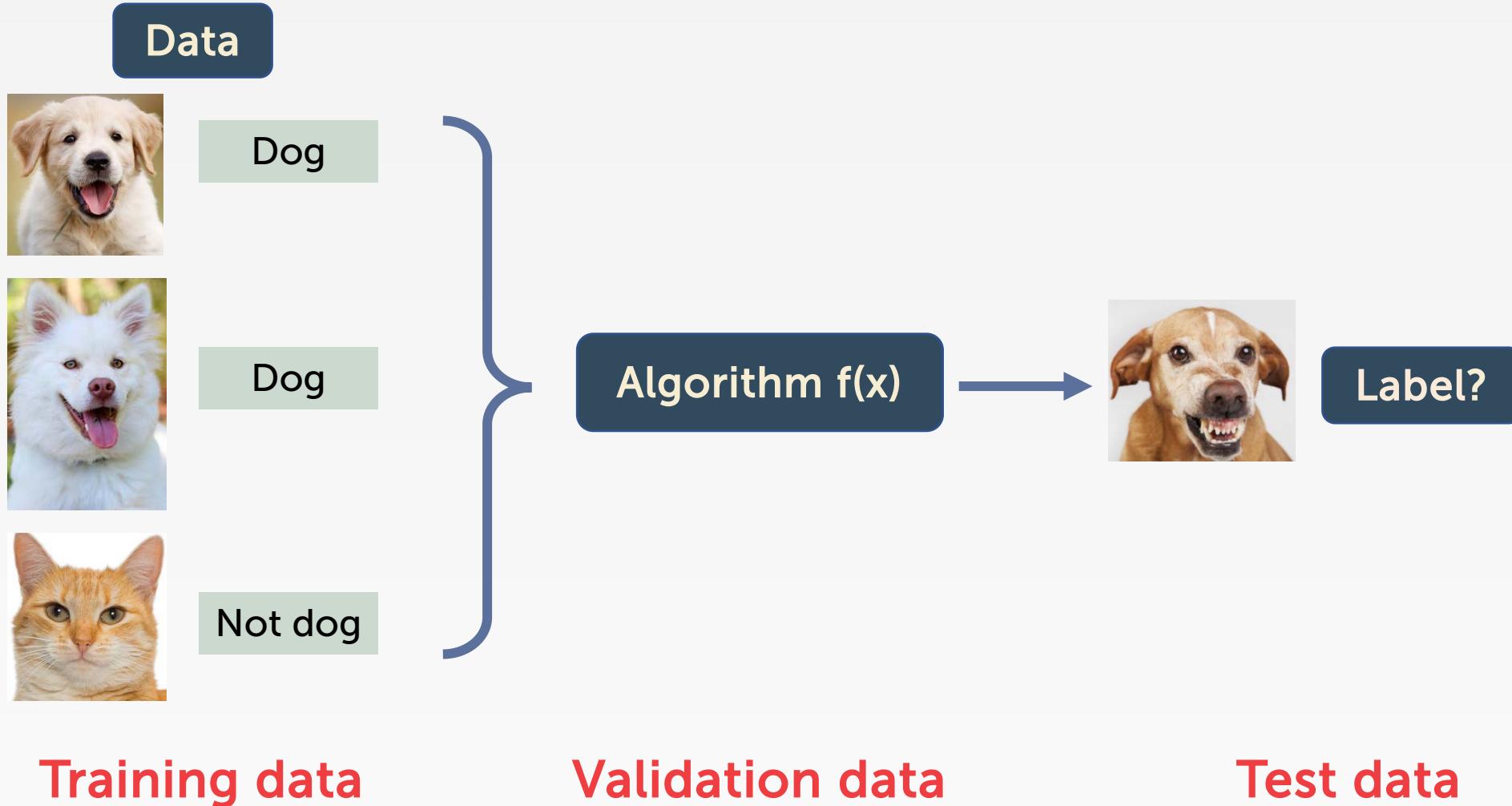
Supervised Learning

- Regressions
- KNN
- Decision Trees
- SVM
- Naïve Bayes
- ...

Supervised Learning



Supervised Learning

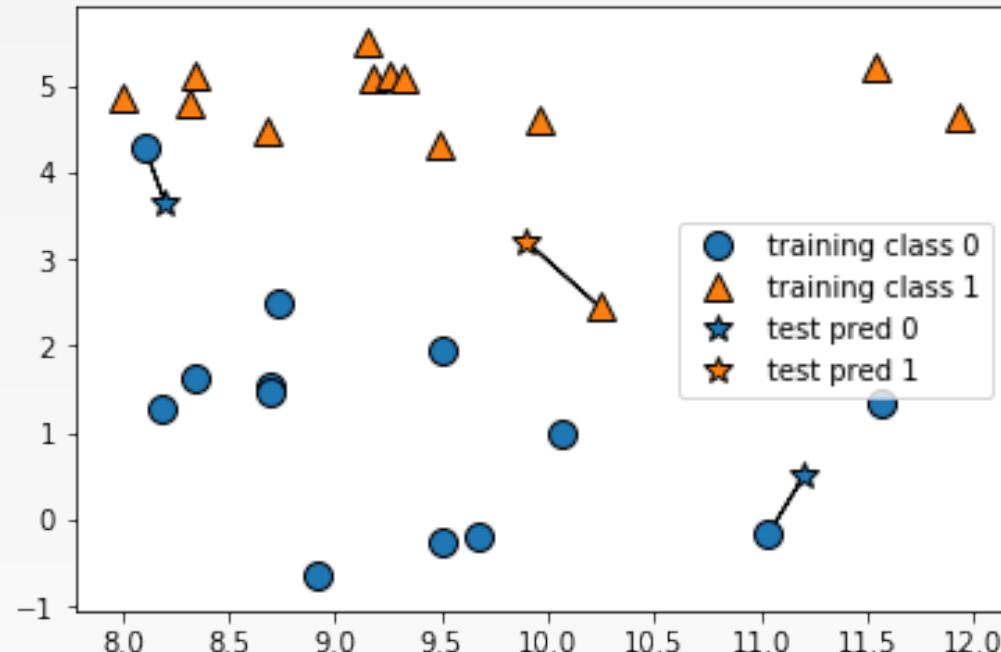


Data Division

- **Training dataset**: the sample of data used to fit the model
- **Validation Dataset**: the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
- **Test Dataset**: a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier

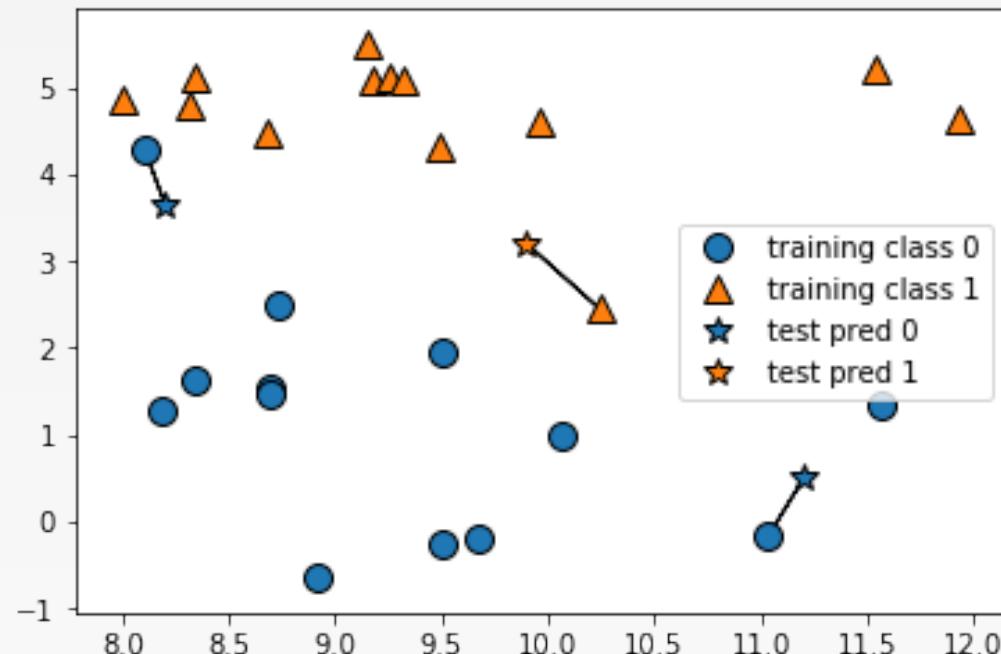
K-Nearest Neighbors

An Illustrative Example

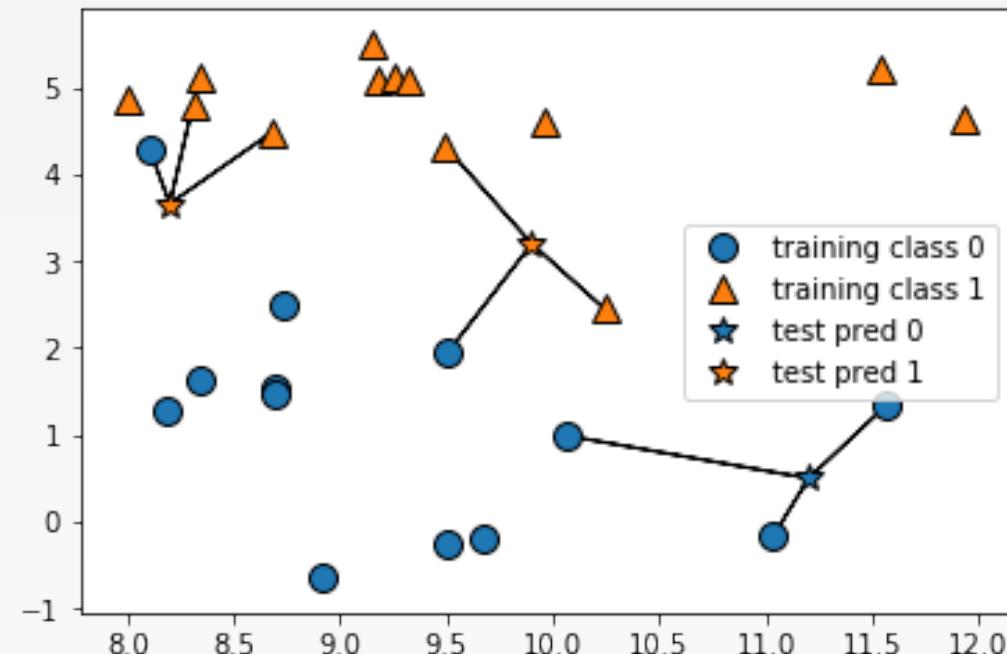


$K = 1$

An Illustrative Example



$K = 1$



$K = 3$

KNN: Classification Algorithm

KNN: Classification Algorithm

- Training: Store all the examples (X_{train}, Y_{train})

KNN: Classification Algorithm

- Training: Store all the examples (X_{train}, Y_{train})
- Prediction: X_{new}
 - Let X_1, \dots, X_k be the k most similar examples to X_{new}
 - Use certain method (e.g., majority vote) to determine Y_{new} based on (Y_1, \dots, Y_k)

KNN: Classification Algorithm

- Training: Store all the examples (X_{train}, Y_{train})
- Prediction: X_{new}
 - Let X_1, \dots, X_k be the k most similar examples to X_{new}
 - Use certain method (e.g., majority vote) to determine Y_{new} based on (Y_1, \dots, Y_k)

Keys

1. A distance metric

$$\text{Euclidean distance } d(x_j, x_k) = \sqrt{\sum_i (x_{j,i} - x_{k,i})^2}$$
$$\text{Manhattan distance } d(x_j, x_k) = \sum_i |x_{j,i} - x_{k,i}|$$

KNN: Classification Algorithm

- Training: Store all the examples (X_{train}, Y_{train})
- Prediction: X_{new}
 - Let X_1, \dots, X_k be the k most similar examples to X_{new}
 - Use certain method (e.g., majority vote) to determine Y_{new} based on (Y_1, \dots, Y_k)

Keys

1. A distance metric

Euclidean distance $d(x_j, x_k) = \sqrt{\sum_i (x_{j,i} - x_{k,i})^2}$
 Manhattan distance $d(x_j, x_k) = \sum_i |x_{j,i} - x_{k,i}|$

2. Value of "K"

Cross validation: larger k? smaller k?

KNN: Classification Algorithm

- Training: Store all the examples (X_{train}, Y_{train})
- Prediction: X_{new}
 - Let X_1, \dots, X_k be the k most similar examples to X_{new}
 - Use certain method (e.g., majority vote) to determine Y_{new} based on (Y_1, \dots, Y_k)

Keys

1. A distance metric

Euclidean distance $d(x_j, x_k) = \sqrt{\sum_i (x_{j,i} - x_{k,i})^2}$
 Manhattan distance $d(x_j, x_k) = \sum_i |x_{j,i} - x_{k,i}|$

2. Value of "K"

Cross validation: larger k? smaller k?

3. Aggregation of the classes of neighbor points

Majority vote

KNN: Pros and Cons

KNN: Pros and Cons

- Advantages:

- Very simple and intuitive
- The cost of the learning process is zero
- No assumption about the characteristics/distributions
- Works on both classification and regression tasks

KNN: Pros and Cons

- Advantages:
 - Very simple and intuitive
 - The cost of the learning process is zero
 - No assumption about the characteristics/distributions
 - Works on both classification and regression tasks
- Drawbacks:
 - Computationally expensive when the dataset is very large
 - Need to calculate the compare distance from new example to all other examples
 - Sensitive to outliers

Python...

Python Quick Checks

- I can read python codes...
- I can write python functions...
- Errors and debugging...

Coding Tips

- Comments?
- Printed messages?
- Functions?

iPython

- Command: jupyter notebook
- Install: Anaconda
- Programming in the browser
- Codes, instructions, and outputs are displayed “in-line”
- Useful for writing codes that tells a story
- Used by scientists and researchers
- ...

Python Packages

- Scikit-learn: Python Machine Learning Library

```
from sklearn.tree import DecisionTreeClassifier
```

- Numpy: Scientific Computing Library

- Typically, data input to scikit-learn will be in the form of a Numpy array

```
import numpy as np
```

- Pandas: Data Manipulation

```
import pandas as pd
```

- Matplotlib: Plotting Library

```
import matplotlib.pyplot as plt
```

- Others: mlearn; graphviz; seaborn

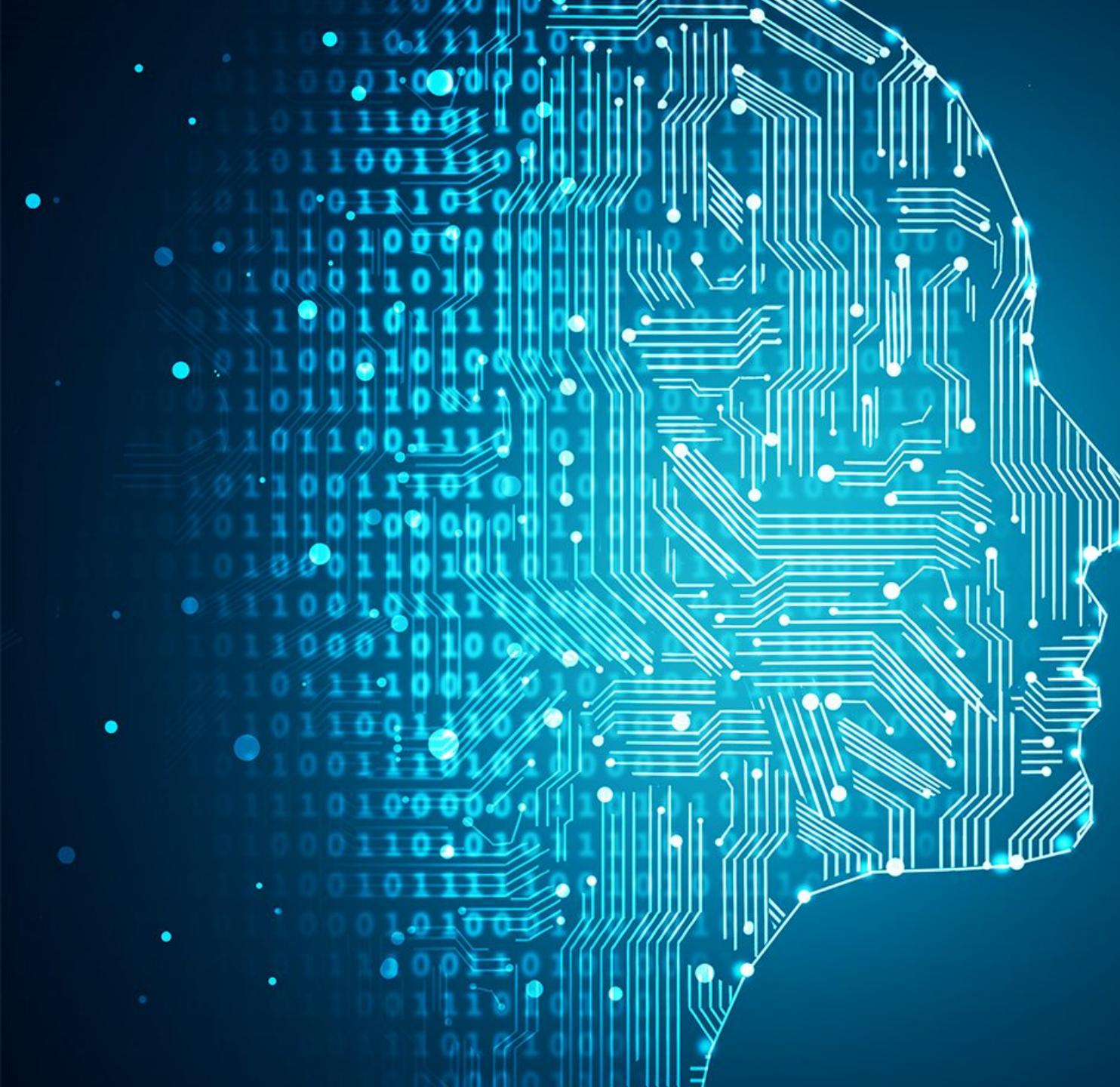


Python Practice

Questions?

For Next Week...

- Python Review
- Regressions
- Bring your laptop with Python (and packages) installed



机器学习与人工智能

Machine Learning and Artificial Intelligence

Lecture 2 Regressions

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

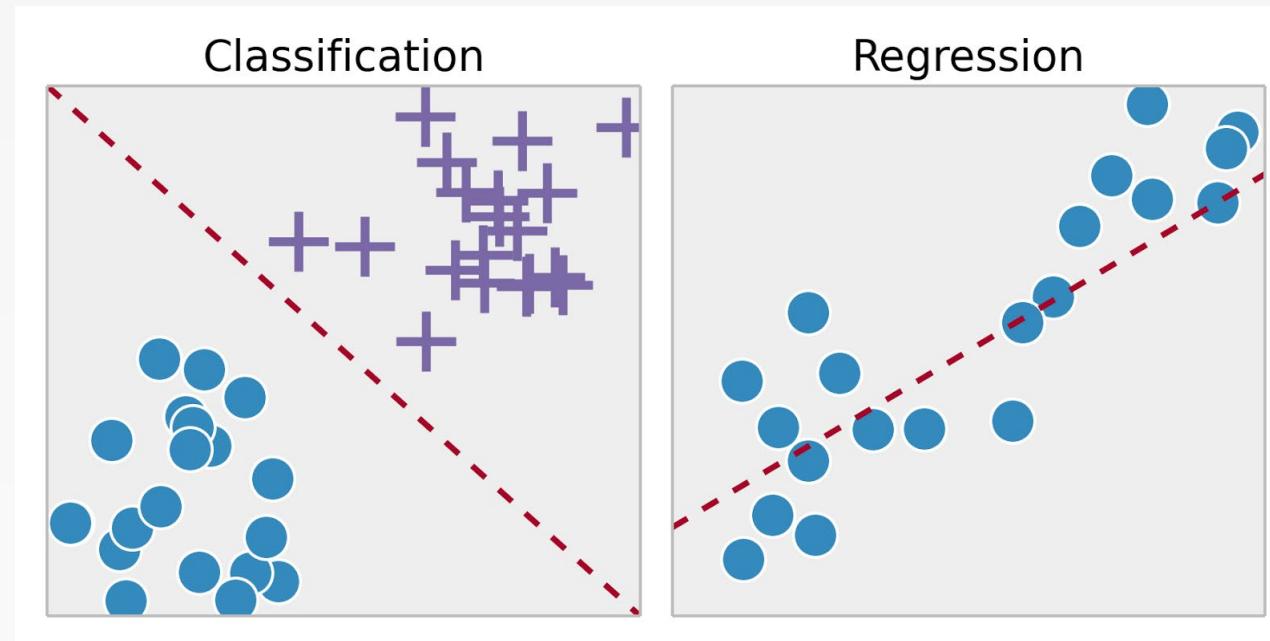
2021 Fall

Group Project

- Sign up the team by **Sept 26** (email/msg TAs)
- Proposal: Due on Oct 10, 2021, 11:59pm (one submission per team)
 - Team members (at most 5 students)
 - Project goals (with a real-world business question and available datasets)
 - Models to address the questions (at least one supervised and one unsupervised learning models)
 - Advanced model applications and deeper analyses are encouraged
- In-Class Presentation (Nov 25)
- Final reports (Due Dec 9, 2021)

Classification vs. Regression

- **Classification**: the goal is to predict a *class label*, which is a choice from a predefined list of possibilities
- **Regression**: the goal is to predict a continuous number, or a *floating-point number (real number)* in programming (math) terms

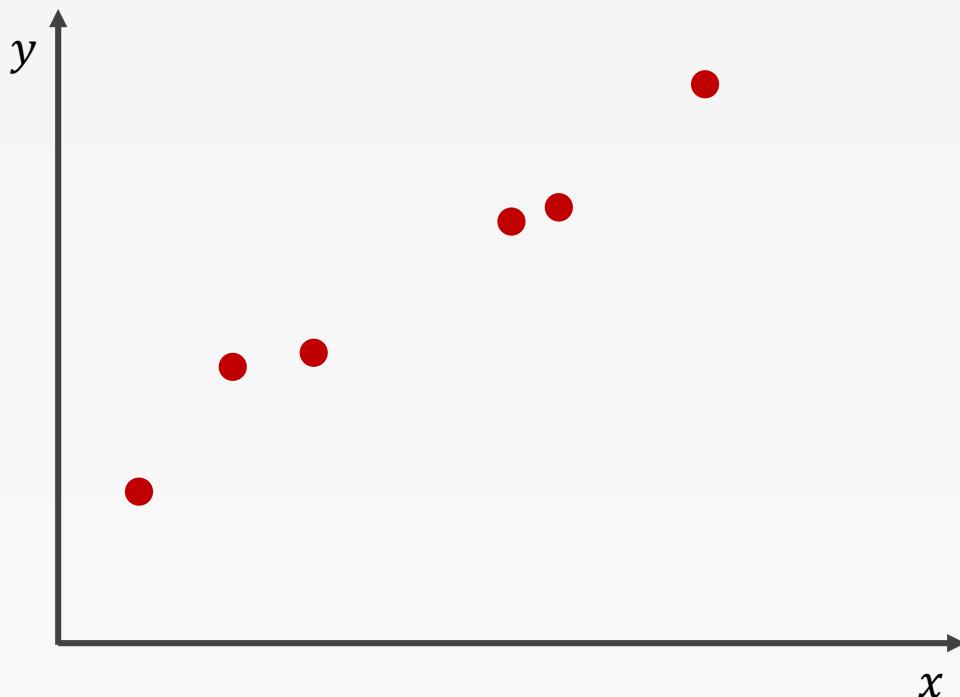


Regression

- Given the value of an input X , the output Y belongs to the set of real value R.
- Evaluation: predict output accurately
- Examples:
 - Predict housing price
 - Forecast precipitation

Regression

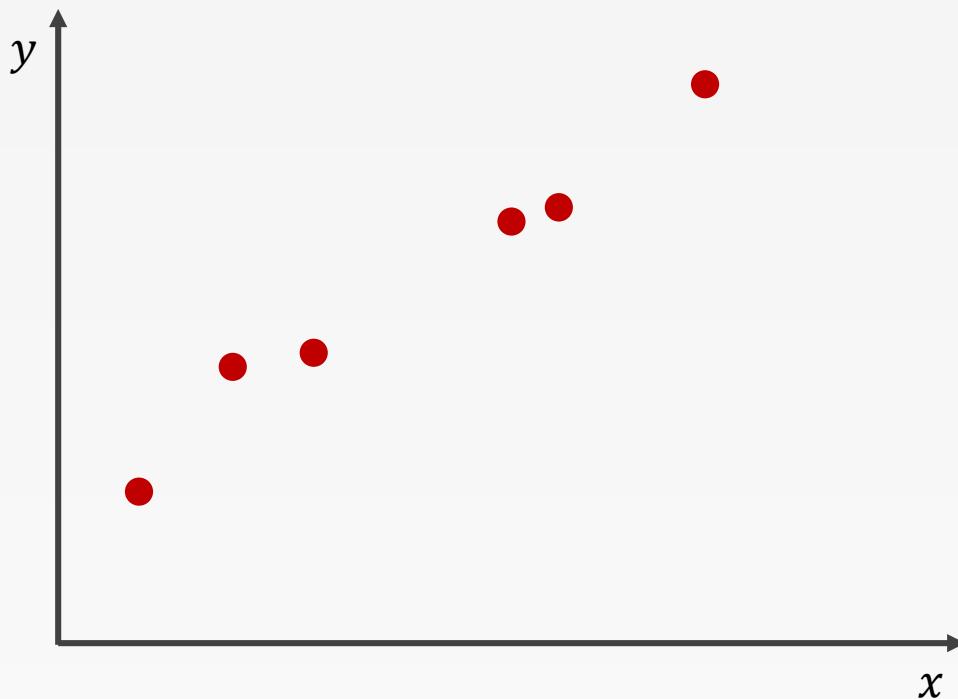
Example: Dataset with only one feature x and one scalar output y



Q: What is the function that best fits these points?

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



$k = 1$

- *Train:* store all (x, y) pairs
- *Predict:* pick the nearest x in the training data and return its y

$k = 2$ Nearest Neighbor Distance Weighted Regression

- *Train:* store all (x, y) pairs
- *Predict:* pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

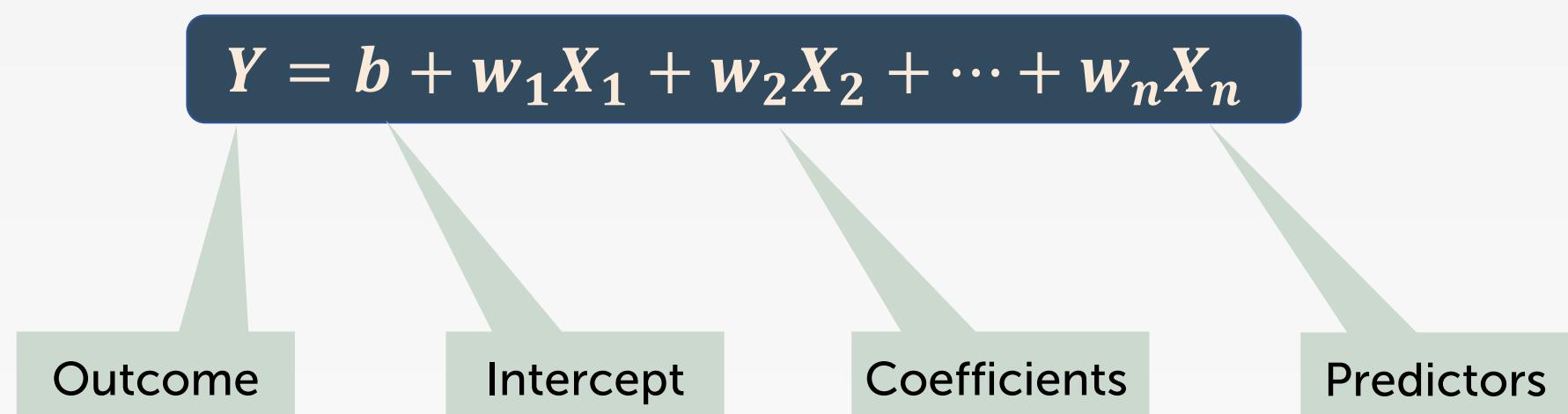
Linear Regression

Agenda

- Definition of Regression
- Linear functions
- Residuals
- Estimations (Optimization)
- Regularization

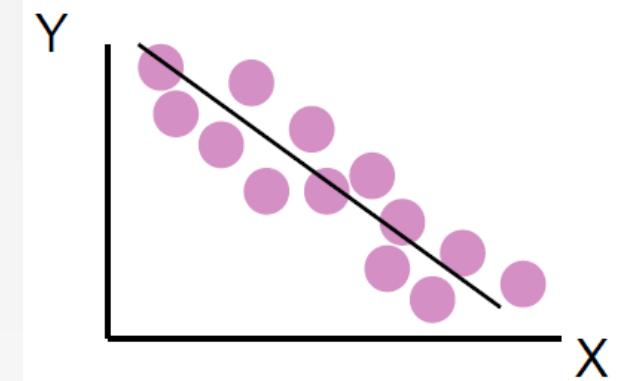
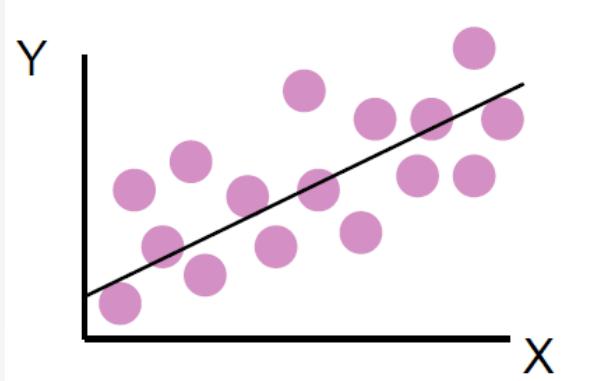
Linear Regression

- Linear relationship: outcome (dependent) variable is a linear combination of predictor (independent) variables.



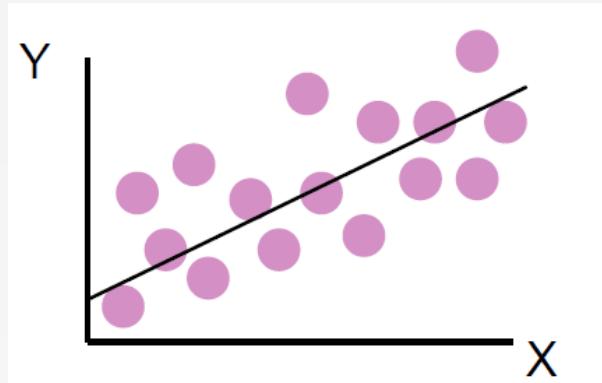
Linear Model?

$$Y_i = b + wX_i$$

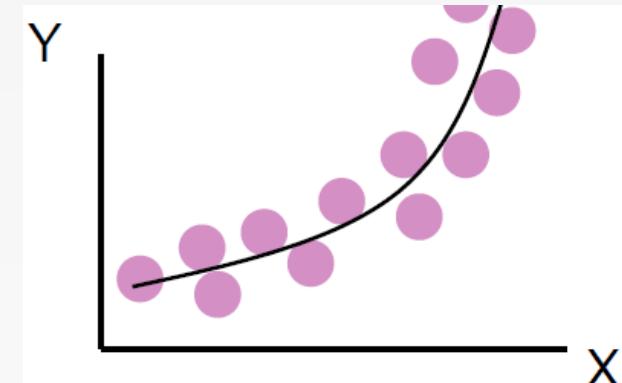
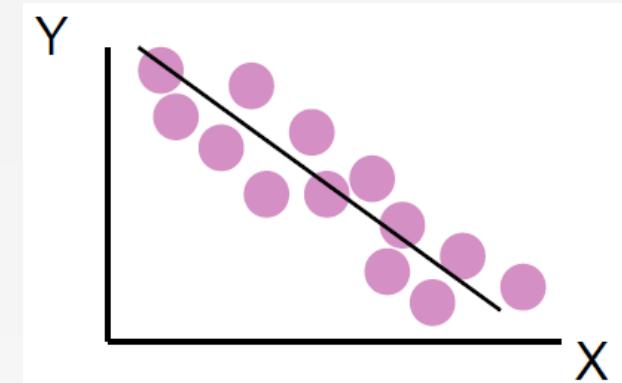
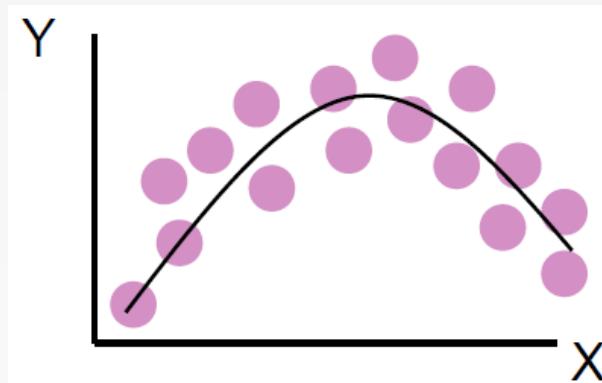


Linear Model?

$$Y_i = b + wX_i$$

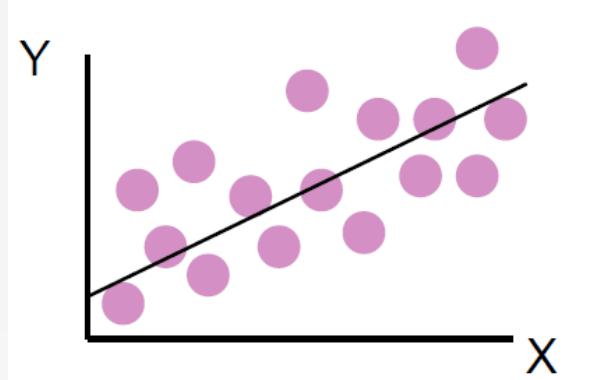


$$Y_i = b + w_1X_i + w_2X_i^2$$

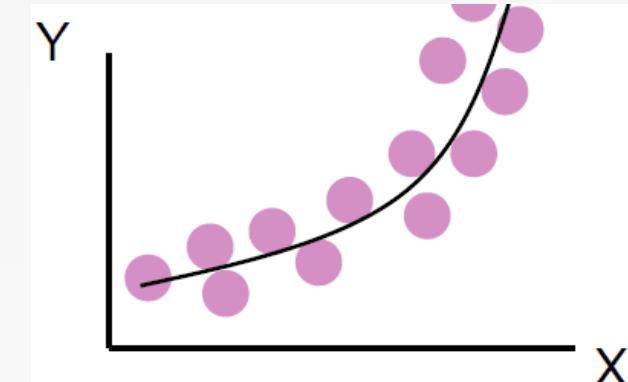
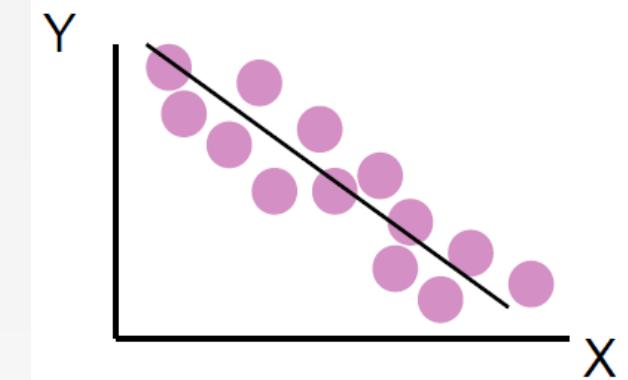
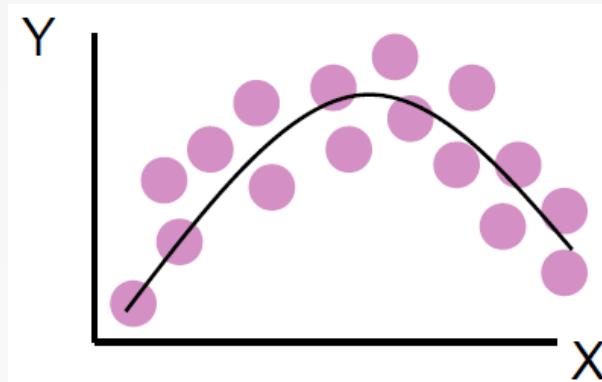


Linear Model?

$$Y_i = b + wX_i$$



$$Y_i = b + w_1X_i + w_2X_i^2$$



A linear model means linear in parameters (not X)!

Linear Regression?

- $y = \sum_i \omega_i f_i(x)$
- $y = \sum_i \omega_i x^i$
- $y = \sum_i e^{w_i x}$
- $y = \sum_i w_i \sin(i^2 x^7)$

Results Interpretation

Level-Level

$$y = b + wx + \varepsilon$$

One unit increase of x
→ w unit increase of y

Results Interpretation

Level-Level

$$y = b + wx + \varepsilon$$

One unit increase of x
→ w unit increase of y

Log-Level

$$\log(y) = b + wx + \varepsilon$$

One unit increase of x →
100w% increase of y

Example: y – income; x – tenured year; $w = 0.04$. One more tenured year increases 4% in income

Results Interpretation

Level-Level

$$y = b + wx + \varepsilon$$

One unit increase of x
 $\rightarrow w$ unit increase of y

Log-Level

$$\log(y) = b + wx + \varepsilon$$

One unit increase of x \rightarrow
 $100w\%$ increase of y

Example: y – income; x – tenured year; $w = 0.04$. One more tenured year increases 4% in income

Log-Log

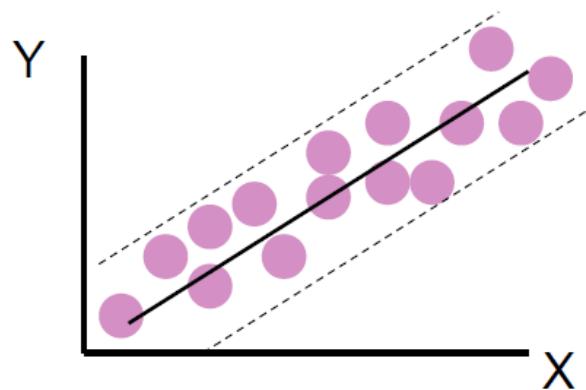
$$\log(y) = b + w \log(x) + \varepsilon$$

One percent increase of x
 $\rightarrow w\%$ increase of y

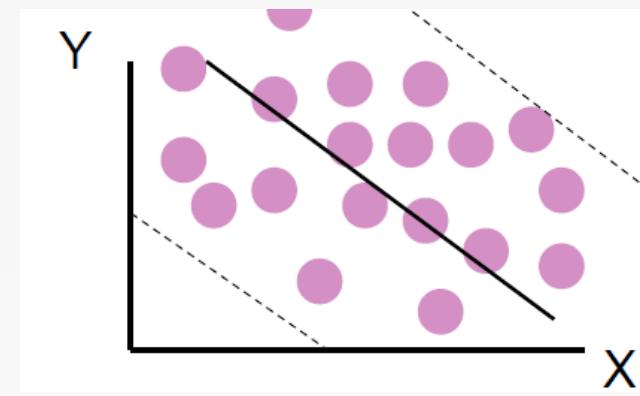
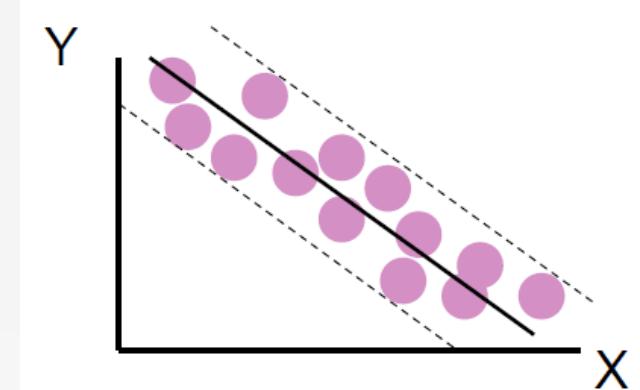
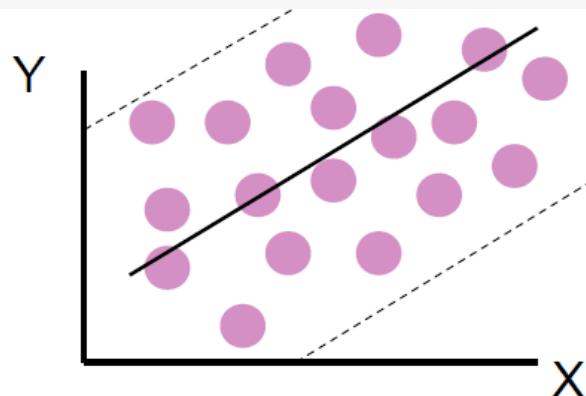
Example: y – demand; x – price; $w = -0.6$. 1% increase in price leads to 0.6% decrease in demand

Linear Model?

Strong relationship

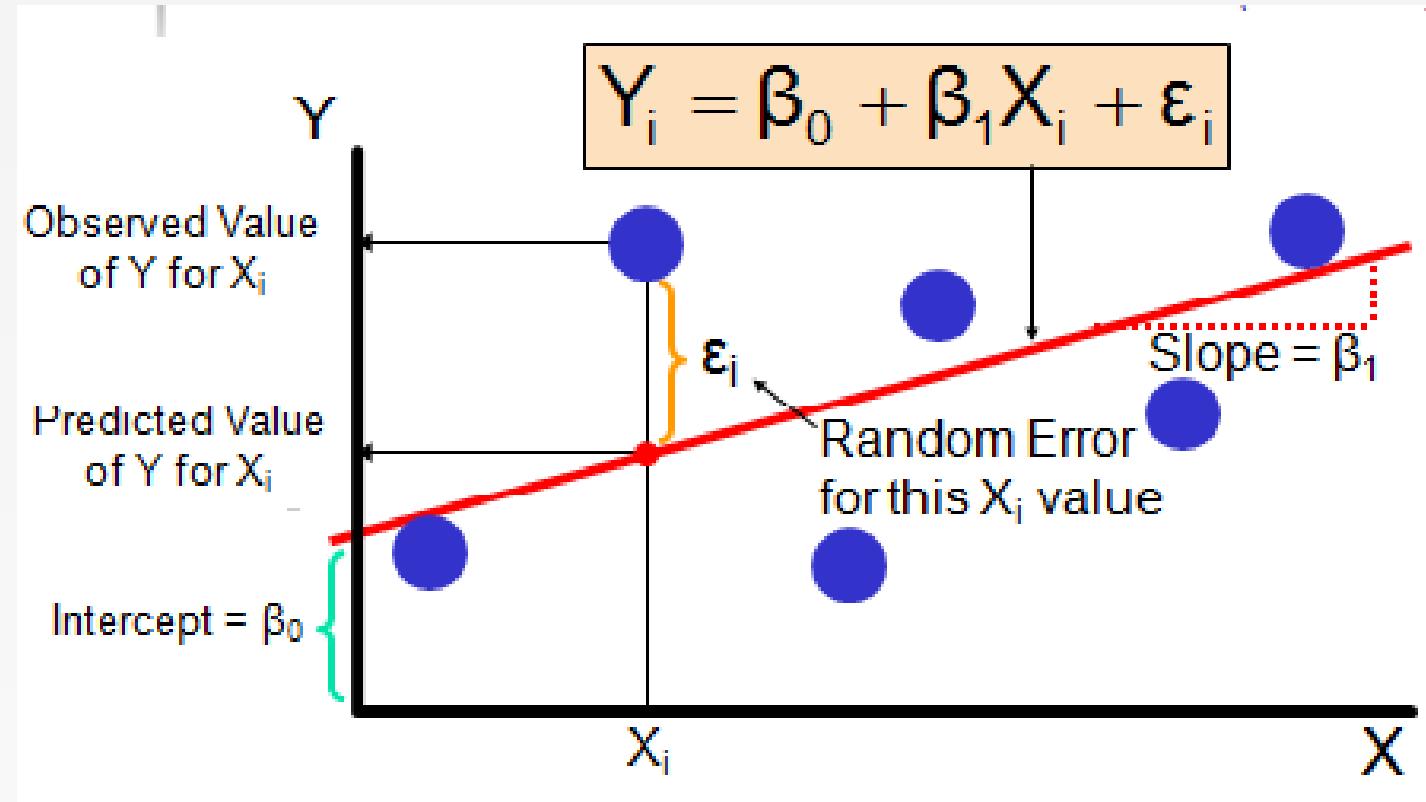


Weak relationship



How to evaluation?

Residuals



Residuals $e = \text{observed } (y) - \text{predicted } (y)$

Train a Linear Model

- Goal: minimize the Error
- Potential ways:
 - Sum (mean) of absolute errors $|e_1| + |e_2| + |e_3| + \dots$
 - Sum (mean) of squared errors $e_1^2 + e_2^2 + e_3^2 + \dots$

Function Approximation

- Objective function: mean squared error (MSE)

$$\begin{aligned} J(\theta) &= \frac{1}{N} \sum_{i=1}^N e_i^2 \\ &= \frac{1}{N} \sum_{i=1}^N (y^{(i)} - \theta^T x^{(i)})^2 \end{aligned}$$

$$\begin{aligned} x' &= [1, x_1, x_2, \dots, x_M]^T \\ \theta &= [b, w_1, \dots, w_M]^T \end{aligned}$$

- Solve the unconstrained optimization problem

- Closed form
- Gradient descent
- Stochastic gradient descent

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

- Test time:
 - Given a new x , make a prediction $\hat{y} = \hat{\theta}^T x$

Closed-form Solution

Criteria

Minimize the sum of squared errors

$$\min \sum_n (y_i - \hat{y}_i)^2$$

Solution

$$y_i = b + w_1 x_i + \varepsilon_i$$

w_1 : slope for the estimated regression equation

$$w_1 = \frac{\sum_n (x_i - \bar{x})(y_i - \bar{y})}{\sum_n (x_i - \bar{x})^2}$$

b : intercept for the estimated regression equation

$$b = \bar{y} - w_1 \bar{x}$$

$$\theta = (X^T X)^{-1} X^T y$$

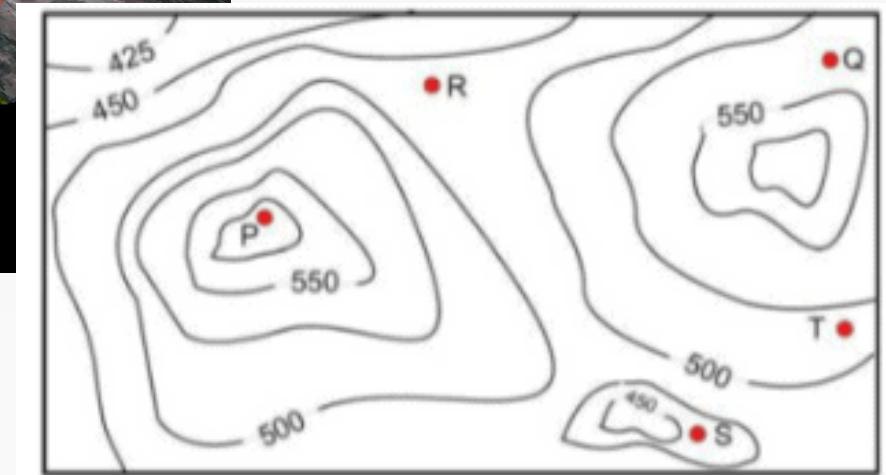
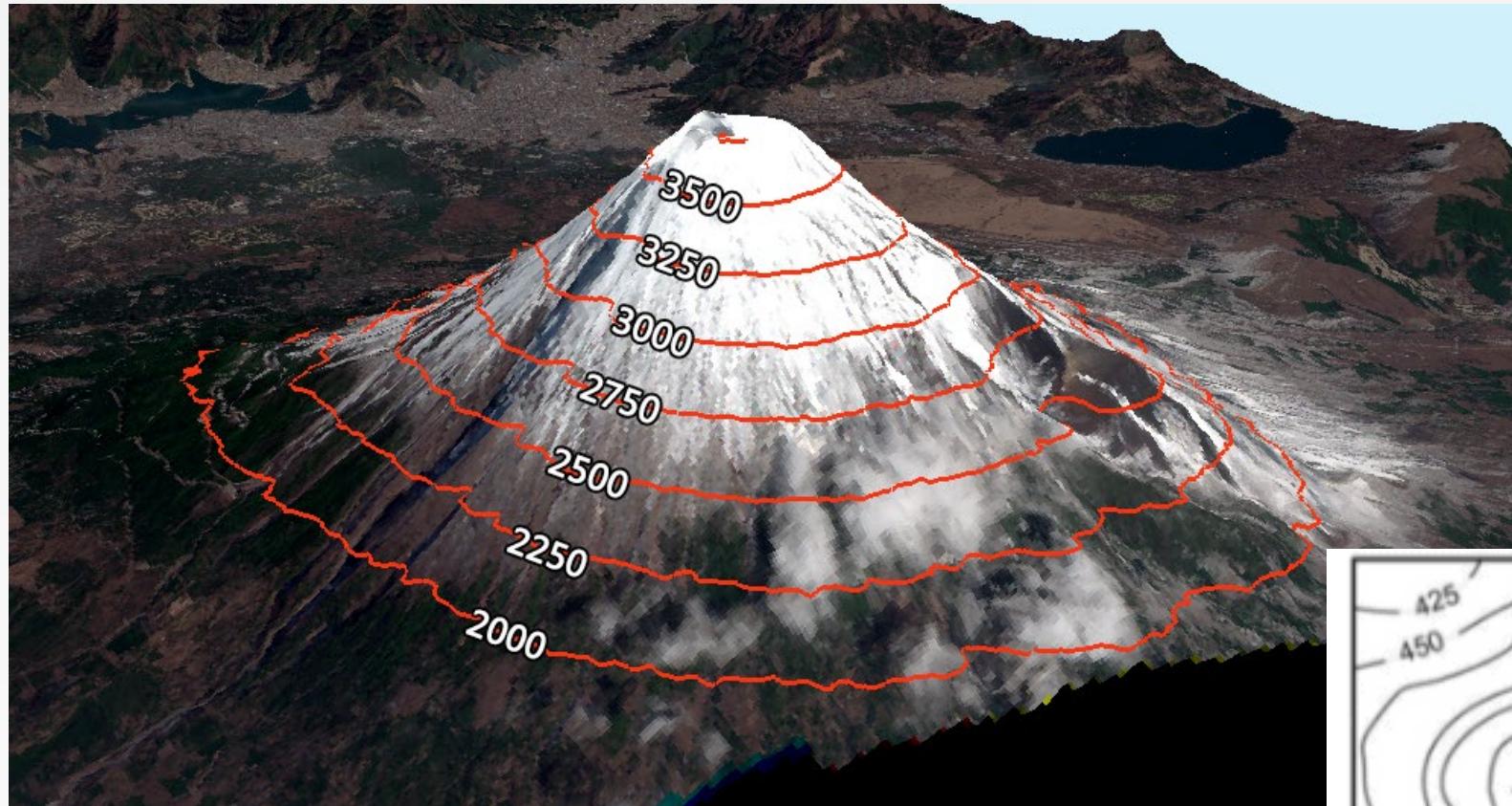
Gradient Descent

Linear Regression Solutions

- Closed-form solution:
 - Computational complexity
 - Stability
- Gradient Descent for Linear Regression

$$\beta = (X^T X)^{-1} X^T y$$

Contour Plots



Contour Plots

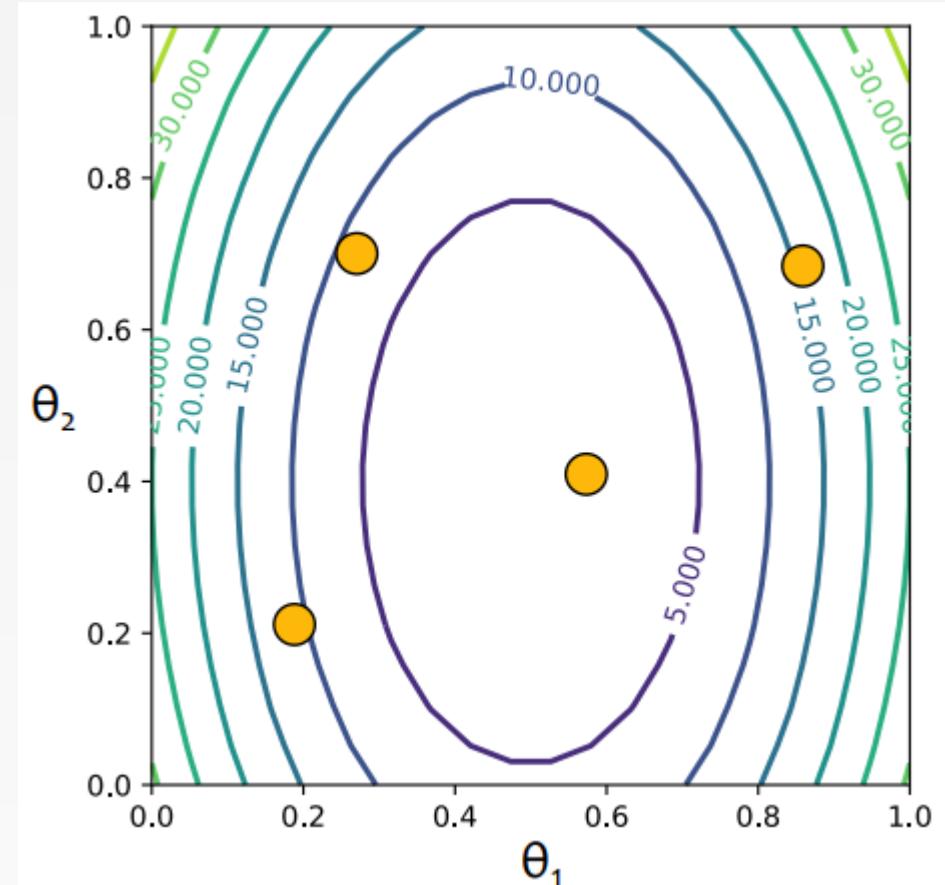
1. Each level curve labeled with value
2. Value label indicates the value of the function for all points lying on that level curve

Optimization by Random Guessing

Random guessing:

1. Pick a random θ
2. Evaluate $J(\theta)$
3. Repeat steps 1 and 2 many times
4. Return θ that gives smallest $J(\theta)$

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = (10(\theta_1 - 0.5))^2 + (6(\theta_2 - 0.4))^2$$



Optimization by Random Guessing

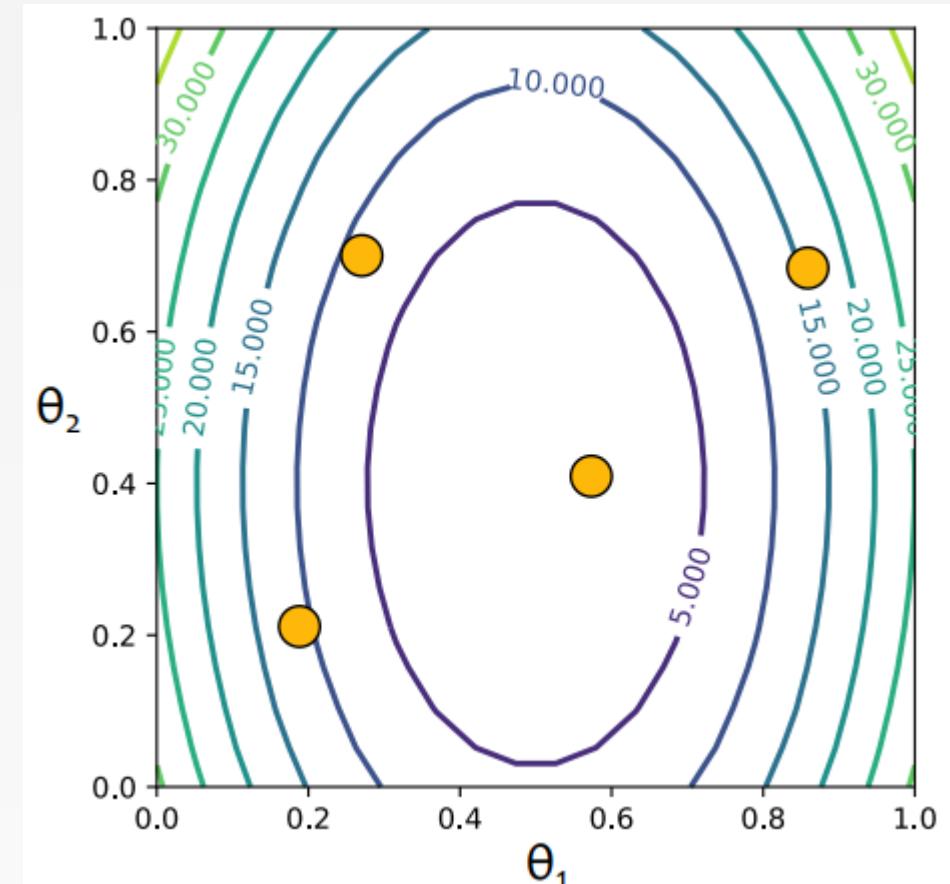
Random guessing:

1. Pick a random θ
2. Evaluate $J(\theta)$
3. Repeat steps 1 and 2 many times
4. Return θ that gives smallest $J(\theta)$

Linear Regression:

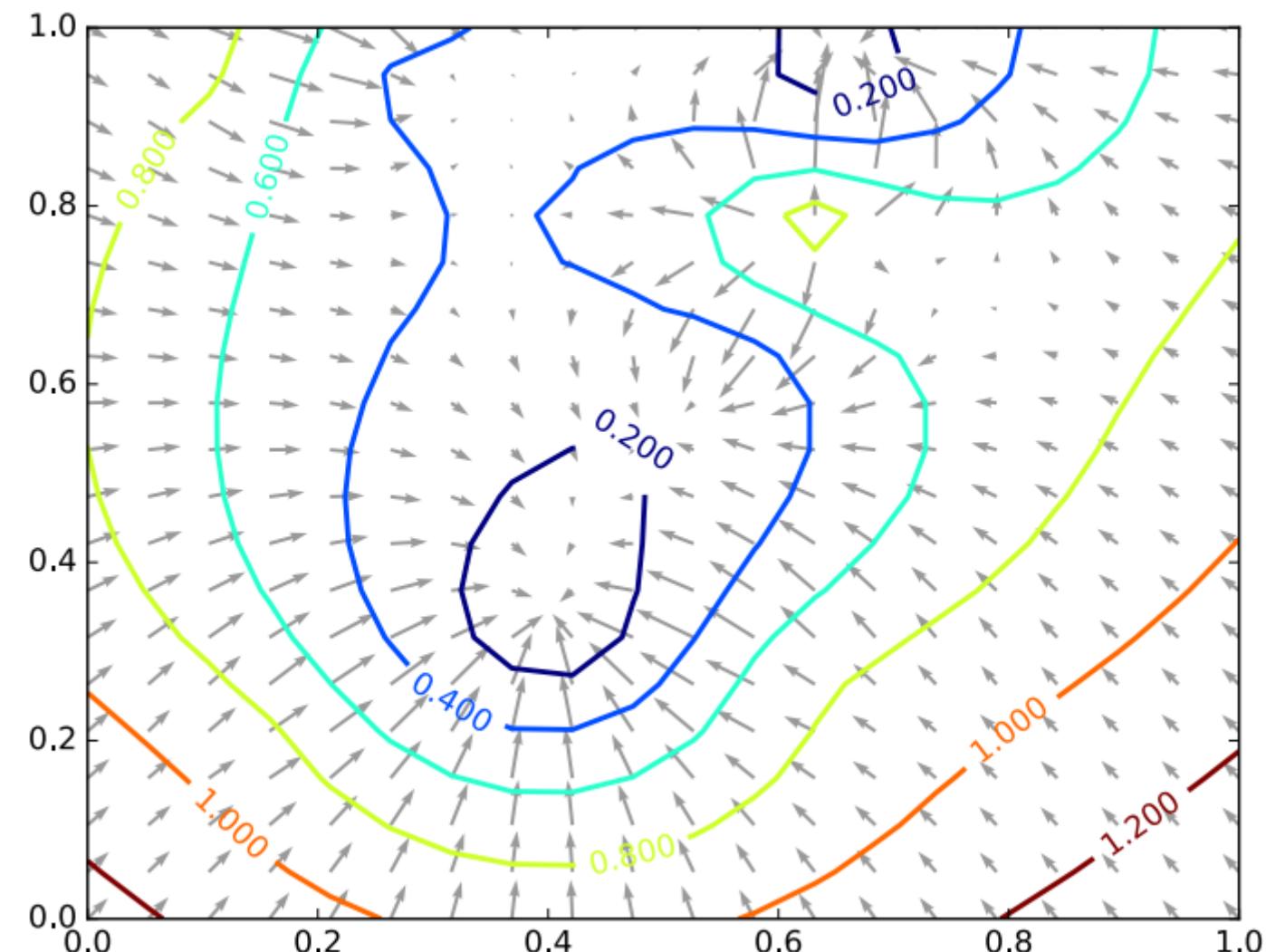
1. Objective function: MSE
2. contour plot: each line labeled with MSE
– lower means a better fit
3. **minimum** corresponds to parameters
 $(w, b) = (\theta_1, \theta_2)$ that **best fit** some training dataset

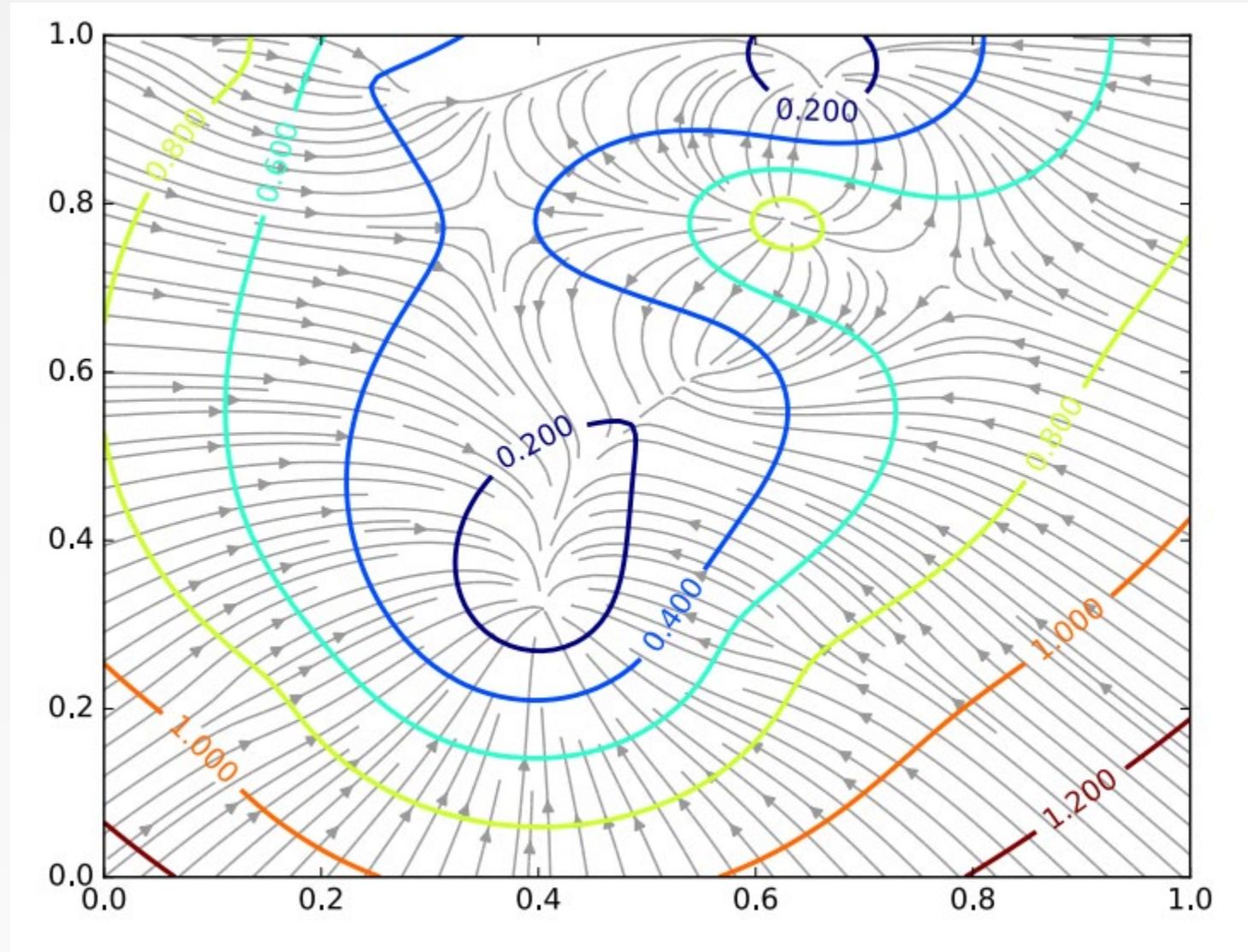
$$J(\theta) = J(\theta_1, \theta_2) = (10(\theta_1 - 0.5))^2 + (6(\theta_2 - 0.4))^2$$





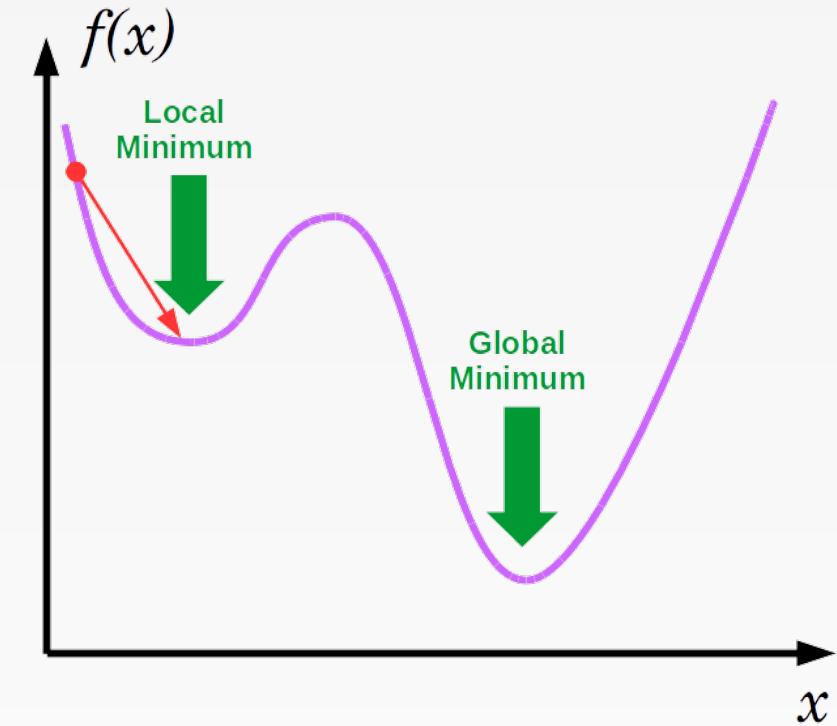
Gradient Descent





Pros and Cons

- Advantages:
 - Simple and often quite effective on ML tasks
 - Often very scalable
- Drawbacks
 - Might find a local minimum
 - Only applies to smooth function (differentiable)

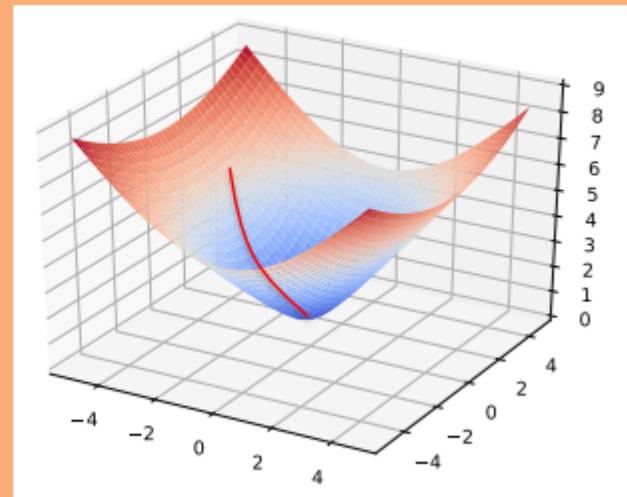


Algorithm

Algorithm 1 Gradient Descent

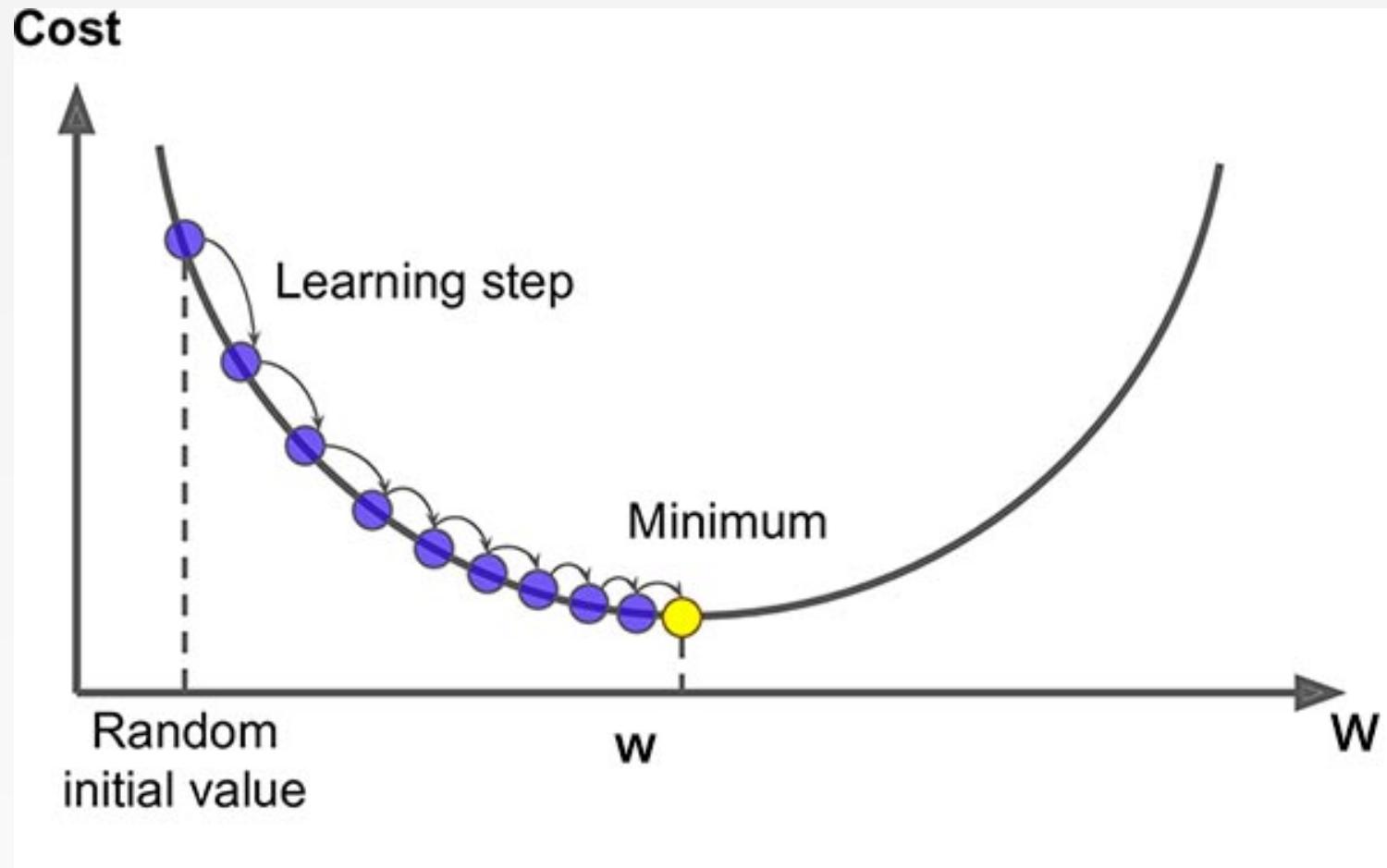
```

1: procedure GD( $\mathcal{D}$ ,  $\theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:      $\theta \leftarrow \theta - \gamma \nabla_{\theta} J(\theta)$ 
5:   return  $\theta$ 
```

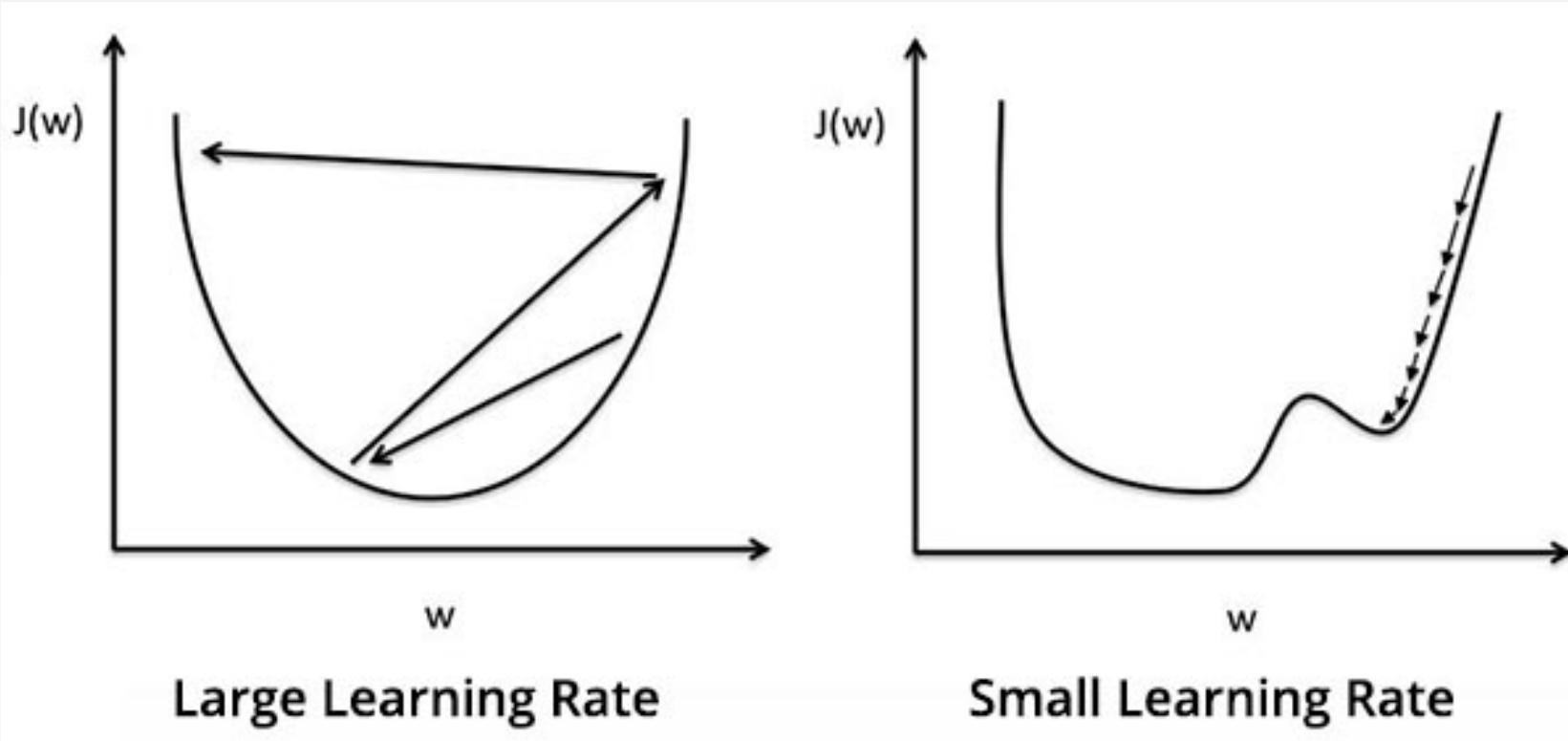


Convergence Criteria (one example): $\|\nabla_{\theta} J(\theta)\|_2 \leq \epsilon$

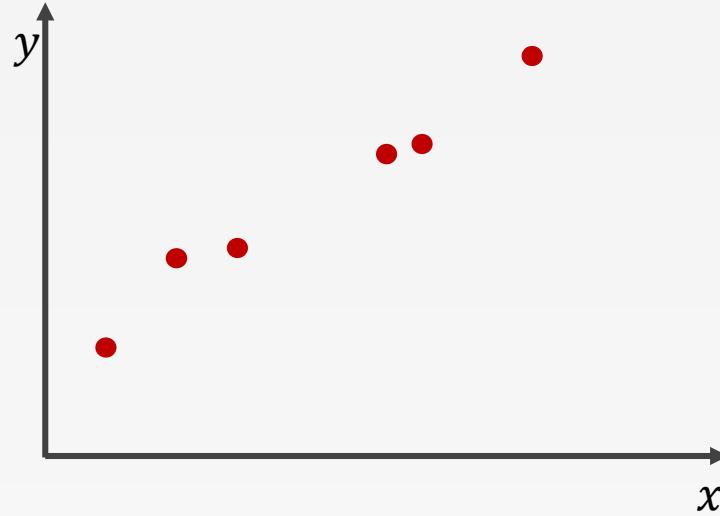
Gradient Descent



Learning Rate

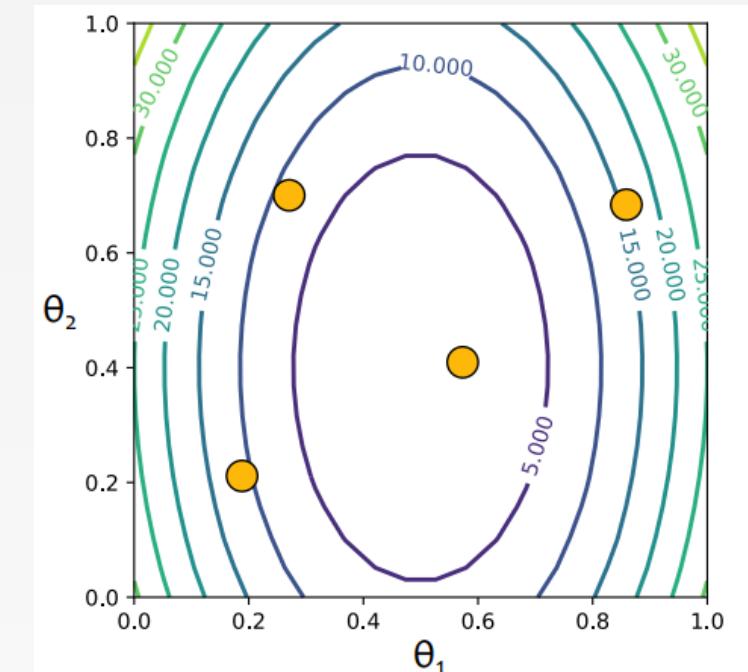


GD for Linear Regression

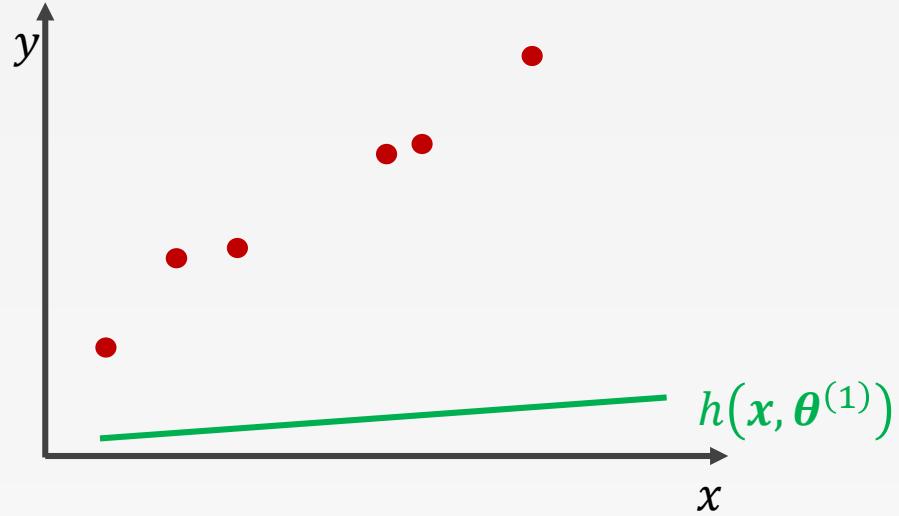


t	θ_1	θ_2	J

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N} \sum (y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)})^2$$

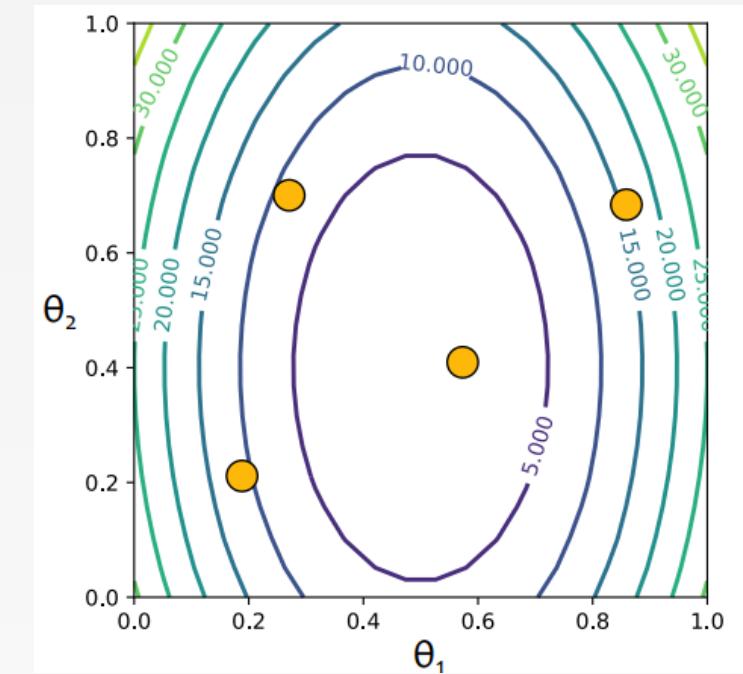


GD for Linear Regression

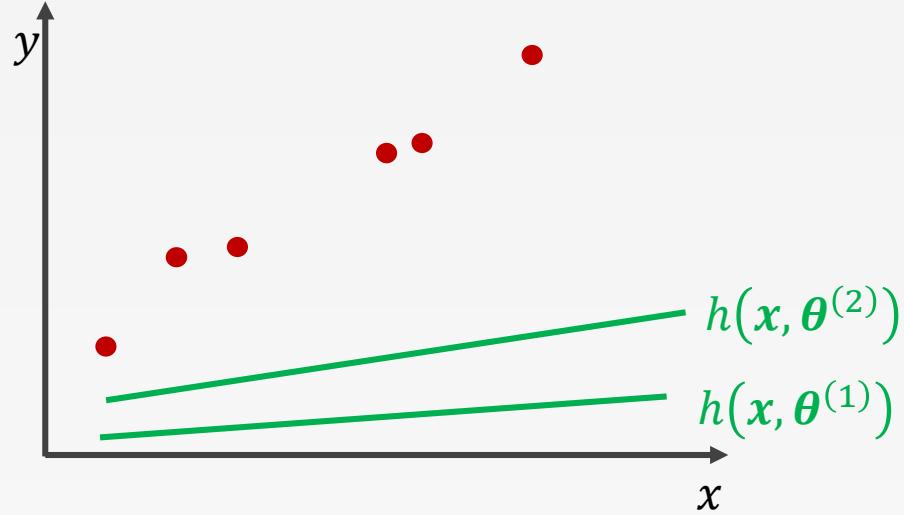


t	θ_1	θ_2	J
1	0.01	0.02	25.2

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N} \sum (y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)})^2$$

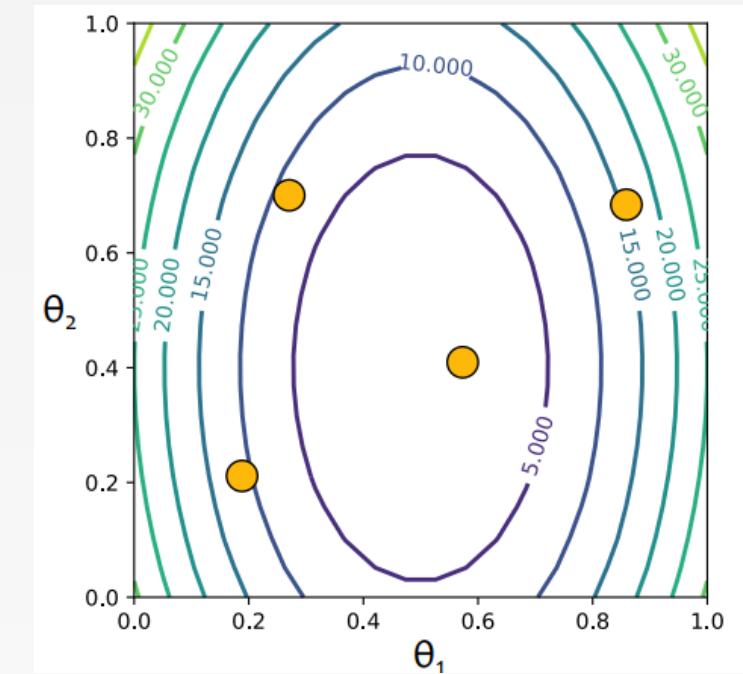


GD for Linear Regression

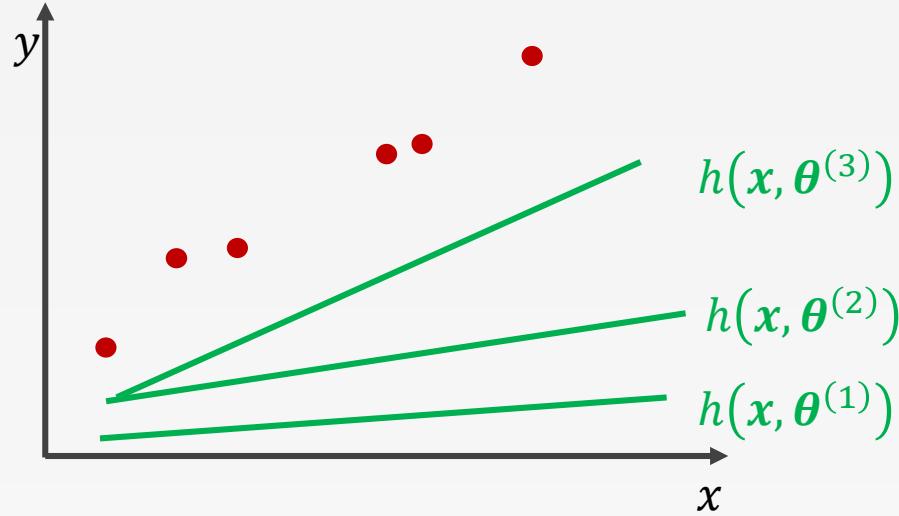


t	θ_1	θ_2	J
1	0.01	0.02	25.2
2	0.30	0.12	8.7

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N} \sum (y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)})^2$$

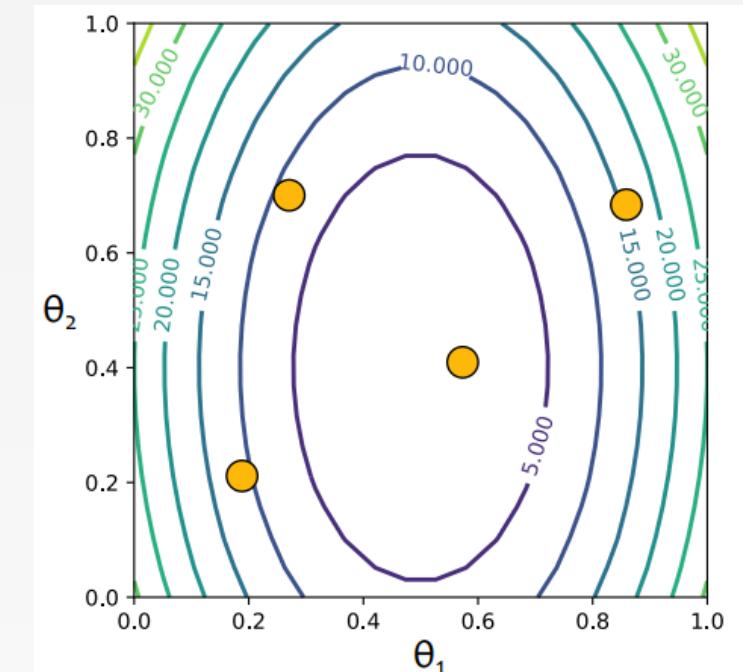


GD for Linear Regression

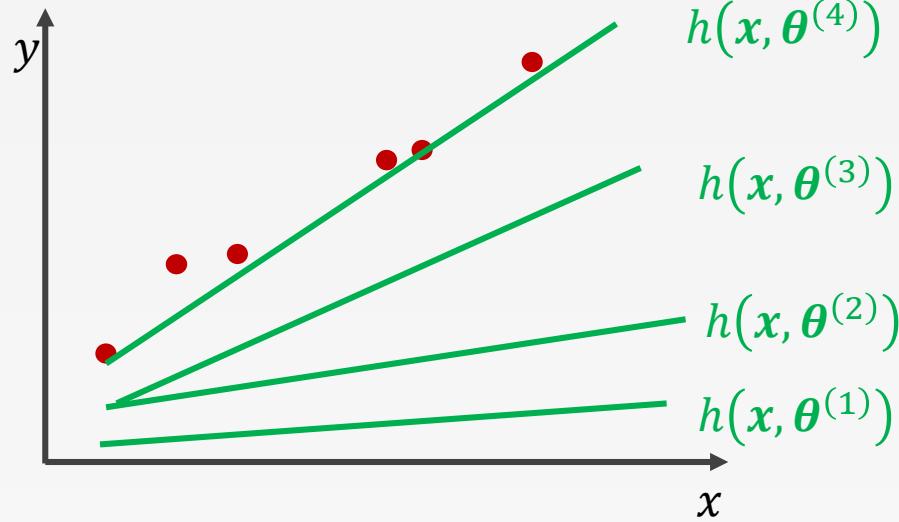


t	θ_1	θ_2	J
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N} \sum (y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)})^2$$

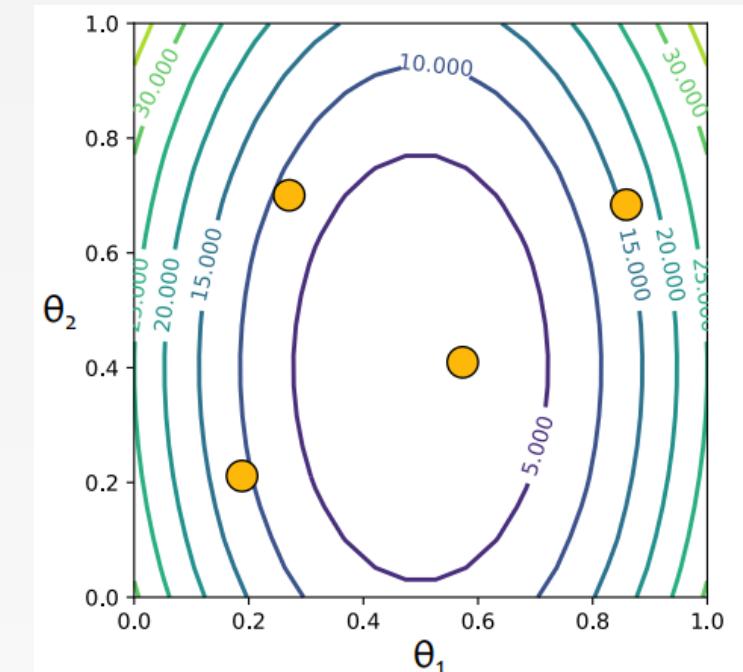


GD for Linear Regression

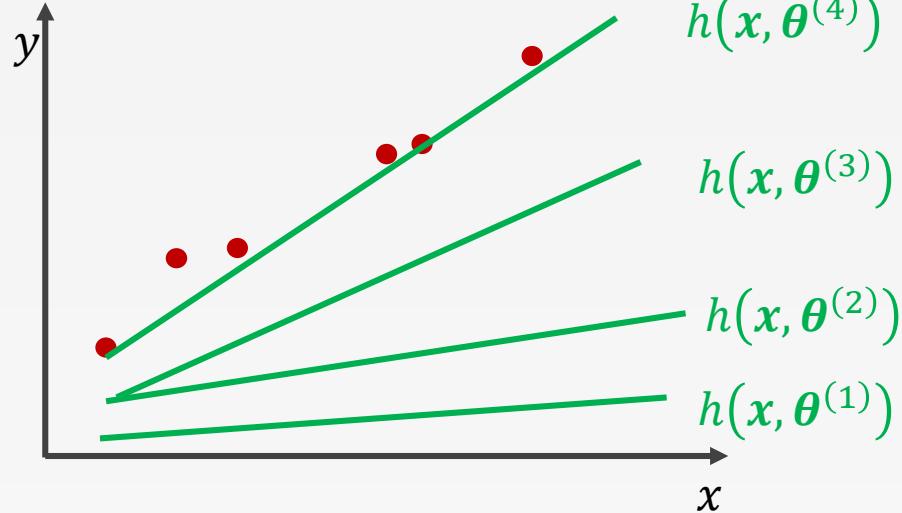


t	θ_1	θ_2	J
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

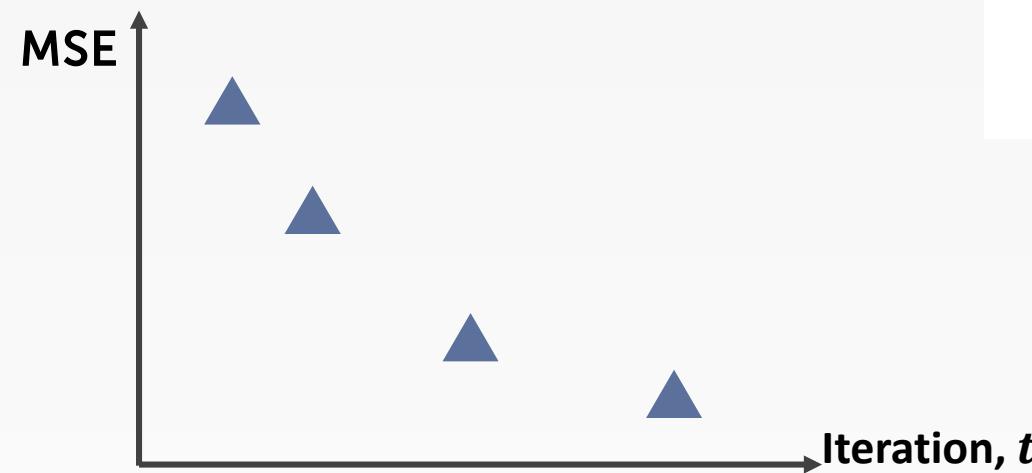
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N} \sum (y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)})^2$$



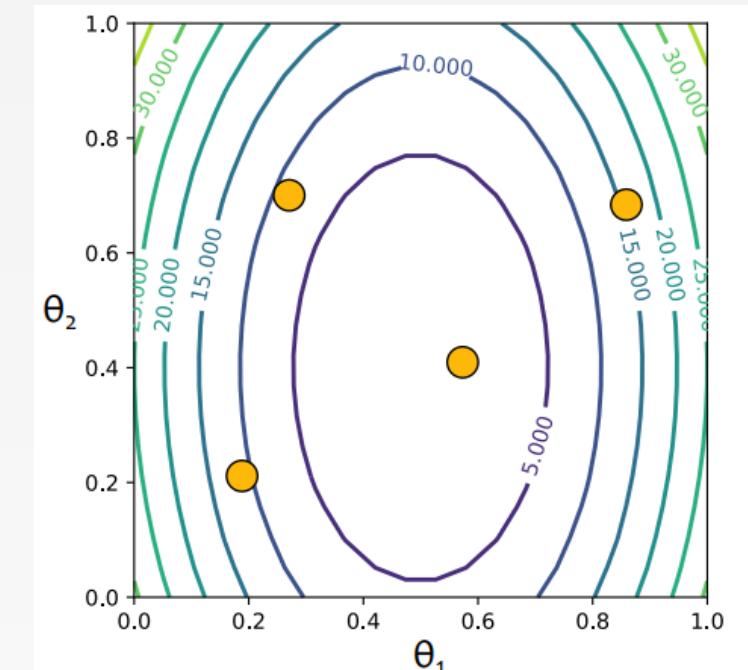
GD for Linear Regression



t	θ_1	θ_2	J
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2



$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N} \sum (y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)})^2$$



GD for Linear Regression

Algorithm 1 GD for Linear Regression

```
1: procedure GDLR( $\mathcal{D}, \theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$                                  $\triangleright$  Initialize parameters
3:   while not converged do
4:      $\mathbf{g} \leftarrow \sum_{i=1}^N (\theta^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$      $\triangleright$  Compute gradient
5:      $\theta \leftarrow \theta - \gamma \mathbf{g}$                                  $\triangleright$  Update parameters
6:   return  $\theta$ 
```

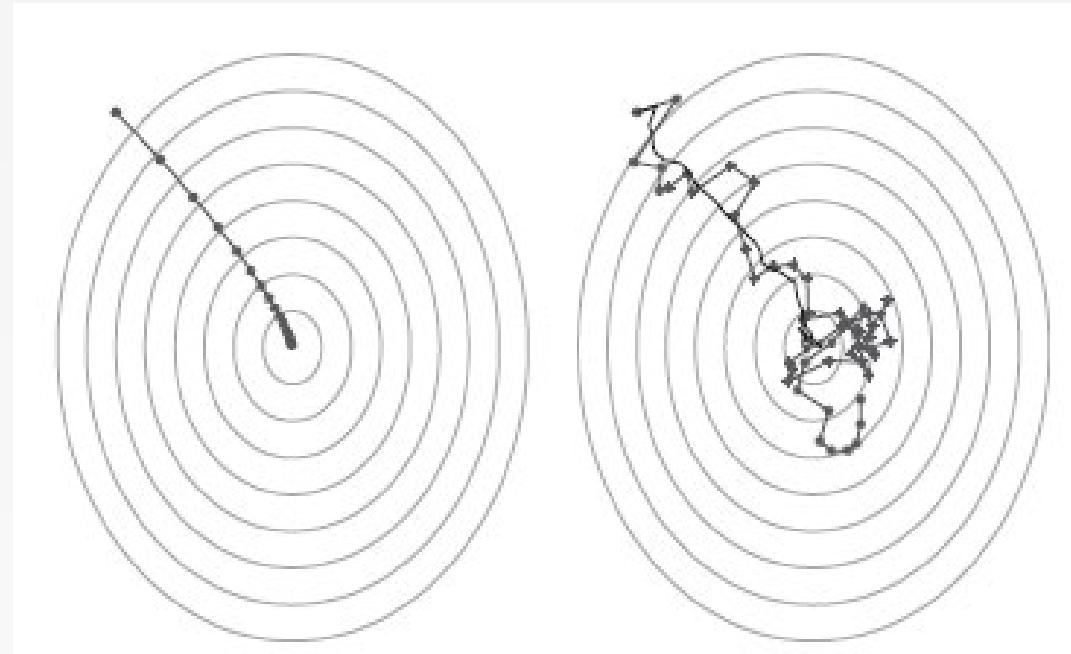
Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

```
1: procedure SGD ( $D, \theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:     for  $i \sim \text{Uniform}(\{1, 2, 3, \dots, N\})$ 
5:        $\theta \leftarrow \theta - \gamma \nabla_{\theta} J^{(i)}(\theta)$ 
6:   return  $\theta$ 
```

Stochastic Gradient Descent

- Just picks a random instance (or sampling) in the training set to compute the gradient



Mini-Batch SGD

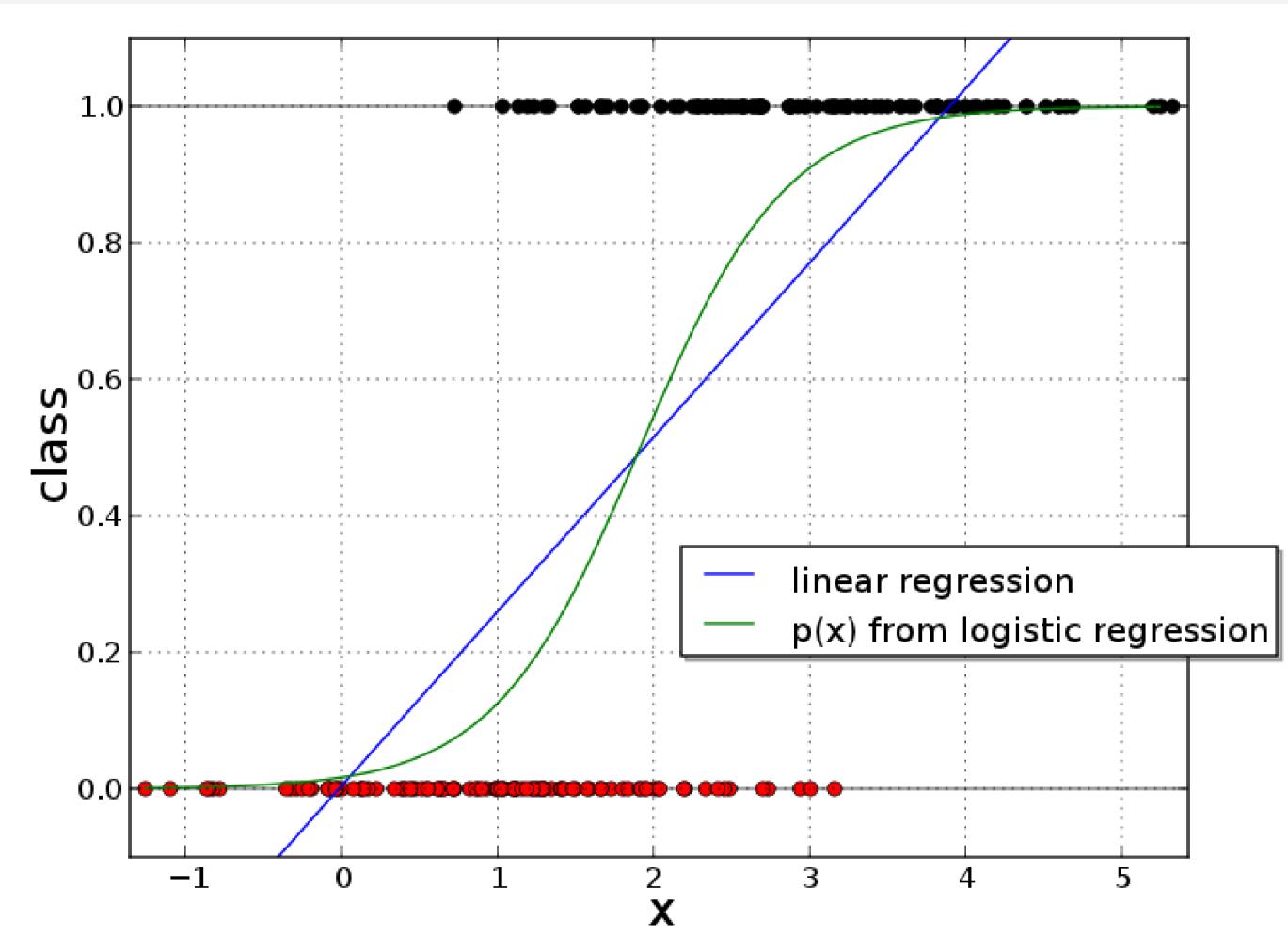
- Gradient Descent:
 - Compute true gradient exactly from all N examples
- Stochastic Gradient Descent (SGD):
 - Approximate true gradient by the gradient of one randomly chosen example
- Mini-Batch SGD:
 - Approximate true gradient by the average gradient of K randomly chosen examples

Logistic Regression

General View

- Logistic regression is a **classification** algorithm
- It predicts the probability of occurrence of an event by fitting data to a *logit* function
- Outcome: categorial variables

Logistic vs. Linear



Logistic Regression

- **Data:** Inputs are continuous vectors of length M. Outputs are discrete.

$$D = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^N$$

- **Model:** Logistic function applied to dot product of parameters with input vector.

$$p_{\theta}(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x})}$$

- **Learning:** finds the parameters that minimize some objective function.

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- **Prediction:** Output is the most probable class.

$$\hat{y} = \operatorname{argmax}_{y \in \{0,1\}} p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$$

Additional Information

- Estimates of coefficients (θ) are derived through an iterative process called Maximum Likelihood Estimation (MLE)
- If estimated probability > Cutoff \rightarrow Classify as class “1”
- Probability of success (Odds) $\hat{\pi} = P(y = 1|x) = \frac{e^u}{1+e^u}$
- **Odds-Ratio** for success $\frac{\hat{\pi}}{1-\hat{\pi}} = e^u$
- Log Odds-Ratio $\ln\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) = u = b + w_1X_1 + w_2X_2 + \dots$

MLE

Principle of Maximum Likelihood Estimation:

Choose the parameters that maximize the likelihood of the data.

$$\boldsymbol{\theta}^{\text{MLE}} = \operatorname{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^N p(\mathbf{x}^{(i)} | \boldsymbol{\theta})$$

Maximum Likelihood Estimate (MLE)

MLE tries to allocate as much probability mass as possible to the things we have observed... ...at the expense of the things we have not observed

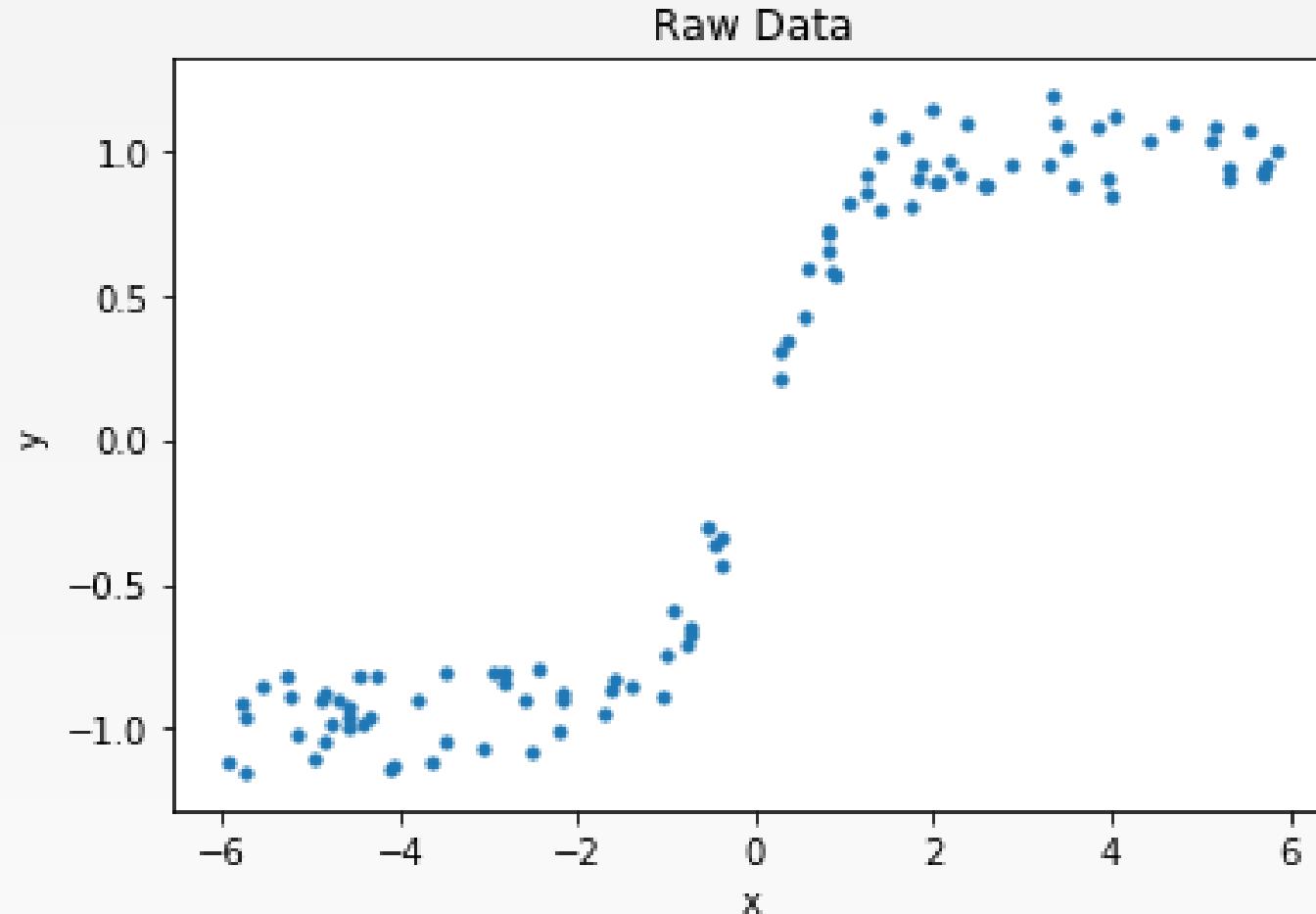
Polynomial Regression

Polynomial Regression

- Generate new features consisting of all polynomial combinations of the original features
- A linear model
- An application of non-linear transformations

$$\begin{aligned} X &= (x_0, x_1) \longrightarrow X' = (x_0, x_1, x_0x_1, x_0^2, x_1^2) \\ Y &= w_0x_0 + w_1x_1 + w_{01}x_0x_1 + w_{00}x_0^2 + w_{11}x_1^2 \end{aligned}$$

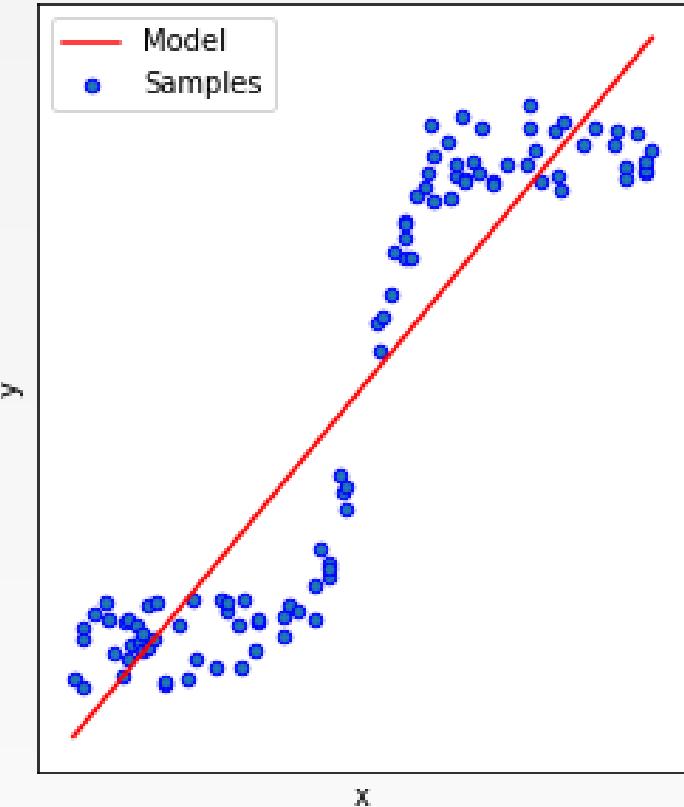
Example I: Polynomial Features



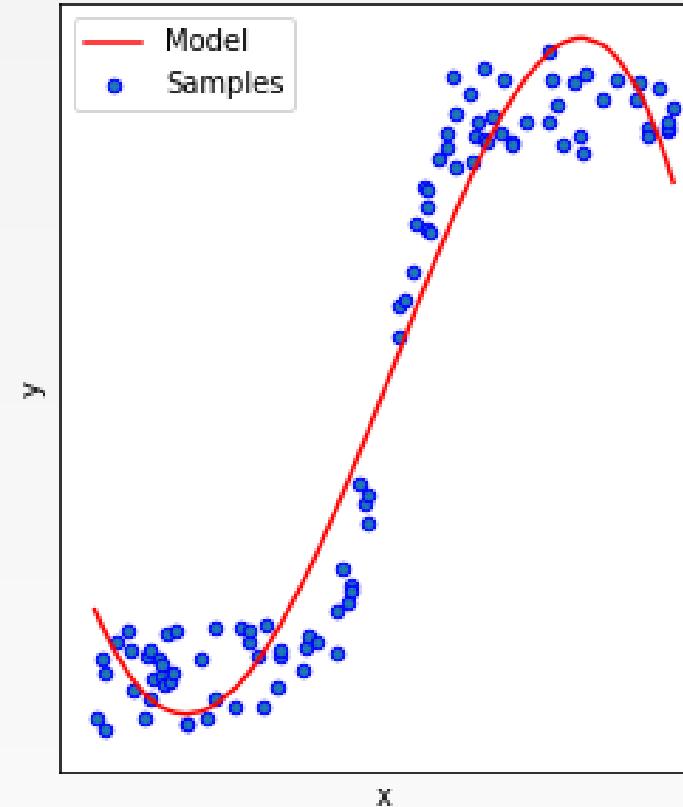
True: $y = \tanh(x) + \text{randn}$

Example I: Polynomial Features

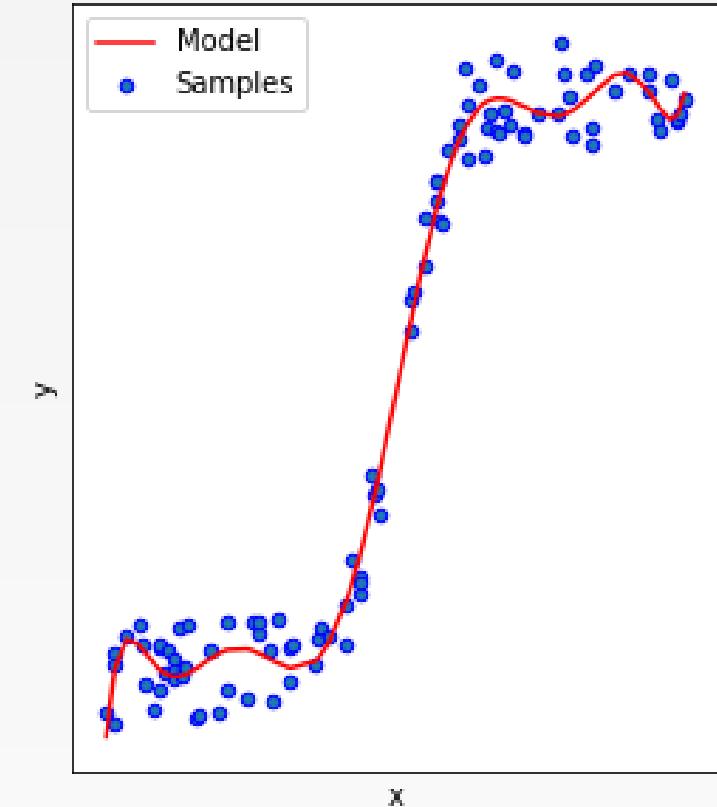
Polynomial Regression (poly = 1)



Polynomial Regression (poly = 4)

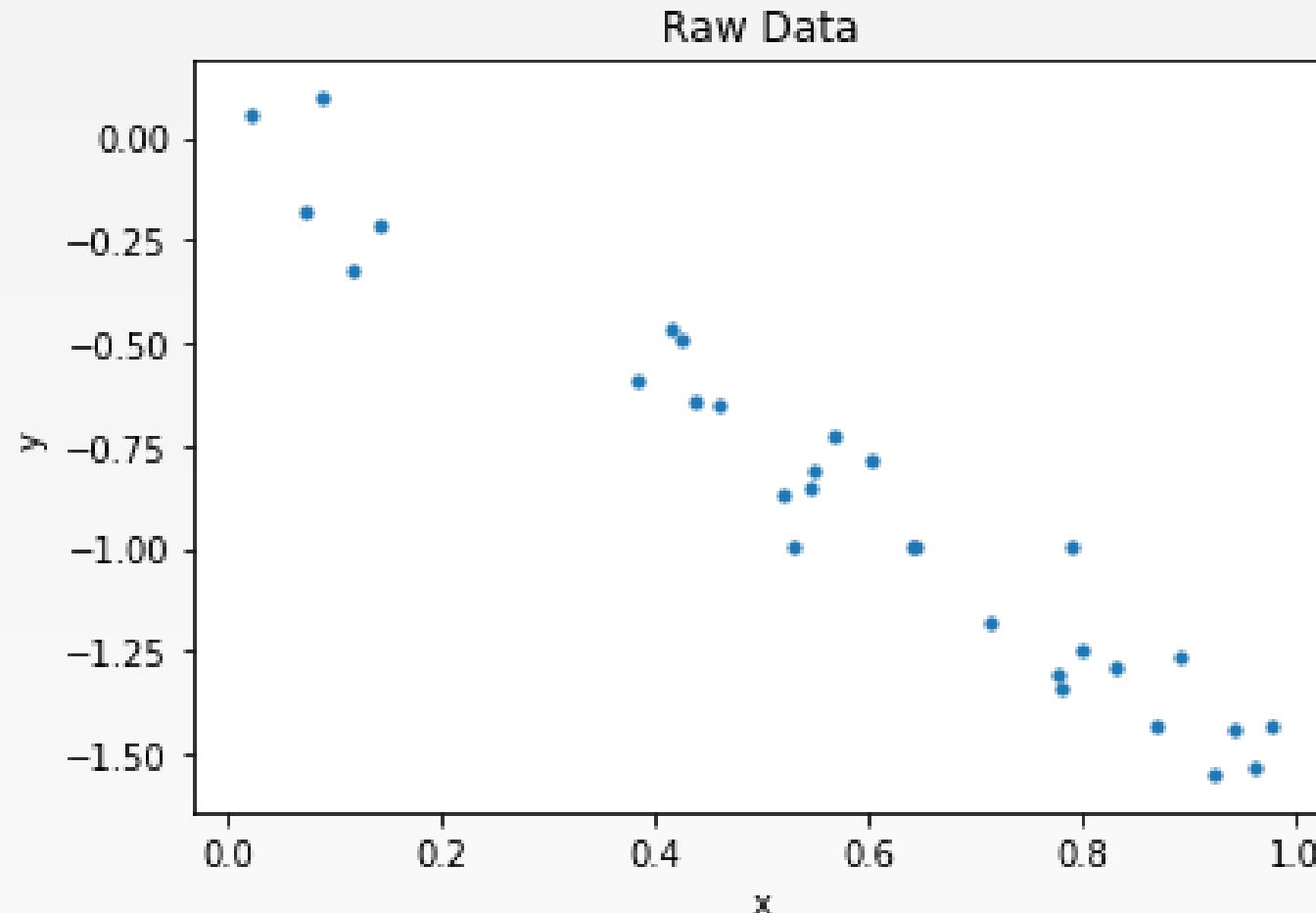


Polynomial Regression (poly = 9)



True: $y = \tanh(x) + \text{randn}$

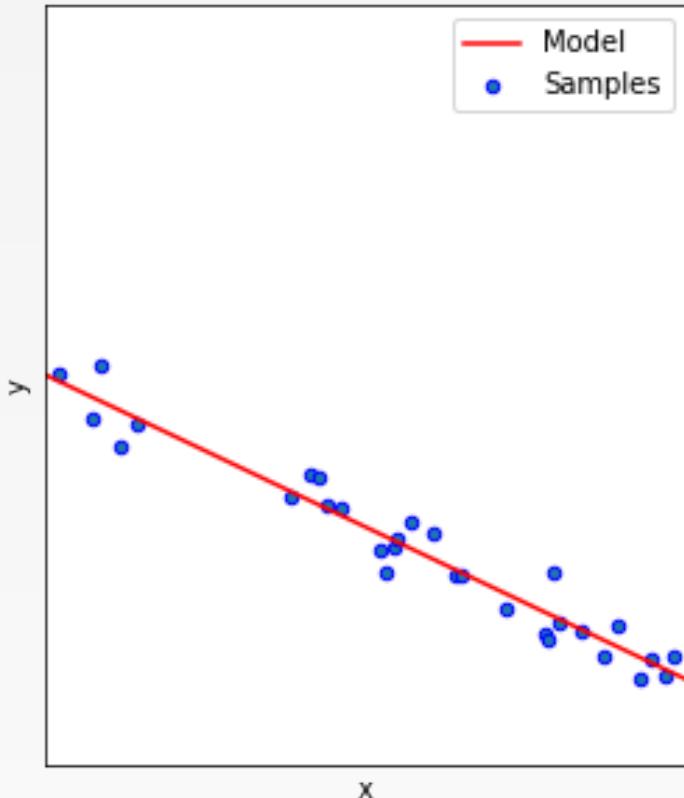
Example II: Polynomial Features



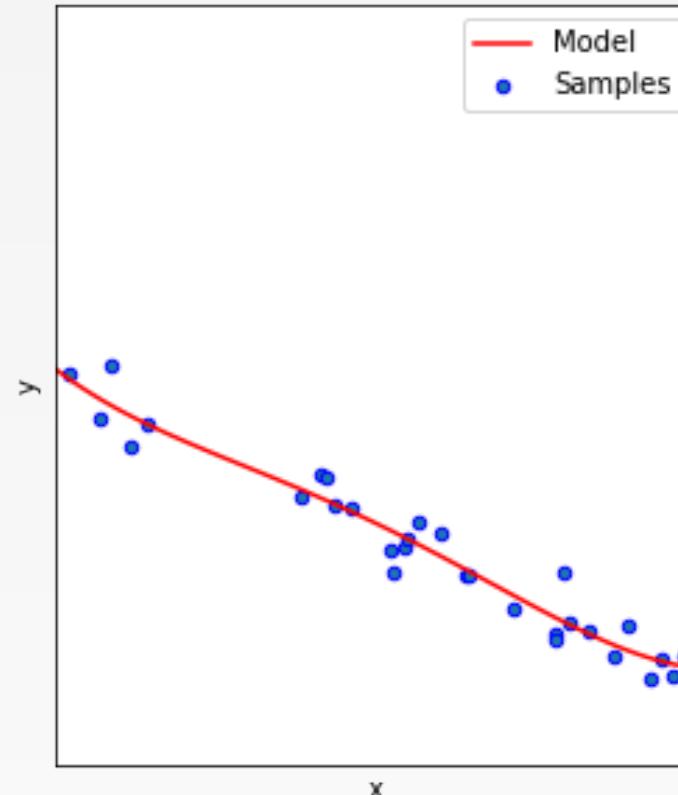
True: $y = -1.5 \cdot x + \text{randn}$

Example II: Polynomial Features

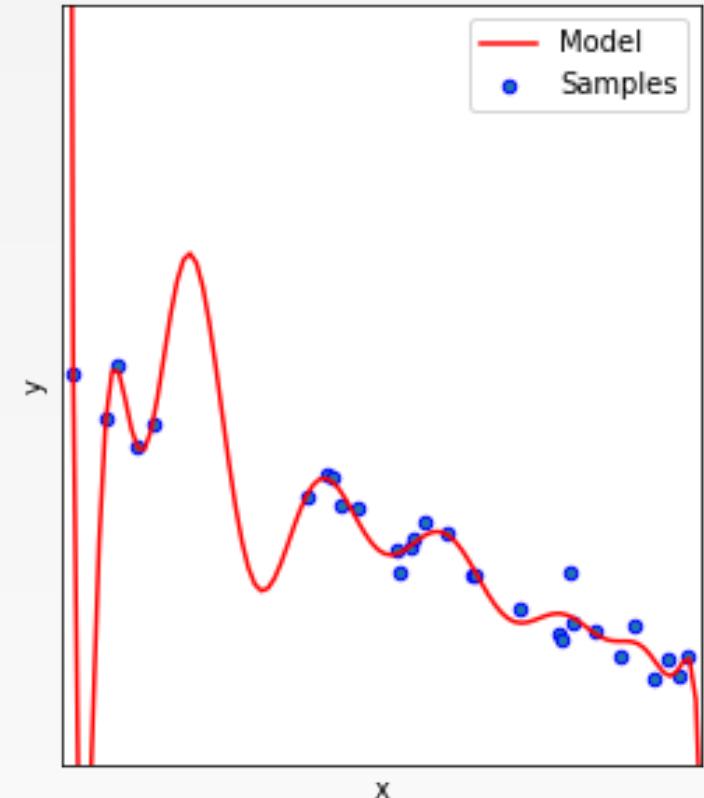
Polynomial Regression (poly = 1)



Polynomial Regression (poly = 4)



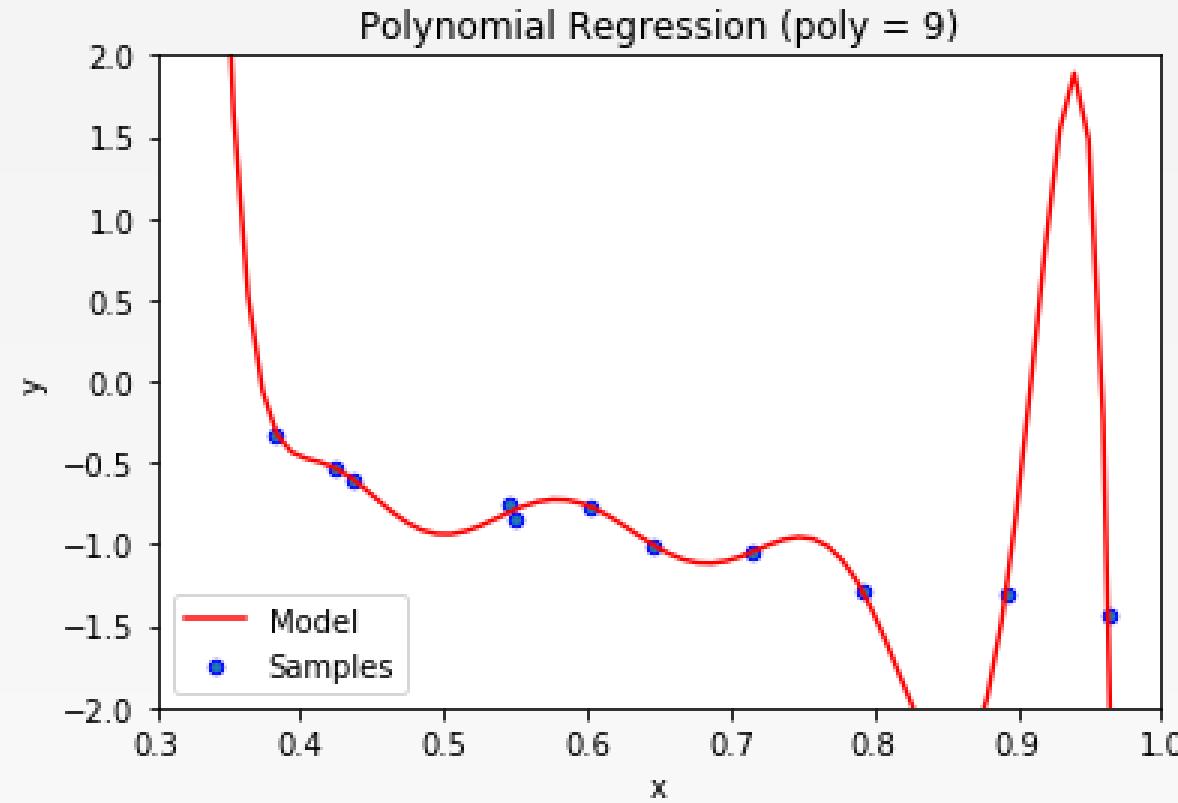
Polynomial Regression (poly = 15)



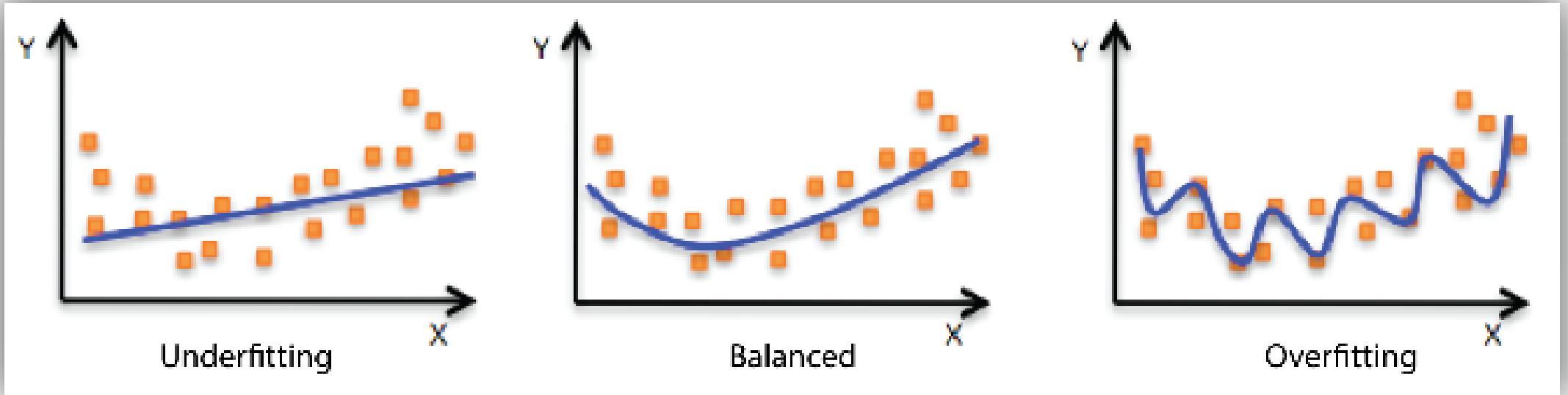
True: $y = -1.5 \cdot x + \text{randn}$

Overfitting

True:
 $y = -1.5 \cdot x + randn$



Overfitting Definition: when the model captures the noise in the training data instead of the underlying structure.



Regularization

Regularization

- Goal: optimize some combination of fit and simplicity
 - Penalize the magnitude of coefficients of features
 - Minimize the error between predicted and actual examples
- Ridge Regression:
 - L2-norm: adds penalty equivalent to square of the magnitude of coefficients
- Lasso Regression:
 - L1-norm: adds penalty equivalent to absolute value of the magnitude of coefficients

Ridge Regression

- Recall: in a linear regression with the least square estimation

$$RSS = \sum_{i=1}^n (y_i - (\omega \cdot x_i + b))^2$$

- Ridge regression

$$RSS = \sum_{i=1}^n (y_i - (\omega \cdot x_i + b))^2 + \alpha \sum_{j=1}^p \omega_j^2$$

Shrinkage penalty

Ridge Regression: λ

- Ridge regression

$$RSS = \sum_{i=1}^n (y_i - (\omega \cdot x_i + b))^2 + \alpha \sum_{j=1}^p \omega_j^2$$

α : tuning parameter

- $\alpha = 0$:
 - A simple linear regression
- $\alpha = \infty$:
 - Coefficients ω will be zero
- As α increases, the flexibility of the model fit decreases

LASSO

Least Absolute Shrinkage and Selection Operator Regression

- L1 Regularization

$$RSS = \sum_{i=1}^n (y_i - (\omega \cdot x_i + b))^2 + \alpha \sum_{j=1}^p |\omega_j|$$

- Lasso combines some of the shrinking advantages of ridge with **variable selection**
 - The L1 penalty has the effects of forcing some coefficient estimates to be exactly equal to zero

Tip: Techniques such as cross validation are recommended to determine which approach is better on a particular dataset

Questions?



机器学习与人工智能 Machine Learning and Artificial Intelligence

Lecture 3 Decision Trees

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

Decision Trees

Classification: Terminology

- **Inputs** = Predictors = Independent Variables = Attributes
- **Output** = Responses = Dependent Variables = Class labels = Target variables
- **Models** = Classifiers
- With a classification algorithm, we aim to build a model to predict what output will be obtained from given inputs.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Sample Data

Outlook	Temp	Humidity	Windy	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Will I Play Tennis Today?

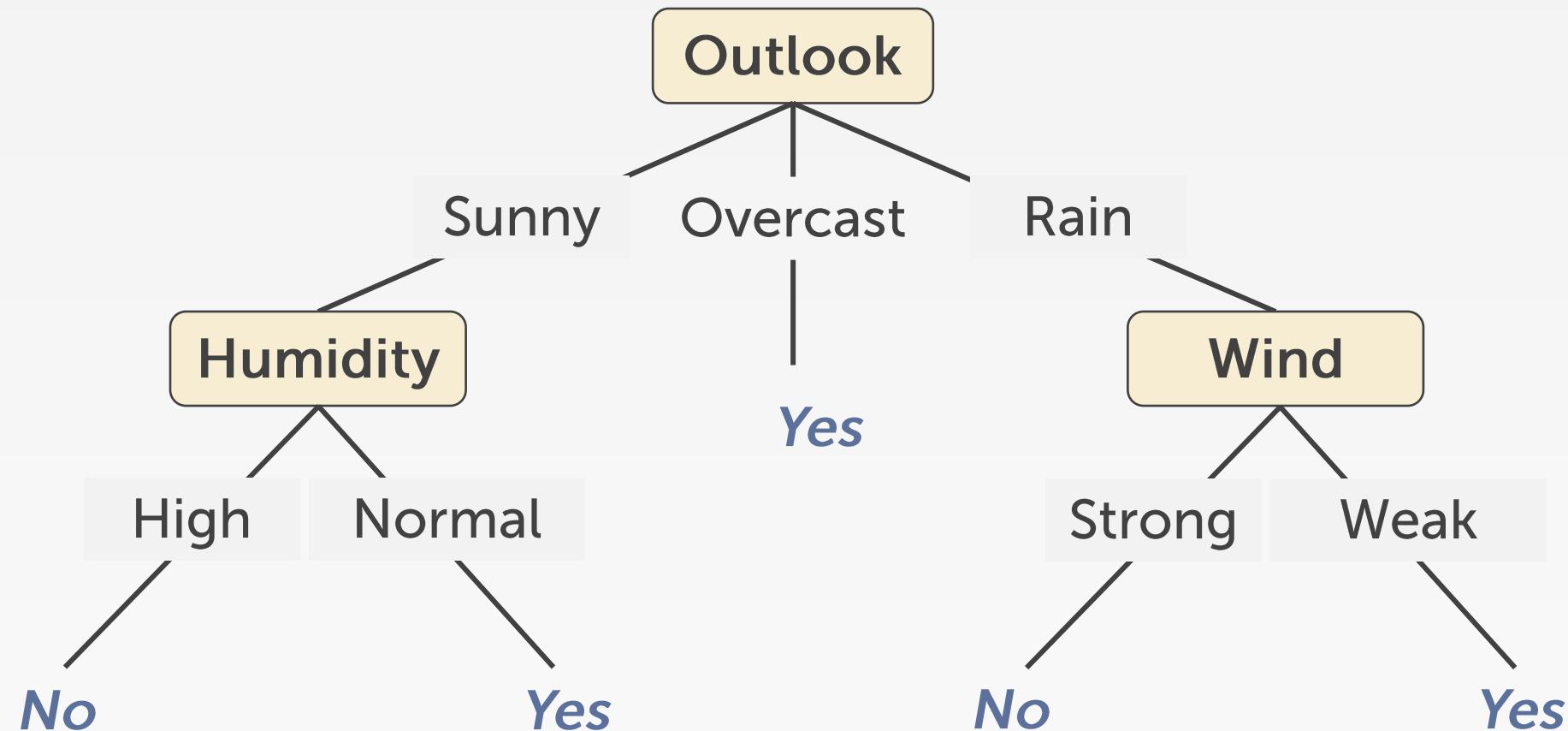
Input

Outlook:	{Sunny, Overcast, Rain}
Temperature:	{Hot, Mild, Cool}
Humidity:	{High, Normal, Low}
Wind:	{Strong, Weak}

Labels

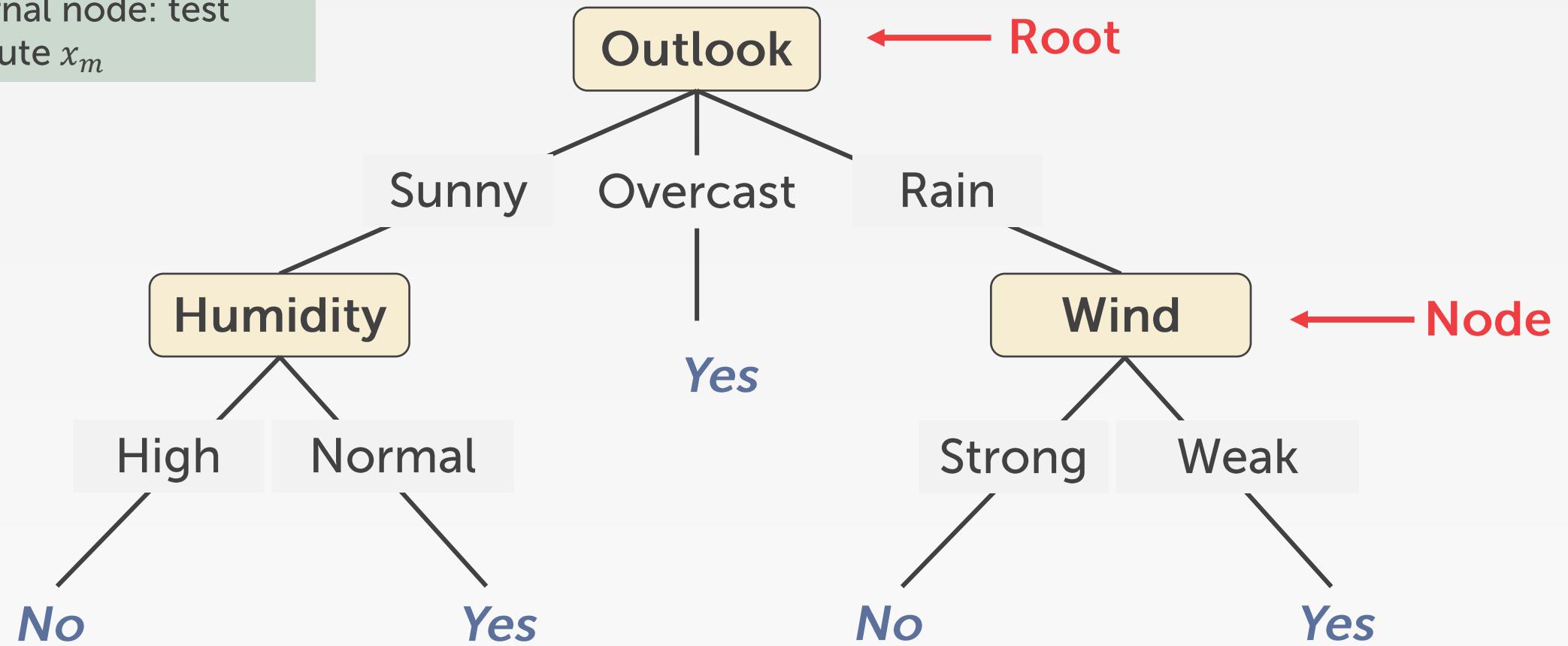
Binary classification task: $Y = \{\text{Yes}, \text{No}\}$

A Decision Tree



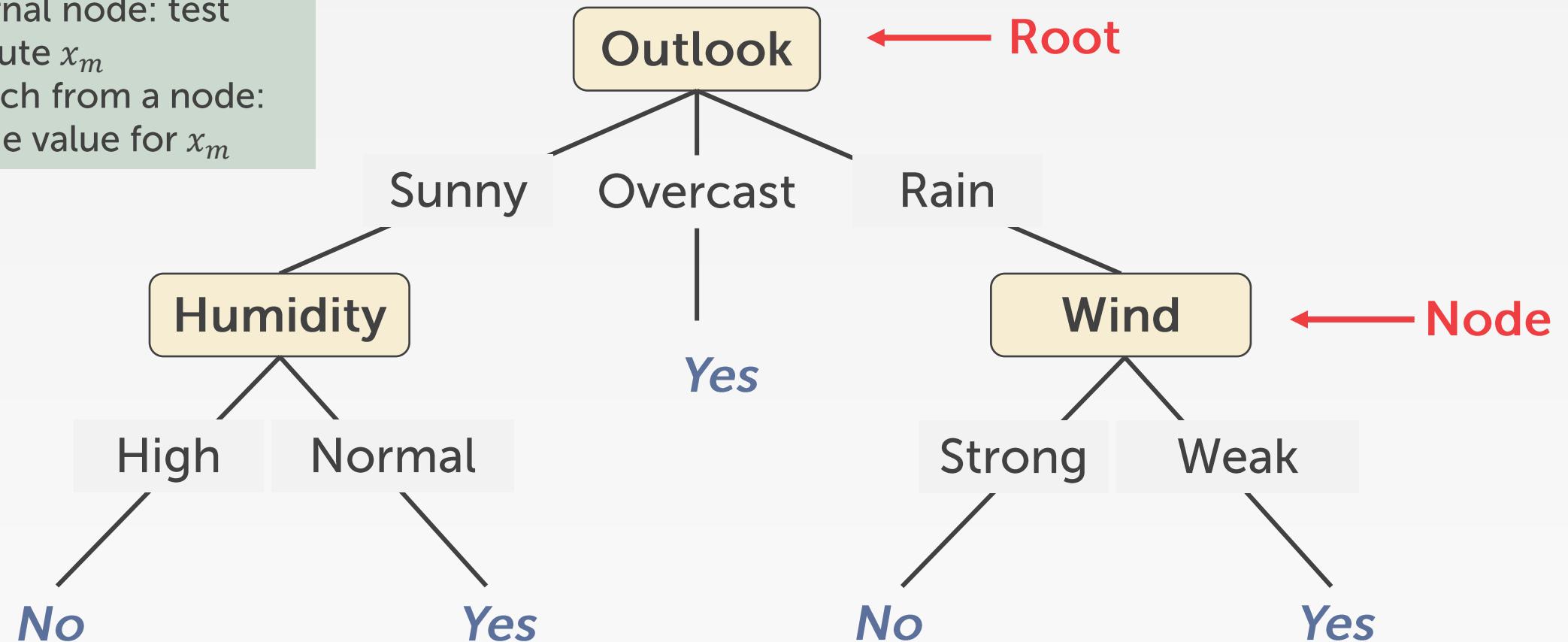
A Decision Tree

- Each internal node: test one attribute x_m



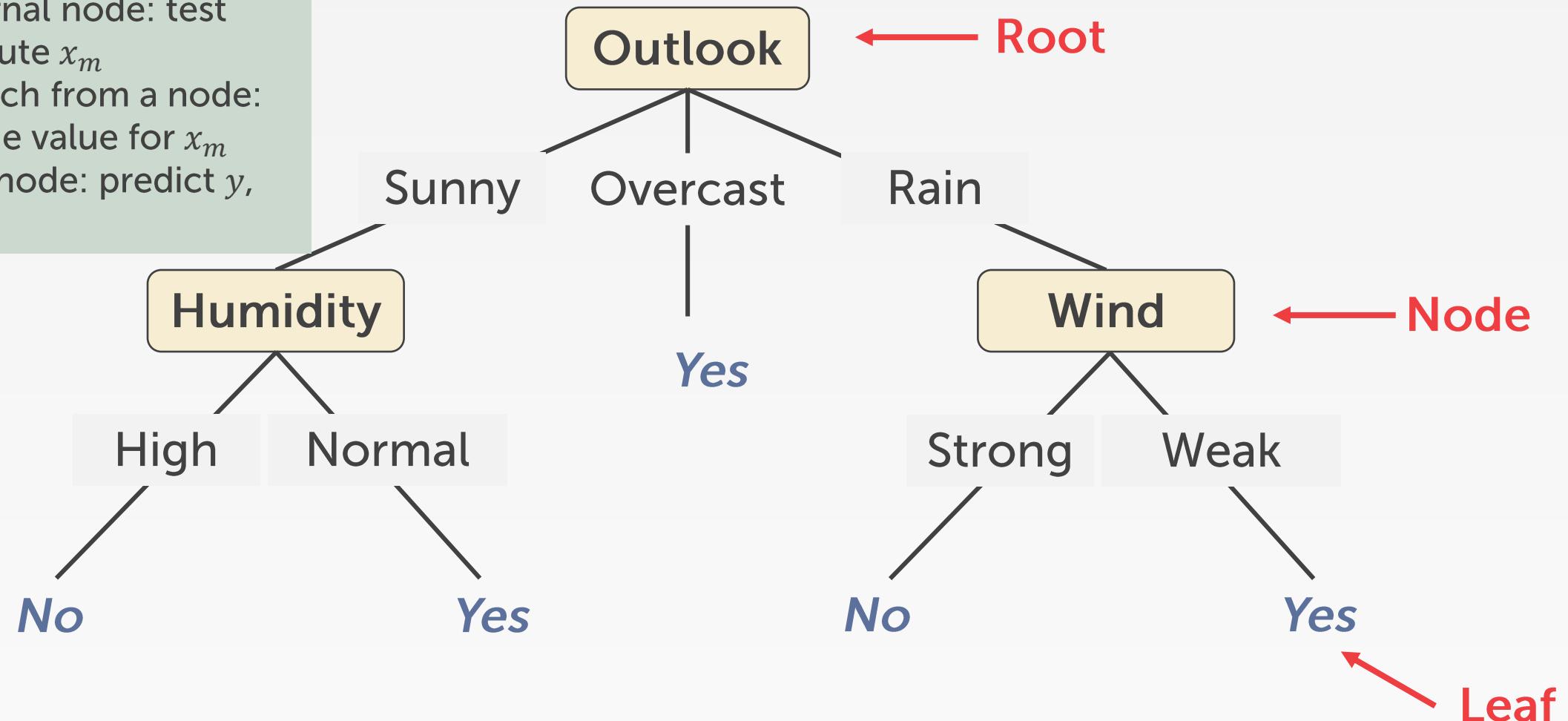
A Decision Tree

- Each internal node: test one attribute x_m
- Each branch from a node: selects one value for x_m



A Decision Tree

- Each internal node: test one attribute x_m
- Each branch from a node: selects one value for x_m
- Each leaf node: predict y , or $p(y|\vec{x})$



Build a Decision Tree?

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m , create a branch labeled with v

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m , create a branch labeled with v
4. Partition node’s data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in \text{node's data} | x_m = v\}, \forall v$

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m , create a branch labeled with v
4. Partition node’s data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in \text{node's data} | x_m = v\}, \forall v$
5. Recurse on each branch (for each v)

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m , create a branch labeled with v
4. Partition node’s data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in \text{node's data} | x_m = v\}, \forall v$
5. Recurse on each branch (for each v)
6. If training examples are perfectly classified, stop and return node with label = majority vote (node’s data).

Basic Algorithm (ID3, C4.5)

```
root = new Node (data = D)
```

```
Return train_tree(root)
```

```
def train_tree(node):
```

1. $X_m \leftarrow$ the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m , create a branch labeled with v
4. Partition node’s data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in \text{node's data} | x_m = v\}, \forall v$
5. Recurse on each branch (for each v)
6. If training examples are perfectly classified, stop and return node with label = majority vote (node’s data).

How do we choose which attribute is best?

Choose the Best Attribute

Key problem: choosing which attribute to split a given set of examples

Some possibilities are:

- Error rate (or accuracy if we want to pick the tree that maximizes the criterion)
- Information gain
- Gini gain
- Random
- ...

Toy Examples (DT with error rates)

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	0
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Toy Examples (DT with error rates)

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	0
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Y	A	B
-	1	0
-	1	0
+	1	0
+	1	0
+	1	1
+	1	1
+	1	1
+	1	1

Decision Tree Algorithm

- ID3 (Iterative Dichotomiser 3)
 - Information gain
- C4.5 (successor of ID3)
 - Gain ratio
- CART (Classification and Regression Tree)
 - Gini impurity ...

Gini

- Gini impurity:

$$G(D) = 1 - \sum_{k=1}^K [p(y = k)]^2$$

- Given an attribute:

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

Gini

- Gini impurity:

$$G(D) = 1 - \sum_{k=1}^K [p(y=k)]^2$$

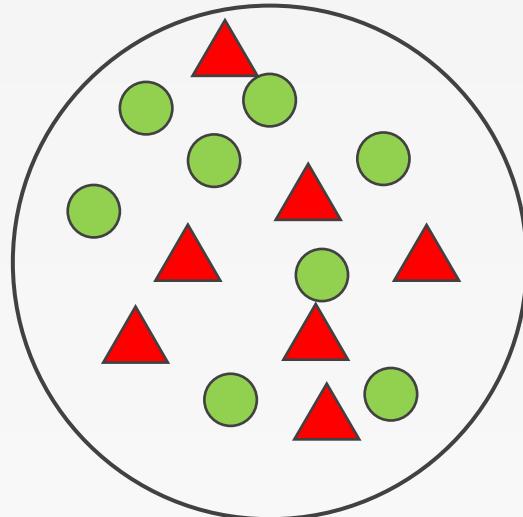
- Given an attribute:

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

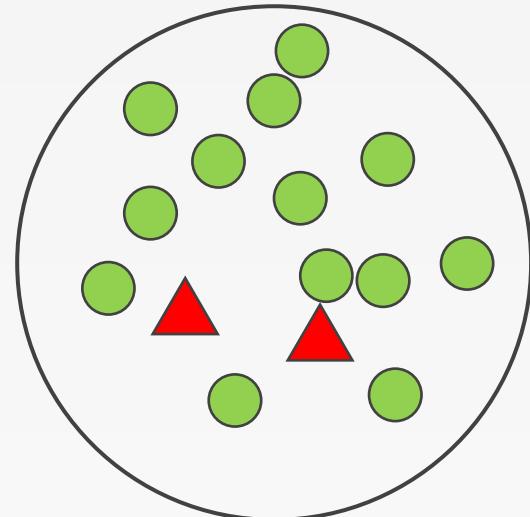
Y	A	B
-	1	0
-	1	0
+	1	0
+	1	0
+	1	1
+	1	1
+	1	1
+	1	1

Impurity

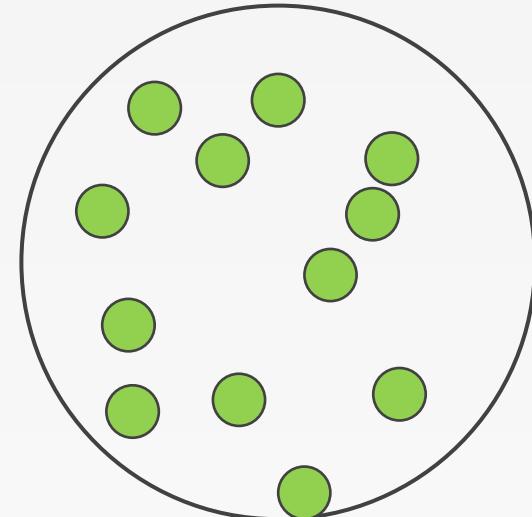
Very Impure Group



Less Impure



Minimum Impurity



Entropy

- Entropy (impurity, disorder) of a set of examples, D, relative to a binary classification is:

$$\text{Entropy}(D) = -p_+ \log(p_+) - p_- \log(p_-)$$

- p_+ is the proportion of positive examples in D and p_- is the proportion of negative examples in D
 - If all samples belong to the same category: Entropy = 0
 - If all samples are equally mixed (0.5, 0.5): Entropy = 1
 - Entropy = Level of uncertainty

Entropy

- If there are more than two classes (K classes)...

$$\text{Entropy}(D) = \sum_{k=1}^K -p_k \log(p_k)$$

p_k is the proportion of D belonging to class k

Information Gain

- The information gain of an attribute X_m is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

- D_v is the subset of D of which attribute X_m has value v
- Partitions of **low** entropy (imbalanced splits) lead to **high** gain

Sample Data

Outlook	Temp	Humidity	Windy	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

An Illustrative Example

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

? examples, ? with no and ? with yes

Before splitting

Entropy = ?

An Illustrative Example

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

14 examples, 5 with no and 9 with yes

Before splitting

$$\text{Entropy} = -\frac{9}{14} \cdot \log_2 \frac{9}{14} - \frac{5}{14} \cdot \log_2 \frac{5}{14} = 0.940$$

Information Gain: Outlook

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Outlook = Sunny: $P(Y=1) = 2/5$, $Entropy_{sunny} = 0.971$

Outlook = Overcast: $P(Y=1) = 4/4$, $Entropy_{Overcast} = 0$

Outlook = Rain: $P(Y=1) = 3/5$, $Entropy_{rain} = 0.971$

Expected entropy:

$$\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 = 0.694$$

Information gain:

$$0.940 - 0.694 = \mathbf{0.246}$$

Information Gain

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Information gain:

Outlook: **0.246**

Temperature:

Humidity:

Wind:

Information Gain

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

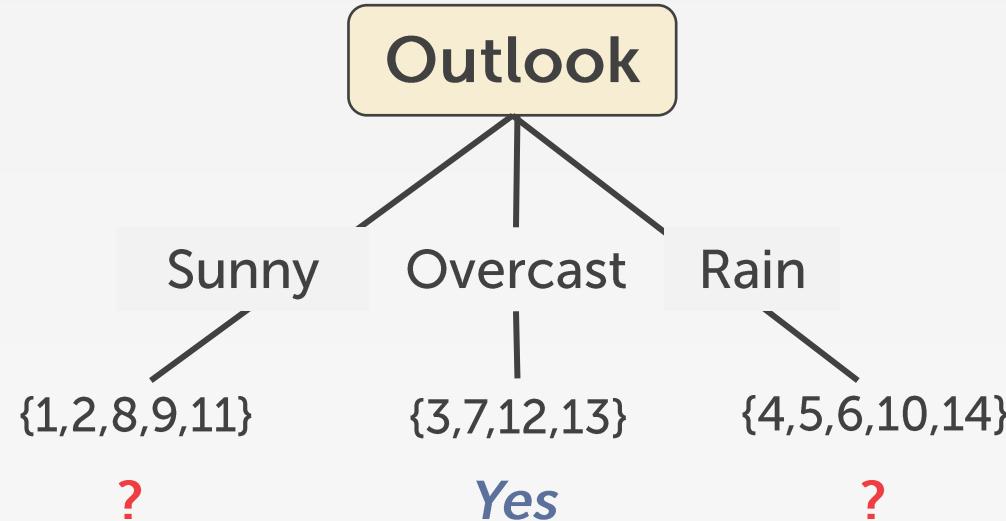
Information gain:

Outlook:	0.246
Temperature:	0.029
Humidity:	0.151
Wind:	0.048

→ Split on Outlook

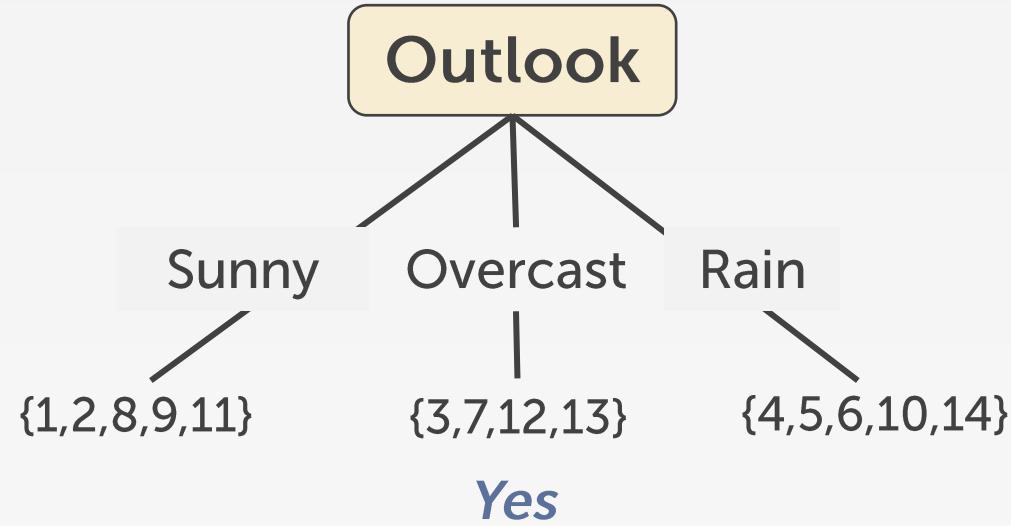
An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
3	O	H	H	W	Yes
4	R	M	H	W	Yes
5	R	C	N	W	Yes
6	R	C	N	S	No
7	O	C	N	S	Yes
8	S	M	H	W	No
9	S	C	N	W	Yes
10	R	M	N	W	Yes
11	S	M	N	S	Yes
12	O	M	H	S	Yes
13	O	H	N	W	Yes
14	R	M	H	S	No



An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
8	S	M	H	W	No
9	S	C	N	W	Yes
11	S	M	N	S	Yes



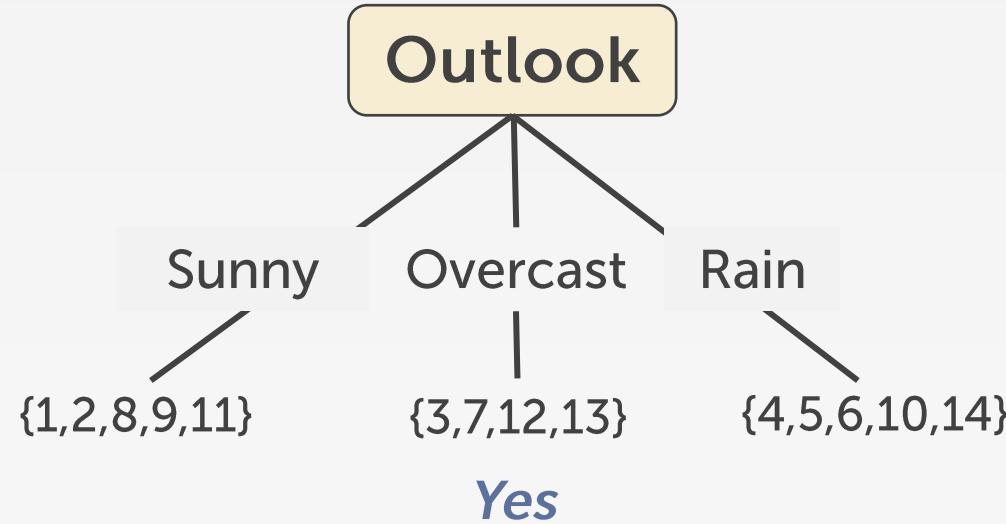
$$Gain(S_{sunny}, Temperature) =$$

$$Gain(S_{sunny}, Humidity) =$$

$$Gain(S_{sunny}, Wind) =$$

An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
8	S	M	H	W	No
9	S	C	N	W	Yes
11	S	M	N	S	Yes



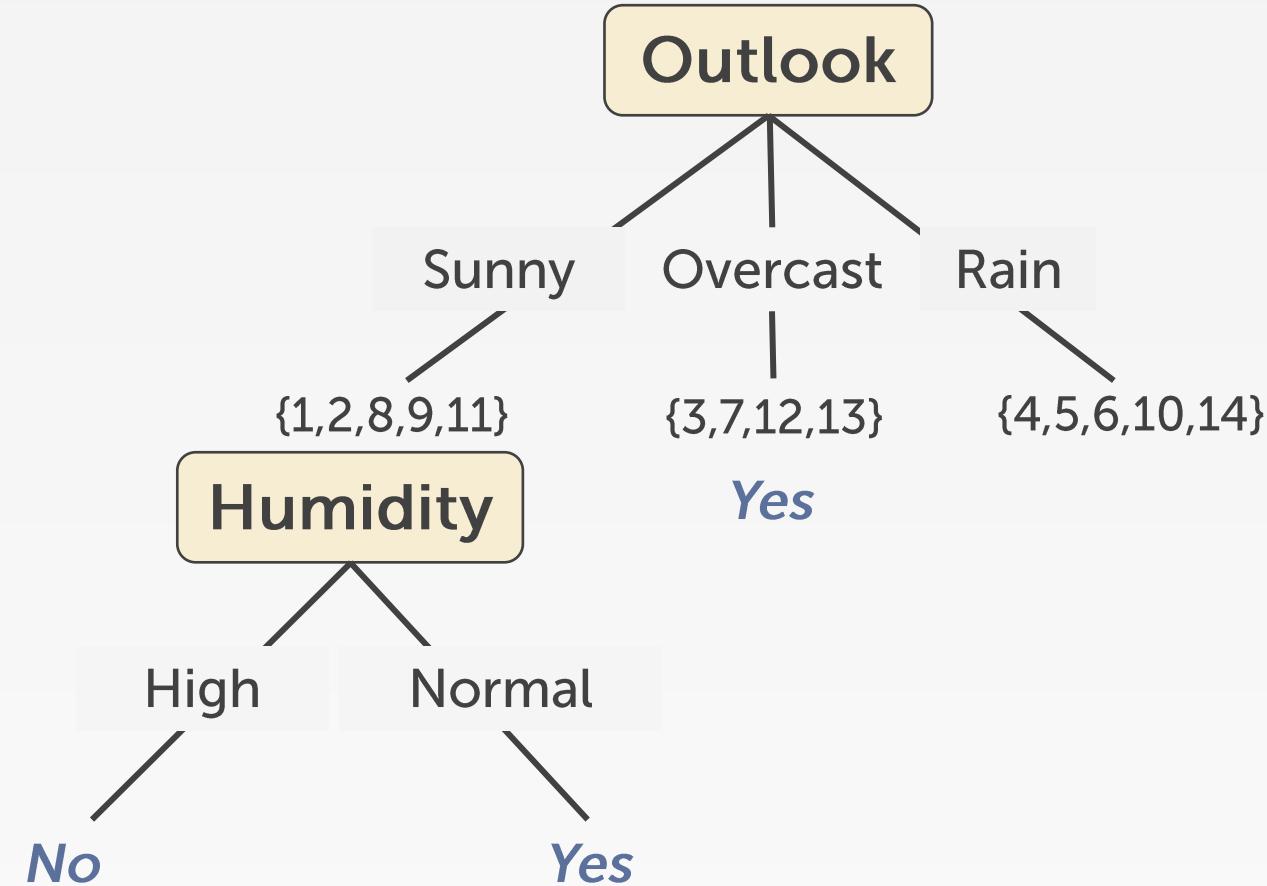
$$Gain(S_{sunny}, Temperature) = 0.97 - \frac{2}{5} \times 0 - \frac{1}{5} \times 0 - \frac{2}{5} \times 1 = 0.57$$

$$Gain(S_{sunny}, Humidity) = 0.97 - \frac{3}{5} \times 0 - \frac{2}{5} \times 0 = \mathbf{0.97}$$

$$Gain(S_{sunny}, Wind) = 0.97 - \frac{3}{5} \times 0.92 - \frac{2}{5} \times 1 = 0.02$$

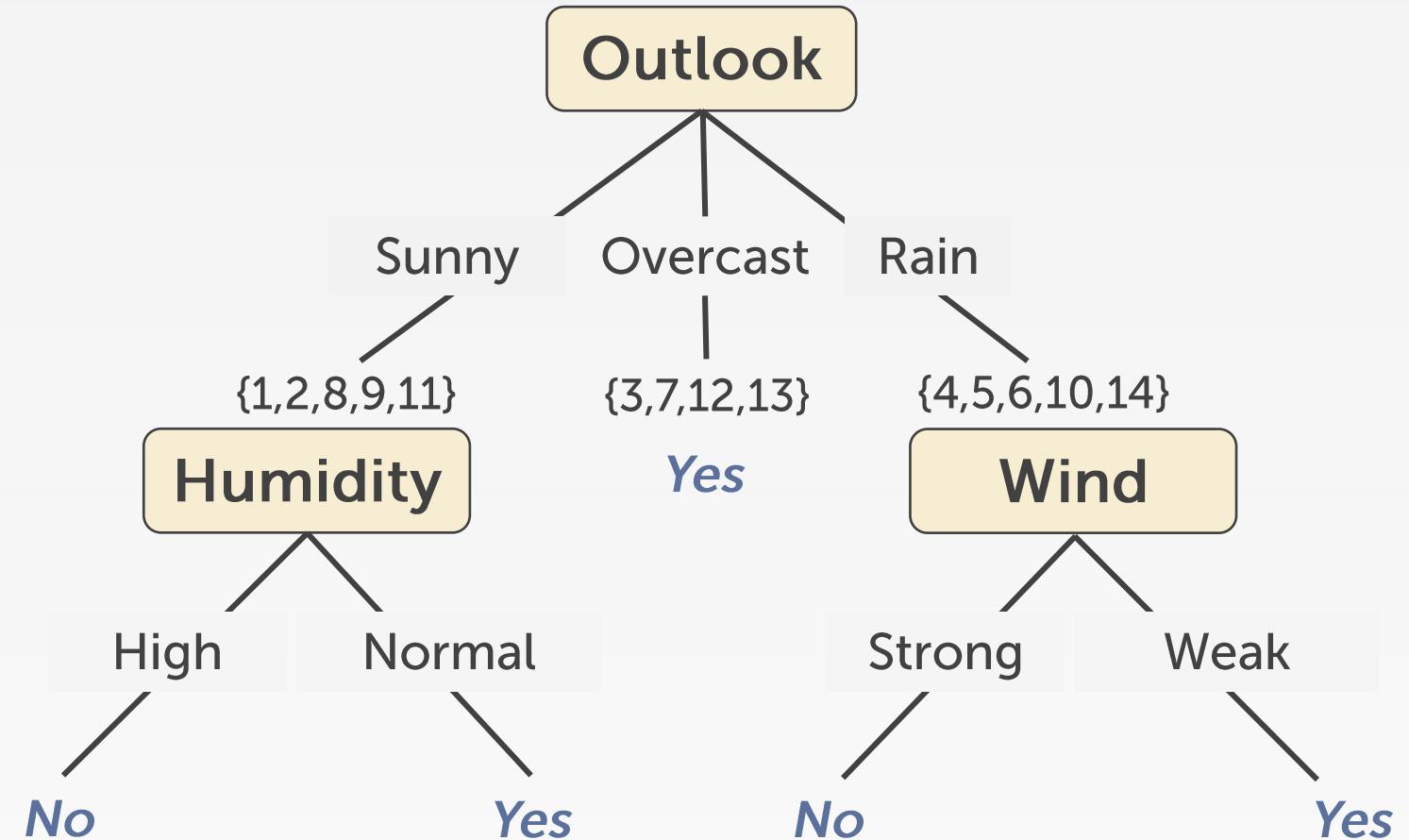
An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
8	S	M	H	W	No
9	S	C	N	W	Yes
11	S	M	N	S	Yes



An Illustrative Example

	O	T	H	W	Play
4	R	M	H	W	Yes
5	R	C	N	W	Yes
6	R	C	N	S	No
10	R	M	N	W	Yes
14	R	M	H	S	No



Gain Ratio (C4.5)

$$GainRatio(D, X_m) = \frac{Gain(D, X_m)}{IV(X_m)}$$

Intrinsic value (or split info):

$$IV(X_m) = -\sum_{v \in Value(X_m)} \frac{|D_v|}{|D|} \log_2 \frac{|D_v|}{|D|}$$

Comparison

Features	ID3	C4.5	CART
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression

Comparison

Features	ID3	C4.5	CART
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression

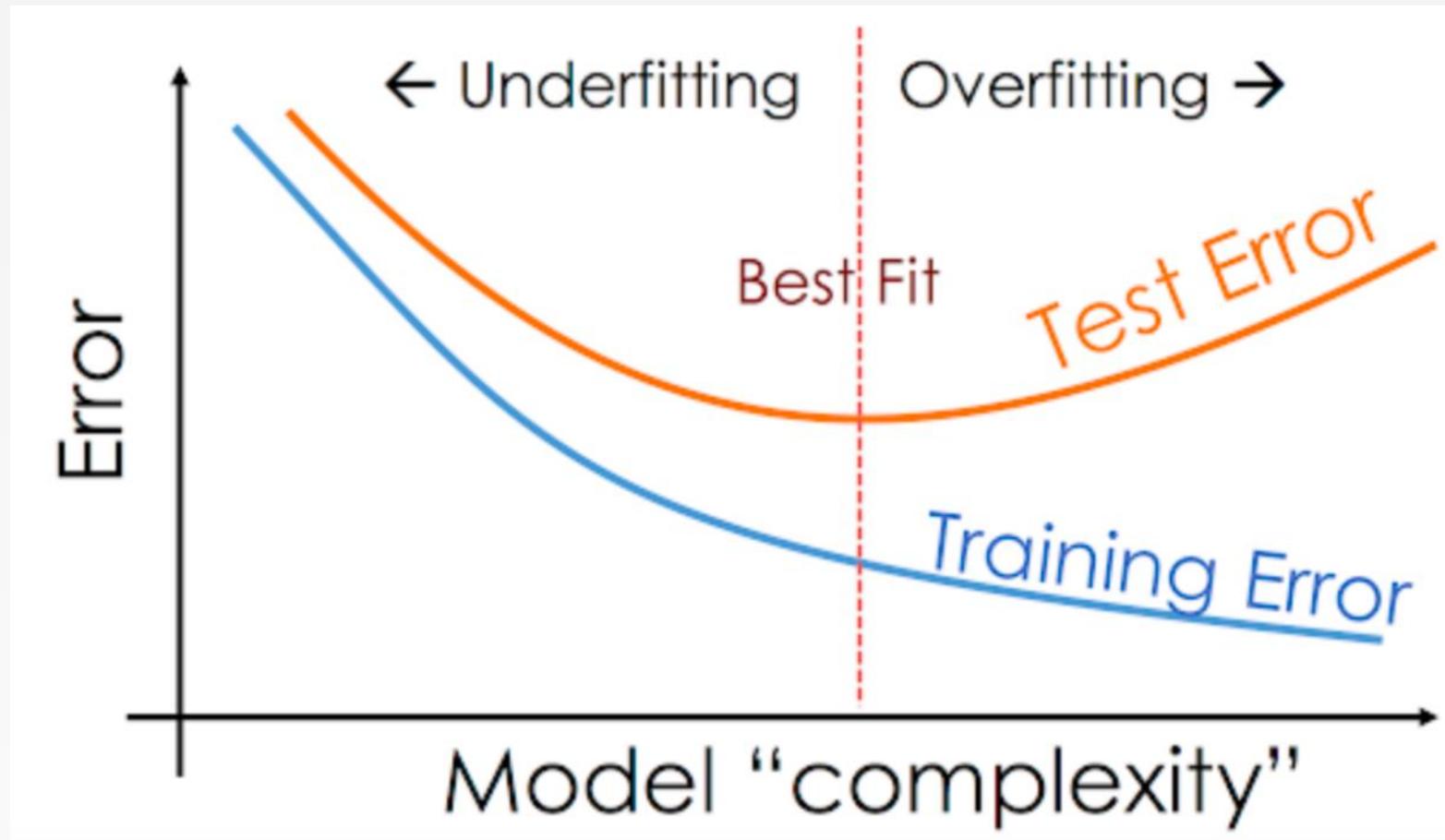
The smallest decision tree
that correctly classifies all of
the training examples is best

Occam's Razor: Prefer the simplest hypothesis that
explains the data

Why Overfitting?

- Too much noise in the training data
 - Two examples have same attribute/value pairs, but different classifications
 - Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
 - The instance was labeled incorrectly (+ instead of -)
- Too much variance in the training data
 - Training data is not a representative sample of the instance space
 - We split on features that are actually irrelevant
 - Too little training data

Overfitting

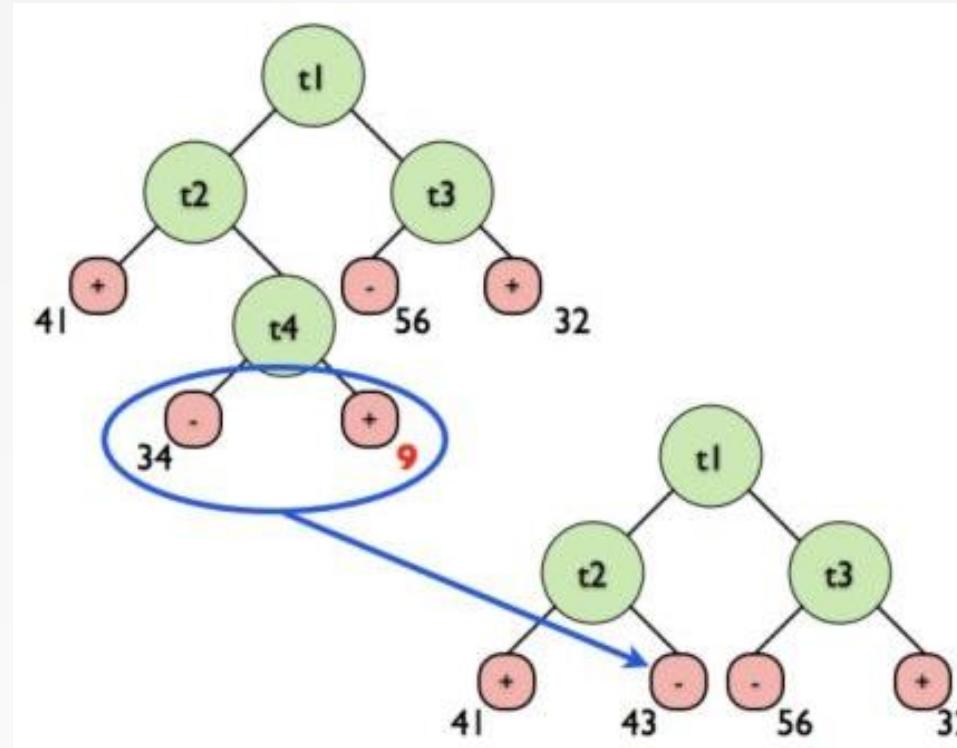


Avoid Overfitting

- How can we avoid overfitting?
 - Acquire more training data
 - Remove irrelevant attributes (manual process – not always possible)
 - Do not grow tree beyond some maximum depth
 - Do not split if splitting criterion (e.g. information gain) is below some threshold
 - Stop growing when data split is not statistically significant
 - Grow the full tree, then post-prune

Pruning a Tree

- Prune = remove leaves and assign majority labels of the parent to all items



Pruning

- Two basic approaches
 - **Pre-pruning** stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices
 - Set a depth cut-off (maximum tree depth) a priori
 - Set a minimum number of data points for each node
 - Stop growing if a split is not statistically significant
 - **Post-pruning** grow the full tree and then remove nodes that seem not to have sufficient evidence

Post-pruning

- Split data into training set and validation set
- Train a tree to classify training set as well as possible
- Do until further pruning reduces validation set accuracy:
 - 1. For each internal tree node, consider making it a leaf node (pruning the tree below it)
 - 2. Greedily choose the above pruning step that best improves error over validation set
- Produces smallest version of the most accurate pruned tree

Continuous Attributes

- Discretize real-values attributes into ranges: e.g., big, medium, small
- Alternatively, based on the thresholds: $A < c$ vs. $A \geq c$
- How to find the split **with the highest gain?** (Bi-partition in C4.5)

Temperature	40	48	60	72	80	90
Play Tennis	No	No	Yes	Yes	Yes	No

Missing Values

- Data D and attribute a
- \tilde{D} is the data that do not have missing values on a
- Value of a : $\{a^1, a^2, \dots, a^V\}$
- $\tilde{D}^\nu, \tilde{D}_k$ where $k = 1, 2, \dots, |Y|$
- $\rho = \frac{\sum_{x \in \tilde{D}} w_x}{\sum_{x \in D} w_x}; \quad \tilde{p}_k = \frac{\sum_{x \in \tilde{D}_k} w_x}{\sum_{x \in \tilde{D}} w_x}; \quad \tilde{r}_\nu = \frac{\sum_{x \in \tilde{D}^\nu} w_x}{\sum_{x \in \tilde{D}} w_x}$
- $Gain(D, a) = \rho \times Gain(\tilde{D}, a) = \rho \times (Ent(\tilde{D}) - \sum_{\nu=1}^V \tilde{r}_\nu Ent(\tilde{D}^\nu))$

$$Ent(\tilde{D}) = - \sum_{k=1}^{|Y|} \tilde{p}_k \log_2 \tilde{p}_k$$
- *Split info* is calculated the same as before but with the missing values considered a separate state that an attribute can take.

Missing Value

Day	Outlook	Temp	Humidity	Windy	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	?	Hot	High	Strong	No
D3	?	?	High	?	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	?	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	?	Mild	High	?	No
D9	?	Cool	Normal	Weak	Yes
D10	?	?	Normal	?	Yes
D11	?	Mild	Normal	?	Yes
D12	Overcast	Mild	?	Strong	Yes
D13	Overcast	Hot	?	Weak	Yes
D14	Rain	Mild	High	Strong	No

Comparison

Features	ID3	C4.5	CART
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression

CART (Regression)

- $R_1 = \{x|x \leq s\}, R_2 = \{x|x > s\}$
- $c_1 = \frac{1}{N_1} \sum_{x_i \in R_1} y_i, c_2 = \frac{1}{N_2} \sum_{x_i \in R_2} y_i$
- $\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2} (y_i - c_2)^2 \right]$
- j^{th} variable; partition s

Model Selection

Model Selection

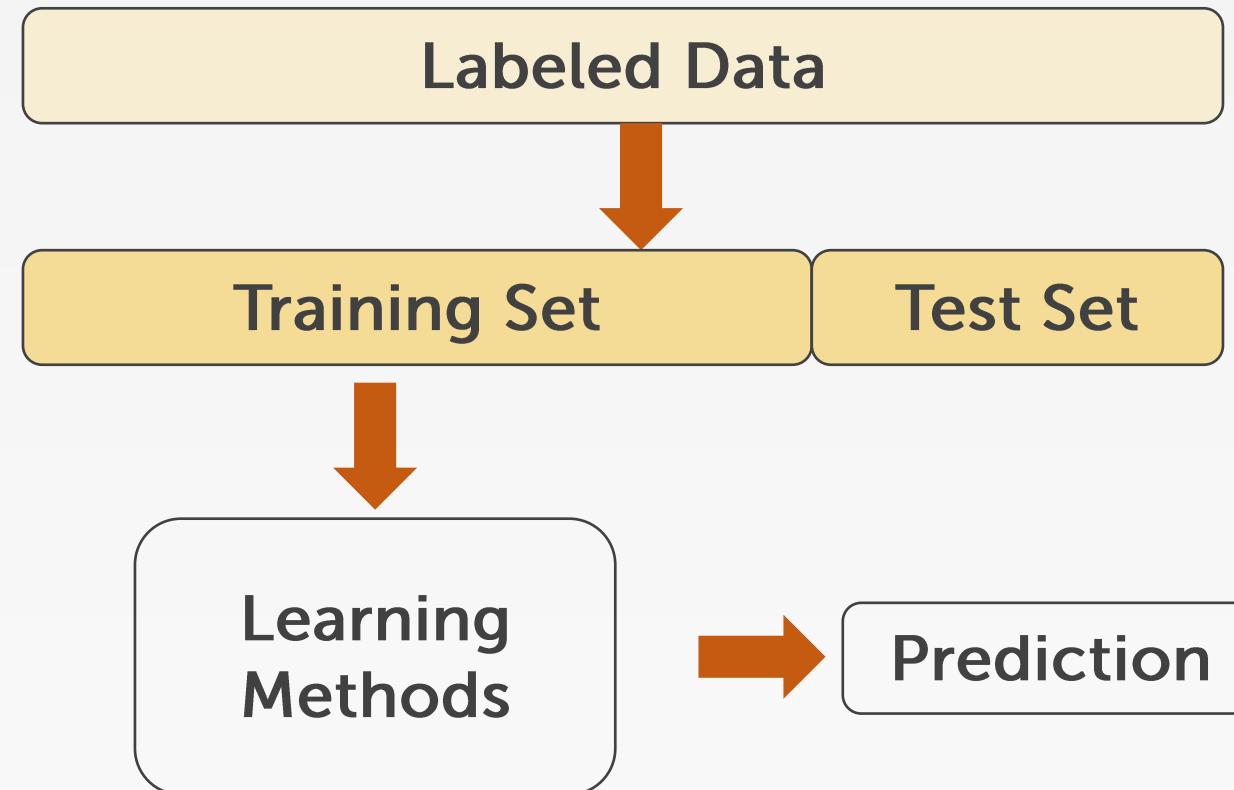
- Machine Learning
 - **Def:** (loosely) a model defines the hypothesis space over which learning performs its search
 - **Def:** model parameters are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
 - **Def:** the learning algorithm defines the data-driven search over the hypothesis space (i.e. search for good parameters)
 - **Def:** hyperparameters are the tunable aspects of the model, that the learning algorithm does not select

- Example

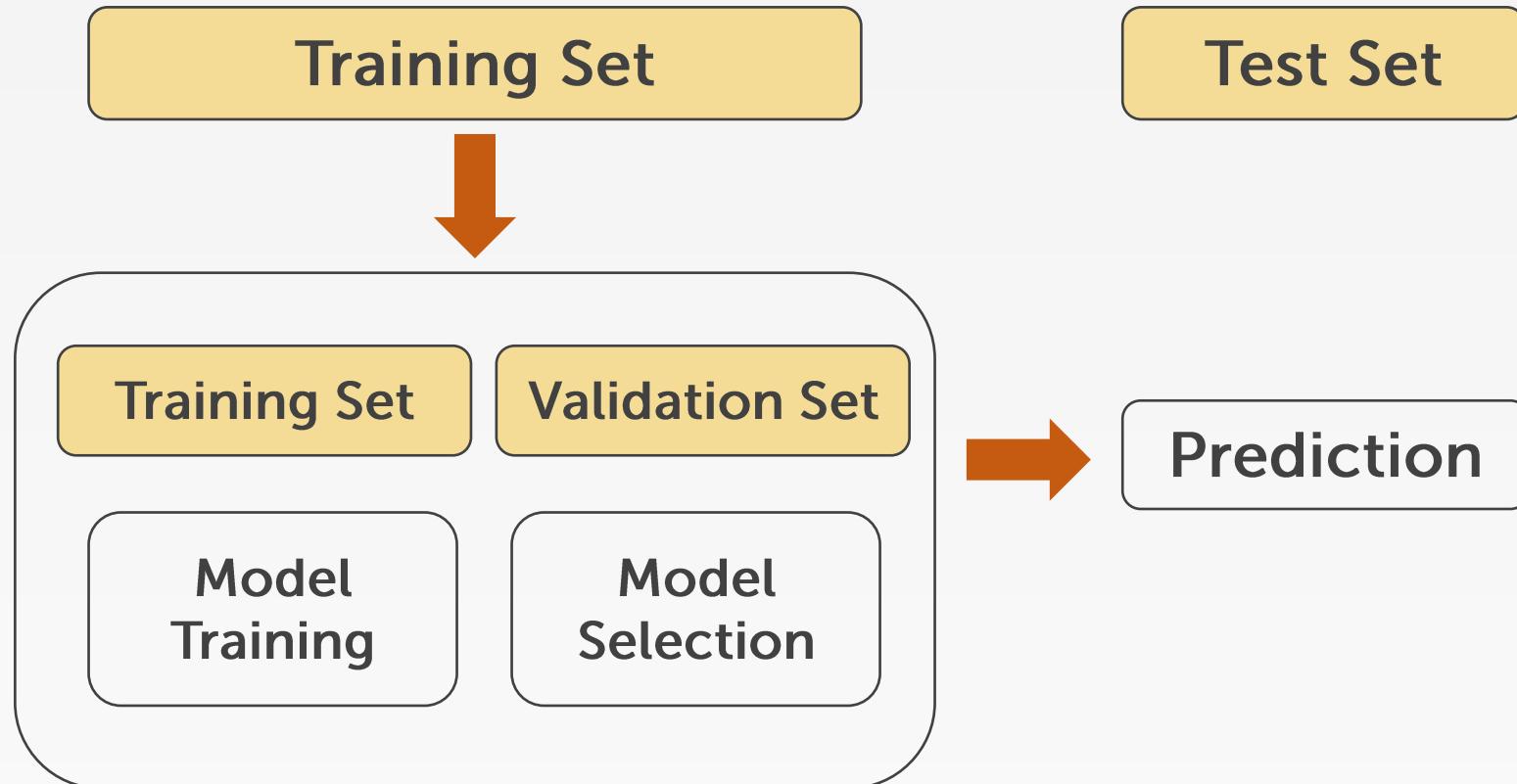
- model = set of all possible trees, possibly restricted by some hyperparameters (e.g. max depth)
- parameters = structure of a specific decision tree
- learning algorithm = ID3, CART, etc.
- hyperparameters = max depth, threshold for splitting criterion, etc.

Cross Validation

Train a Supervised Model



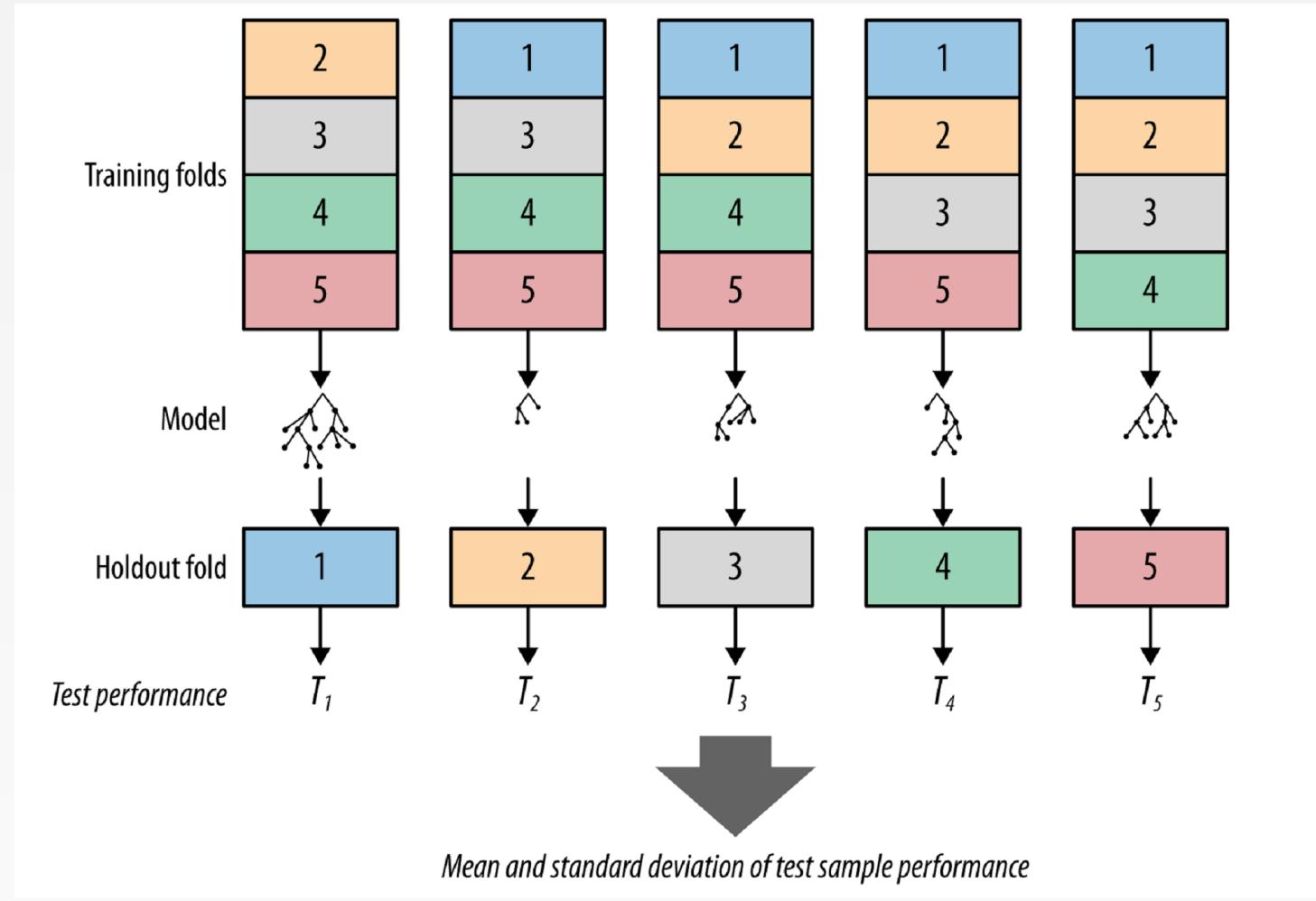
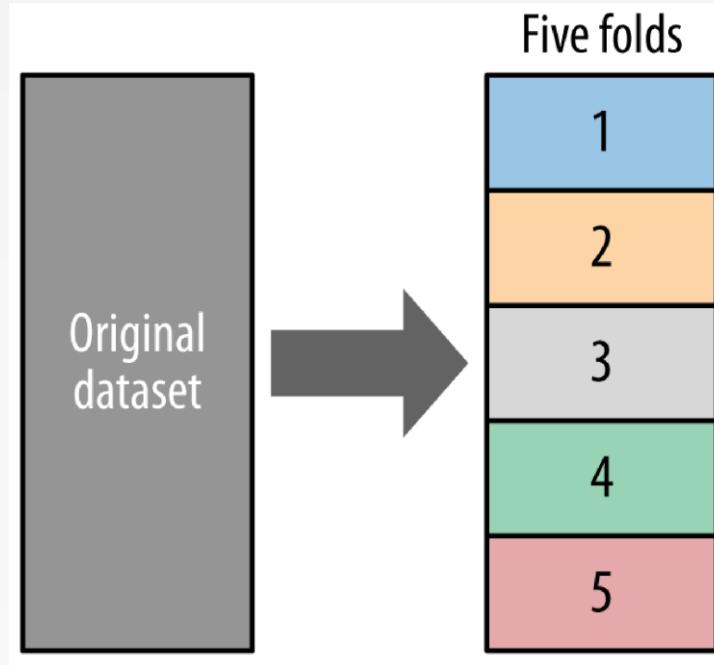
Select a Model



Limitation of a Single Partition

- We may not have enough data to have both sufficiently large training and test set:
 - A large test set: more reliable estimate of prediction accuracy
 - A large training set: more representative for learning process
- Solution: **Cross Validations**
 - Sometime also called “N-fold Cross-Validation”
 - N: number of folds used

N-fold Cross Validation



Grid Search

- Approaches to hyperparameter tuning:
 - **Grid search**
 - Random search
 - Bayesian optimization
 - ...

Grid Search

	C=0.001	C=0.01	C=10
$\alpha = 0.001$	M (C=0.001, $\alpha = 0.001$)	M (C=0.01, $\alpha = 0.001$)	M (C=10, $\alpha = 0.001$)
$\alpha = 0.1$	M (C=0.001, $\alpha = 0.1$)	M (C=0.01, $\alpha = 0.1$)	M (C=10, $\alpha = 0.1$)
...			
$\alpha = 100$	M (C=0.001, $\alpha = 100$)	M (C=0.01, $\alpha = 100$)	M (C=10, $\alpha = 100$)

Grid Search with CV

- Manually set a grid of discrete hyperparameter values.
- Set a metric for scoring model performance.
- Search exhaustively through the grid.
- For each set of hyperparameters, evaluate each model's CV score.
- The optimal hyperparameters are those of the model achieving the best CV score.

Tuning is expensive

- Hyperparameter tuning:
 - Computationally expensive,
 - Sometime leads to very slight improvement
- Weight the impact of tuning on the whole project

Model Evaluation (Binary Classification)

Accuracy

Number of correct predictions/Data size

Is Accuracy Enough?

Is Accuracy Enough?

- Potential Issues:
 - There is a large class skew
 - E.g., Is 98% accuracy good if 97% of the instances are negative?

Is Accuracy Enough?

- Potential Issues:
 - There is a large class skew
 - E.g., Is 98% accuracy good if 97% of the instances are negative?
 - There are differential misclassification costs
 - E.g., Medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

Is Accuracy Enough?

- Potential Issues:
 - There is a large class skew
 - E.g., Is 98% accuracy good if 97% of the instances are negative?
 - There are differential misclassification costs
 - E.g., Medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
- Require better and more evaluation measurements!

Confusion Matrix

A Two-class Problem

		Predicted class	
		Positive	Negative
Actual class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Additional Metrics

Precision

% of those cases the model classified as positive that are correct

$$\frac{TP}{TP + FP}$$

Measures model's accuracy for those it classified as the class of interest

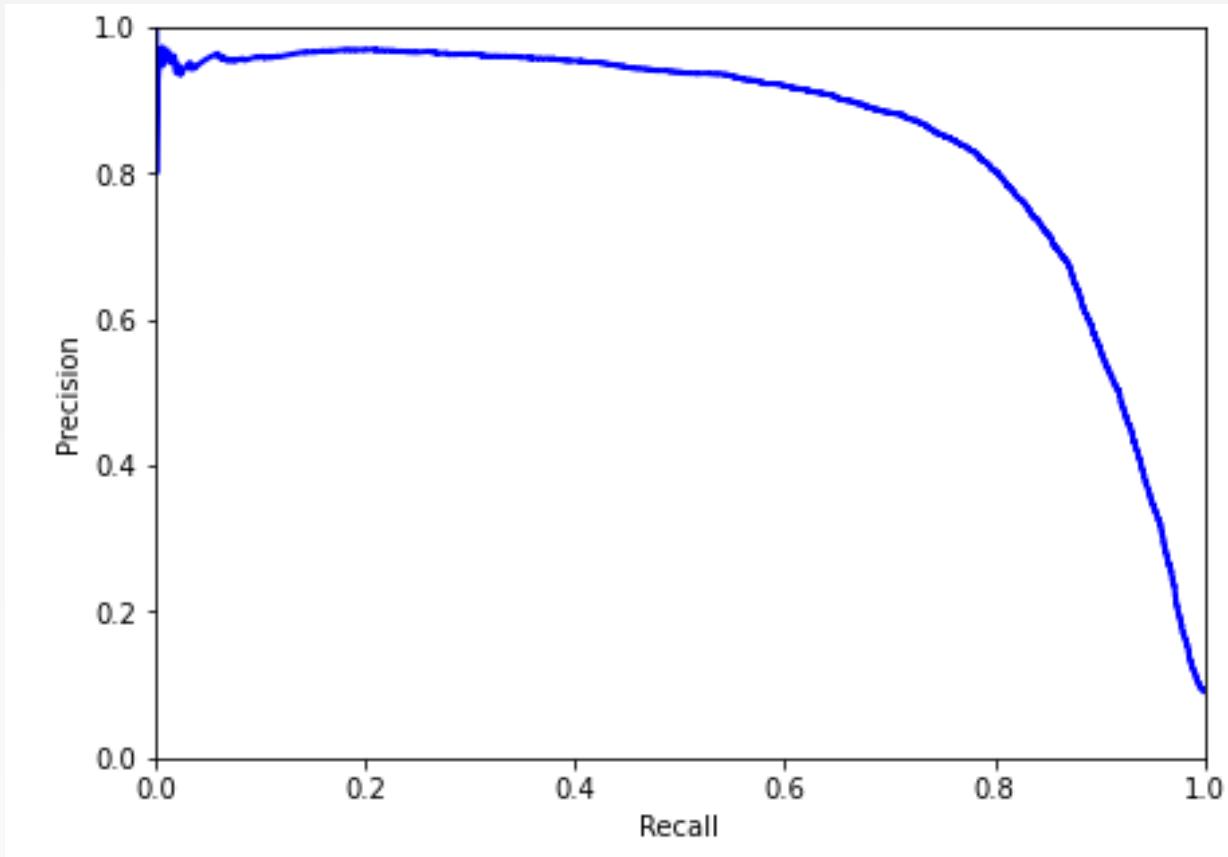
Recall

(sensitivity): % of actual positive class (class of interest) correctly classified

$$\frac{TP}{TP + FN}$$

Measures how well the model detect the class of interest

Precision-Recall Curve



Additional Metrics

Precision

% of those cases the model classified as positive that are correct

$$\frac{TP}{TP + FP}$$

Measures model's accuracy for those it classified as the class of interest

Recall

(sensitivity): % of actual positive class (class of interest) correctly classified

$$\frac{TP}{TP + FN}$$

Measures how well the model detect the class of interest

Specificity

% of negative class correctly classified

$$\frac{TN}{TN + FP}$$

Measures how well the model rules out the other class correctly

F_1 score

$$2 \times (Precision \times recall) / (Precision + recall)$$

Practice

- Scenario 1: Mining systems to identify sites
 - Classifier 1: low recall, high precision
 - Classifier 2: high recall, low precision

Practice

- Scenario 1: Mining systems to identify sites
 - Classifier 1: low recall, high precision
 - Classifier 2: high recall, low precision
- Scenario 2: Identify terrorists
 - Classifier 1: low recall, high precision
 - Classifier 2: high recall, low precision

→ expensive

Illustrative Examples

Model I

		Predicted Class	
Actual Class		1 (Yes)	0 (No)
(Yes) 1	500	0	
(No) 0	200	300	

Model II

		Predicted Class	
Actual Class		1 (Yes)	0 (No)
(Yes) 1	300	200	
(No) 0	0	500	

An Illustrative Example I

		Predicted Class	
Actual Class		1 (Yes)	0 (No)
(Yes) 1		500	0
(No) 0		200	300

Sensitivity (recall) – how well it detects class 1

Specificity – how well it rules out class 0

Precision – accuracy when identifying class 1

F_1 – balance between precision and recall

An Illustrative Example I

$$\begin{aligned} Accuracy &= \frac{500 + 300}{1000} \\ &= 80\% \end{aligned}$$

		Predicted Class	
		1 (Yes)	0 (No)
		(Yes) 1	0
		500	0
		200	300

Sensitivity (recall) – how well it detects class 1

Specificity – how well it rules out class 0

Precision – accuracy when identifying class 1

F_1 – balance between precision and recall

An Illustrative Example I

$$\begin{aligned} Accuracy &= \frac{500 + 300}{1000} \\ &= 80\% \end{aligned}$$

		Predicted Class	
		1 (Yes)	0 (No)
Actual Class	(Yes) 1	500	0
	(No) 0	200	300

Sensitivity (recall) – how well it detects class 1 $500/(500+0) = 1.0$

Specificity – how well it rules out class 0 $300/(200+300) = 0.6$

Precision – accuracy when identifying class 1 $500/(500+200) = 0.7143$

F_1 – balance between precision and recall

$$2 \times \frac{0.7143 * 1}{0.7143 + 1} = 0.8333$$

An Illustrative Example II

$$\begin{aligned} Accuracy &= \frac{500 + 300}{1000} \\ &= 80\% \end{aligned}$$

		Predicted Class	
		1 (Yes)	0 (No)
Actual Class	(Yes) 1	300	200
	(No) 0	0	500

Sensitivity (recall) – how well it detects class 1 $300/(300+200) = 0.6$

Specificity – how well it rules out class 0 $500/500 = 1$

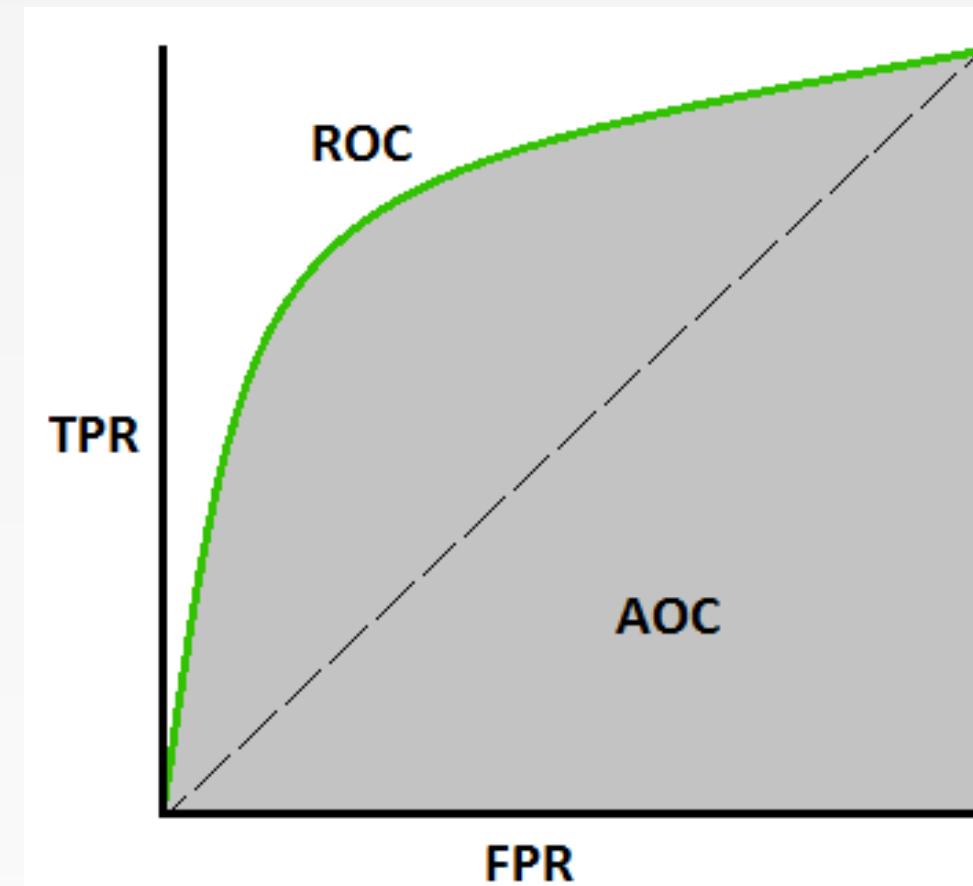
Precision – accuracy when identifying class 1 $300/300 = 1$

F_1 – balance between precision and recall $2 \times \frac{0.6 * 1}{0.6 + 1} = 0.75$

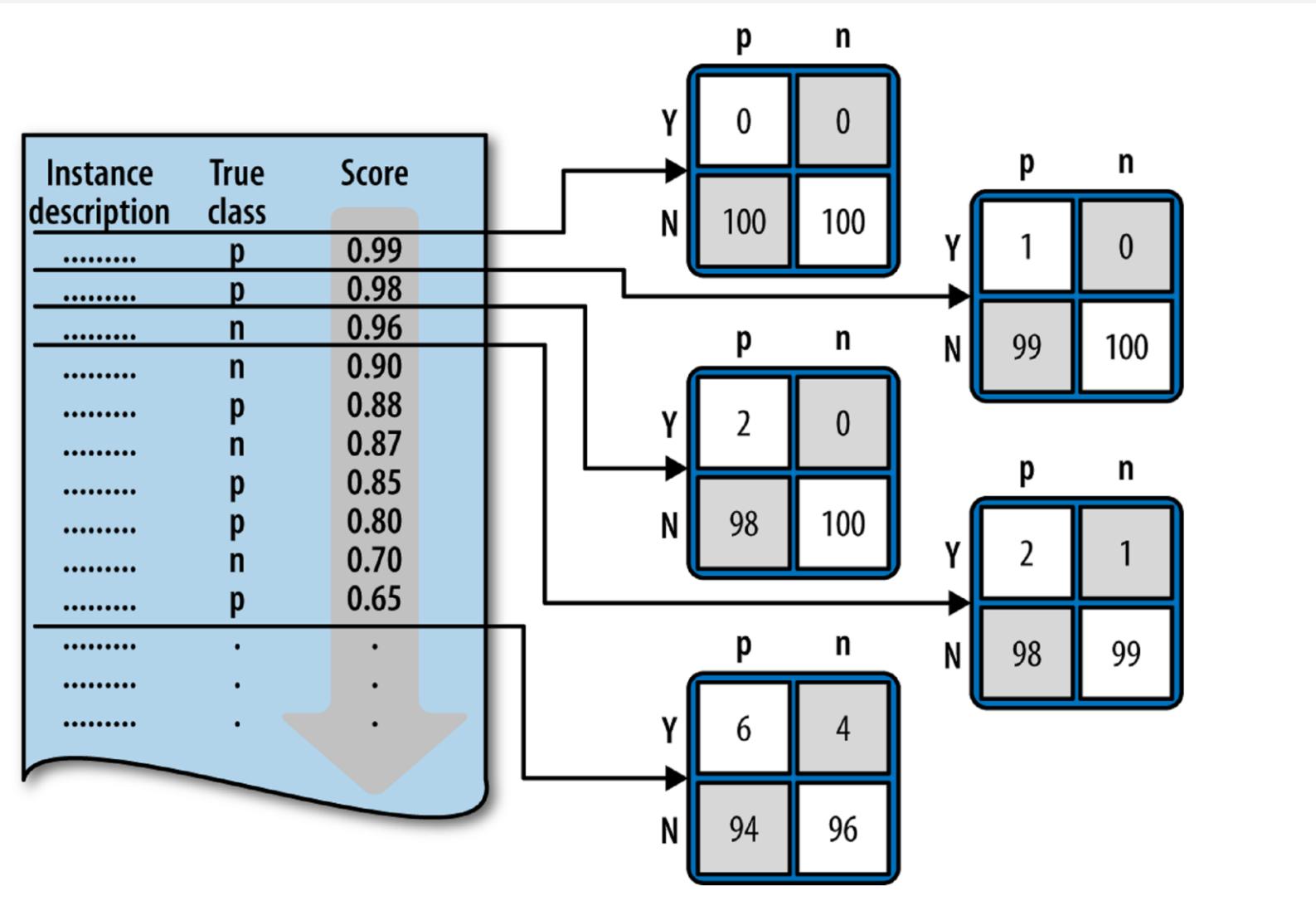
ROC

ROC Curves

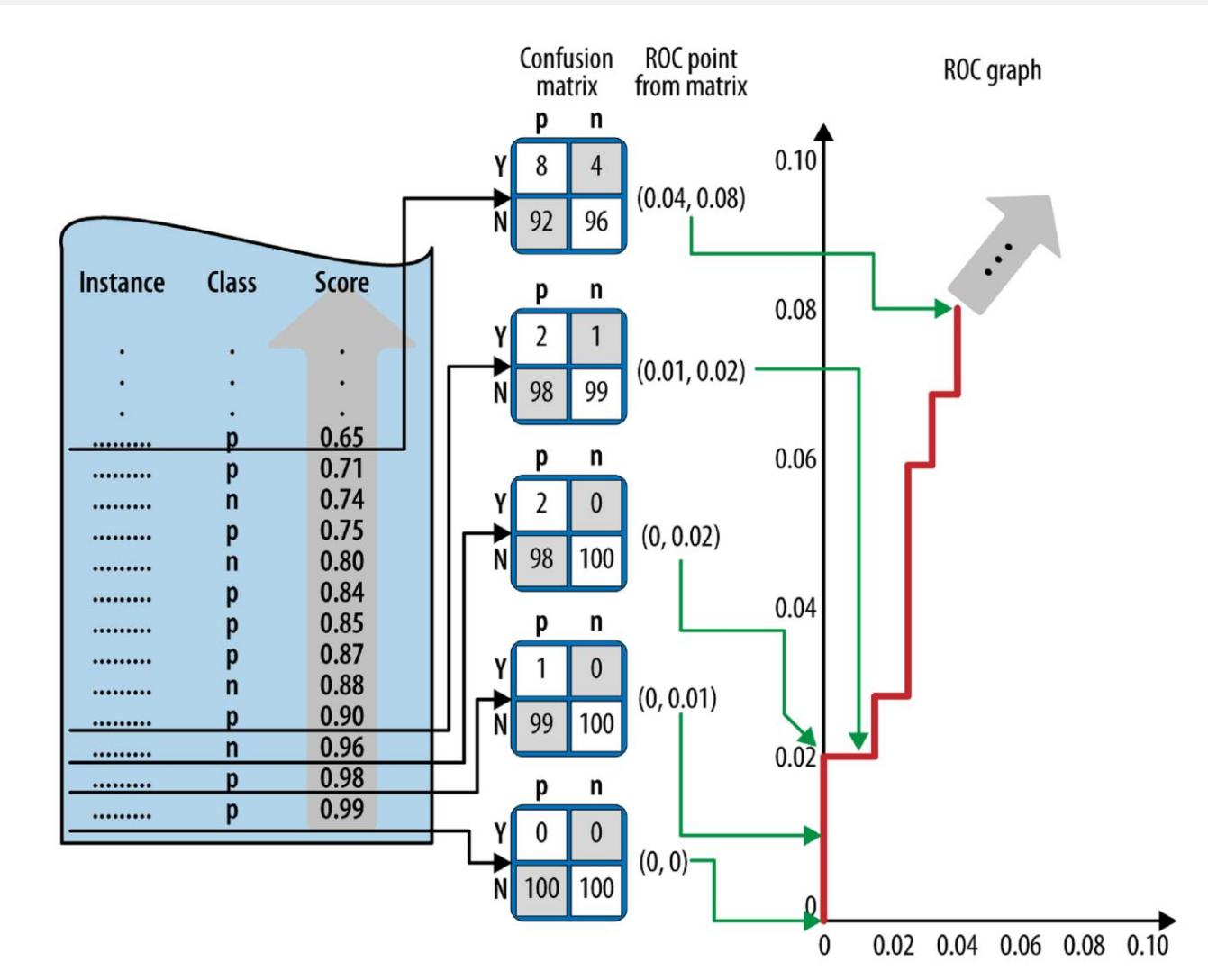
- A Receiver Operating Characteristic (ROC) curve plots the TP-rate (sensitivity) vs. the FP-rate (1-specificity) as a threshold on the confidence of an instance being positive is varied



Generate a ROC curve



Generate a ROC curve



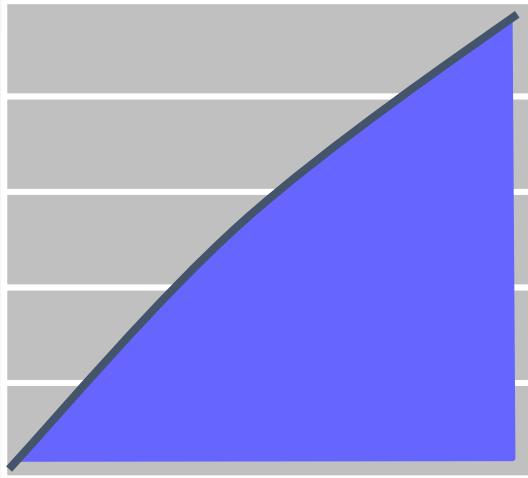
Algorithm Detail

- sort test-set predictions according to probability that each instance is positive
- step through sorted list from high to low probability
 - locate a threshold between instances with opposite classes (keeping instances with the same confidence value on the same side of threshold)
 - compute TPR, FPR for instances above threshold
 - output (FPR, TPR) coordinate

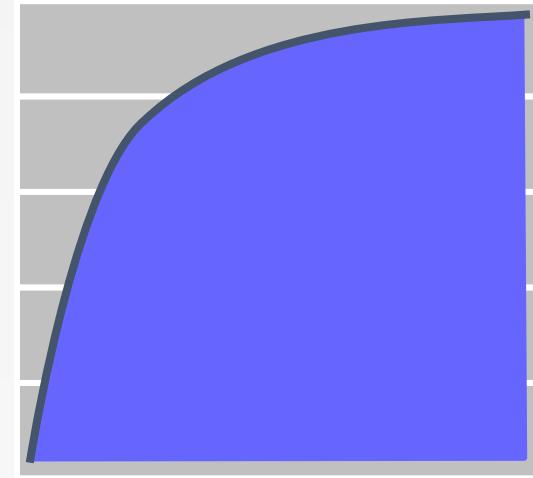
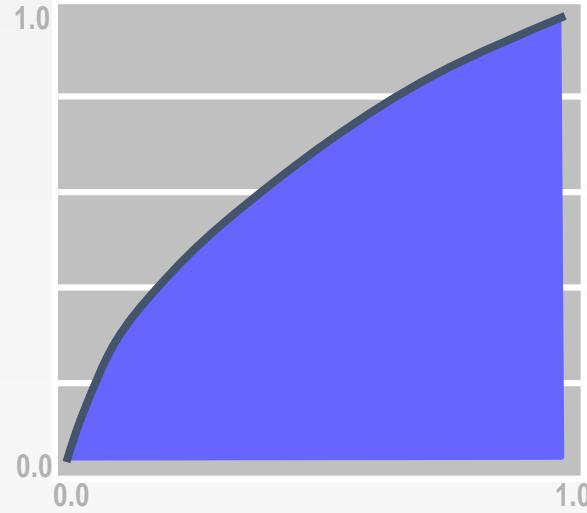
Area Under the ROC Curve (AUC)

- The area under a classifier's curve expressed as a fraction of the unit square
- The AUC is useful when a single number is needed to summarize performance
 - A ROC curve provides more information than its area

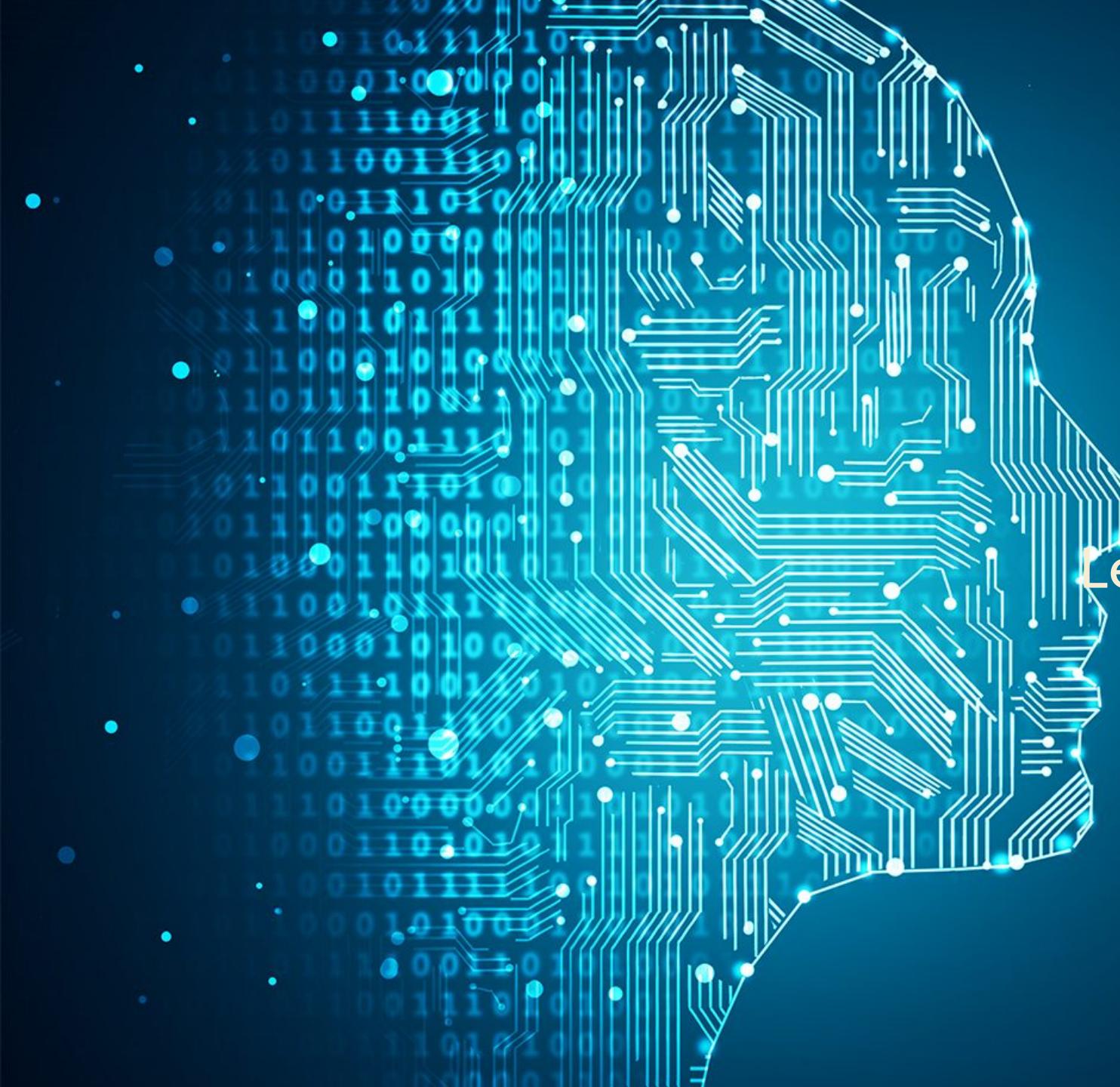
AUC



weak model
ROC Index < 0.6



strong model
ROC Index > 0.7



机器学习与人工智能 Machine Learning and Artificial Intelligence

Lecture 5 SVM and Naïve Bayes

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

Missing Value in DT

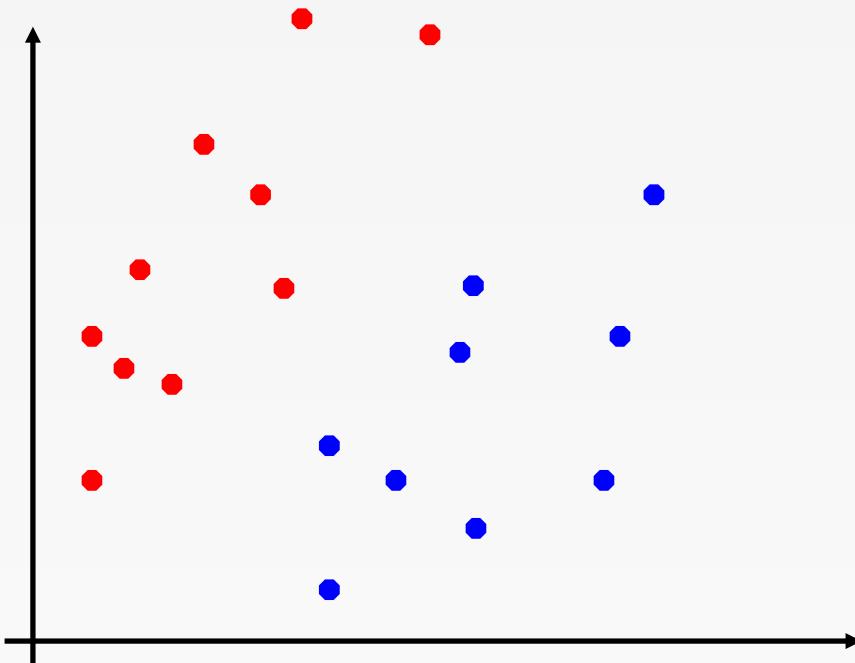
- Data D and attribute a
- \tilde{D} is the data that do not have missing values on a
- Value of a : $\{a^1, a^2, \dots, a^V\}$
- $\tilde{D}^\nu, \tilde{D}_k$ where $k = 1, 2, \dots, |Y|$
- $\rho = \frac{\sum_{x \in \tilde{D}} w_x}{\sum_{x \in D} w_x}; \quad \tilde{p}_k = \frac{\sum_{x \in \tilde{D}_k} w_x}{\sum_{x \in \tilde{D}} w_x}; \quad \tilde{r}_\nu = \frac{\sum_{x \in \tilde{D}^\nu} w_x}{\sum_{x \in \tilde{D}} w_x}$
- $Gain(D, a) = \rho \times Gain(\tilde{D}, a) = \rho \times (Ent(\tilde{D}) - \sum_{\nu=1}^V \tilde{r}_\nu Ent(\tilde{D}^\nu))$

$$Ent(\tilde{D}) = - \sum_{k=1}^{|Y|} \tilde{p}_k \log_2 \tilde{p}_k$$
- *Split info* is calculated the same as before but with the missing values considered a separate state that an attribute can take.

Support Vector Machine

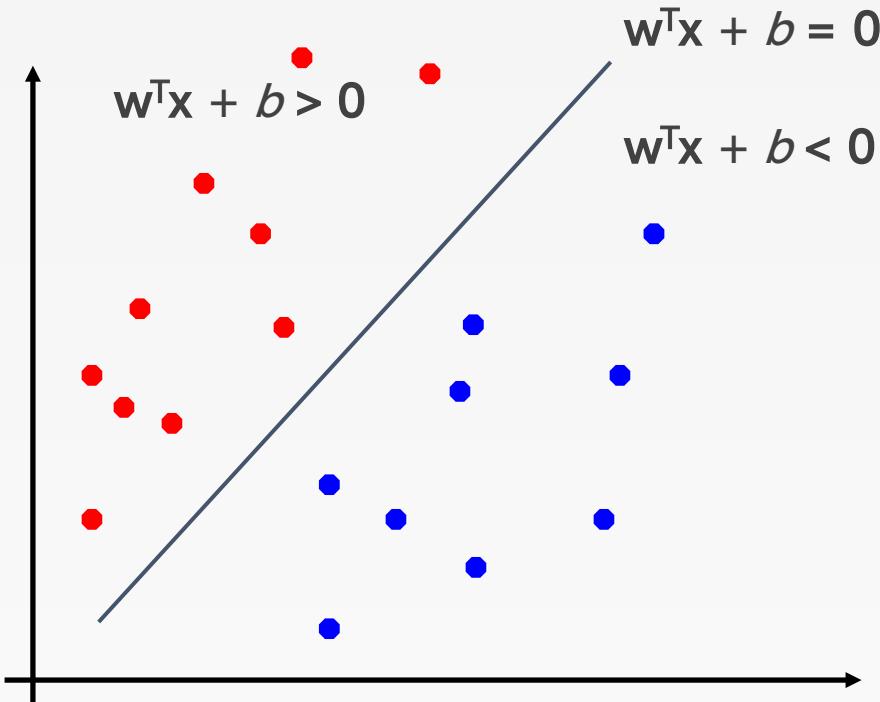
Linear SVM Classification

- Binary classification can be viewed as the task of separating classes in feature space:



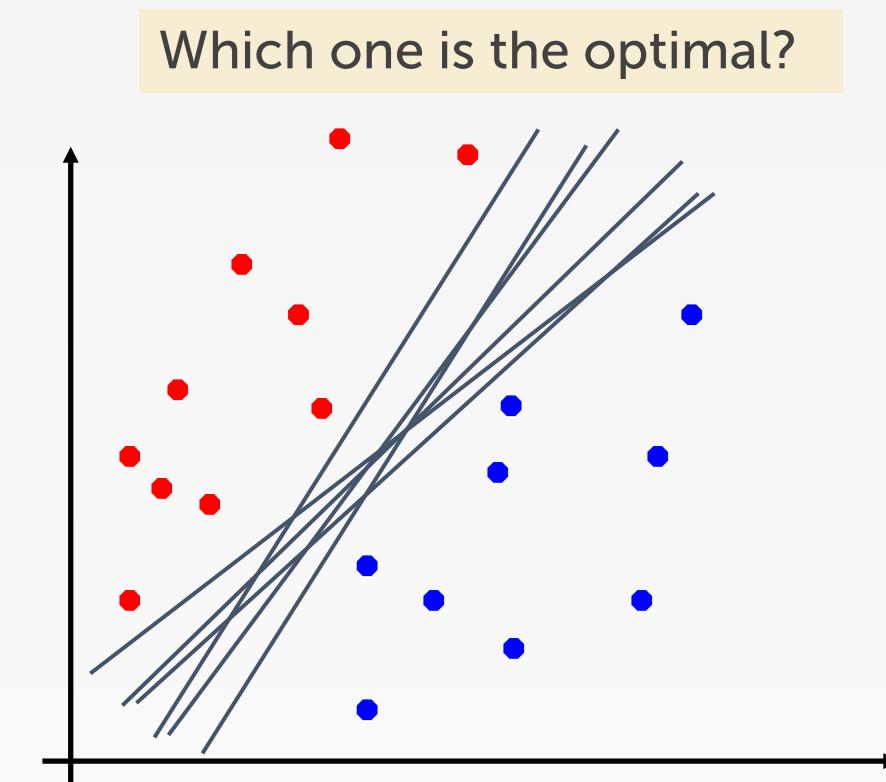
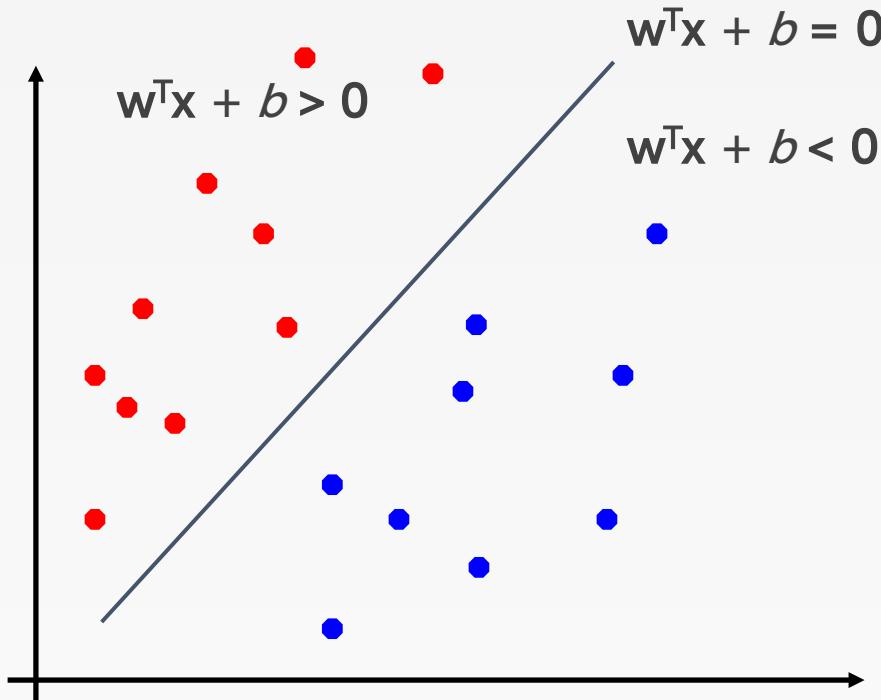
Linear SVM Classification

- Binary classification can be viewed as the task of separating classes in feature space:



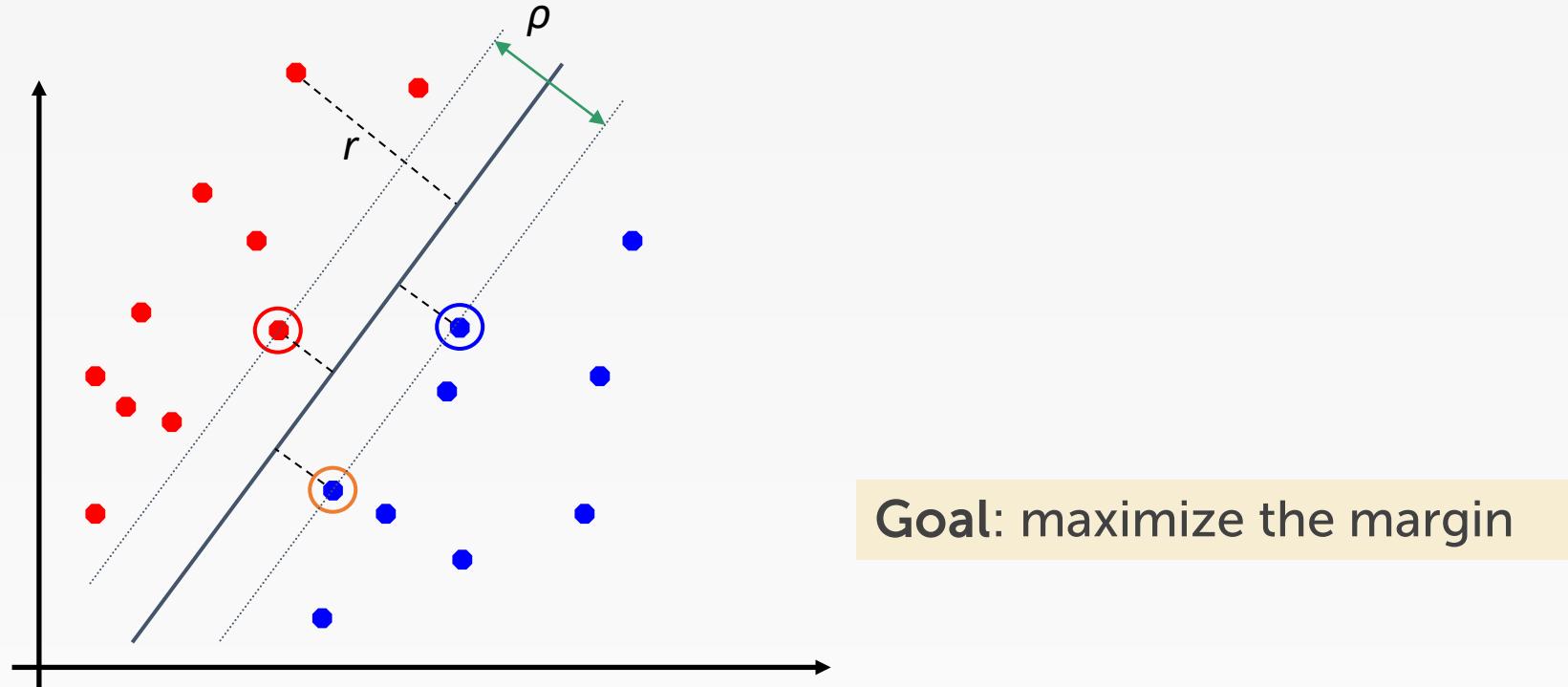
Linear SVM Classification

- Binary classification can be viewed as the task of separating classes in feature space:



Classification Margin

- Distance from example X_i to the separator is $r = \frac{|w^T X_i + b|}{\|w\|}$
- Examples closest to the hyperplane are *support vectors*.
- *Margin* ρ of the separator is the distance between support vectors.



SVM Optimization

Hard-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ \forall i = 1, \dots, N$$

SVM Optimization

Hard-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ \forall i = 1, \dots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} \cdot x^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N$$

$$\sum_{i=1}^N \alpha_i y^{(i)} = 0$$

Definition: **support vectors** are those points $x^{(i)}$ for which $\alpha_i \neq 0$

Method of Lagrange Multipliers

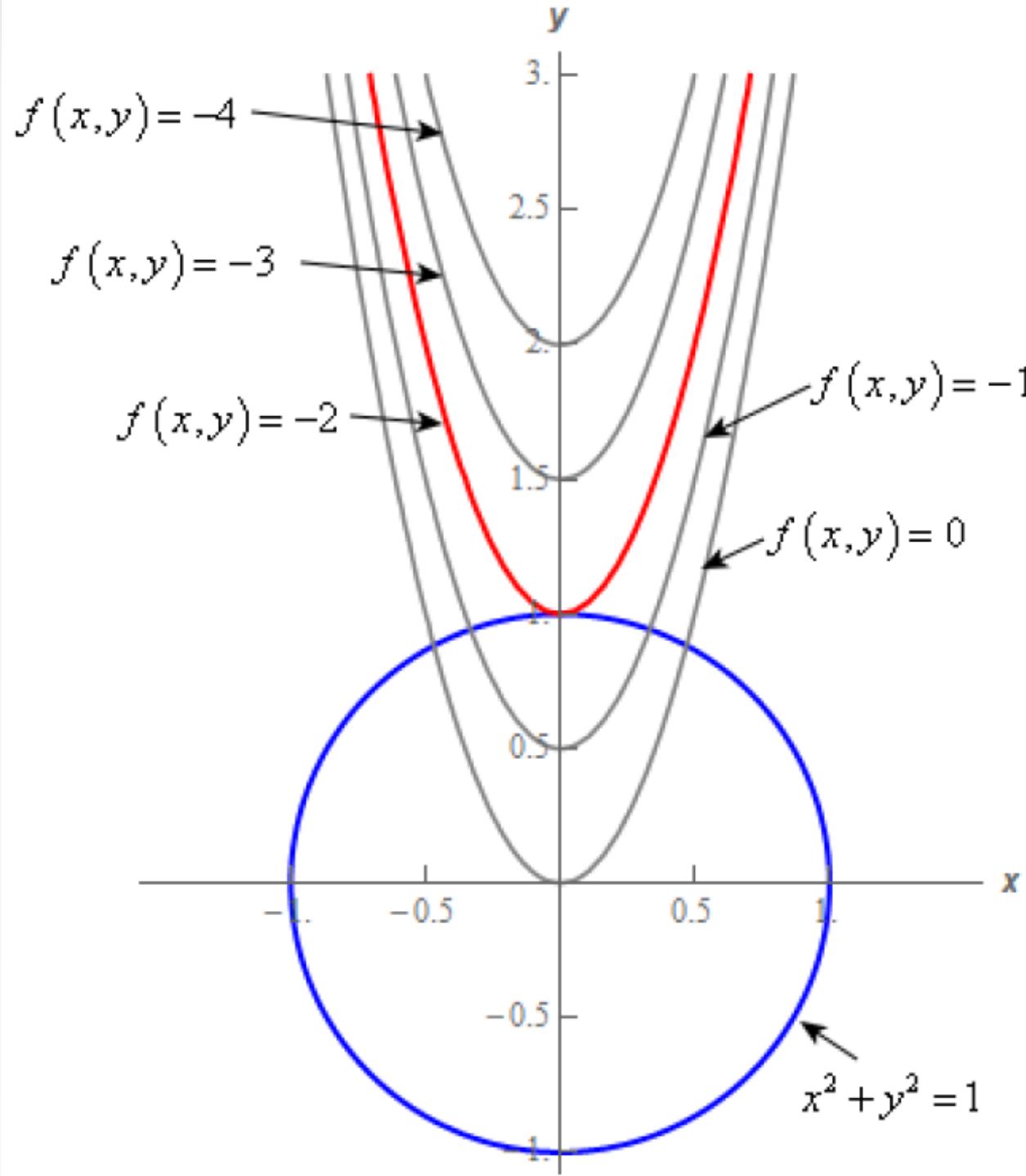
- Goal: $\min f(\boldsymbol{x})$ s.t., $g(\boldsymbol{x}) \leq c$

- Step 1: construct Lagrangian

$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda(g(\boldsymbol{x}) - c)$$

- Step 2: Solve $\min_x \max_{\lambda} L(\boldsymbol{x}, \lambda)$

$$\nabla f(\boldsymbol{x}) = \lambda \nabla g(\boldsymbol{x}), \text{ s.t. } \lambda \geq 0, g(\boldsymbol{x}) \leq c$$

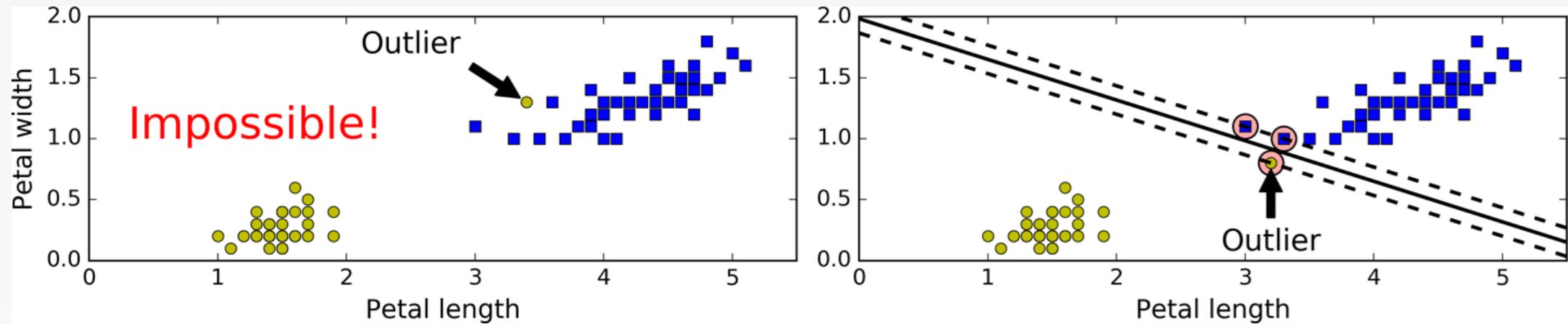


Hard Margin Classification

- Hard margin classification: all instances be off the decision boundary

Hard Margin Classification

- Hard margin classification: all instances be off the decision boundary
- Potential issues:
 - Only works if the data is linearly separable
 - Sensitive to outliers



Soft Margin Classification

- Key idea: balance between keeping the decision boundary as large as possible and limiting the margin violations

SVM Optimization

Hard-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ \forall i = 1, \dots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} \cdot x^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N$$

$$\sum_{i=1}^N \alpha_i y^{(i)} = 0$$

SVM Optimization

Hard-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ \forall i = 1, \dots, N$$

Soft-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C(\sum_{i=1}^N e_i)$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - e_i, \\ e_i \geq 0 \\ \forall i = 1, \dots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} \cdot x^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N$$

$$\sum_{i=1}^N \alpha_i y^{(i)} = 0$$

SVM Optimization

Hard-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ \forall i = 1, \dots, N$$

Soft-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C(\sum_{i=1}^N e_i)$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - e_i, \\ e_i \geq 0 \\ \forall i = 1, \dots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} \cdot x^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N \\ \sum_{i=1}^N \alpha_i y^{(i)} = 0$$

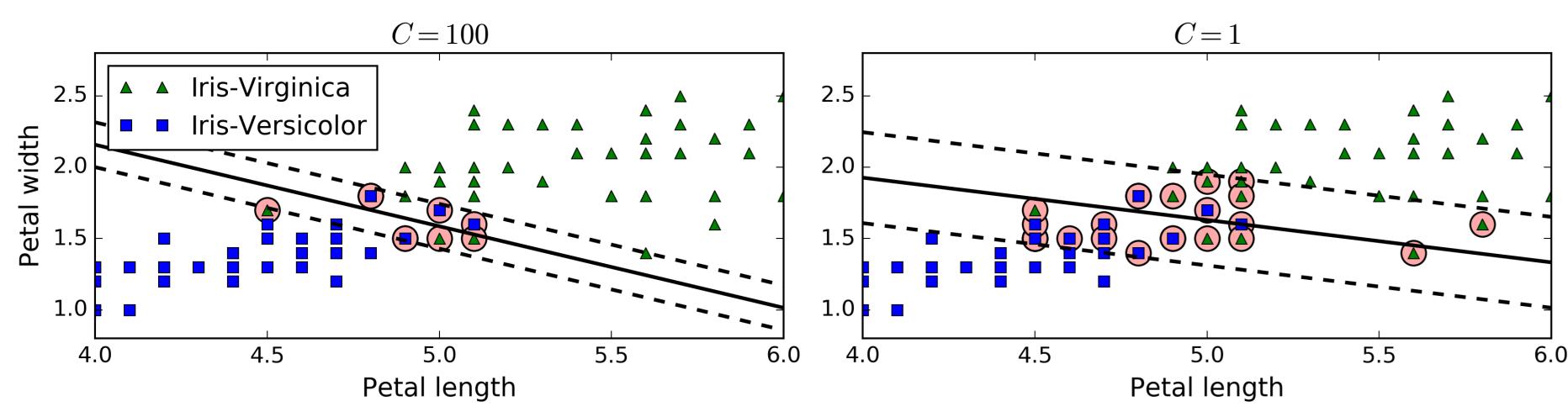
Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} \cdot x^{(j)}$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \forall i = 1, \dots, N \\ \sum_{i=1}^N \alpha_i y^{(i)} = 0$$

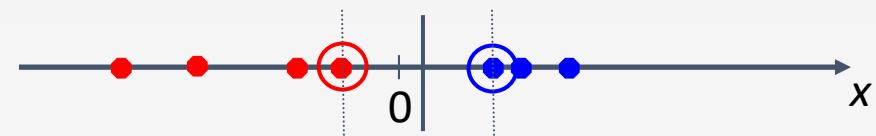
Soft Margin Classification

- **Key idea**: balance between keeping the decision boundary as large as possible and limiting the margin violations
- **C**: Regularization parameter
 - Small C \rightarrow large margin
 - Large C \rightarrow narrow margin
 - $C = \infty \rightarrow$ hard margin



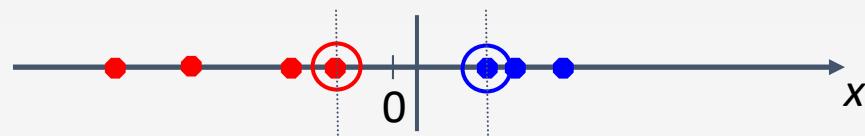
Non-linear SVMs

Datasets that are linearly separable with some noise work out great



Non-linear SVMs

Datasets that are linearly separable with some noise work out great

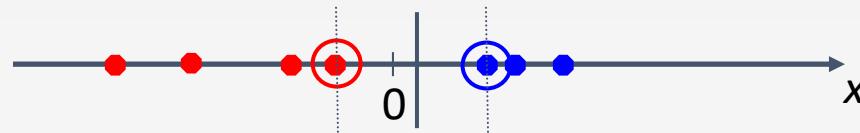


But what if the dataset is not that perfect?

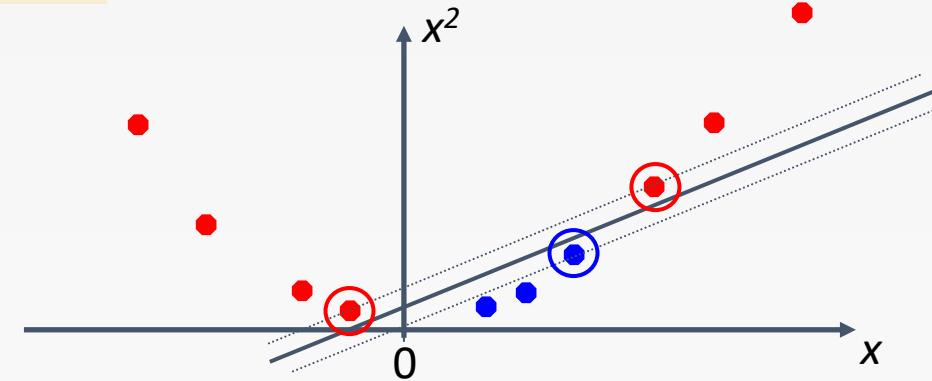


Non-linear SVMs

Datasets that are linearly separable with some noise work out great



But what if the dataset is not that perfect?



General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable

Kernel Method

- Motivation #1: Inefficient Features
 - Non-linearly separable data requires high dimensional representation
 - Might be prohibitively expensive to compute or store
- Motivation #2: Memory-based Methods
 - KNN
- Key idea:
 - Rewrite the algorithm so that we only work with dot product $x^T z$ of feature vectors
 - Replace the dot products $x^T z$ with a kernel function $k(x, z)$

SVM Optimization

Hard-margin SVM (Primal)

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ \forall i = 1, \dots, N$$

$$\min_{w,b} \frac{1}{2} \|w\|_2^2$$

$$\text{s.t. } y^{(i)}(w^T \phi(x^{(i)}) + b) \geq 1, \\ \forall i = 1, \dots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} \cdot x^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N \\ \sum_{i=1}^N \alpha_i y^{(i)} = 0$$

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} \phi(x^{(i)}) \cdot \phi(x^{(j)})$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N \\ \sum_{i=1}^N \alpha_i y^{(i)} = 0$$

SVM Kernel Trick

Hard-margin SVM (Lagrangian Dual)

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y^{(i)} y^{(j)} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$$\text{s.t. } \alpha_i \geq 0, \forall i = 1, \dots, N$$

$$\sum_{i=1}^N \alpha_i y^{(i)} = 0$$

The “Kernel Trick”

- If every data point is mapped into high-dimensional space via some transformation: $\Phi: x \rightarrow \psi(x)$, the inner product becomes:

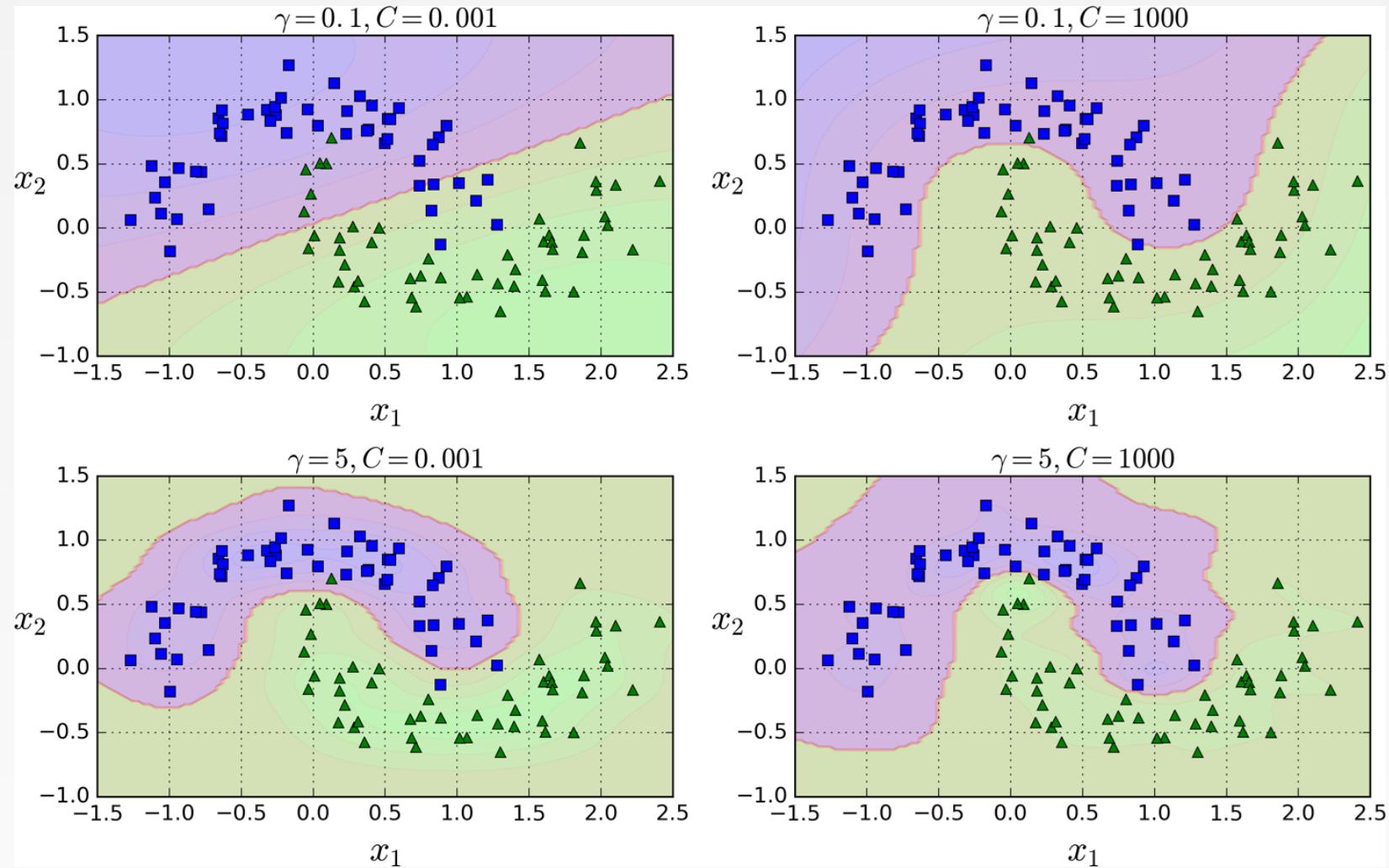
$$K(x, z) = \psi(x)^T \psi(z)$$

- A kernel function implicitly maps data to a high-dimensional space (without the need to compute each $\psi(x)$ explicitly)
- Can be applied to many algorithms:
 - Classification: SVM, ...
 - Regression: ridge regression, ...
 - Clustering: K-means,...

Kernel Example

Name	Kernel Function (Implicit dot product)	Feature Space (Explicit dot product)
Linear	$K(x, z) = x^T z$	Same as original input
Polynomial	$K(x, z) = (x^T z)^d$	All polynomials of degree d
Gaussian	$K(x, z) = \exp\left(-\frac{\ x-z\ _2^2}{2\sigma^2}\right)$	Infinite dimensional space
Sigmoid Kernel	$K(x, z) = \tanh(\alpha x^T z + c)$	With SVM, this is equivalent to a 2-layer neural network

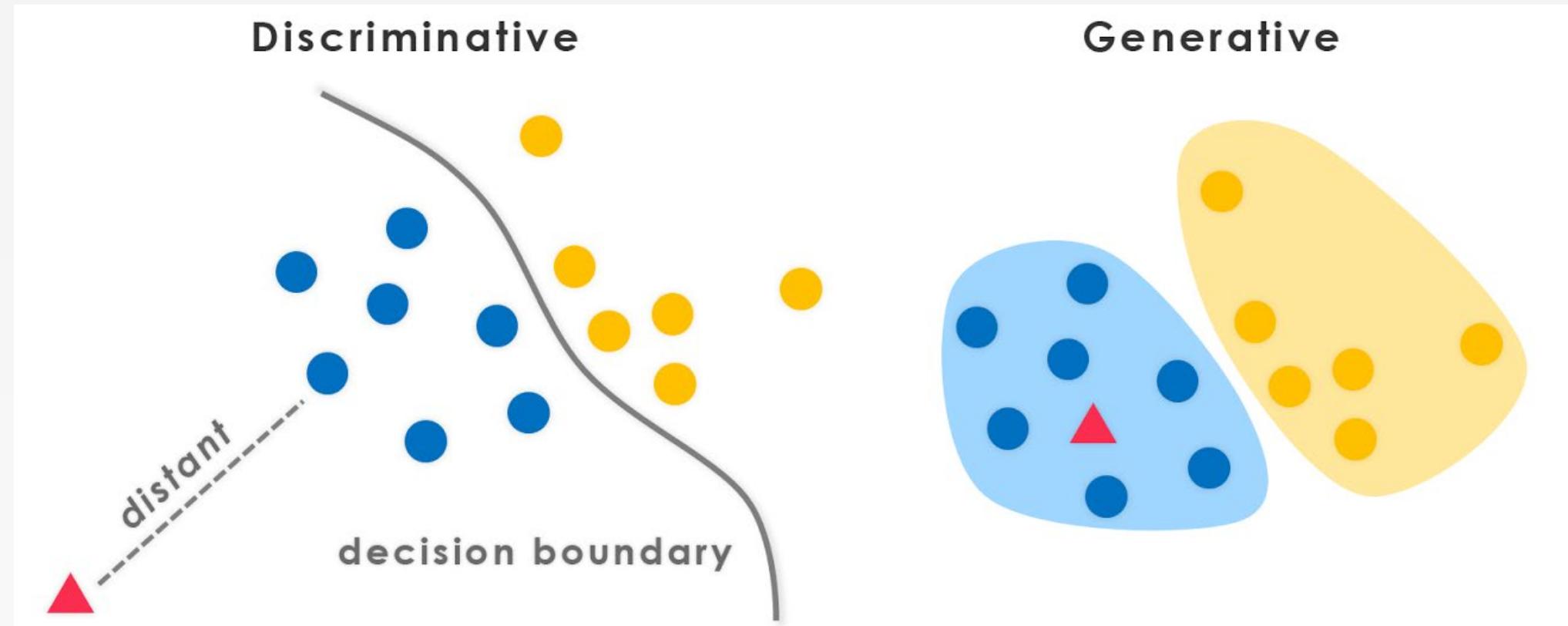
RBF Kernel



RBF Kernel: $K(x, z) = \exp(-\gamma \|x - z\|_2^2)$

Naïve Bayes

Generative vs. Discriminative



Probability Review

- $P(A) + P(\neg A) = 1$
 - $0 \leq P(A) \leq 1$
 - $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
 - $P(A) = P(A \vee B) + (A \wedge \neg B)$
 - $P(A|B) = \frac{P(A \wedge B)}{P(B)}$
 - Independence: $P(A \wedge B) = P(A) \times P(B)$
 $P(A|B) = P(A)$
 - Bayes' Rule: $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$
- $P(A) = \sum_{i=1}^k P(A \wedge B = v_i)$

Bayes Theorem

Heart of Bayes' theorem

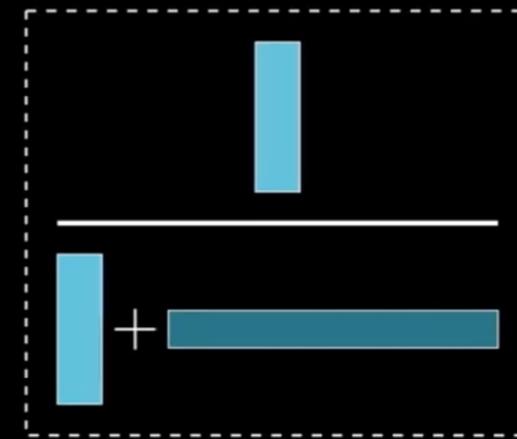
All possibilities



All possibilities
fitting the evidence



P (Librarian given)
the evidence



Naïve Bayes Assumption

Naïve Bayes classifiers assume that the effect of a variable value on a given class membership is independent of the values of other variables

$$P(X_1, X_2 | Y) = P(X_1 | X_2, Y)P(X_2 | Y) = P(X_1 | Y)P(X_2 | Y)$$

$$\text{More generally, } P(X_1, \dots, X_n | Y) = \prod_i P(X_i | Y)$$

$$\text{Use Bayes' Rule: } P(Y_j | X_1, \dots, X_N) = \frac{P(Y_j) \cdot \prod_i P(X_i | Y_j)}{\sum_k P(Y_k) \cdot \prod_i P(X_i | Y_k)}$$

Fake News Detector

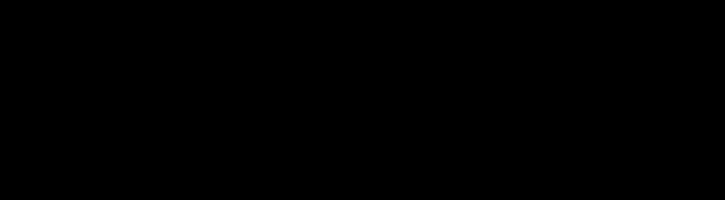
CNN News

cnn politics 45 CONGRESS SUPREME COURT 2018 ELECTION RESULTS

THE RUSSIA INVESTIGATION

Email pointed Trump campaign to WikiLeaks documents

By Manu Raju and Jeremy Herb, CNN
Updated 2305 GMT (0705 HKT) December 8, 2017



NEWS & BUZZ

- Erin Burnett scorches Trump's 'unpresidential' letter
- Reporter says Trump called after Putin meeting to side

Fake News

Michelle Obama Deletes Hillary Clinton From Twitter

When Hillary goes low, Michelle goes insta!

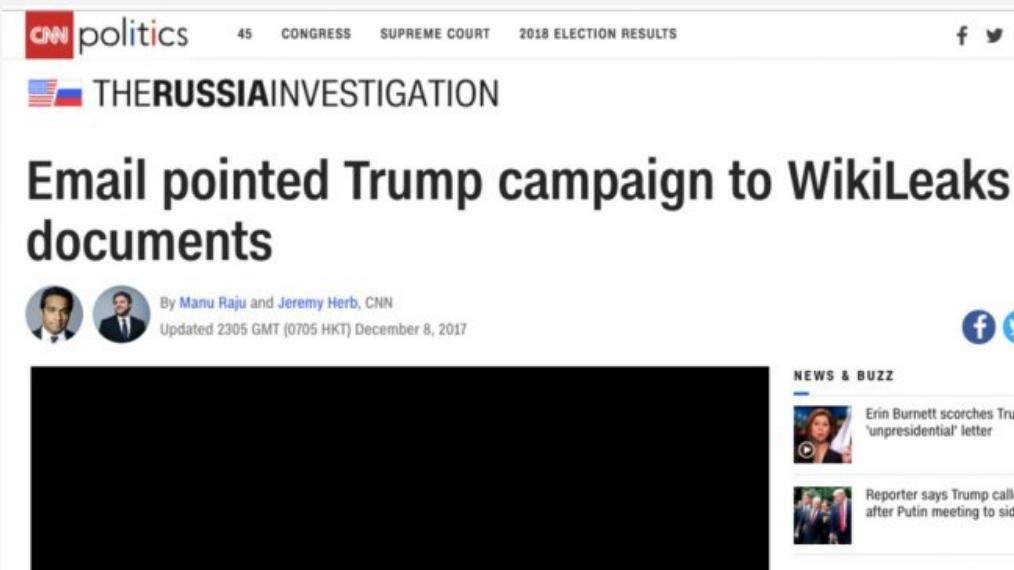
Published November 1, 2017 by Karen DeWitt in News, US, Politics



Michelle Obama has scrubbed all references to Hillary Clinton from both of her Twitter accounts as news breaks that Clinton is under two different FBI investigations involving four FBI offices.

Fake News Detector

CNN News



Fake News

Michelle Obama Deletes Hillary Clinton From Twitter

When Hillary goes low, Michelle goes insta!

Published November 1, 2017 by Steven Dreyfus in News, US, Politics



Michelle Obama has scrubbed all references to Hillary Clinton from both of her Twitter accounts as news breaks that Clinton is under two different FBI investigations involving four FBI offices.

the dog is on the table

Bag of words



Model 1: Bernoulli Naïve Bayes

Flip a weighted coin



Model 1: Bernoulli Naïve Bayes

Flip a weighted coin



If HEADS, flip
each red coin



$y \quad x_1 \quad x_2 \quad x_3 \quad \dots \quad x_M$

If TAILS, flip each
blue coin



Model 1: Bernoulli Naïve Bayes

Flip a weighted coin



If HEADS, flip
each red coin



If TAILS, flip each
blue coin



y	x_1	x_2	x_3	\dots	x_M
0	1	0	1	\dots	1

Model 1: Bernoulli Naïve Bayes

Flip a weighted coin



If HEADS, flip
each red coin



If TAILS, flip each
blue coin



y	x_1	x_2	x_3	\dots	x_M
0	1	0	1	\dots	1
1	0	1	0	\dots	1

Model 1: Bernoulli Naïve Bayes

Flip a weighted coin



If HEADS, flip
each red coin



If TAILS, flip each
blue coin



$y \quad x_1 \quad x_2 \quad x_3 \quad \dots \quad x_M$

0	1	0	1	...	1
---	---	---	---	-----	---

1	0	1	0	...	1
---	---	---	---	-----	---

1	1	1	1	...	1
---	---	---	---	-----	---

0	0	0	1	...	1
---	---	---	---	-----	---

0	1	0	1	...	0
---	---	---	---	-----	---

1	1	0	1	...	0
---	---	---	---	-----	---

What's wrong with the Naïve Bayes Assumption?

The features might not be independent!!

- Example 1:
 - If a document contains the word “Donald”, it’s extremely likely to contain the word “Trump” – These are not independent!
- Example 2:
 - If the petal width is very high, the petal length is also likely to be very high

Model 1: Bernoulli Naïve Bayes

- Data: $x \in \{0,1\}^M, y \in \{0,1\}$

Generative Process:

$$y \sim \text{Bernoulli}(\phi)$$

$$x_1 \sim \text{Bernoulli}(\theta_{y,1})$$

$$x_2 \sim \text{Bernoulli}(\theta_{y,2})$$

...

$$x_M \sim \text{Bernoulli}(\theta_{y,M})$$

Model:

$$p_{\phi,\theta}(x, y) = p_{\phi,\theta}(x_1, x_2, \dots, x_M, y)$$

$$= p_\phi(y) \prod_{m=1}^M p_\theta(x_m | y)$$

$$= \left[(\phi)^y (1 - \phi)^{(1-y)} \prod_{m=1}^M (\theta_{y,m})^{x_m} (1 - \theta_{y,m})^{(1-x_m)} \right]$$

MLE

Training: Find the **class-conditional** MLE parameters

Count Variables

$$N_{y=1} = \sum_{i=1}^N \mathbb{I}(y^{(i)} = 1)$$

$$N_{y=0} = \sum_{i=1}^N \mathbb{I}(y^{(i)} = 0)$$

$$N_{y=0,x_m=1} = \sum_{i=1}^N \mathbb{I}(y^{(i)} = 0 \wedge x_m^{(i)} = 1)$$

Maximum Likelihood Estimators

$$\phi = \frac{N_{y=1}}{N}$$

$$\phi_{0,m} = \frac{N_{y=0,x_m=1}}{N_{y=0}}$$

$$\phi_{1,m} = \frac{N_{y=1,x_m=1}}{N_{y=1}}$$

$$\forall m \in \{1, \dots, M\}$$

An Illustrative Example

ID	Charges?	Size	Outcome
1	Y	Small	Truthful
2	N	Small	Truthful
3	N	Large	Truthful
4	N	Large	Truthful
5	N	Small	Truthful
6	N	Small	Truthful
7	Y	Small	Fraud
8	Y	Large	Fraud
9	N	Large	Fraud
10	Y	Large	Fraud

Goal: new record: small firm, charges = yes

An Illustrative Example

ID	Charges?	Size	Outcome
1	Y	Small	Truthful
2	N	Small	Truthful
3	N	Large	Truthful
4	N	Large	Truthful
5	N	Small	Truthful
6	N	Small	Truthful
7	Y	Small	Fraud
8	Y	Large	Fraud
9	N	Large	Fraud
10	Y	Large	Fraud

Goal: new record: small firm, charges = yes

$$P(\text{size} = \text{small} | \text{Fraud}) = 0.25$$

$$P(\text{charge} = Y | \text{Fraud}) = 0.75$$

$$P(\text{size} = \text{small} | \text{Truthful}) = 4/6$$

$$P(\text{charge} = Y | \text{Truthful}) = 1/6$$

$$P(\text{Fraud}) \times 0.25 \times 0.75 = 0.075$$

$$P(\text{Truthful}) \times \left(\frac{4}{6}\right) \times \left(\frac{1}{6}\right) = 0.067$$

$$P(\text{Fraud} | \text{small, yes}) = \frac{0.075}{0.075 + 0.067} = \mathbf{0.53}$$

Naïve Bayes Model

- **Suppose:** Depends on the choice of event model $P(X_k|Y)$
- **Model:** Product of prior and the model

$$P(X, Y) = P(Y) \prod_{k=1}^K P(X_k|Y)$$

- **Training:** Find the class-conditional MLE parameters
 - For $P(Y)$, we find the MLE using all the data.
 - For each $P(X_k|Y)$, we condition on the data with the corresponding
- **Classification:** Find the class that maximizes the posterior

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_y p(y|x) \\ &= \operatorname{argmax}_y p(x|y)p(y)/p(x) \\ &= \operatorname{argmax}_y p(x|y)p(y)\end{aligned}$$

A shortcoming of MLE

- suppose we never observe the word “**unicorn**” in a real news article?

A shortcoming of MLE

- suppose we never observe the word “unicorn” in a real news article?
- Add-1 Smoothing

$$D = \{(x^{(i)}, y^{(i)})\}_{i=1}^N, D' = D \cup \{(\mathbf{0}, 0), (\mathbf{0}, 1), (\mathbf{1}, 0), (\mathbf{1}, 1)\}$$
$$\theta_{k,0} = \frac{1 + \sum_{i=1}^N \mathbb{I}(y^{(i)} = 0 \wedge x_k^{(i)} = 1)}{2 + \sum_{i=1}^N \mathbb{I}(y^{(i)} = 0)}$$

Other NB Models

- Bernoulli Naïve Bayes:
 - For binary features
- Multinomial Naïve Bayes:
 - For integer features
- Gaussian Naïve Bayes
 - For continuous features

Model 2: Multinomial Naïve Bayes

- Data: $x = [x_1, x_2, \dots, x_M]$, where $x_m \in \{1, \dots, K\}$

Generative Process:

for $i \in \{1, \dots, N\}$:

$y \sim Bernoulli(\phi)$

for $j \in \{1, \dots, M_i\}$:

$x_j^{(i)} \sim Multinomial(\theta_{y^{(i)}}, 1)$

Model:

$$p_{\phi, \theta}(x, y)$$

$$= \left[(\phi)^y (1 - \phi)^{(1-y)} \prod_{j=1}^{M_i} \theta_{y, x_j} \right]$$

Model 3: Gaussian Naïve Bayes

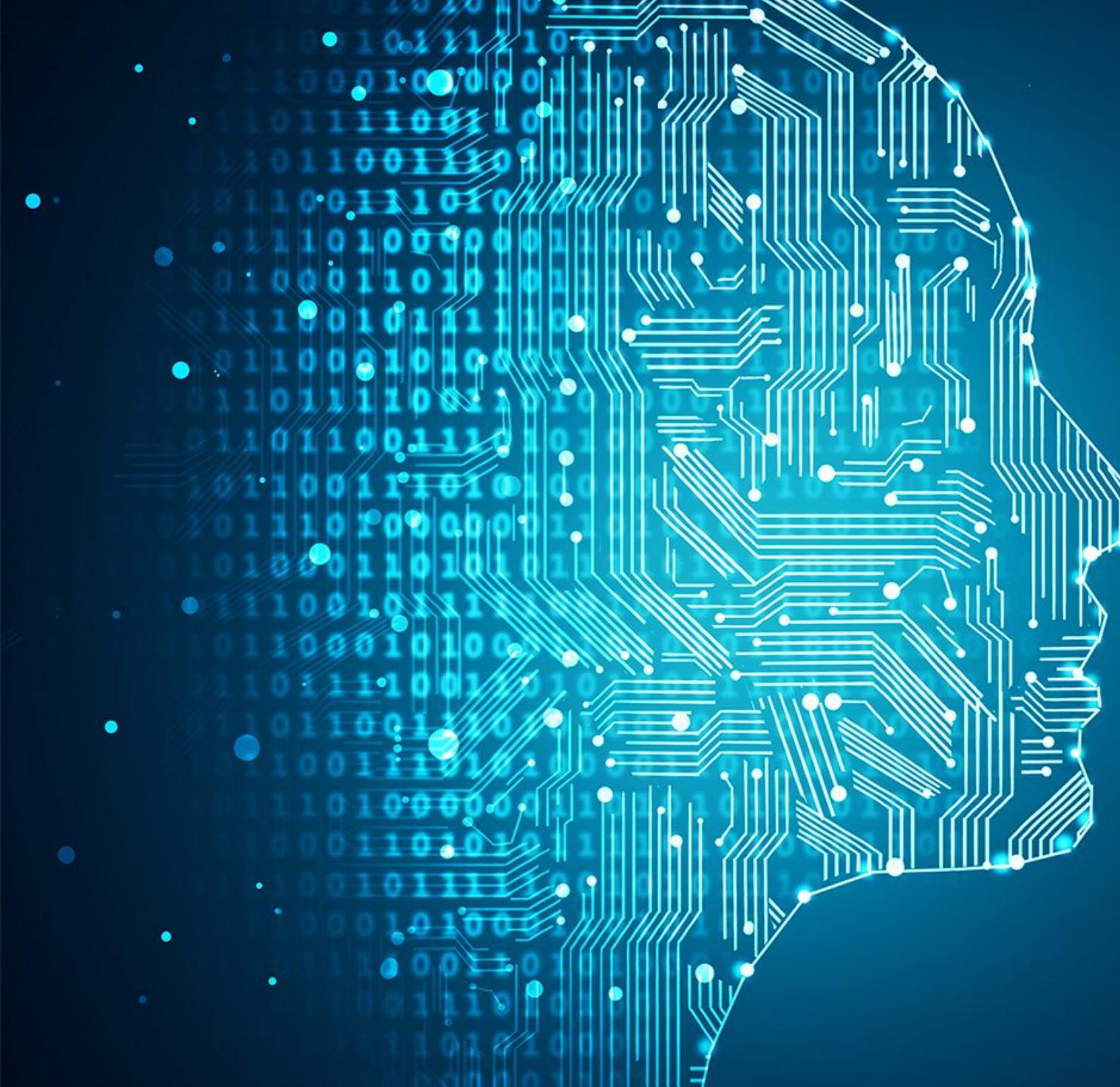
- Data: $x \in \mathbb{R}^M$

Model:

$$p(x, y) = p(x_1, x_2, \dots, x_M, y)$$

$$= p(y) \prod_{k=1}^M p(x_k|y)$$

Gaussian Naïve Bayes assumes that $p(x_k|y)$ is given by a normal distribution.



机器学习与人工智能 Machine Learning and Artificial Intelligence

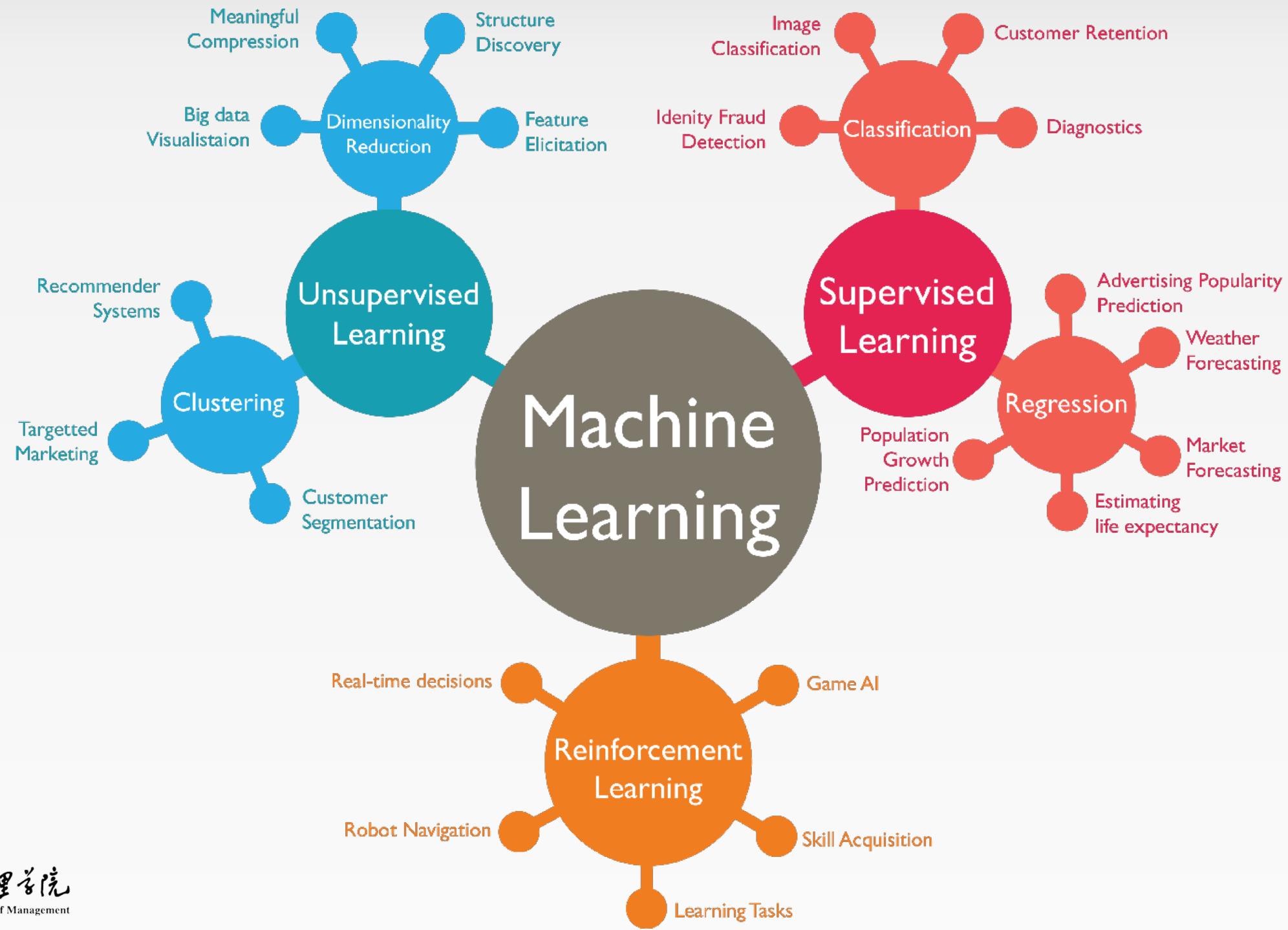
Lecture 6 Ensemble Models, HMM, Clustering

Yingjie Zhang (张颖婕)

Peking University

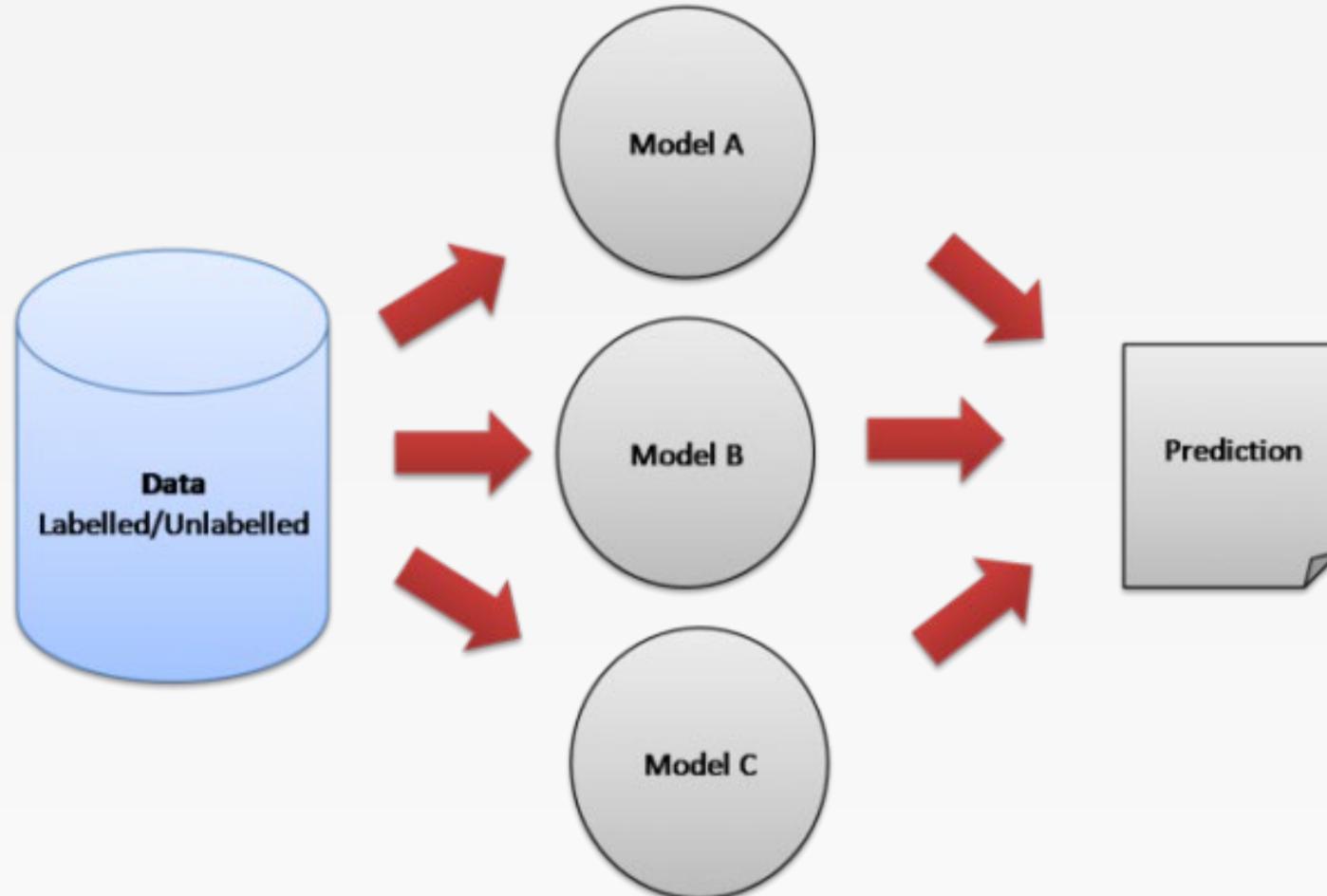
yingjiezhang@gsm.pku.edu.cn

2021 Fall



Ensemble Models

Wisdom of the Crowd



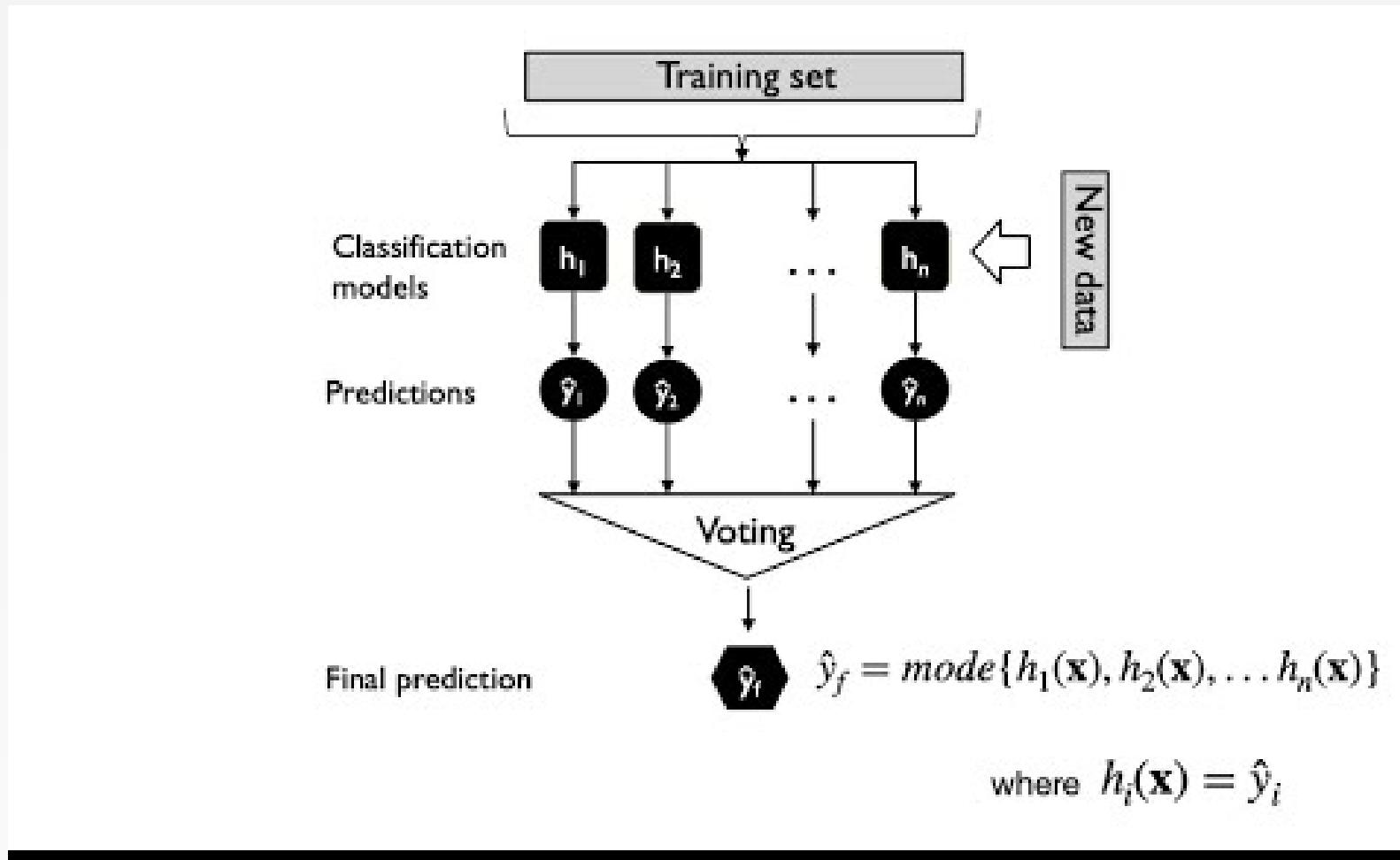
Ensemble Methods

- Why it works:
 - Diversity!
 - Imagine that we have 5 completely independent classifiers; each of them individually is correct 70% of the time
 - Prob(correctly classify a record by a majority vote)
$$= C_{(5,3)}(0.7)^3(0.3)^2 + C_{(5,4)}(0.7)^4(0.3)^1 + C_{(5,5)}(0.7)^5 = 0.837$$
- Downside:
 - Increased complexity, more difficult to interpret
 - Does not always guarantee performance improvements

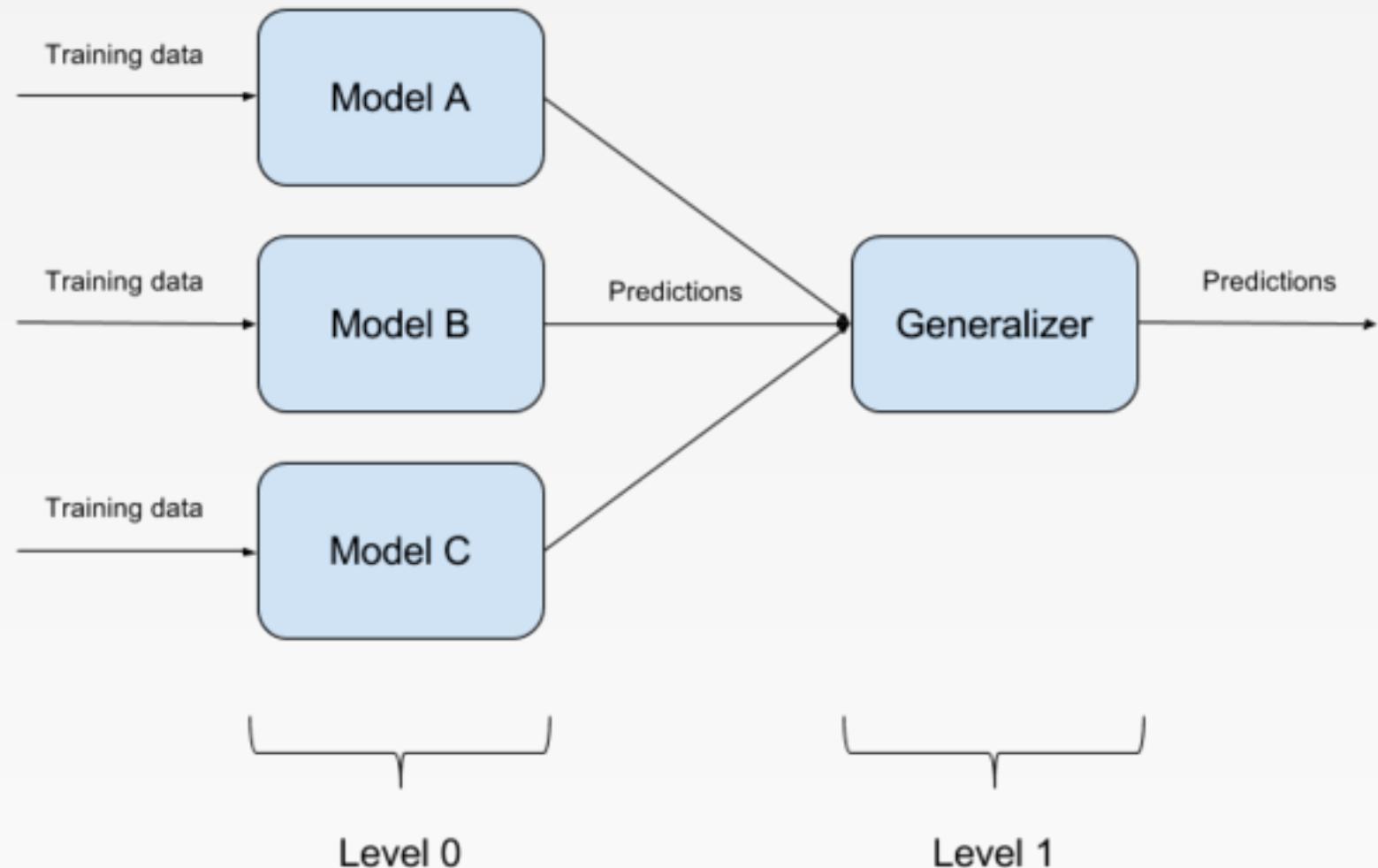
Ensemble Methods

- Voting Classifiers
- Stacking
- Bagging
- Boosting

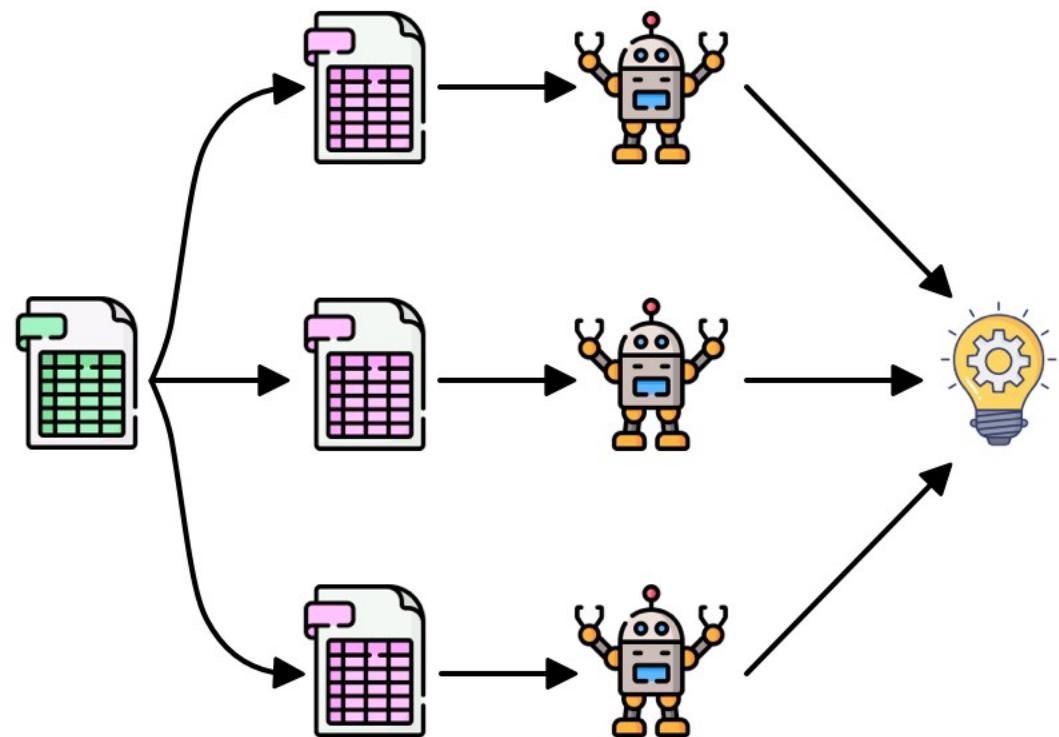
Voting Classifiers



Stacking

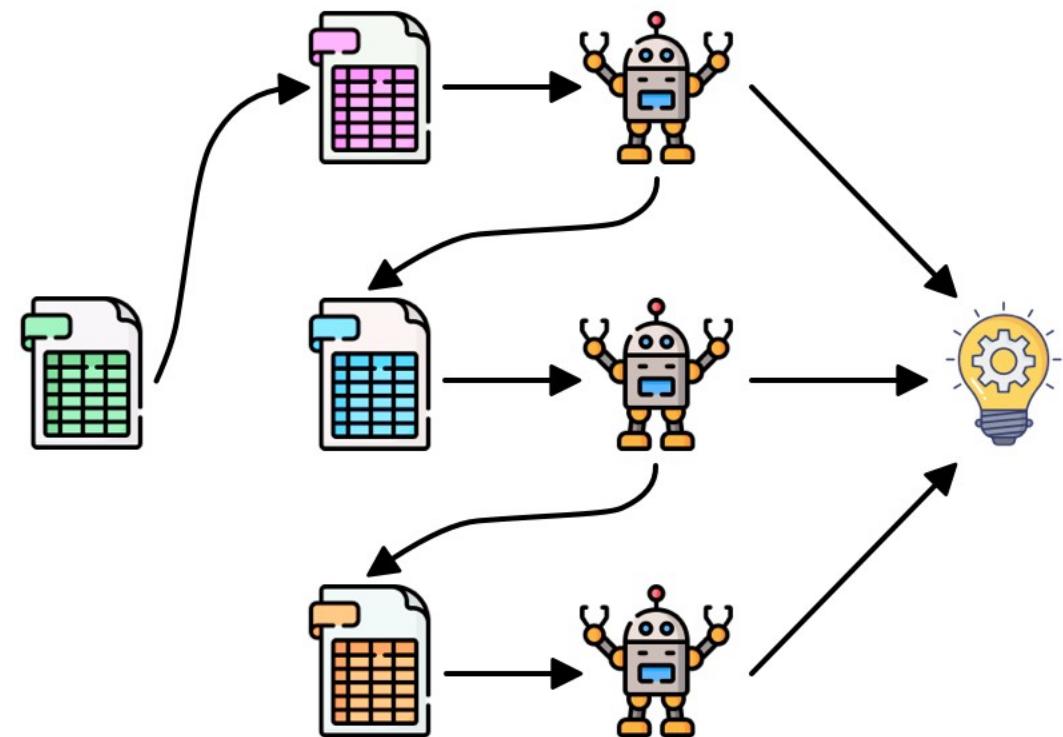


Bagging



Parallel

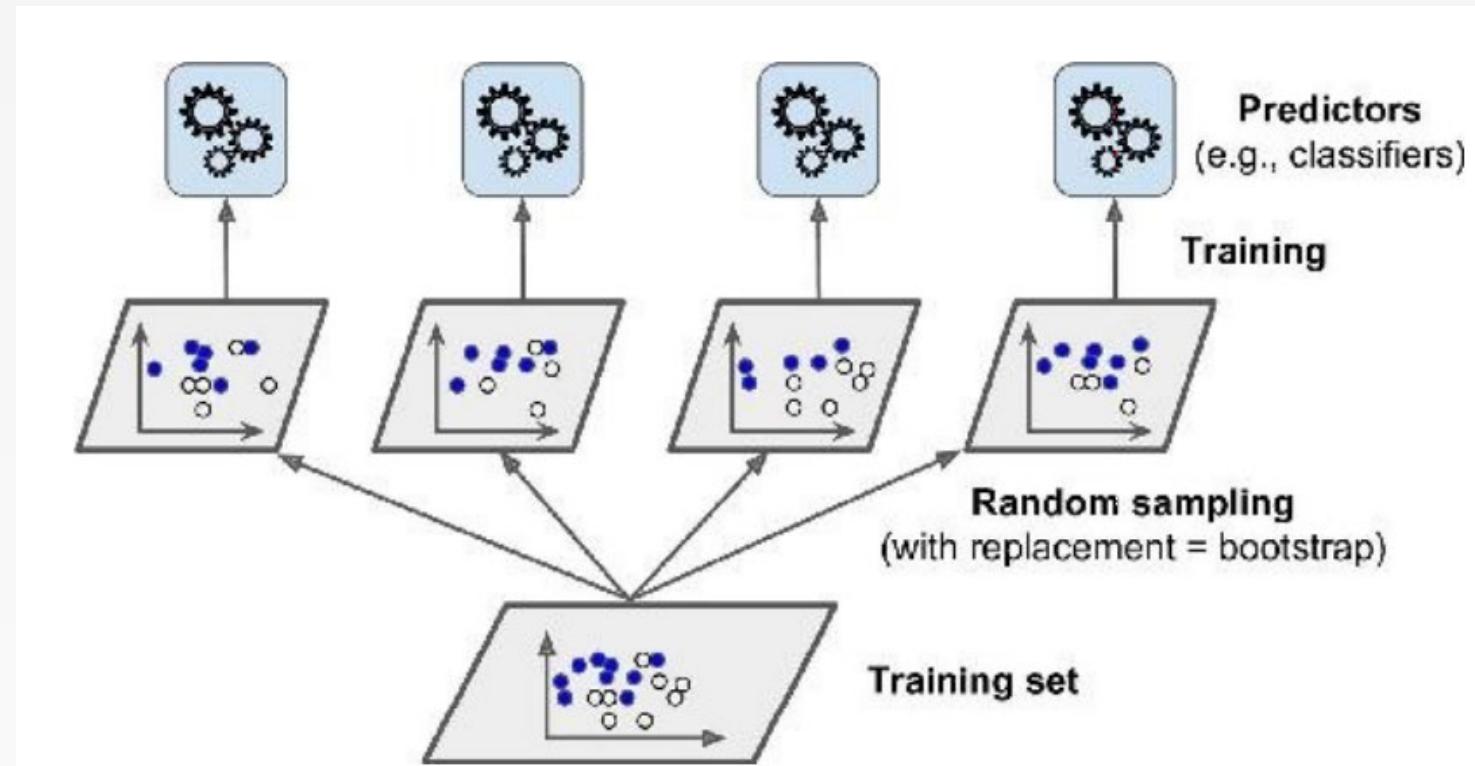
Boosting



Sequential

Bagging: Bootstrap Aggregation

- Ideas:
 - Use the same training algorithm for every predictor, but to train them on different random subsets of the training set



Bagging

- Given
 - Labelled dataset
 - Specific predictive modeling techniques
- Train k models on different training data samples
 - Bootstrap samples: sampled with replacement, typically of the same size as the original training data
- Final prediction is done by combining (i.e., majority vote, averaging) the predictions of k individual models

Overview

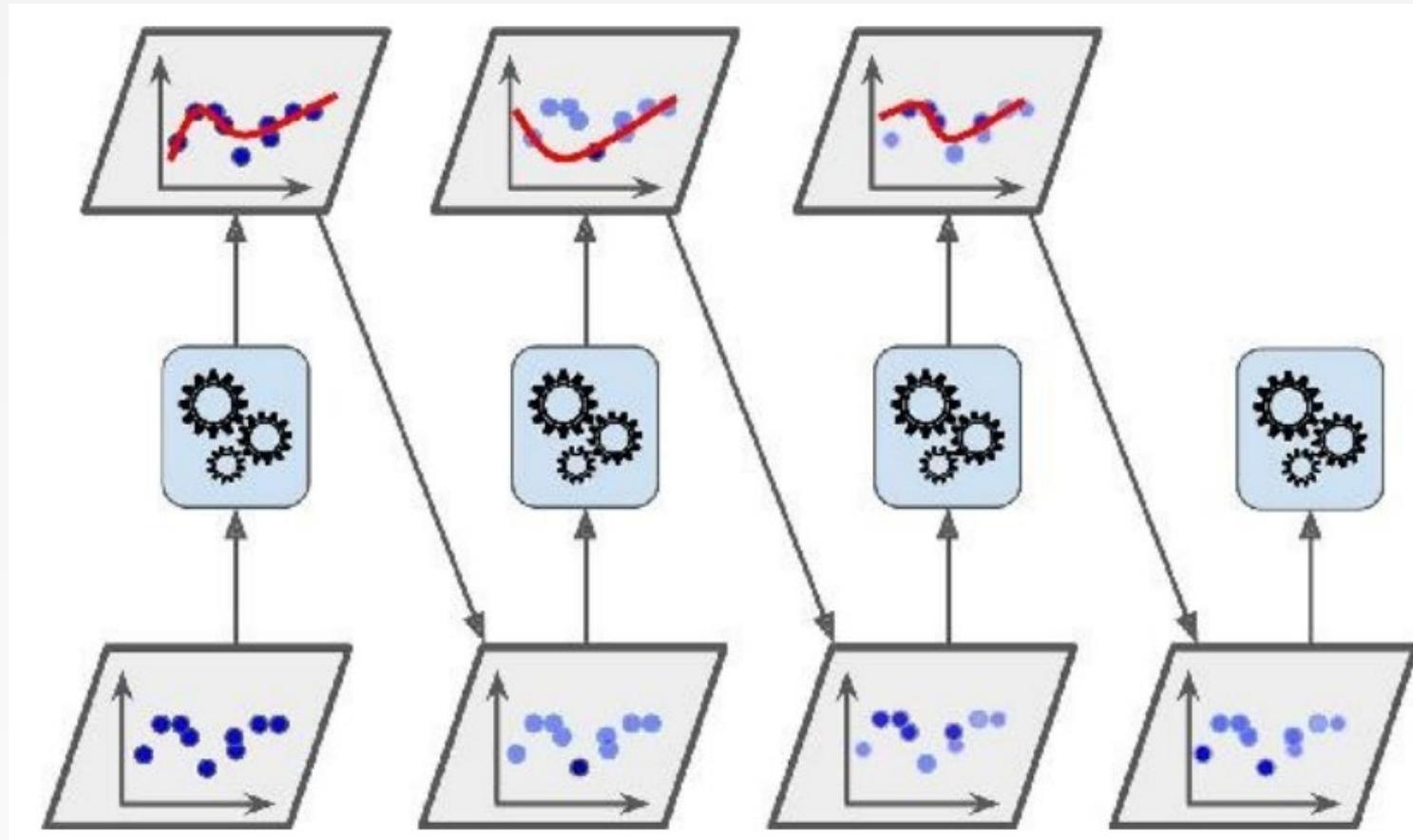
- Definition
 - Collection of unpruned trees
 - Rule to combine individual tree decisions
- Purpose
 - Improve prediction accuracy
 - Improve efficiency
- Principle
 - Encouraging diversity among the tree
- Solution: randomness
 - Bagging
 - Random decision trees

Details

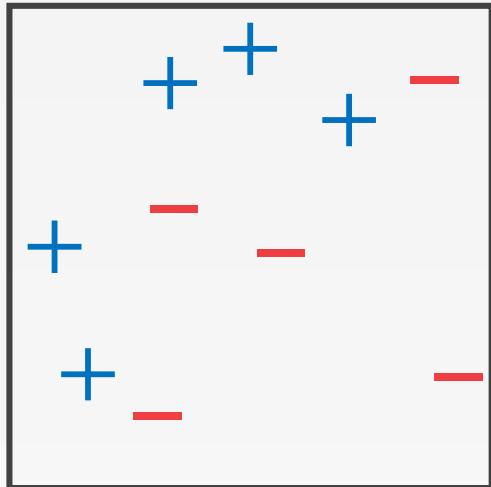
- Build many “random” trees
- Randomness: using only a random sample of m attributes to calculate each split
- For each tree:
 - Choose a different training sample
 - For each node, choose m random attributes and find the best split
 - Trees are often fully grown (not pruned)
- Predication: majority vote among all the trees

Boosting

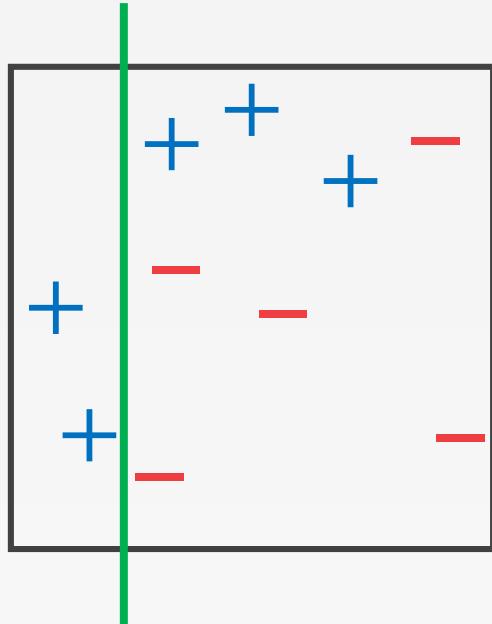
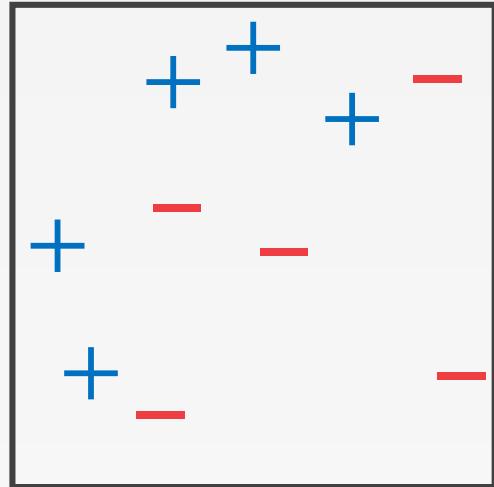
- AdaBoost



AdaBoost: Toy Example



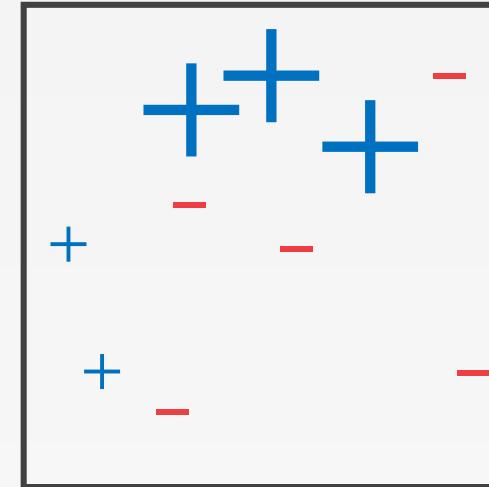
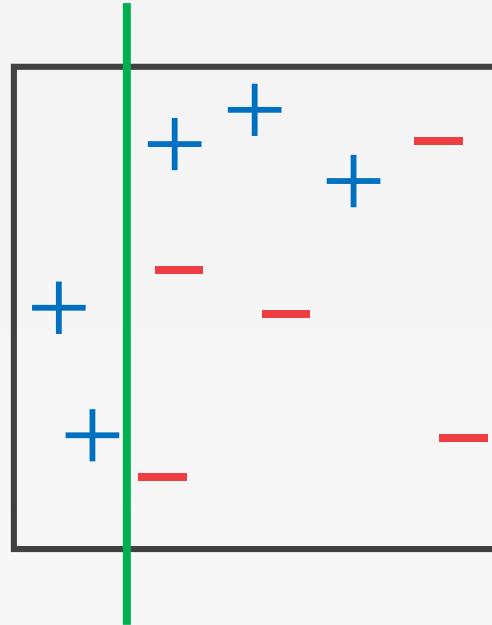
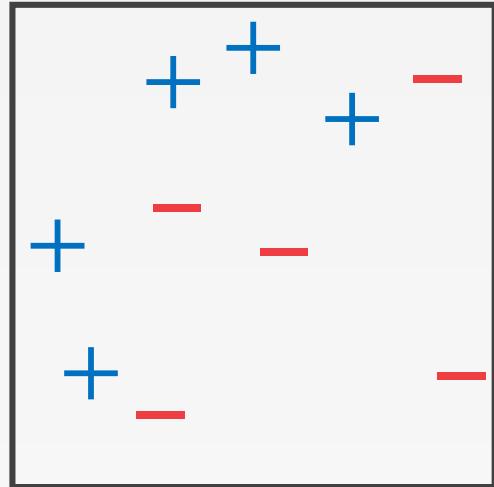
AdaBoost: Toy Example



$$\varepsilon_1 = 0.3$$

$$\alpha_1 = 0.42$$

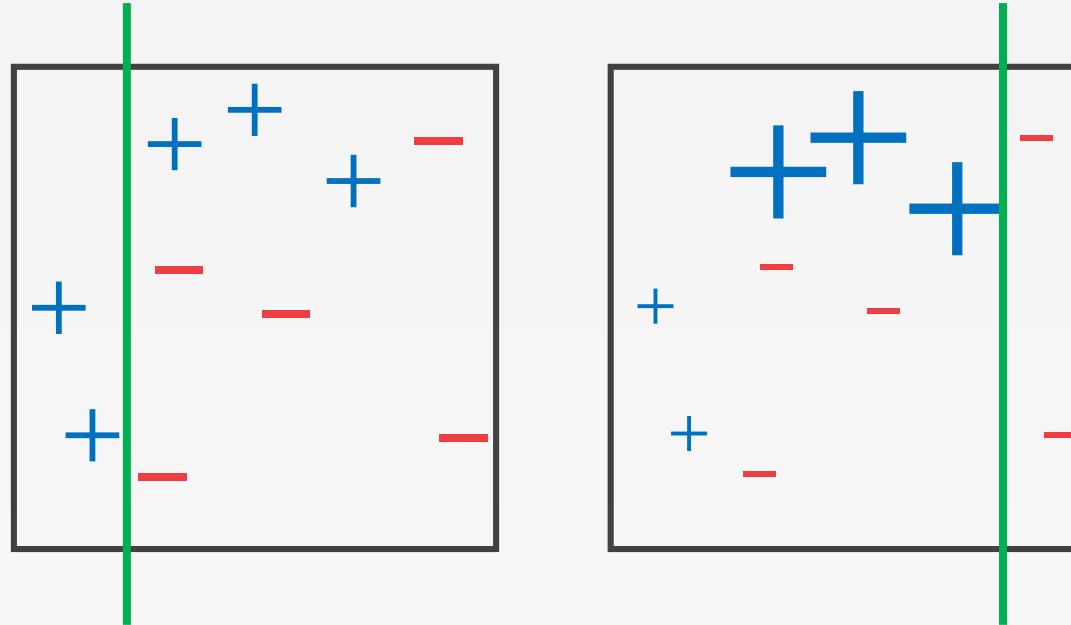
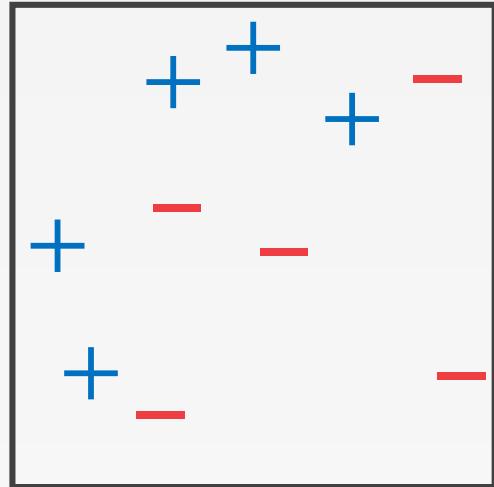
AdaBoost: Toy Example



$$\varepsilon_1 = 0.3$$

$$\alpha_1 = 0.42$$

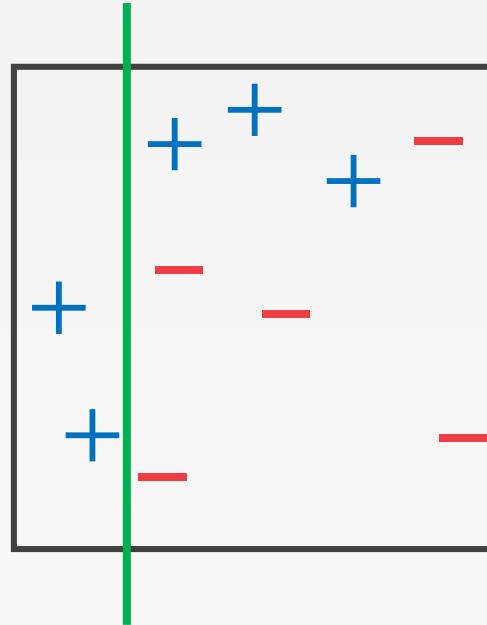
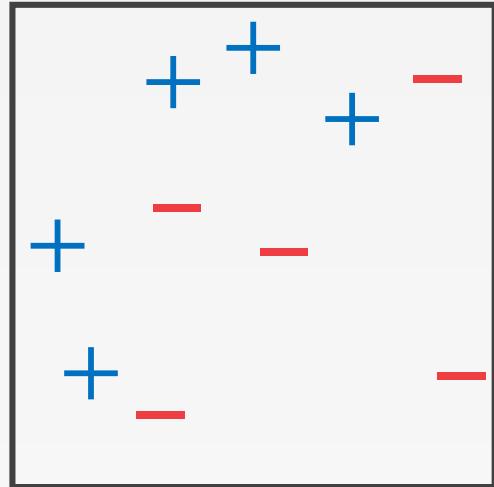
AdaBoost: Toy Example



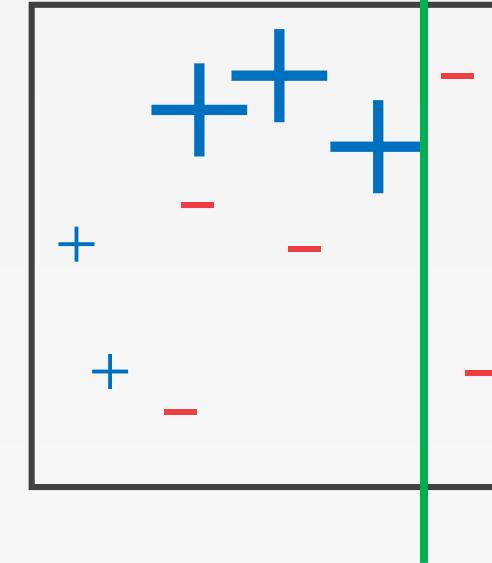
$$\begin{aligned}\varepsilon_1 &= 0.3 \\ \alpha_1 &= 0.42\end{aligned}$$

$$\begin{aligned}\varepsilon_2 &= 0.21 \\ \alpha_2 &= 0.65\end{aligned}$$

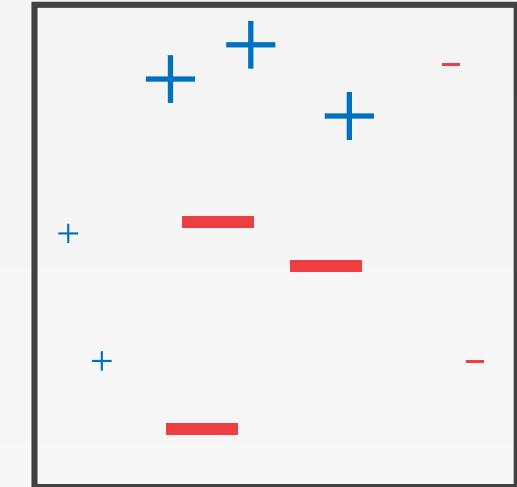
AdaBoost: Toy Example



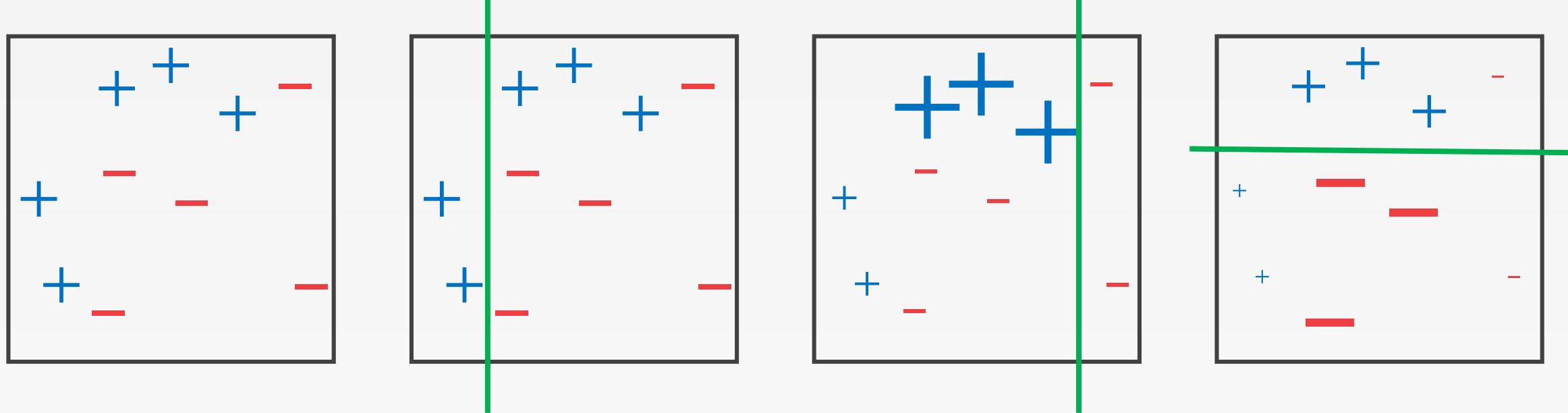
$$\begin{aligned}\varepsilon_1 &= 0.3 \\ \alpha_1 &= 0.42\end{aligned}$$



$$\begin{aligned}\varepsilon_2 &= 0.21 \\ \alpha_2 &= 0.65\end{aligned}$$



AdaBoost: Toy Example



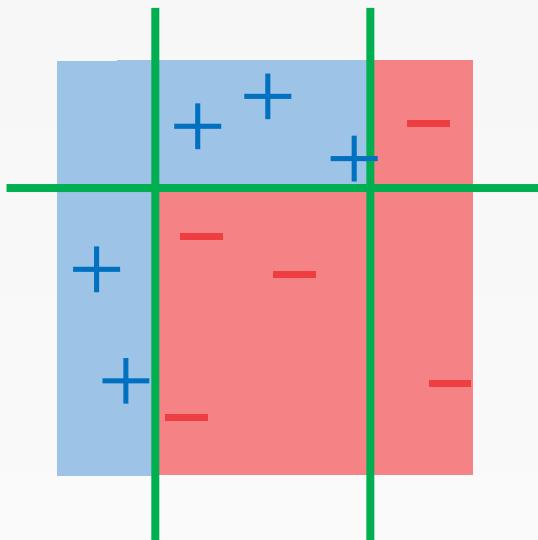
$$\begin{aligned}\varepsilon_1 &= 0.3 \\ \alpha_1 &= 0.42\end{aligned}$$

$$\begin{aligned}\varepsilon_2 &= 0.21 \\ \alpha_2 &= 0.65\end{aligned}$$

$$\begin{aligned}\varepsilon_3 &= 0.14 \\ \alpha_3 &= 0.92\end{aligned}$$

AdaBoost: Toy Example

$H_{final} = \text{sign}$



AdaBoost Algorithm

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For $t=1, \dots, T$:

Train weak learner using distribution D_t

Get week hypothesis $h_t: X \rightarrow \{-1, +1\}$ with error $\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$

Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$

Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)

Output the final hypothesis: $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$

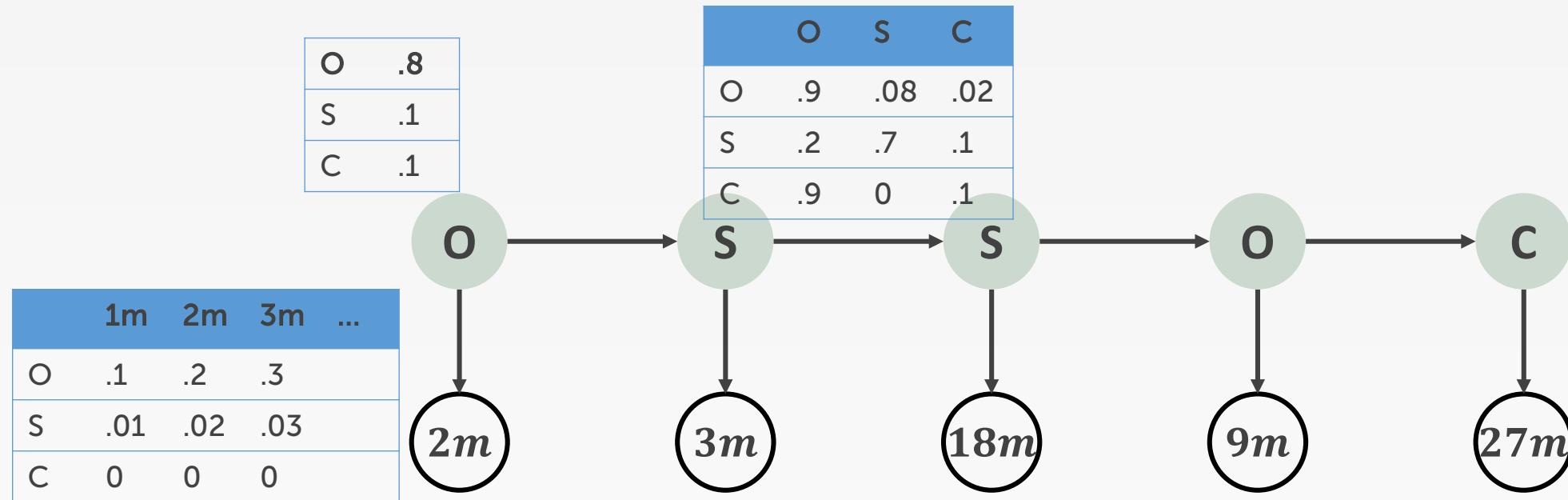
Hidden Markov Models



HMM

- An HMM provides a joint distribution with an assumption of dependence between adjacent states

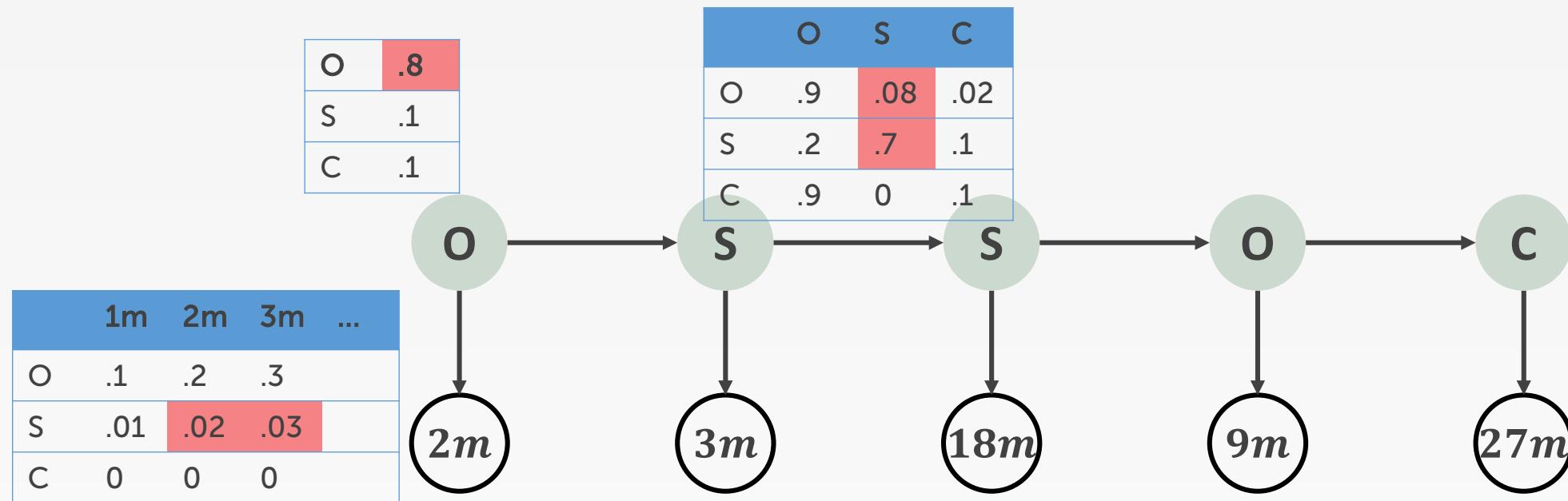
$$p(O, S, S, O, C, 2m, 3m, 18m, 9m, 27m) = (.8 * .2 * .08 * .03 * .7 * \dots)$$



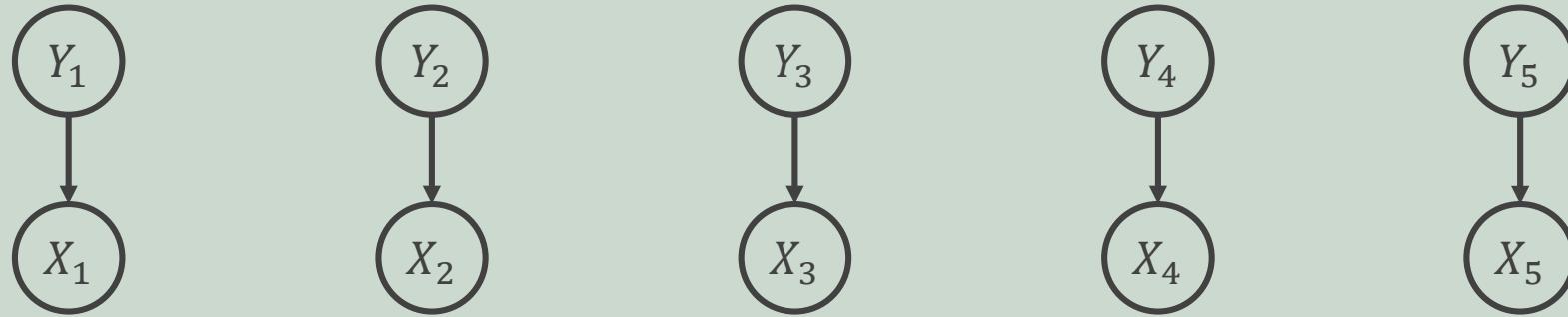
HMM

- An HMM provides a joint distribution with an assumption of dependence between adjacent states

$$p(O, S, S, O, C, 2m, 3m, 18m, 9m, 27m) = (.8 * .2 * .08 * .03 * .7 * \dots)$$

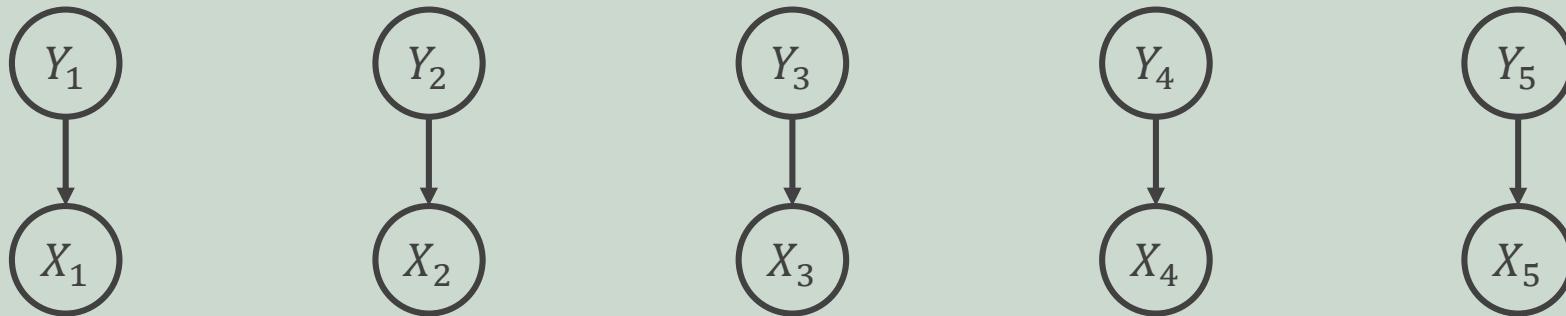


HMM

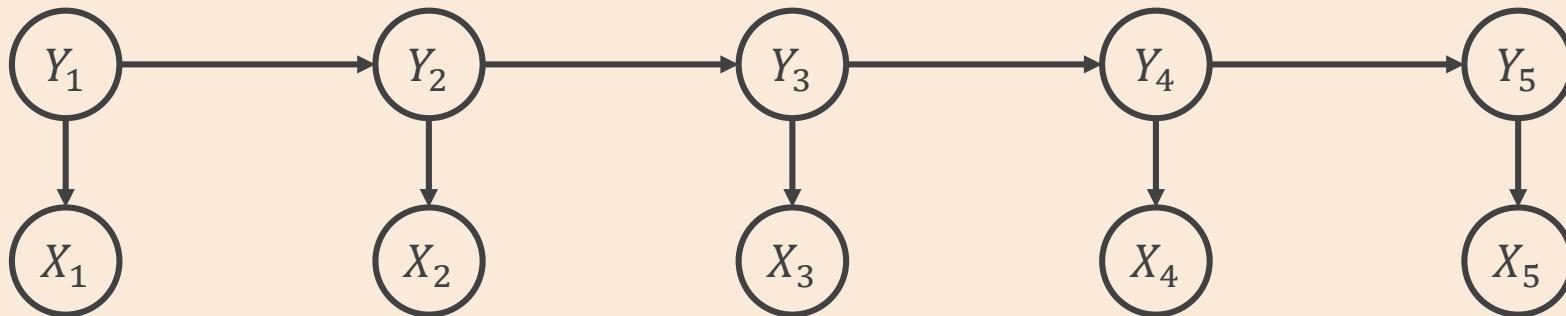


Naïve Bayes: $P(X, Y) = \prod_{t=1}^T P(X_t | Y_t)p(Y_t)$

HMM

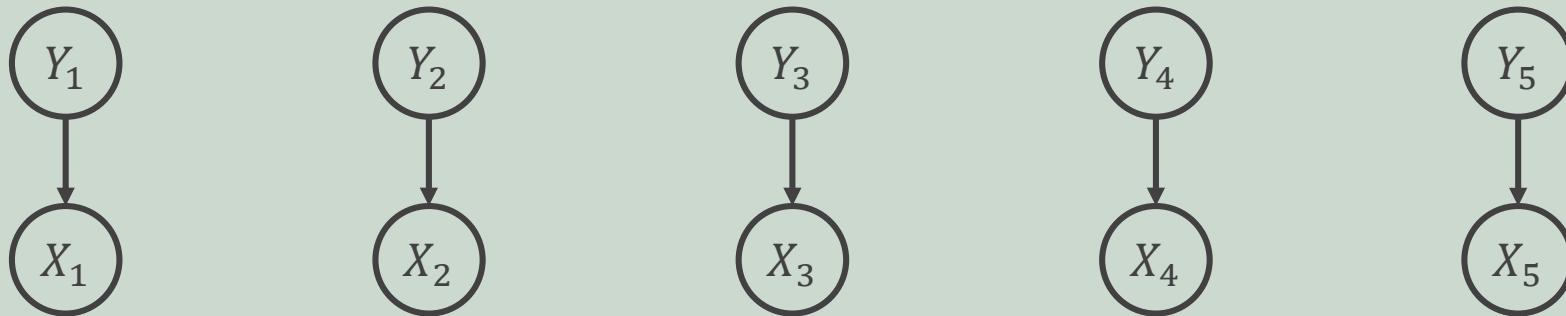


Naïve Bayes: $P(X, Y) = \prod_{t=1}^T P(X_t | Y_t) p(Y_t)$

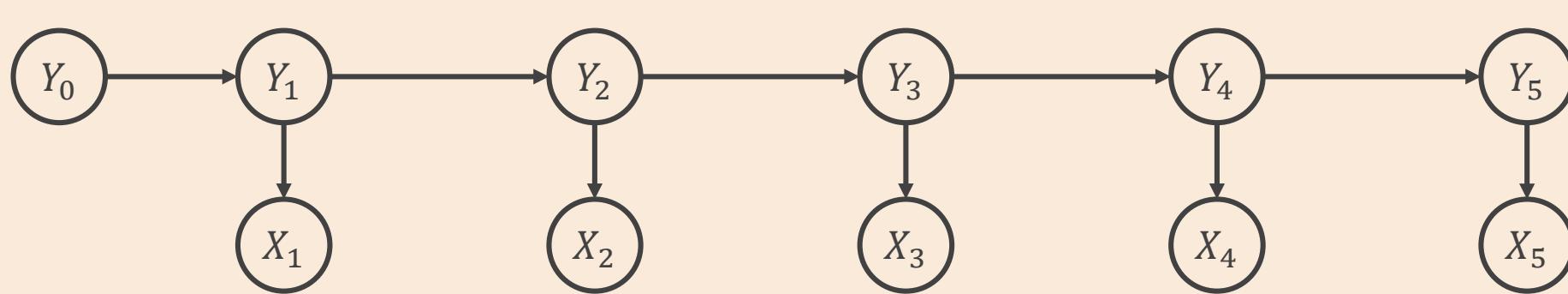


HMM: $P(X, Y | Y_0) = \prod_{t=1}^T P(X_t | Y_t) p(Y_t | Y_{t-1})$

HMM



Naïve Bayes: $P(X, Y) = \prod_{t=1}^T P(X_t | Y_t)p(Y_t)$



HMM: $P(X, Y | Y_0) = \prod_{t=1}^T P(X_t | Y_t)p(Y_t | Y_{t-1})$

Supervised Learning for HMM

- HMM Parameters:
 - Emission matrix, A , where $P(X_t = k|Y_t = j) = A_{jk}, \forall t, k$
 - Transition matrix, B , where $P(Y_t = k|Y_{t-1} = j) = B_{jk}, \forall t, k$
- Assumption: $y_0 = START$
- Generative Story:
 - $Y_t \sim Multinomial(B_{Y_{t-1}}), \forall t$
 - $X_t \sim Multinomial(A_{Y_t}), \forall t$
- Joint Distribution:
 - $p(X, Y|y_0) = \prod_{t=1}^T p(x_t|y_t)p(y_t|y_{t-1}) = \prod_{t=1}^T A_{y_t, x_t}B_{y_{t-1}, y_t}$

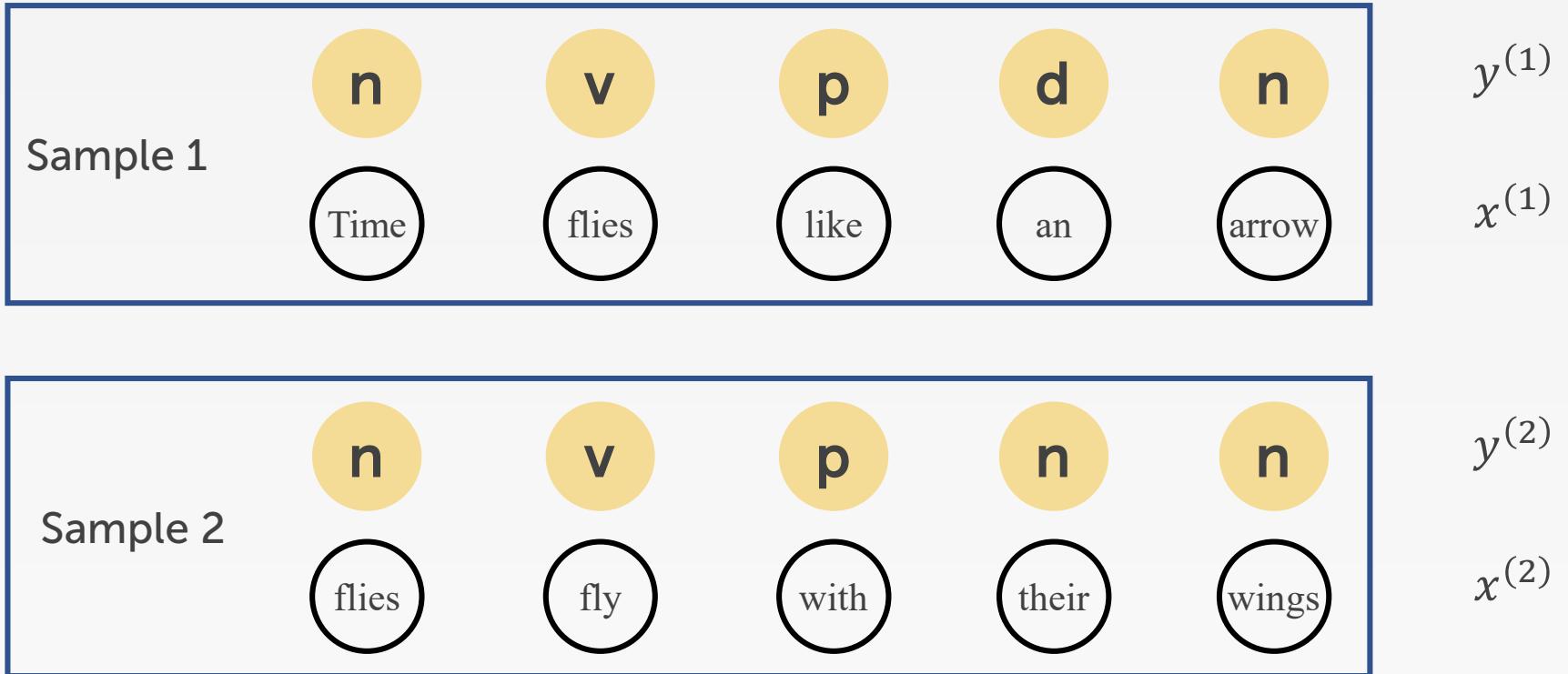
Unsupervised Learning for HMMs

- We don't observe any y 's
- This unsupervised learning setting can be achieved by finding parameters that maximize the marginal likelihood
- We optimize using the Expectation-Maximization (EM) algorithm
 - Marginal probability: $p_\theta(x) = \sum_{y \in \mathbb{Y}} p_\theta(x, y)$
 - $l(\theta) = \log \prod_{i=1}^N p_\theta(x^{(i)}) = \sum_{i=1}^N \log \sum_{y \in \mathbb{Y}} p_\theta(x^{(i)}, y)$

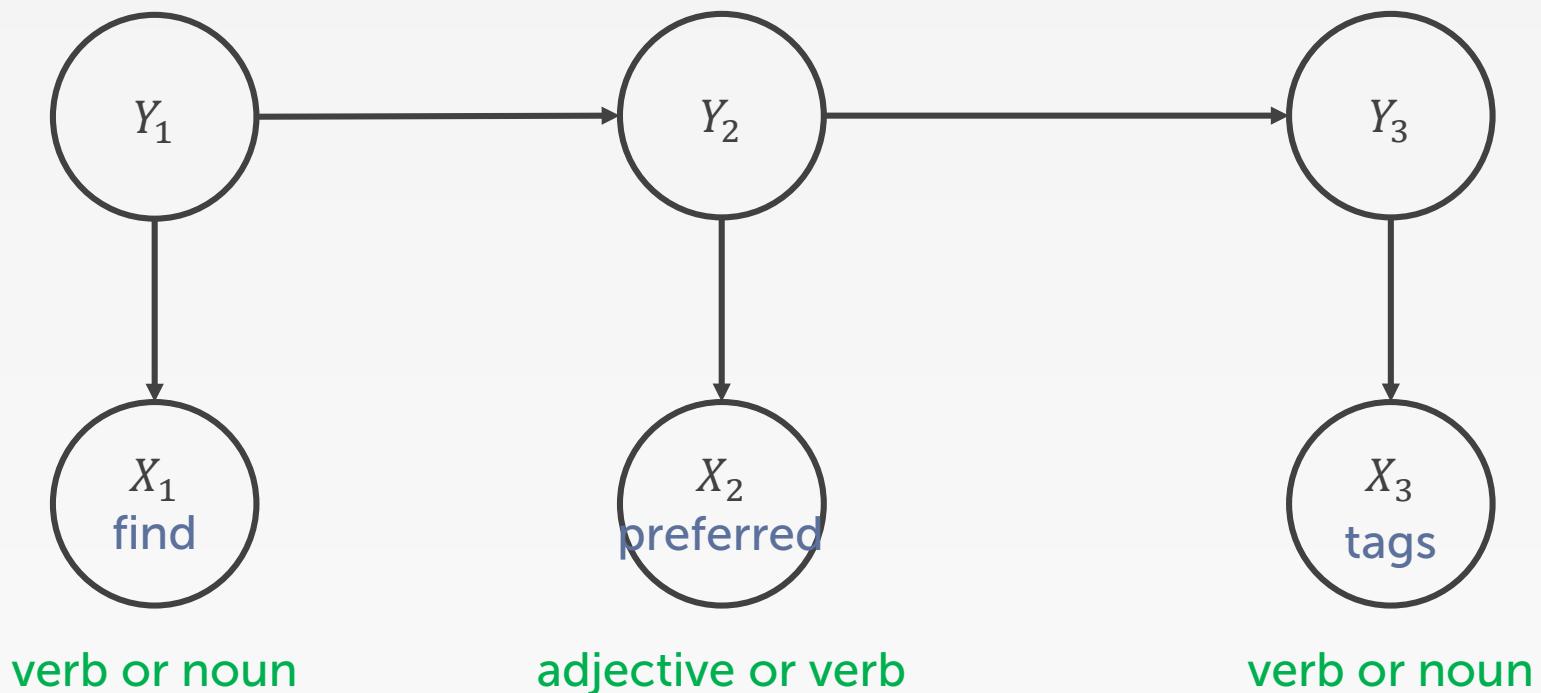
Inference for HMMs

- **Evaluation**: Compute the probability of a given sequence of observations
- **Viterbi Decoding**: Find the most-likely sequence of hidden states, given a sequence of observations
- **Learning**: find the optimal parameters to maximize the probability of the sequence of observations

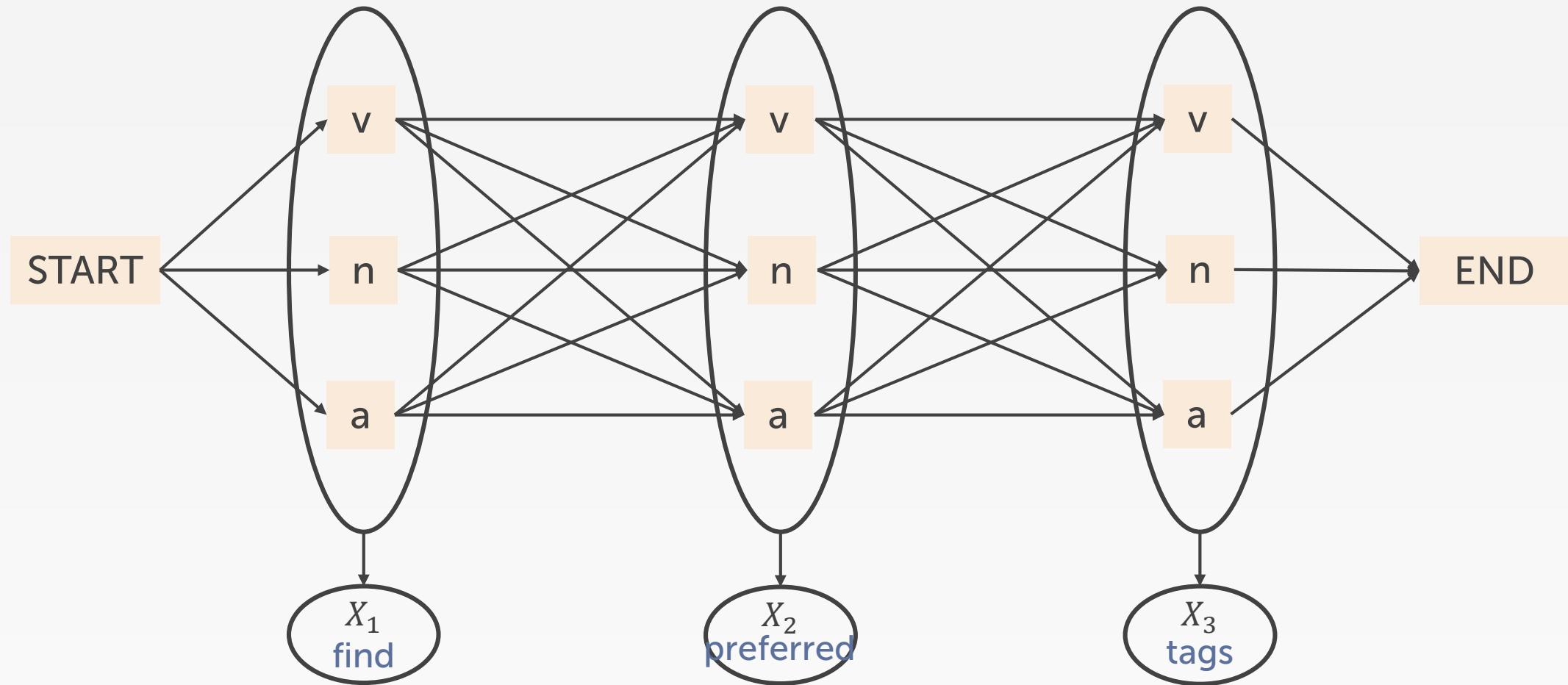
Part-of-Speech (POS) Tagging



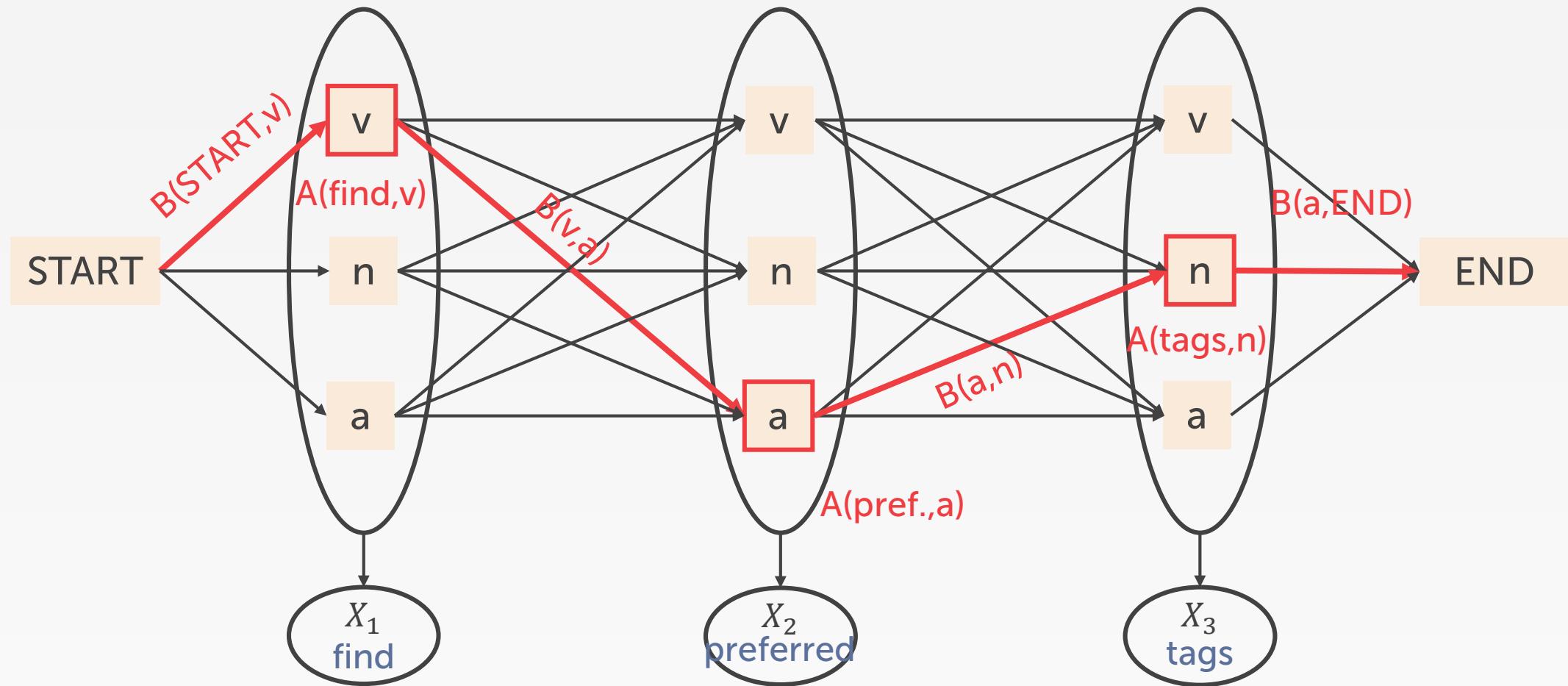
Forward-Backward Algorithm



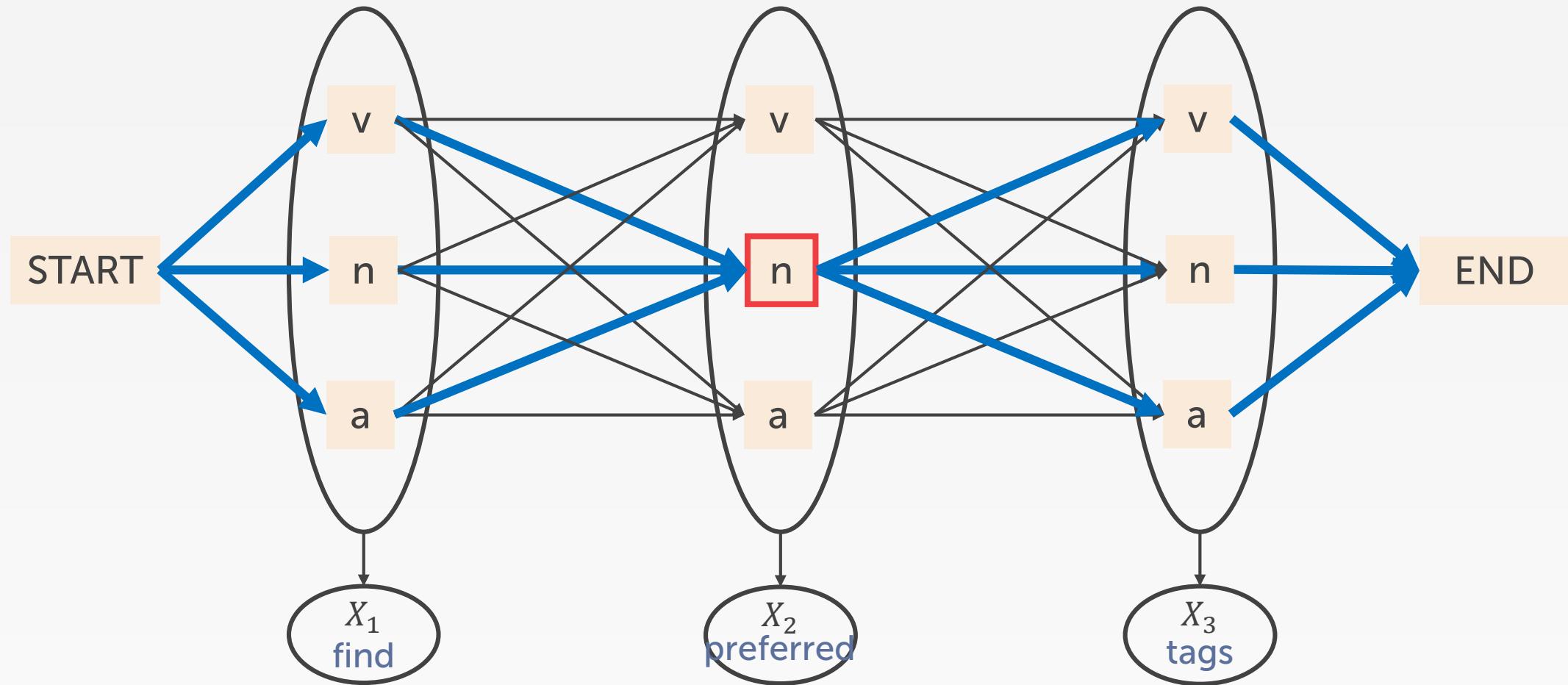
Forward-Backward Algorithm



Forward-Backward Algorithm



Forward-Backward Algorithm: Finds Marginals



Forward-Backward Algorithm

- Define $\alpha_t(k) \triangleq p(x_1, \dots, x_t, y_t = k)$, $\beta_t(k) \triangleq p(x_{t+1}, \dots, x_T | y_t = k)$
 - Assume $y_0 = START$, $y_{T+1} = END$
1. Initialize $\alpha_0(START) = \beta_{T+1}(END) = 1$, $\alpha_0(k) = 0, \forall k \neq START$, $\beta_{T+1}(k) = 0, \forall k \neq END$
 2. Forward algorithm:


```
for t = 1, ..., T:
    for k = 1, ..., K:
         $\alpha_t(k) = p(x_t | y_t = k) \sum_{j=1}^K \alpha_{t-1}(j) p(y_t = k | y_{t-1} = j)$ 
```
 3. Backward algorithm:


```
for t = T, ..., 1:
    for k = 1, ..., K:
         $\beta_t(k) = \sum_{j=1}^K p(x_{t+1} | y_{t+1} = j) \beta_{t+1}(j) p(y_{t+1} = j | y_t = k)$ 
```
 4. Evaluation: $p(\vec{x}) = \alpha_{T+1}(END)$
 5. Marginal: $p(y_t = k | \vec{x}) = \frac{\alpha_t(k) \beta_t(k)}{p(\vec{x})}$

Viterbi Algorithm (Decoding)

- Define $\omega_t(k) \triangleq \max_{y_1, \dots, y_{t-1}} p(x_1, \dots, x_t, y_1, \dots, y_t = k),$

$$b_t(k) \triangleq \operatorname{argmax}_{y_1, \dots, y_{t-1}} p(x_1, \dots, x_t, y_1, \dots, y_t = k)$$

- Assume $y_0 = START$

1. Initialize $\omega_0(START) = 1, \omega_0(k) = 0, \forall k \neq START$

2. For $t = 1, \dots, T$:

for $k = 1, \dots, K$:

$$\omega_t(k) = \max_{j \in \{1, \dots, K\}} p(x_t | y_t = k) \omega_{t-1}(j) p(y_t = k | y_{t-1} = j)$$

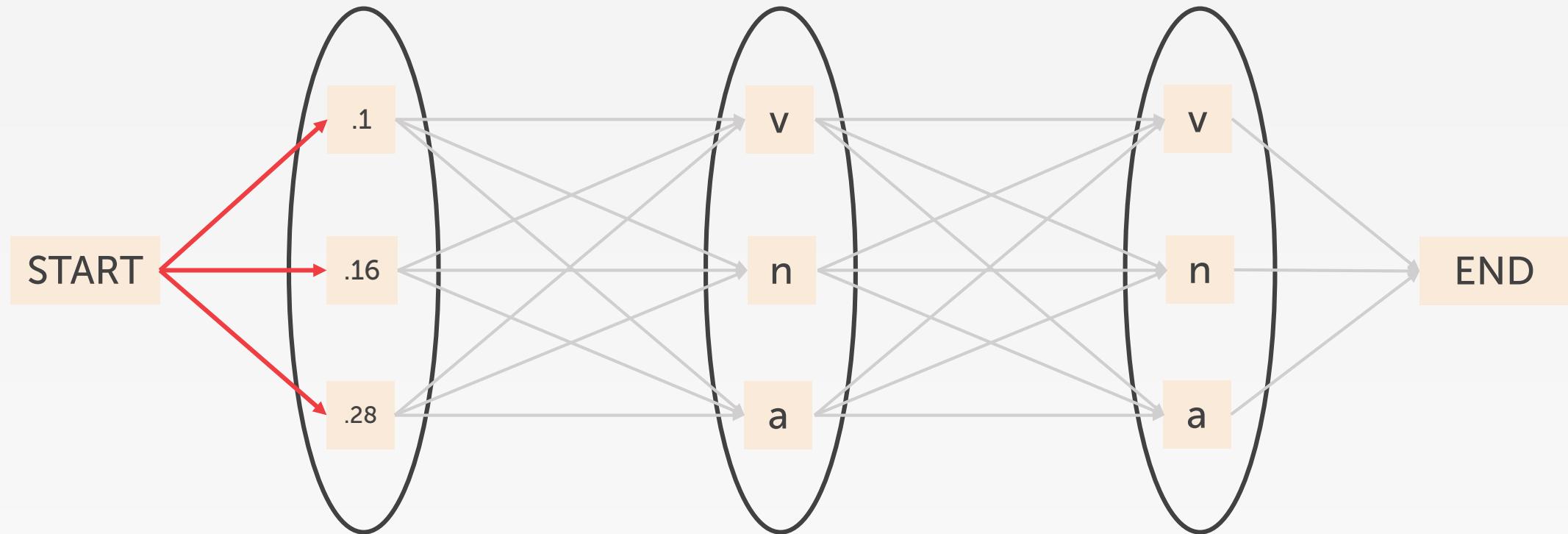
$$b_t(k) = \operatorname{argmax}_{j \in \{1, \dots, K\}} p(x_t | y_t = k) \omega_{t-1}(j) p(y_t = k | y_{t-1} = j)$$

3. Compute most probable assignment

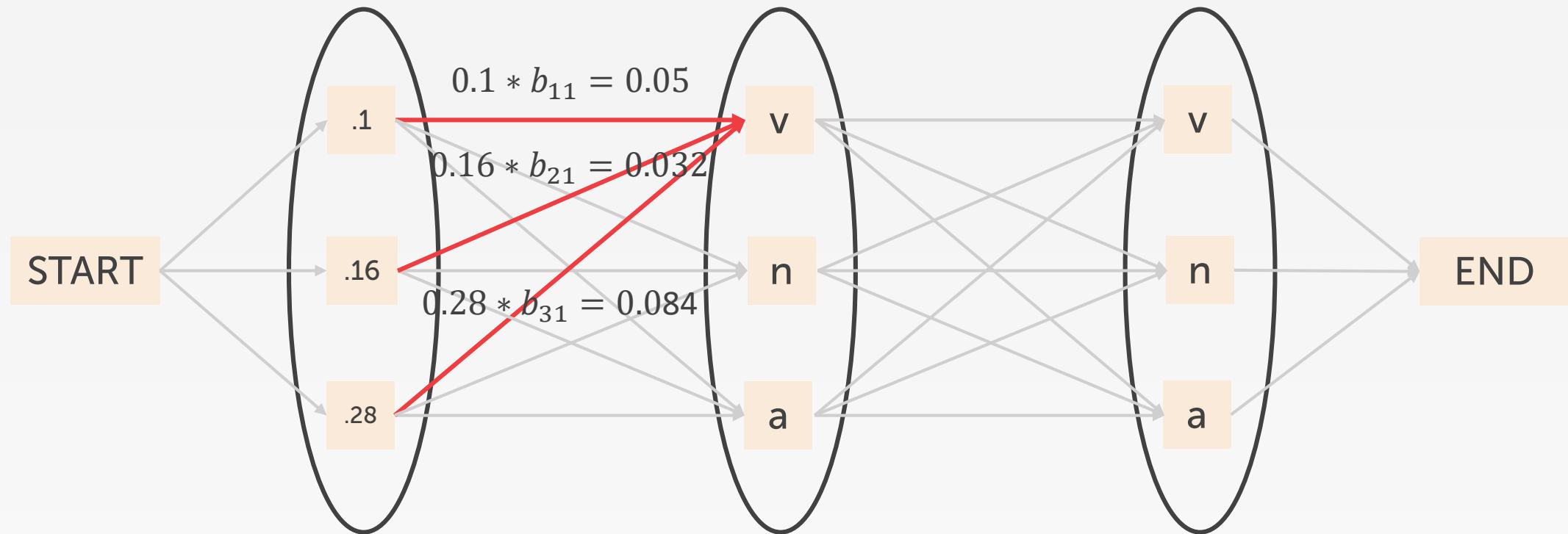
$$\widehat{y}_T = b_{T+1}(END)$$

$$\text{for } t = T - 1, \dots, 1: \widehat{y}_t = b_{t+1}(\widehat{y}_{t+1})$$

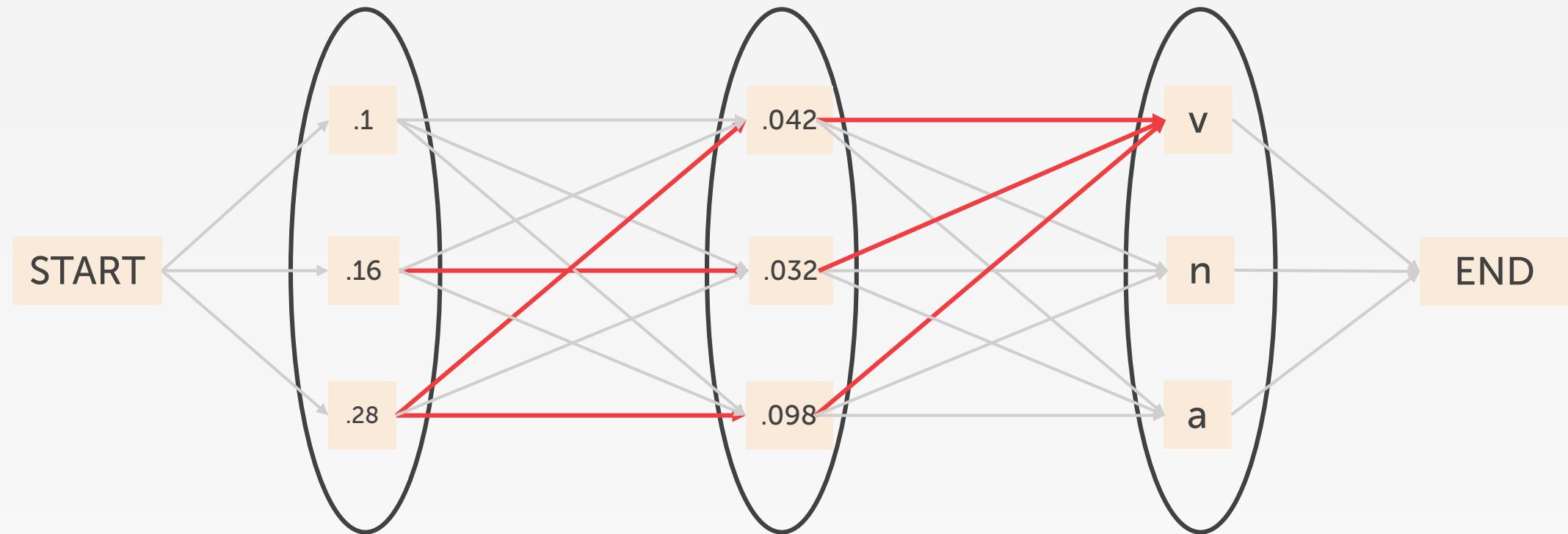
Example: Viterbi Algorithm



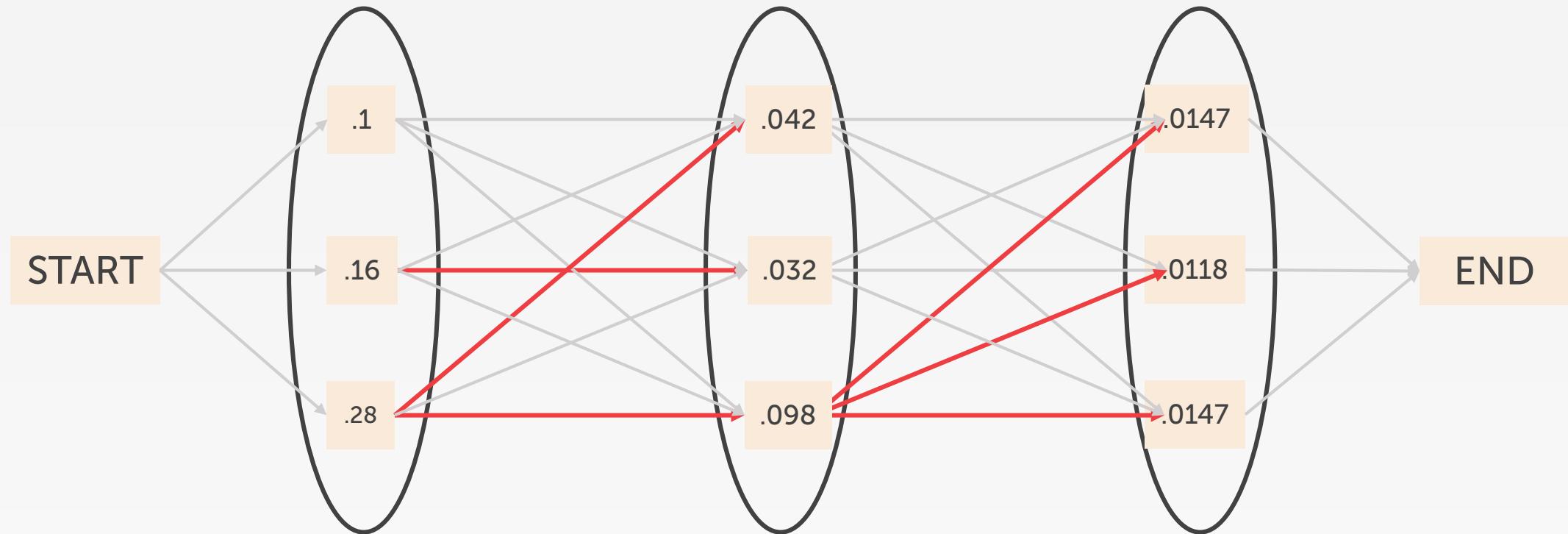
Example: Viterbi Algorithm



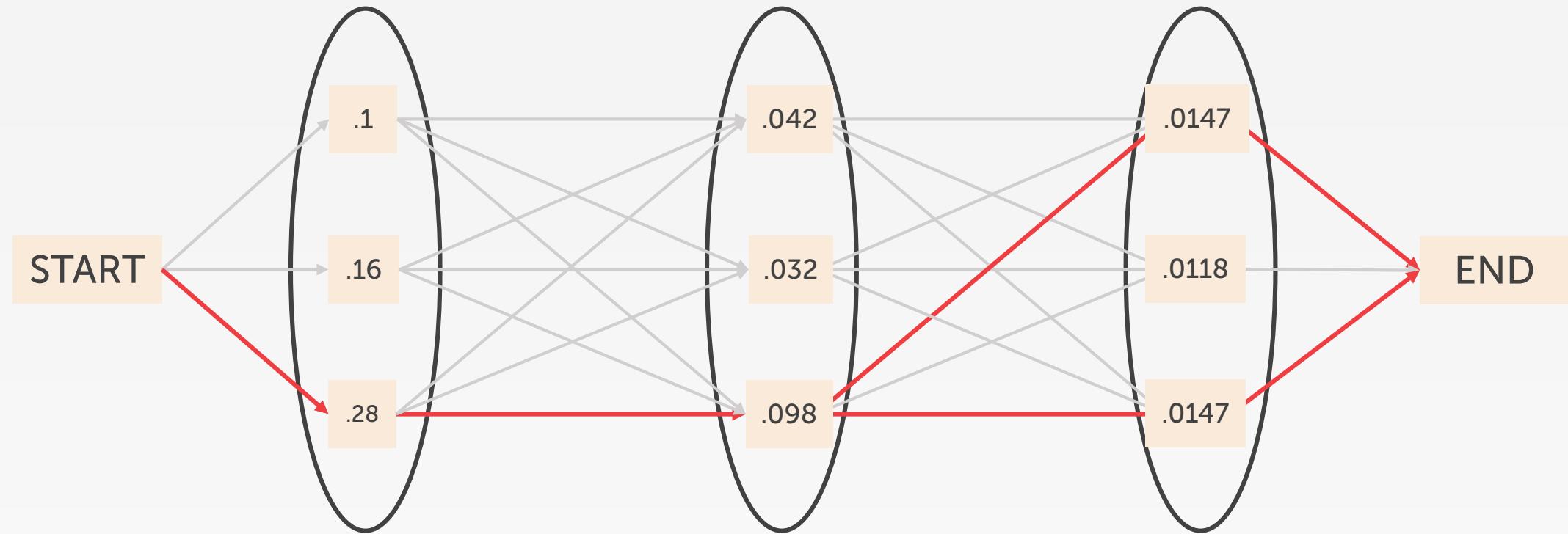
Example: Viterbi Algorithm



Example: Viterbi Algorithm



Example: Viterbi Algorithm



Unsupervised Learning

Learning Paradigms

Paradigm

Supervised

↪ Regression

↪ Classification

↪ Binary classification

Unsupervised

↪ Clustering

↪ Dimensionality Reduction

Semi-supervised

Reinforcement Learning

Data

$$\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^N$$

$$y^{(i)} \in \mathbb{R}$$

$$y^{(i)} \in \{1, \dots, K\}$$

$$y^{(i)} \in \{+1, -1\}$$

$$\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N \quad \boldsymbol{x} \sim p^*(\cdot)$$

Predict $\{z^{(i)}\}_{i=1}^N$ where $z^{(i)} \in \{1, \dots, K\}$

Convert each $\boldsymbol{x}^{(i)} \in \mathbb{R}^M$ to $\boldsymbol{u}^{(i)} \in \mathbb{R}^K$ with $K \ll M$

$$\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\boldsymbol{x}^{(j)}\}_{j=1}^{N_2}$$

$$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \dots\}$$

Goals

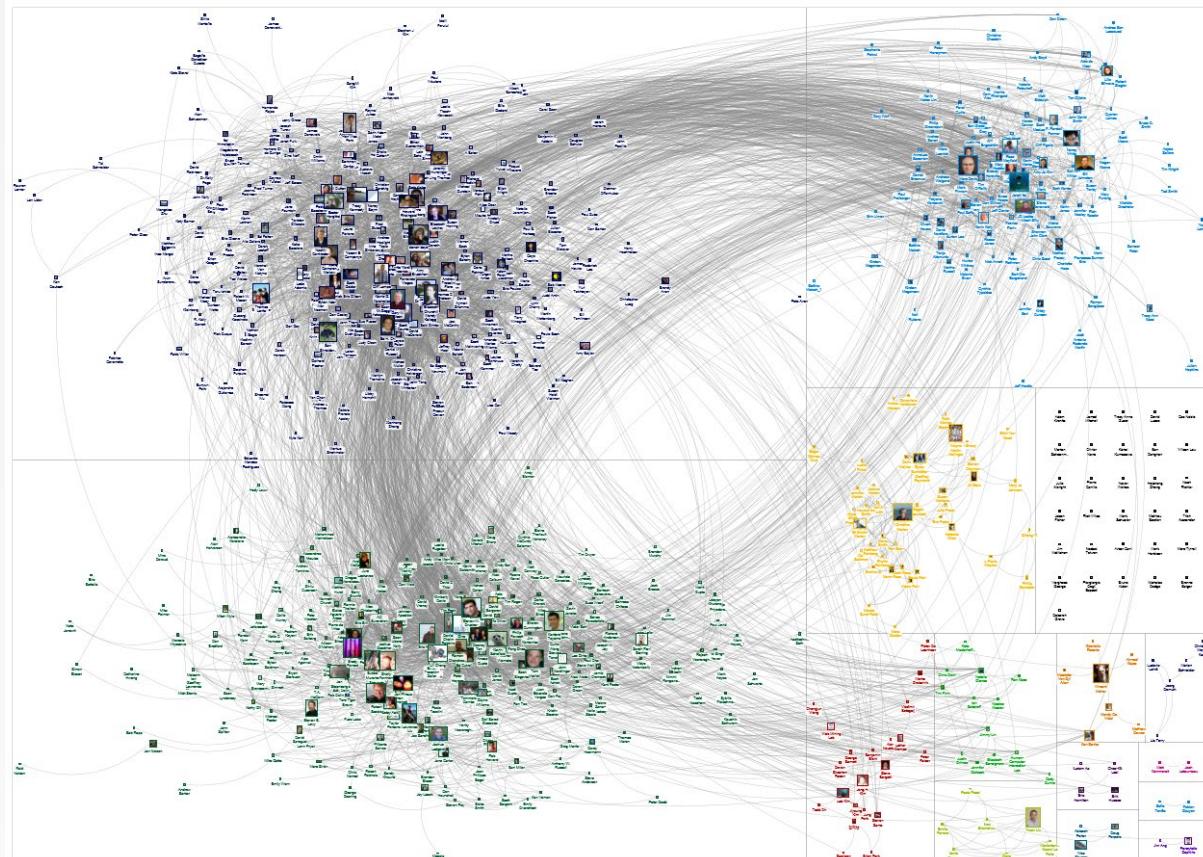
- To discover interesting things from the data:
 - Is there an informative way to visualize the data?
 - Can we discover subgroups among the variables?
- Models:
 - Clustering
 - K-means
 - DBSCAN
 - Hierarchical Clustering

Clustering

Clustering

- Partition **unlabeled** data into groups (clusters)
- Points within a cluster should be “similar”
- Points in different clusters should be “different”

Applications



K-Means

Overview

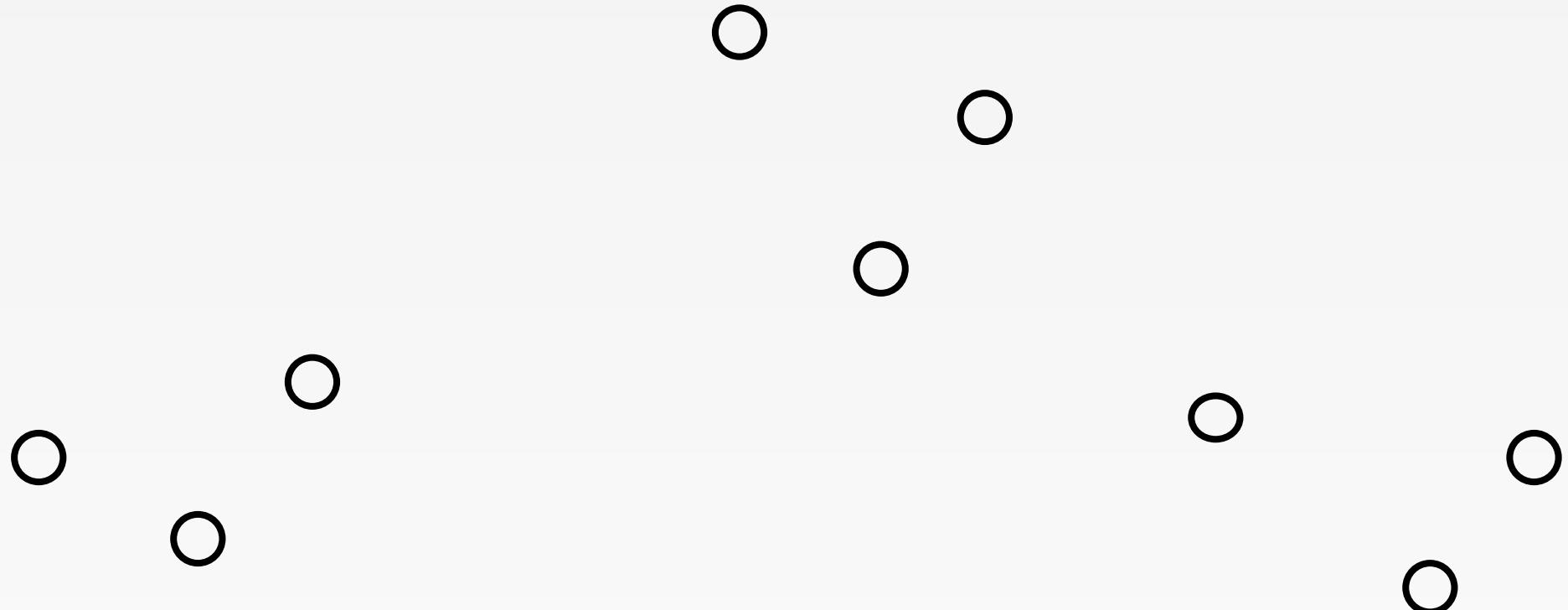
- K-means (MacQueen, 1967)
- Each cluster has a cluster center, called centroid
- K is specified by the user

K-means Algorithm

- Given K and unlabeled feature vectors $D = \{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$
- Initialize cluster center $c = \{c^{(1)}, \dots, c^{(K)}\}$ and cluster assignments $z = \{z^{(1)}, z^{(2)}, \dots, z^{(N)}\}$
- Repeat until convergence:
 - For j in $\{1, \dots, K\}$
 $c^{(j)}$ is the mean of all points assigned to cluster j
 - for i in $\{1, \dots, N\}$
 $z^{(i)}$ is the index j of cluster center nearest to $x^{(i)}$

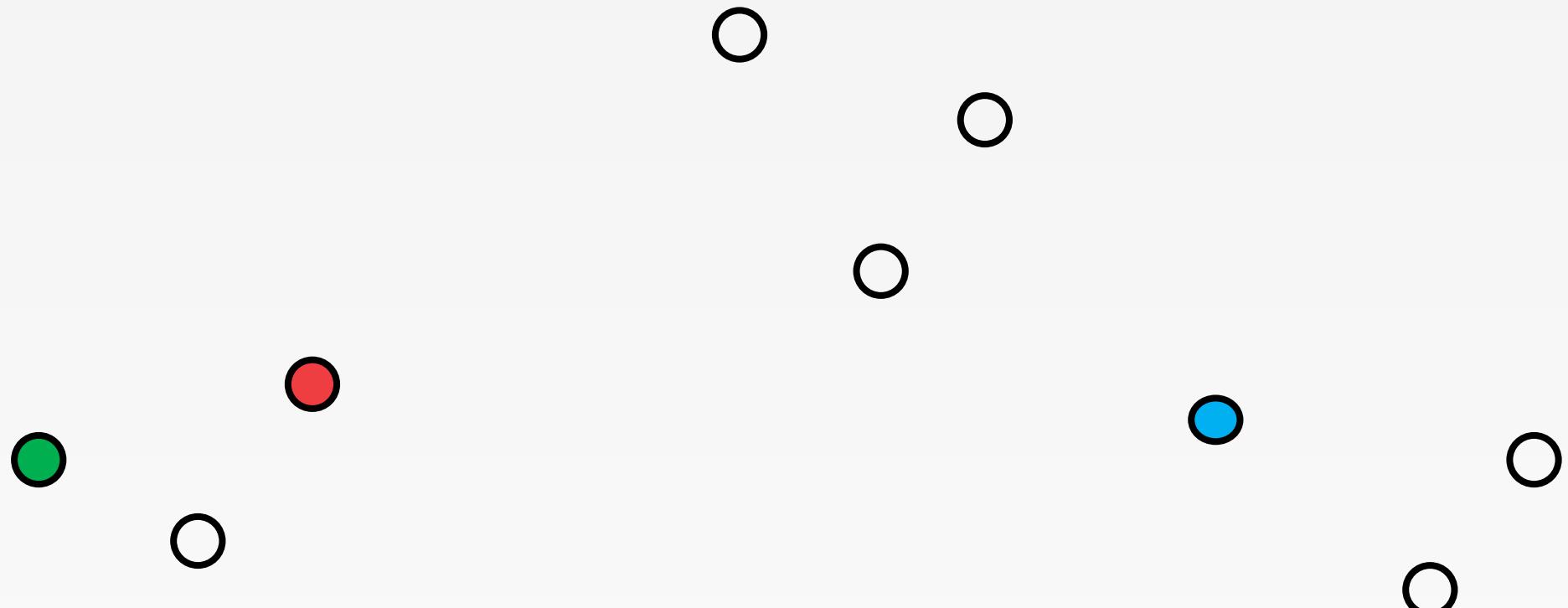
Illustrative Example

Given a set of data points



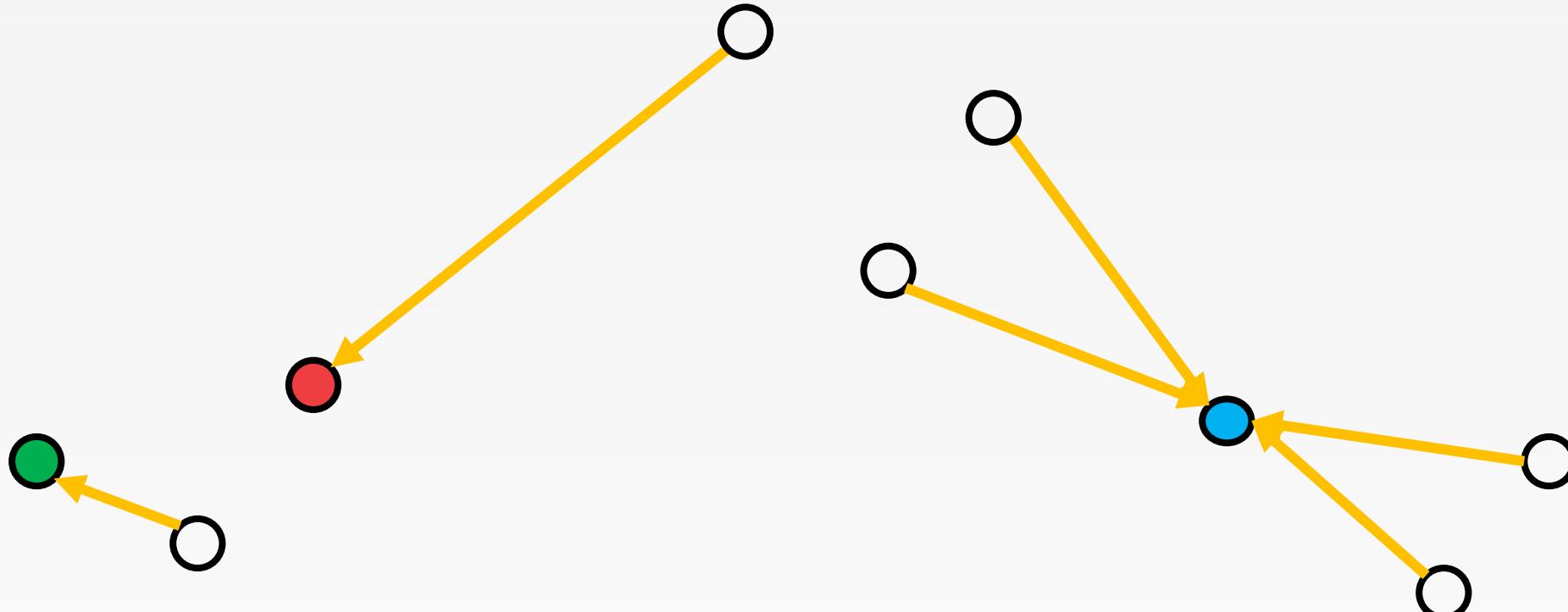
Illustrative Example

Select initial centers at random ($k=3$)



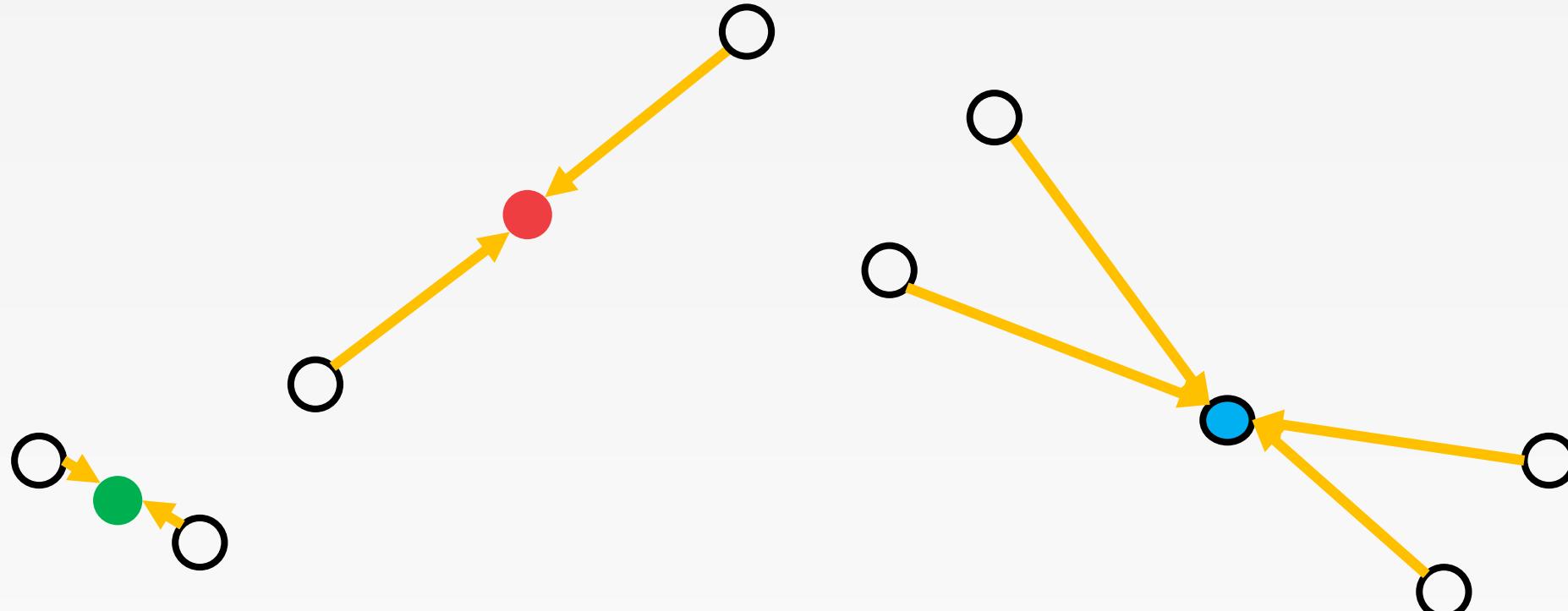
Illustrative Example

Assign each point to its nearest center



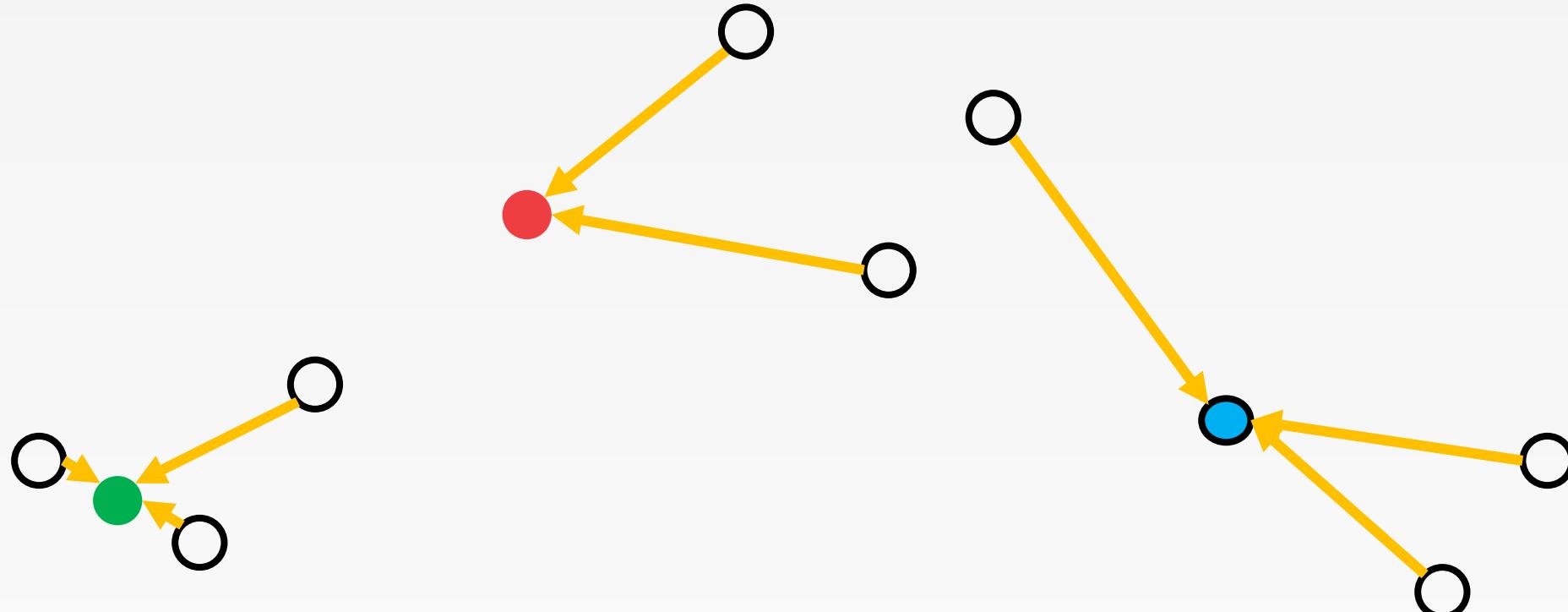
Illustrative Example

Recompute optimal centers given a fixed clustering



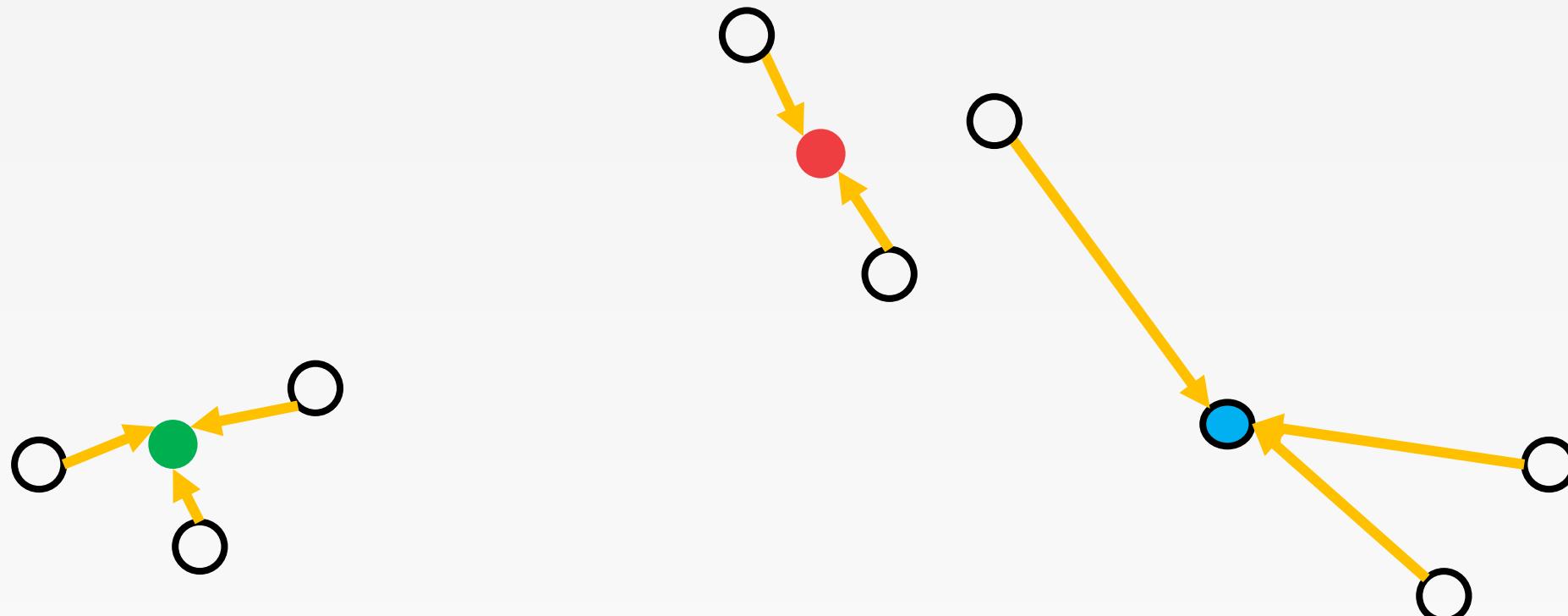
Illustrative Example

Assign each point to its nearest center



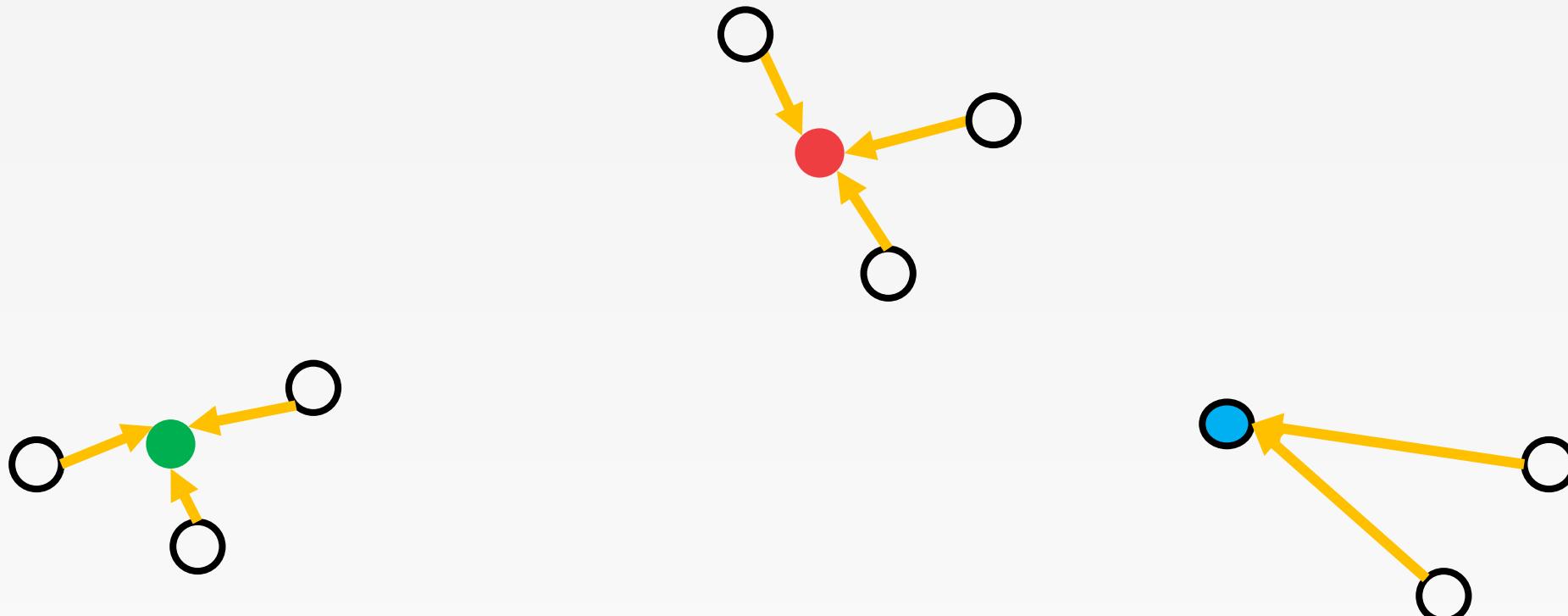
Illustrative Example

Recompute optimal centers given a fixed clustering



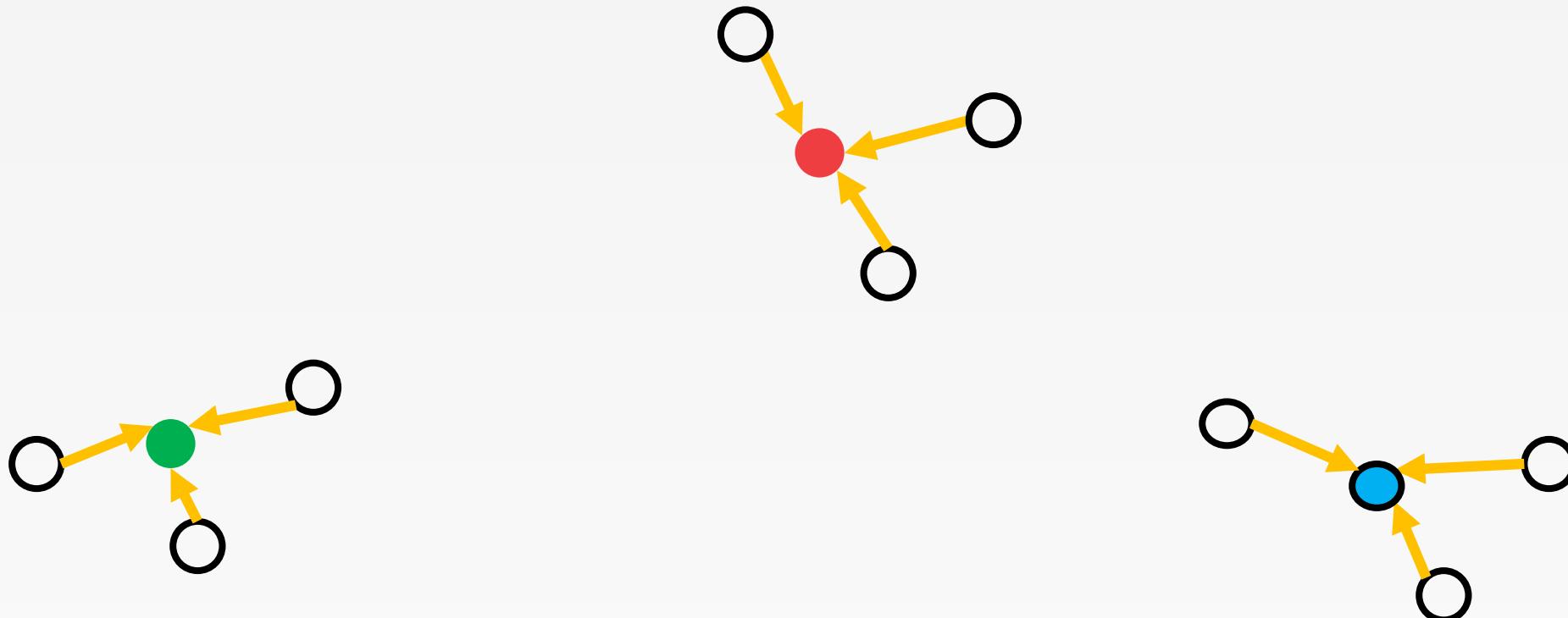
Illustrative Example

Assign each point to its nearest center



Illustrative Example

Recompute optimal centers given a fixed clustering



Measure the Distance

- Similarity measure (distance measure)

- Euclidean distance $d(x, y) = \sqrt{(x - y)^2} = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$
- Manhattan distance $d(x, y) = |x - y| = \sum_{i=1}^d |x_i - y_i|$

Stopping Criterion

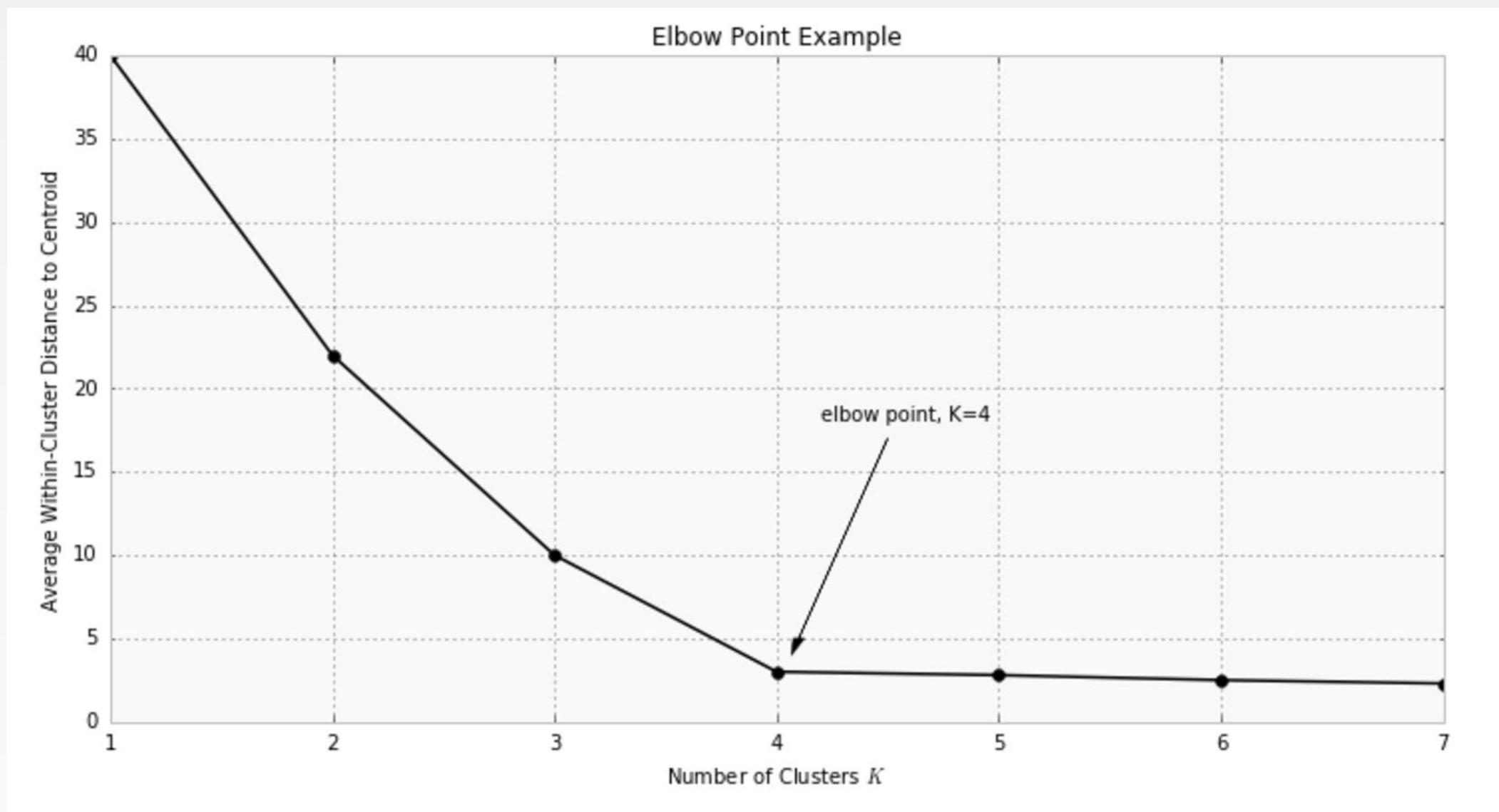
- no (or minimum) re-assignments of data points to different clusters, *or*
- no (or minimum) change of centroids, *or*
- minimum decrease in the **sum of squared error(SSE)**,

How to choose k?

Elbow method:

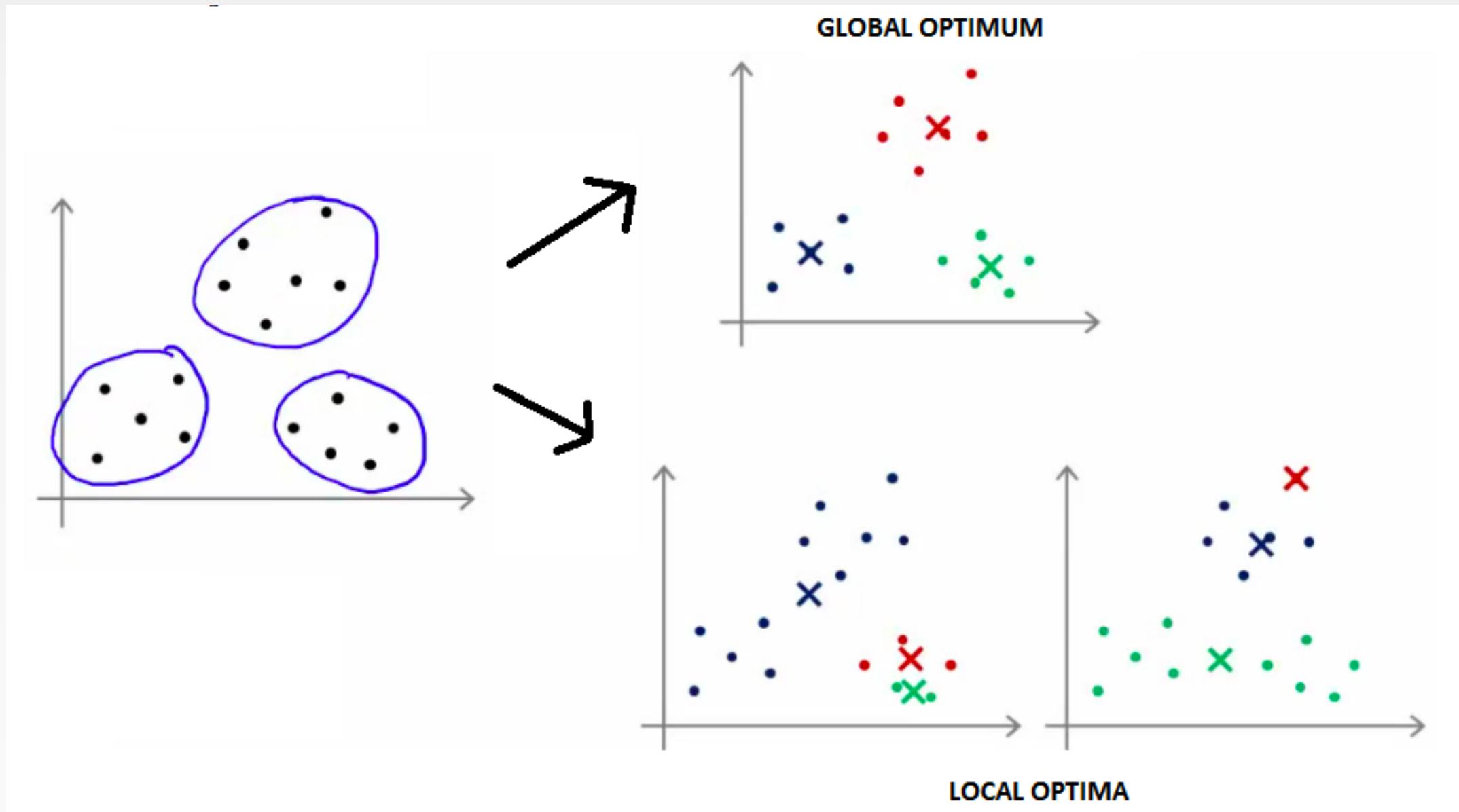
run k-means clustering on the dataset for a range of values of k
for each value of k calculate the sum of squared errors (SSE)

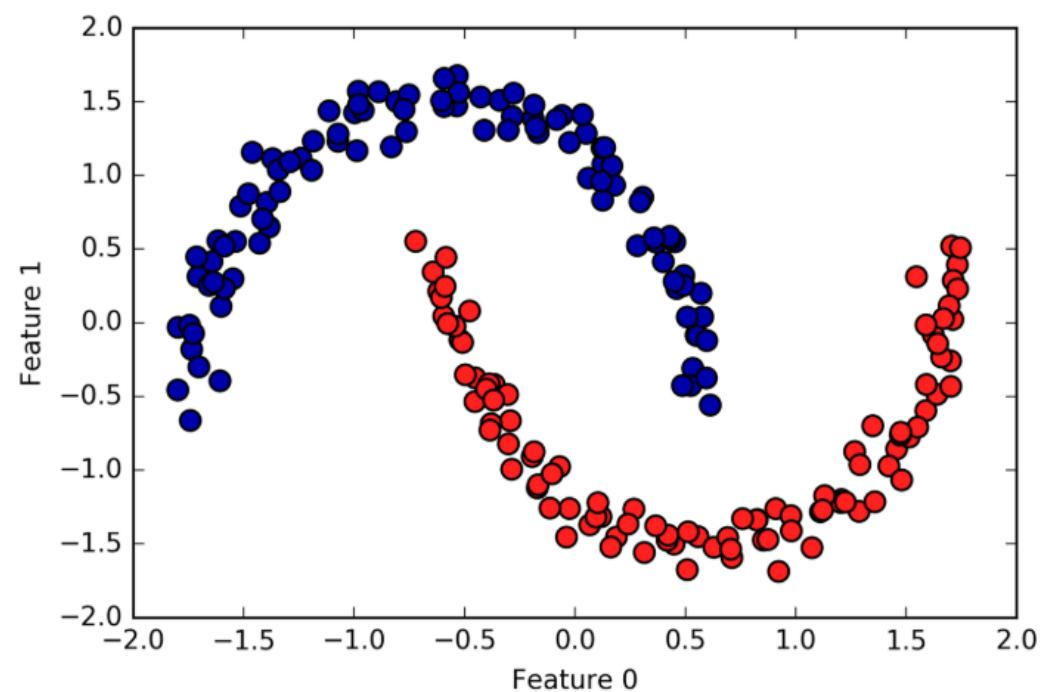
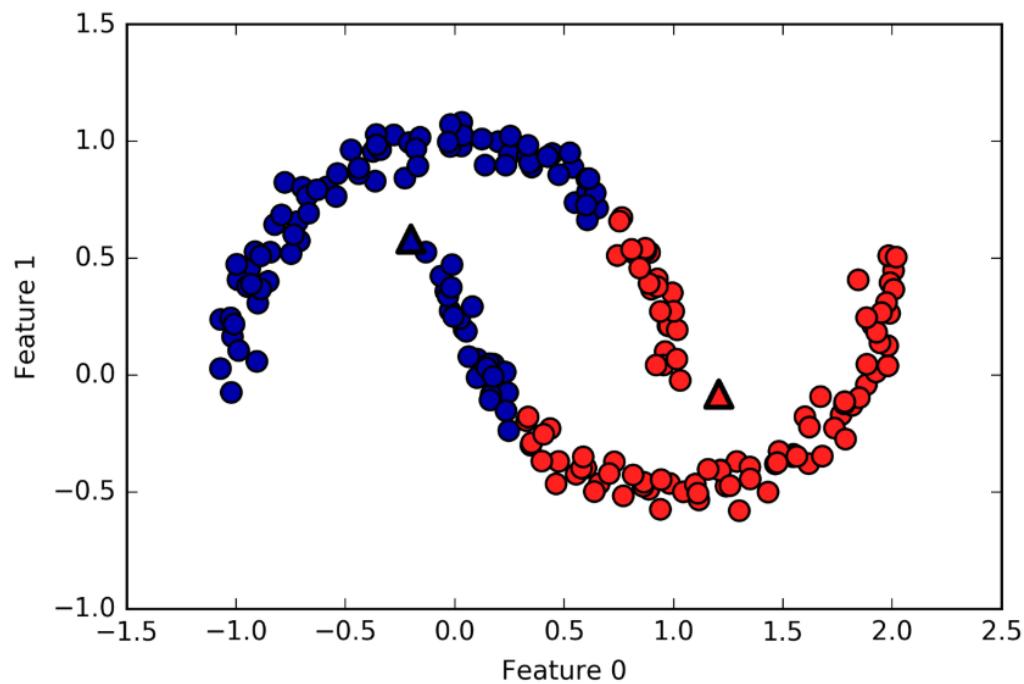
If the line chart looks like an arm, then the "elbow" on the arm is the
value of k that is the best



Pros and Cons

- Strengths:
 - Simple: each to understand and to implement
 - Efficient
- Weakness:
 - The algorithm is sensitive to outliers
 - it terminates at a **local optimum** if SSE is used. The global optimum is hard to find due to complexity
 - Might be sensitive to initial seeds
 - Only simple cluster shapes





DBSCAN

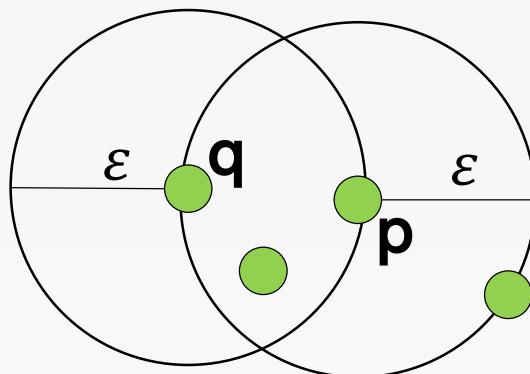
Density-Based Spatial Clustering of Applications with Noise

Density-based Clustering

- Basic Idea:
 - Clusters are dense regions in the data space, separated by regions of lower object density
 - A cluster is defined as a maximal set of density-connected points

Density Definition

- ε -Neighborhood – Objects within a radius of ε from an object
 $N_\varepsilon(p): \{q | d(p, q) \leq \varepsilon\}$
- “High density” -- ε -Neighborhood of an object contains at least $MinPts$ of objects

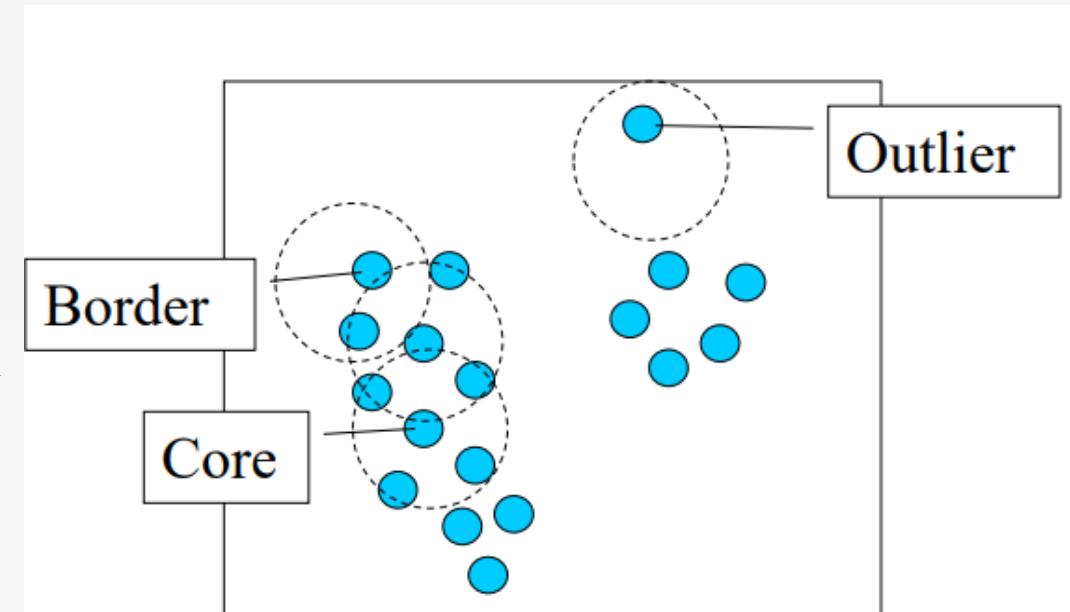


Density of p is “high” ($MinPts = 4$)

Density of q is “low” ($MinPts = 3$)

Core, Border, Outlier

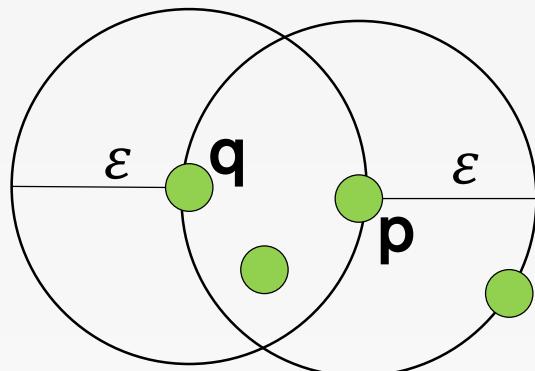
- Given ε and $MinPts$, categorize the objects into three exclusive groups:
 - Core point**: has more than $MinPts$ points within ε (these are points that are at the interior of a cluster)
 - Border point**: has fewer than $MinPts$ within ε , but is the neighborhood of a core point
 - Noise point**: any point that is neither a core nor a border point



$\varepsilon = 1\text{unit}$, $MinPts = 5$

Density-reachability

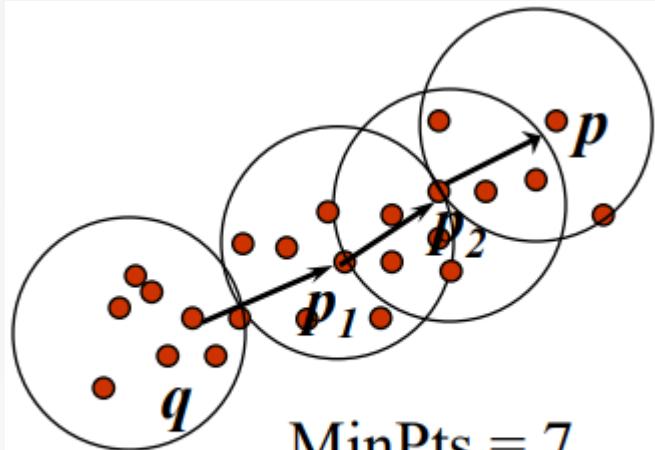
- An object q is directly density-reachable from object p if p is a core object and q is in p's ε -neighborhood.



$MinPts=4$

q is directly density-reachable from p
p is not directly density-reachable from q
Density-reachability is asymmetric

Density-reachability



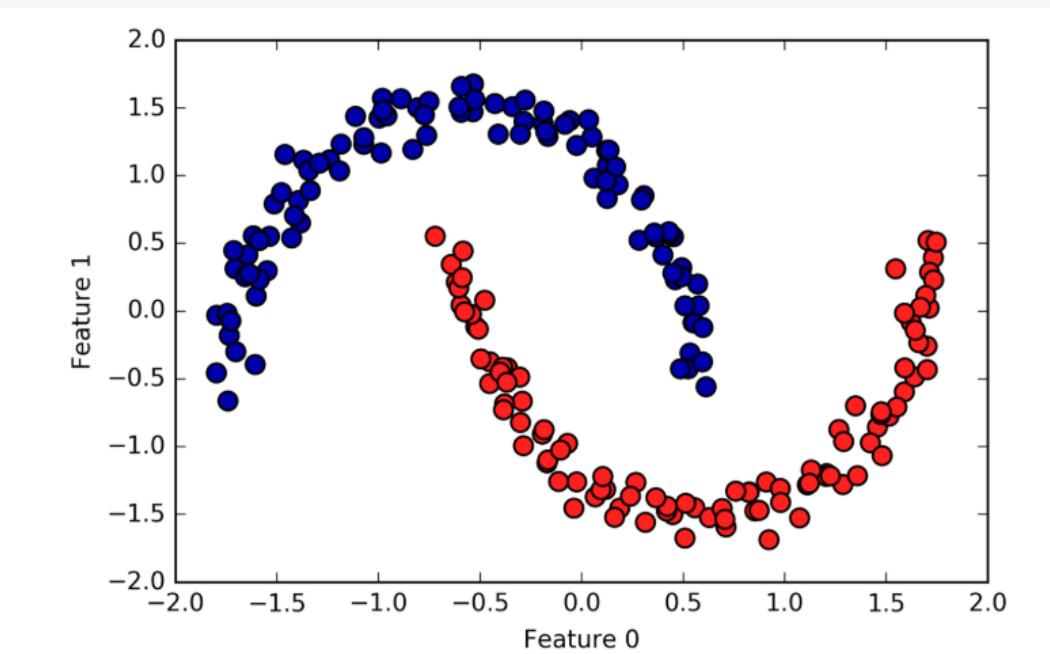
A point p is directly density-reachable from p_2
 p_2 is directly density-reachable from p_1
 p_1 is directly density-reachable from q
 $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain

DBSCAN Algorithm

```
for each  $o \in D$  do
    if  $o$  is not yet classified then
        if  $|o$ 's  $\varepsilon$ -neighborhood| <  $MinPts$ 
            assign  $o$  to  $NOISE$ 
        else
            collect all objects density-reachable from  $o$ 
            and assign them to a new cluster
```

Pros and Cons

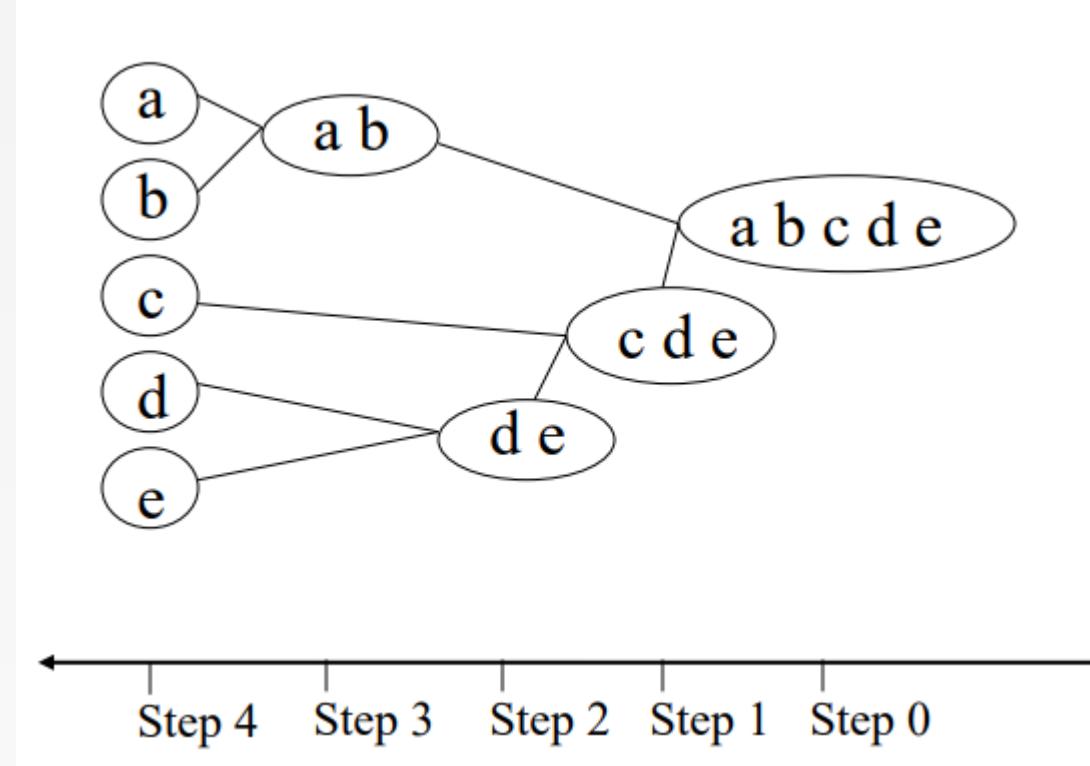
- Can learn arbitrary cluster shapes (resistant to noise)
- Can detect outliers
- Needs two parameters to adjust



Hierarchical Clustering

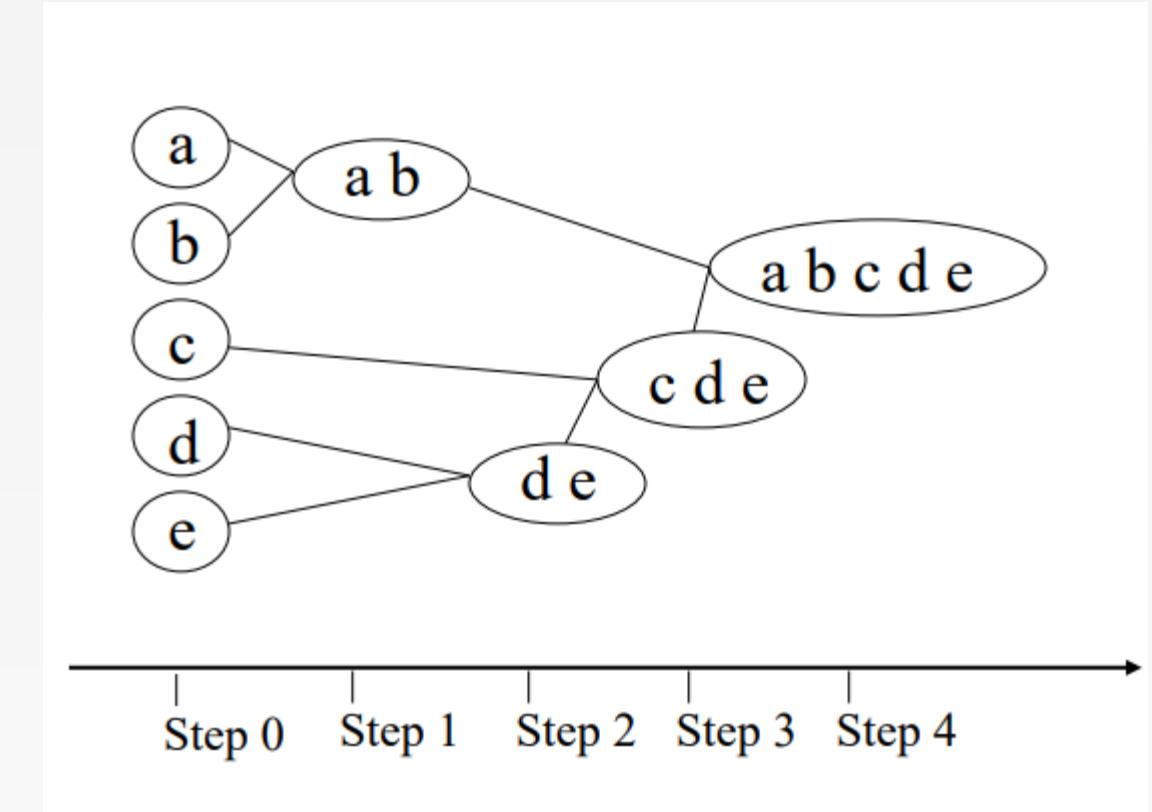
Types

- **Divisive (top-down) clustering**
 - All objects in one cluster
 - Select a cluster and split it into two sub clusters
 - Until each leaf cluster contains only one object



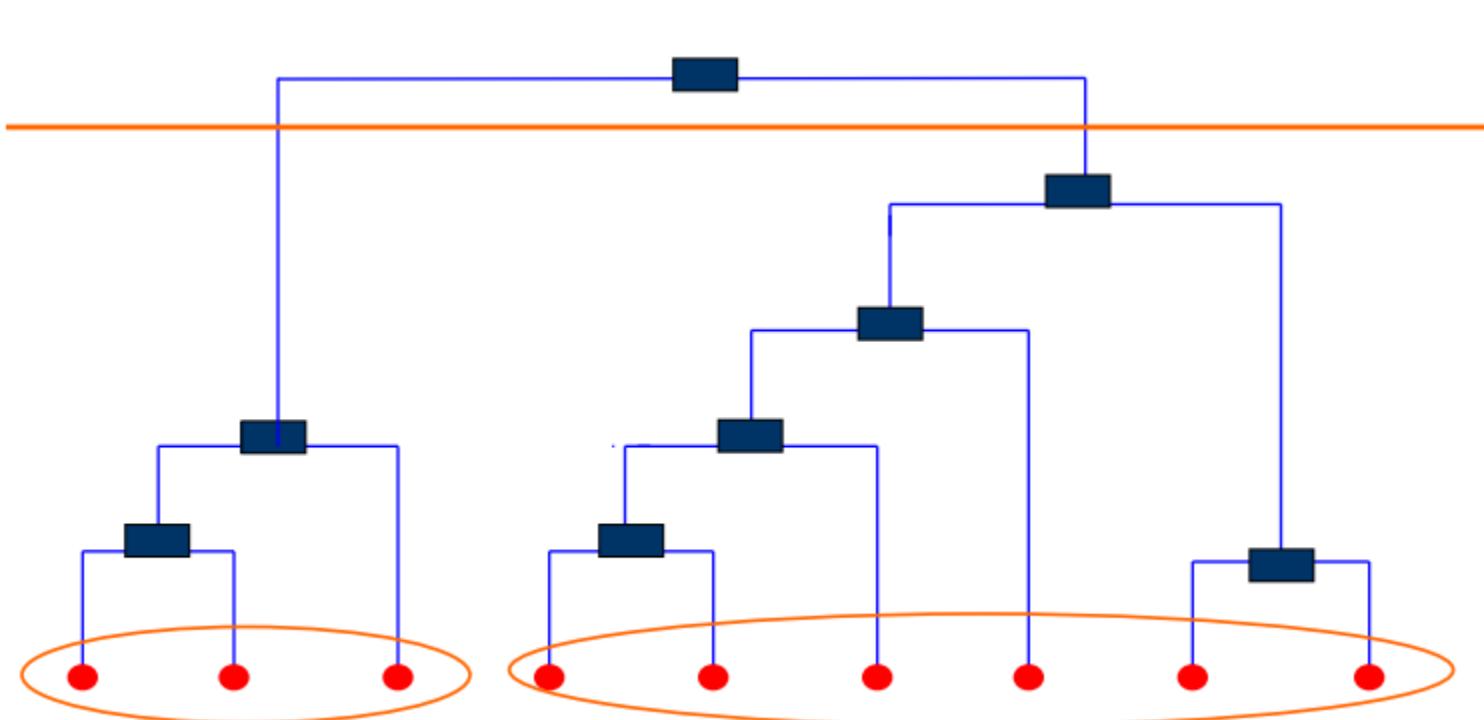
Types

- Agglomerative (bottom-up) clustering
 - Each object is a cluster
 - Merge two clusters which are most similar to each other
 - Until all objects are merged into a single cluster

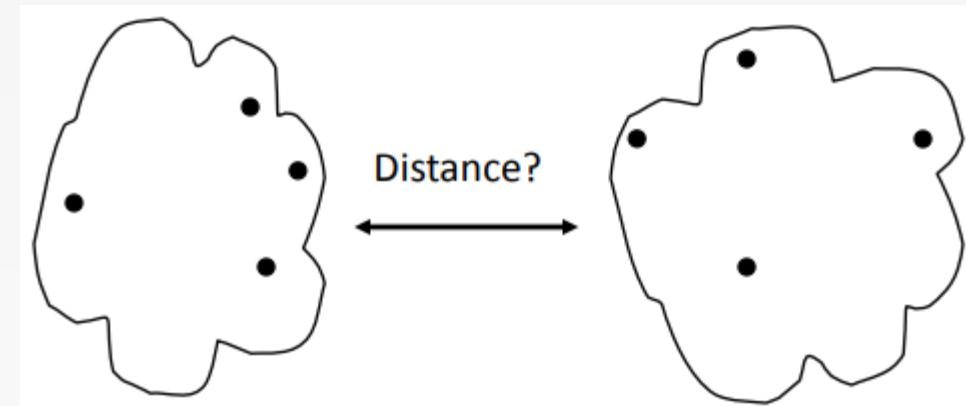


Dendrogram

- A tree that shows how clusters are merged/split hierarchically
- Each node on the tree is a cluster; each leaf node is a singleton cluster
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster

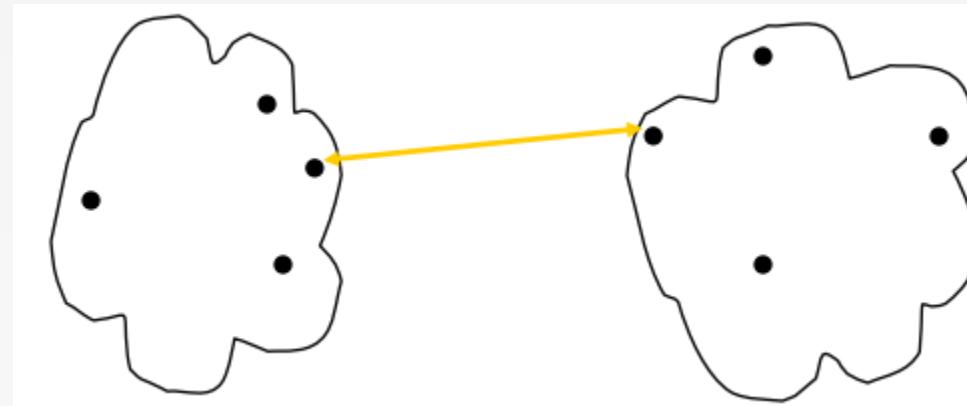


Inter-Cluster Distance

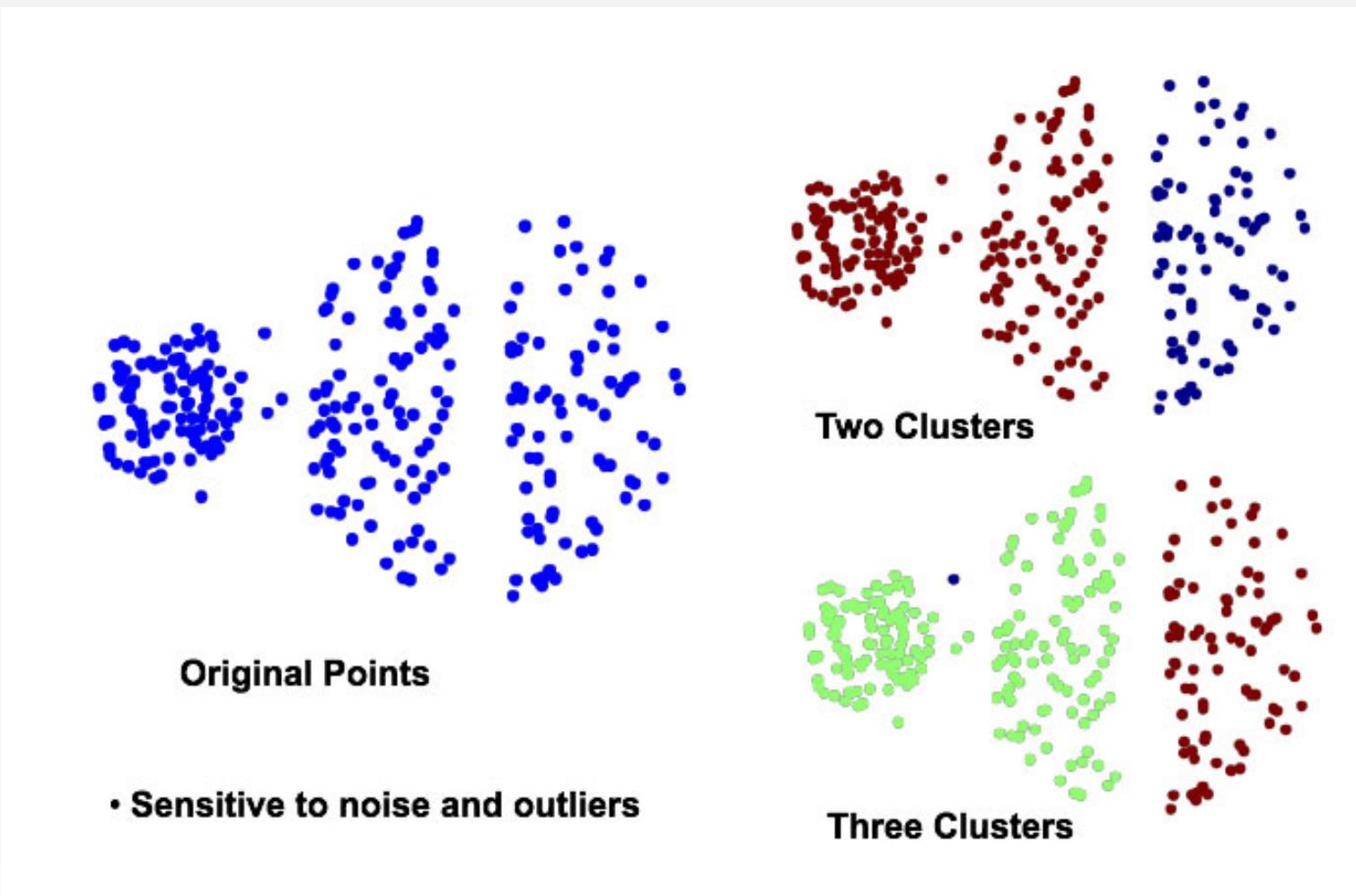


MIN (Single Link)

- The distance between two clusters is represented by the distance of the closest pair of data objects belonging to different clusters.
- Determined by one pair of points, i.e., by one link in the proximity graph

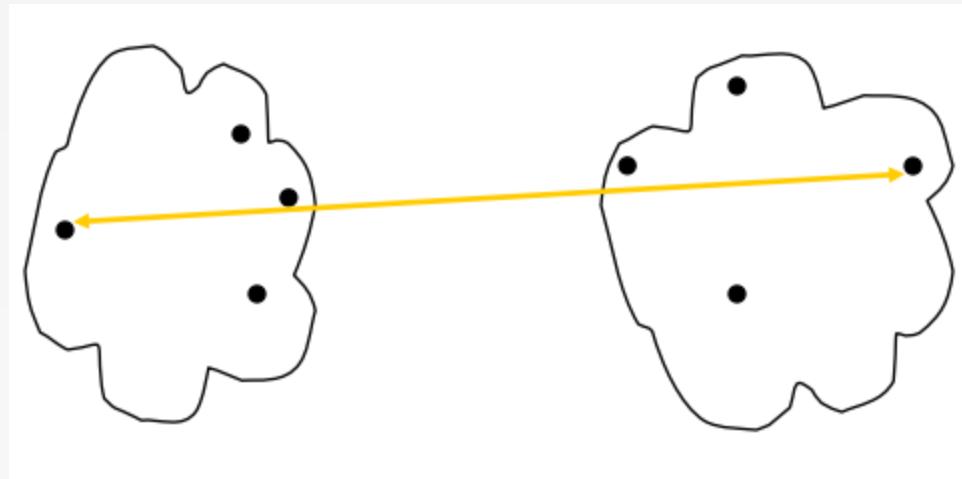


- Limitation: sensitive to noise/outliers



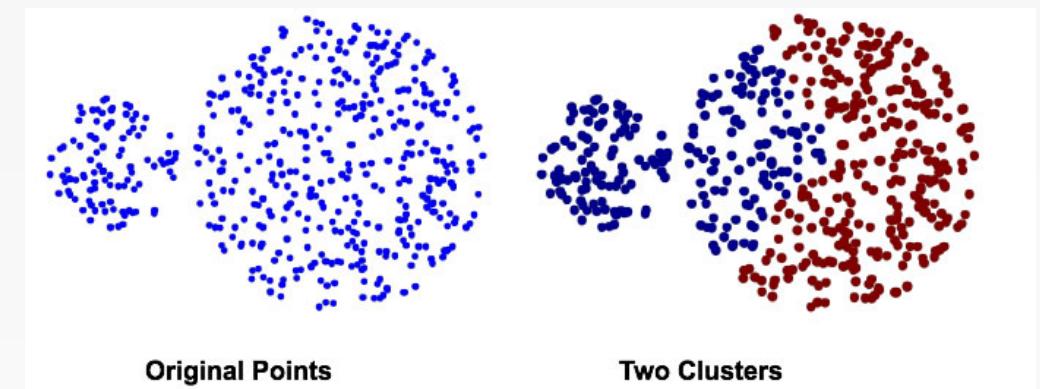
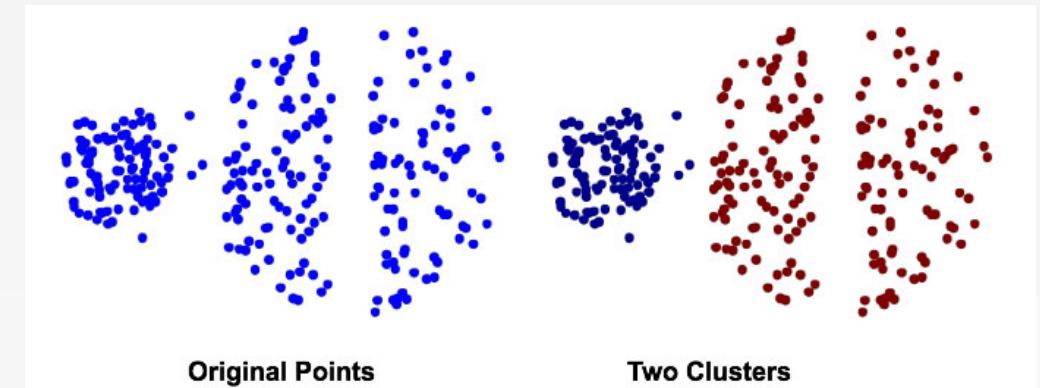
MAX (Complete link)

- The distance between two clusters is represented by the distance of the farthest pair of data objects belonging to different clusters



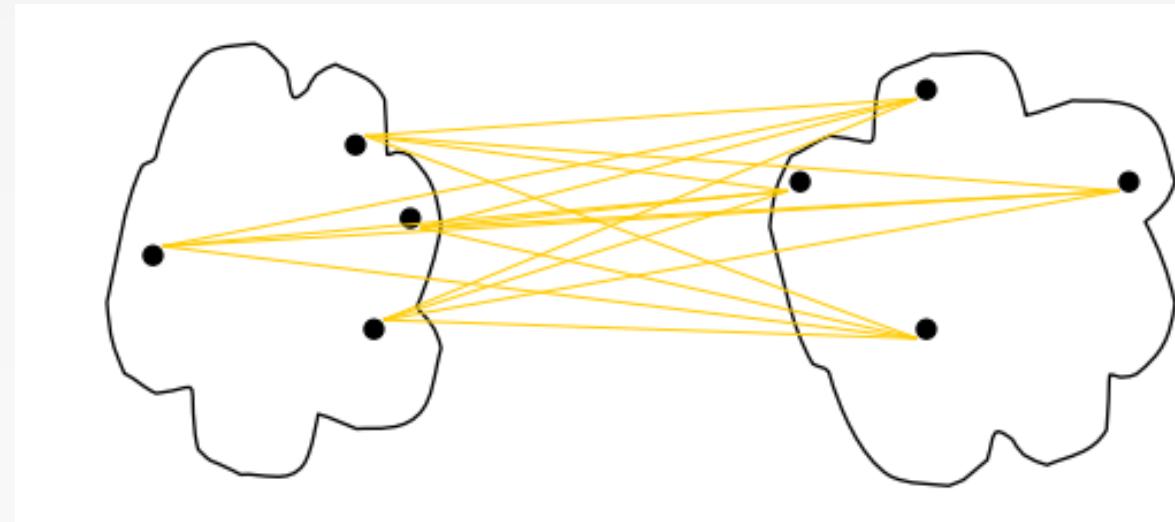
MAX (Complete link)

- Strength: less sensitive to noise/outliers
- Limitations: tends to break large clusters



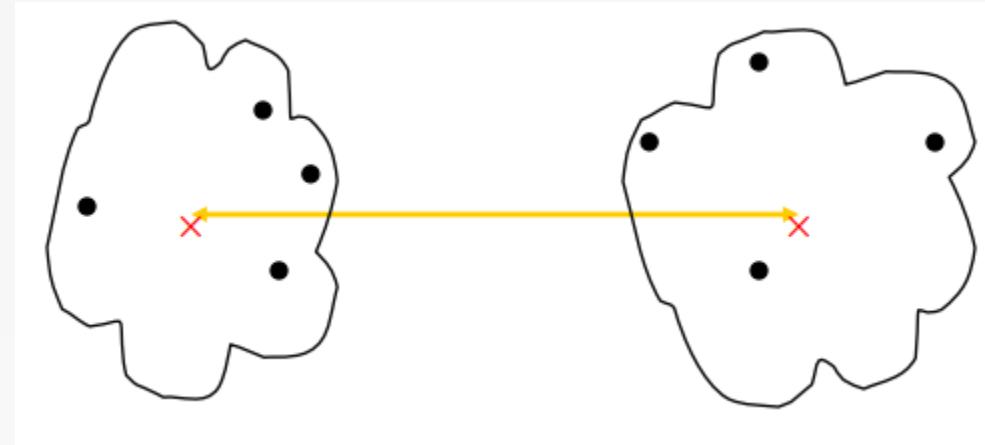
Group average

- The distance between two clusters is represented by the average distance of all pairs of data objects belonging to different clusters
- Determined by all pairs of points in the two clusters



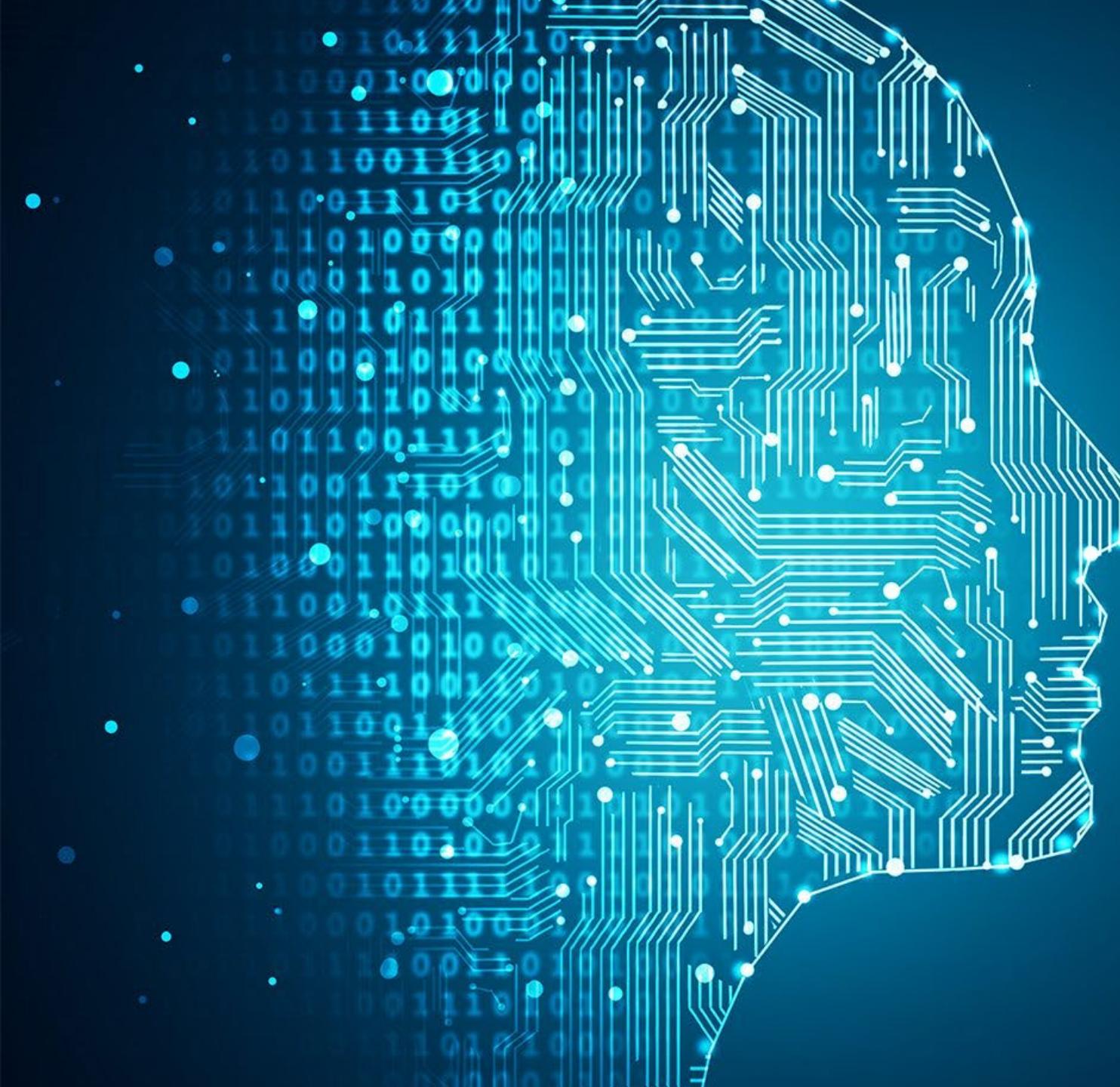
Centroid Distance

- The distance between two clusters is represented by the distance between the centers of the clusters
 - – Determined by cluster centroids



Ward's Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
- Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers



机器学习与人工智能 Machine Learning and Artificial Intelligence

Lecture 7 PCA

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

Principal Component Analysis

High Dimension Data

- High resolution images (millions of pixels)



High Dimension Data

- Customer purchase data



手淘推荐简介

最大流量入口
每天服务数亿用户



首页

最大成交渠道之一
每天成交金额数十亿



首猜

最复杂业务形态
几十种内容和数百场景



直播



最复杂技术场景
大数据+算法驱动



会场

Useful for

- Visualization
- More efficient use of resources
 - (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)
- Further processing by ML algorithms

PCA Overview

- PCA is a technique that can simplify data
- It is a linear transformation that chooses a new coordinate system for the data set such that
 - greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component)
 - the second greatest variance on the second axis, and so on.

Toy Example

Consider the following 3D data points

1
2
3

2
4
6

4
8
12

3
6
9

6
12
18

5
10
15

If each component is stored in a byte,
we need $18 = 3 \times 6$ bytes

Toy Example

Consider the following 3D data points

$$\begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array} = 1 * \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 2 \\ \hline 4 \\ \hline 6 \\ \hline \end{array} = 2 * \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 4 \\ \hline 8 \\ \hline 12 \\ \hline \end{array} = 4 * \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

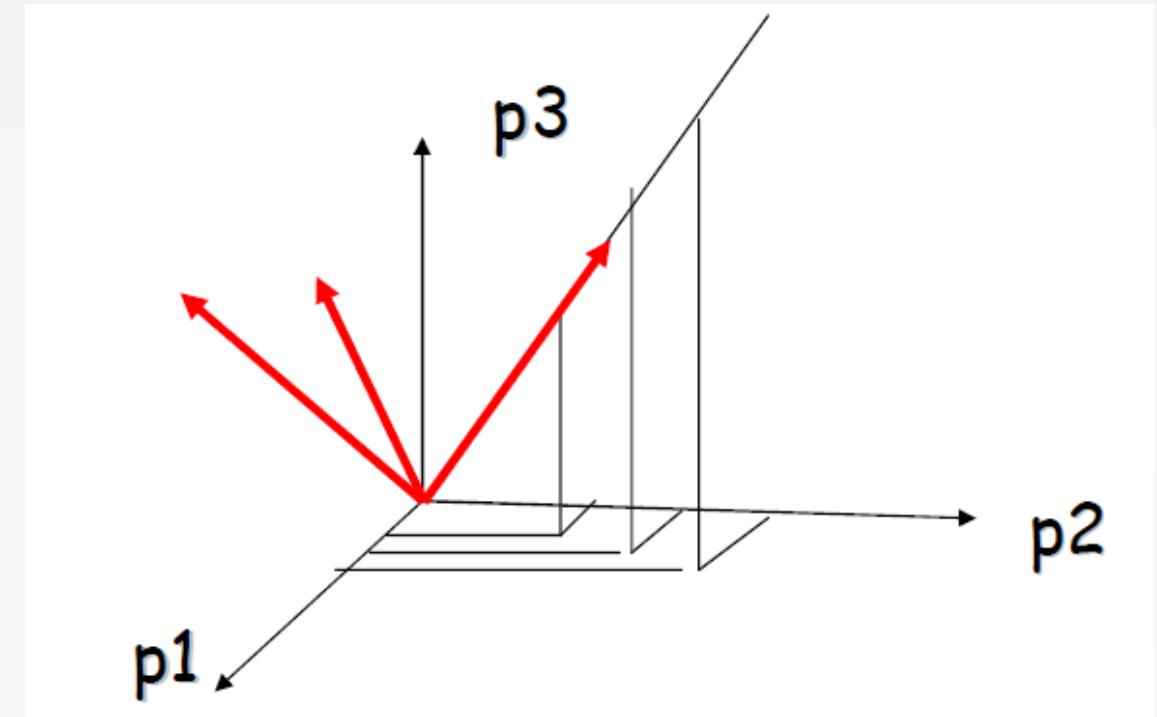
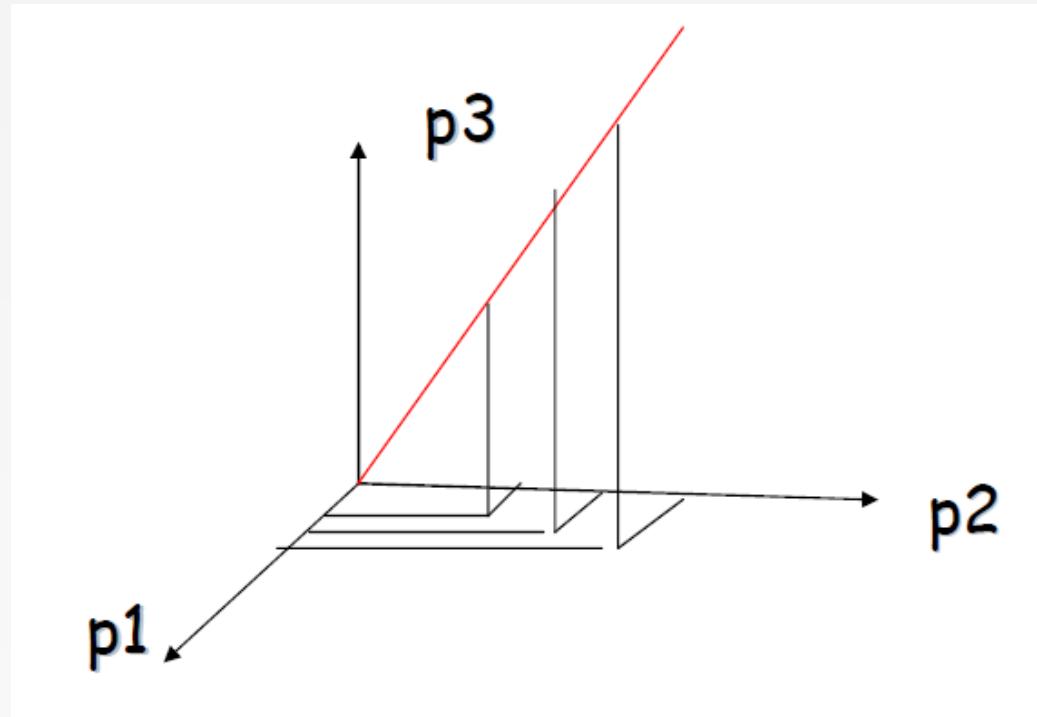
$$\begin{array}{|c|} \hline 3 \\ \hline 6 \\ \hline 9 \\ \hline \end{array} = 3 * \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 6 \\ \hline 12 \\ \hline 18 \\ \hline \end{array} = 6 * \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 5 \\ \hline 10 \\ \hline 15 \\ \hline \end{array} = 5 * \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

They can be stored using only 9 bytes (50% savings!)

Toy Example



Principle Component Analysis

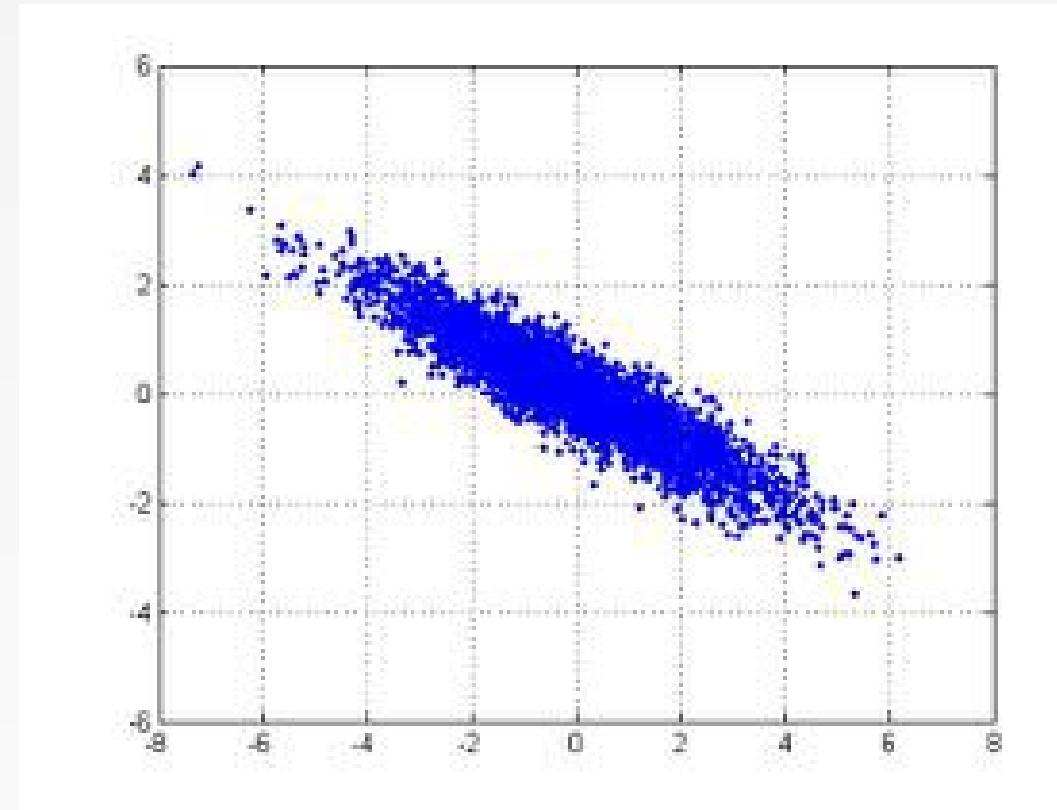
- Identifying the axes is known as Principal Components Analysis, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).
- Data for PCA:

$$\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N \quad X = \begin{bmatrix} (\boldsymbol{x}^{(1)})^T \\ (\boldsymbol{x}^{(2)})^T \\ \dots \\ (\boldsymbol{x}^{(N)})^T \end{bmatrix}$$

We assume the data is centered: $\mu = \frac{1}{N} \sum_{i=1}^N \boldsymbol{x}^{(i)} = \mathbf{0}$

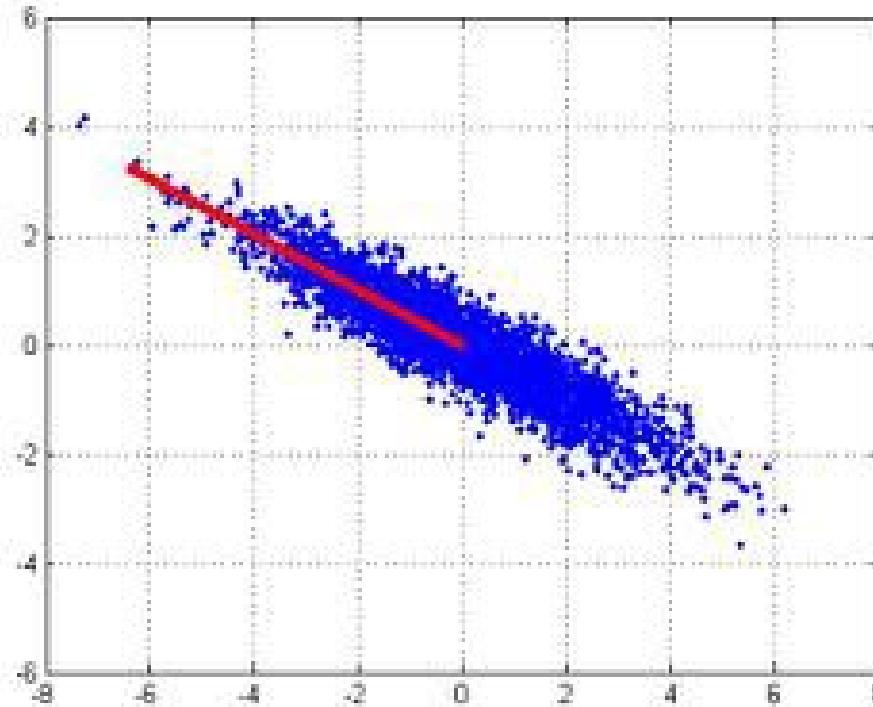
2D Gaussian Dataset

The original dataset:



2D Gaussian Dataset

First find the direction of maximum variance, labeled “Component 1”

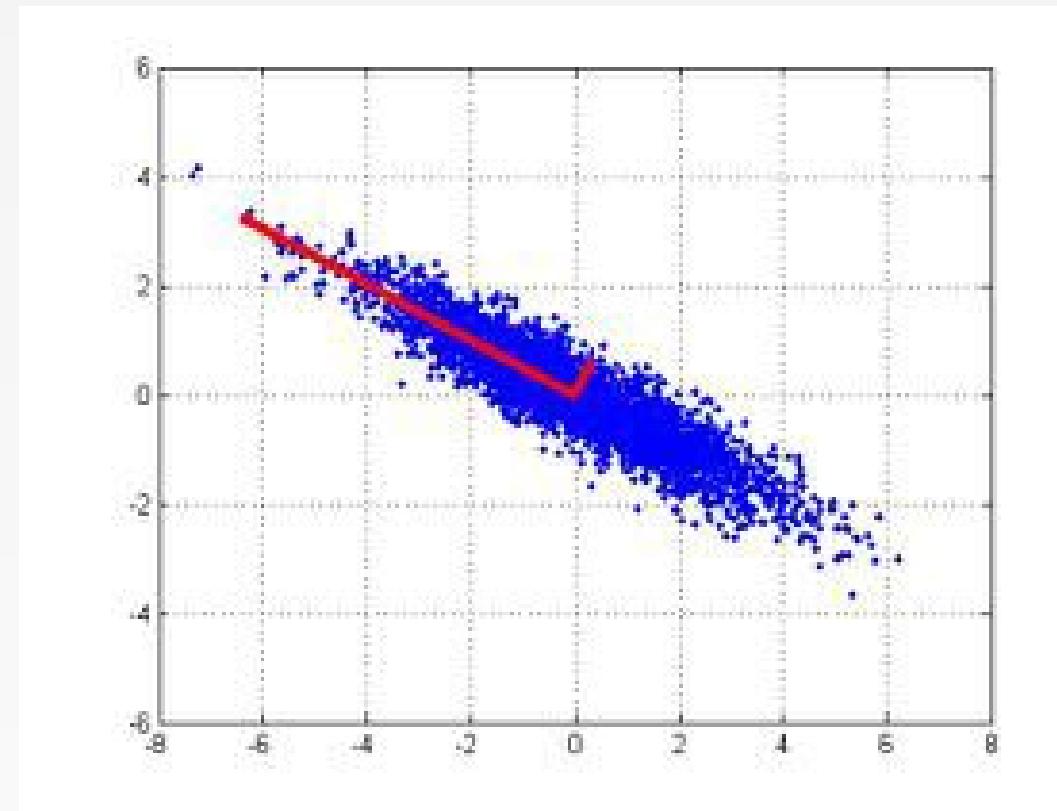


Along this direction:

- Features are most correlated with each other
- Contains the most of the information

2D Gaussian Dataset

Component 2: orthogonal to the first direction & maximized variance



Sample Covariance matrix

- The sample covariance matrix is given by:

$$\Sigma_{jk} = \frac{1}{N} \sum_{i=1}^N (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$$

- Since the data matrix is centered, we rewrite as:

$$\sum = \frac{1}{N} X^T X$$

Definition of PCA

- Given K vectors, $\overrightarrow{v_1}, \overrightarrow{v_2}, \dots, \overrightarrow{v_K}$, the projection of a vector $x^{(i)}$ to a lower K -dimensional space is

$$\vec{u}^{(i)} = \begin{bmatrix} \overrightarrow{v_1}^T \vec{x}^{(i)} \\ \dots \\ \overrightarrow{v_K}^T \vec{x}^{(i)} \end{bmatrix}$$

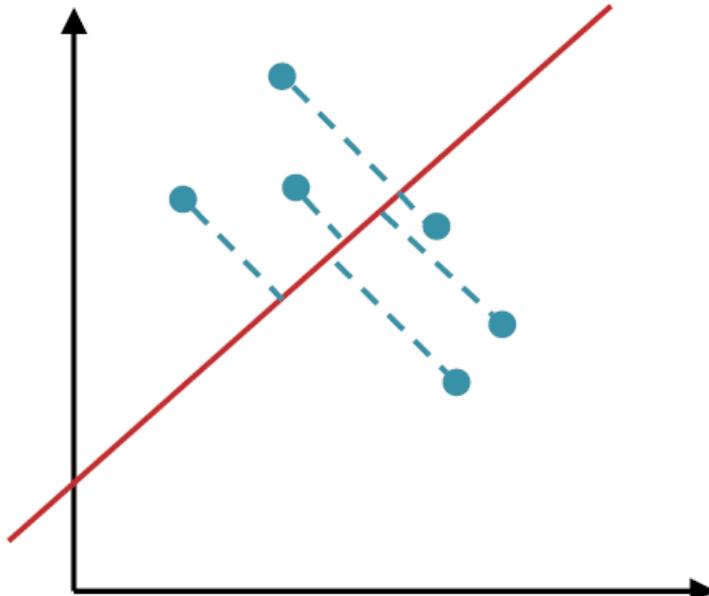
- Def: PCA repeatedly chooses a next $\overrightarrow{v_j}$ that minimize the reconstruction error, s.t., $\overrightarrow{v_j}$ is orthogonal to $\overrightarrow{v_1}, \dots, \overrightarrow{v_{j-1}}$

PCA: Maximize the Variance

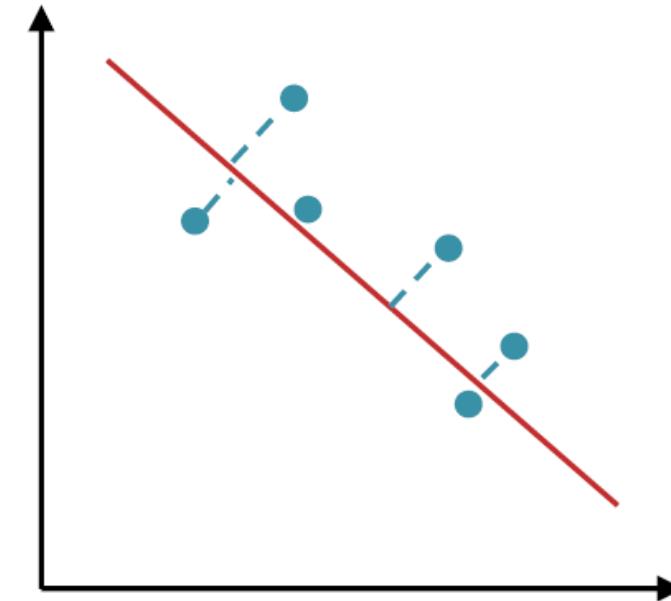
Quiz: Consider the two projections below

1. Which maximizes the variance?
2. Which minimizes the reconstruction error?

Option A



Option B



Eigenvectors and Eigenvalues

- For a square matrix A ($n \times n$), the vector v ($n \times 1$) is an eigenvector iff there exists eigenvalue λ (scalar) such that

$$Ax = \lambda x$$

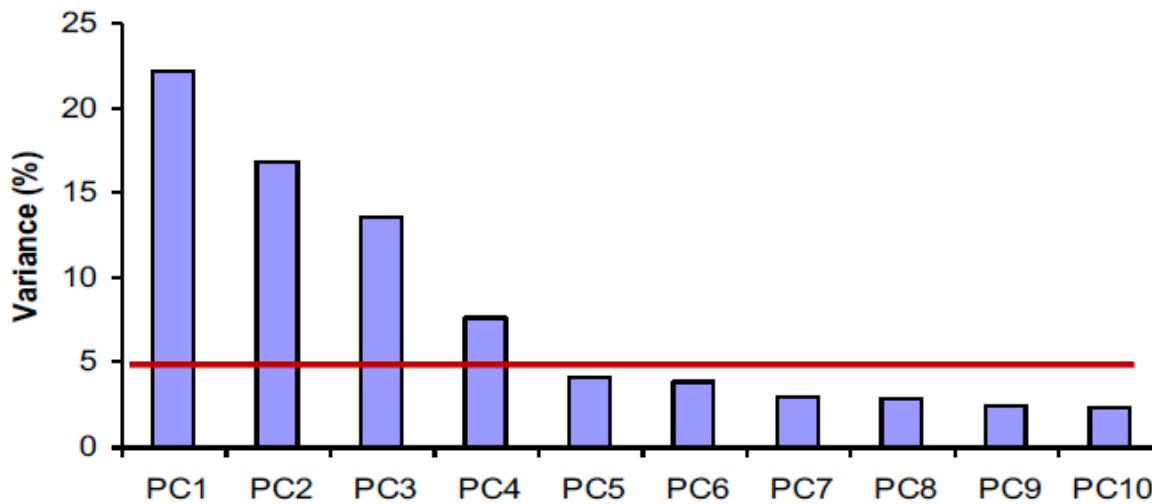
- **Theorem 1:** The vector that maximizes the variance is the eigenvector of Σ with largest eigenvalue
- **Theorem 2:** The eigenvector of a symmetric matrix are orthogonal to each other
- **Fact 1:** Σ is a symmetric matrix

Algorithms for PCA

- Singular Value Decomposition (SVD)
 - Find all the principal components at once
 - Two options:
 - Option A: run SVD on $X^T X$
 - Option B: run SVD on X

How Many PCs?

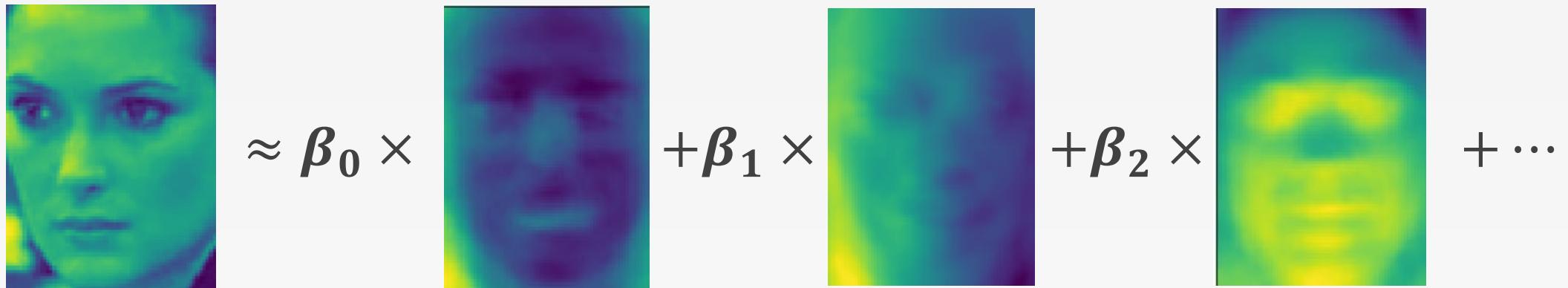
- For M original dimensions, sample covariance matrix is $M \times M$, and has up to M eigenvectors. So M PCs.
- Where does dimensionality reduction come from?
Can *ignore* the components of lesser significance.



- You do *lose some information*, but if the eigenvalues are small, you don't lose much
 - M dimensions in original data
 - calculate M eigenvectors and eigenvalues
 - choose only the first D eigenvectors, based on their eigenvalues
 - final data set has only D dimensions

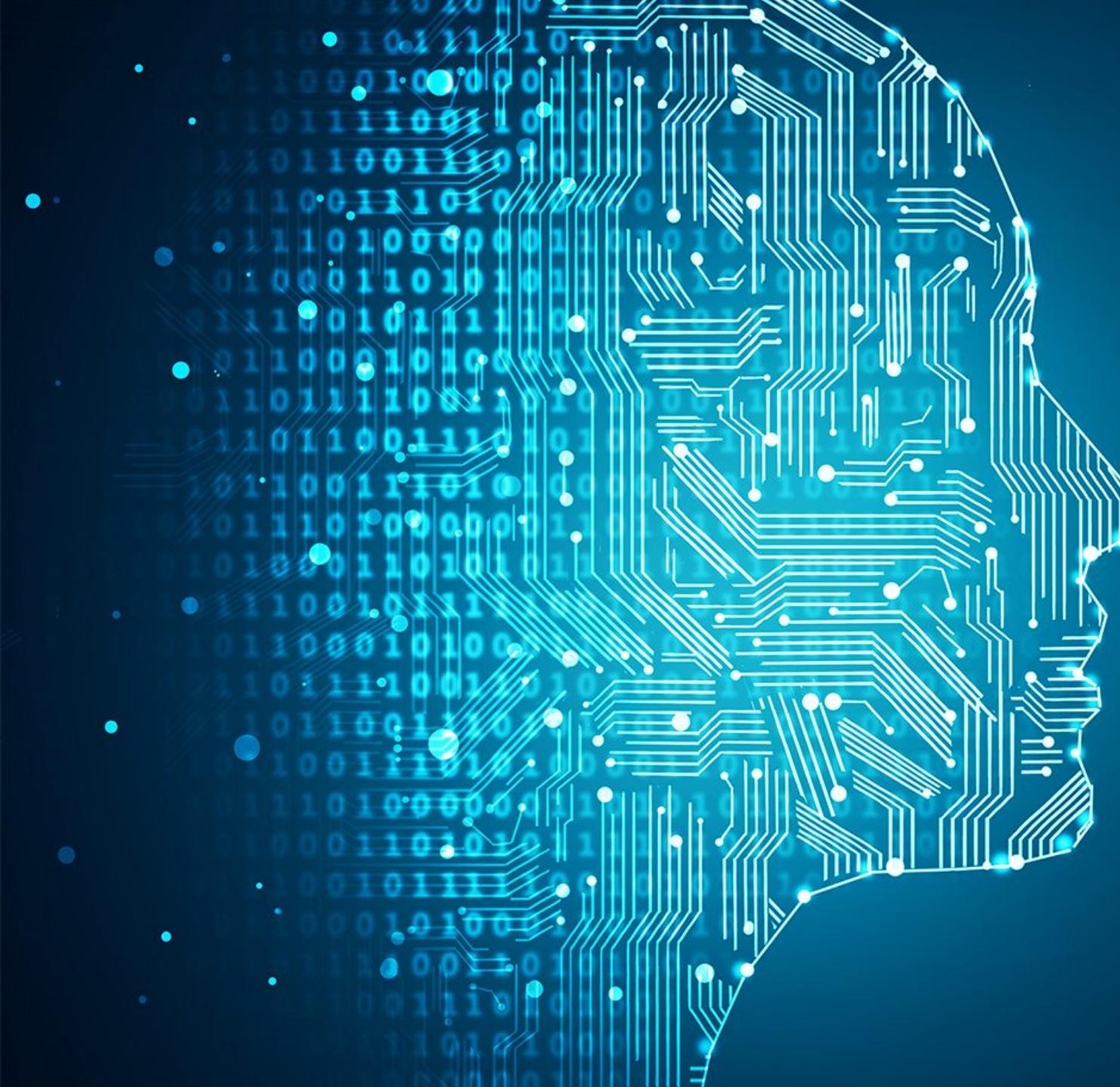
Example: Facial Recognition

PCA Transformation


$$\text{face image} \approx \beta_0 \times \text{component 0} + \beta_1 \times \text{component 1} + \beta_2 \times \text{component 2} + \dots$$

The image shows a face composed of several colored squares. To its right is a mathematical equation: the face is approximated by the sum of the product of coefficients ($\beta_0, \beta_1, \beta_2$) and corresponding components. The components are square images showing different facial features like eyes, nose, and mouth.

β_0, β_1 , and so on are the coefficients of the principal components for this data point.



机器学习与人工智能

Machine Learning

and Artificial

Intelligence

Deep Learning

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

This is an INTRODUCTION!!

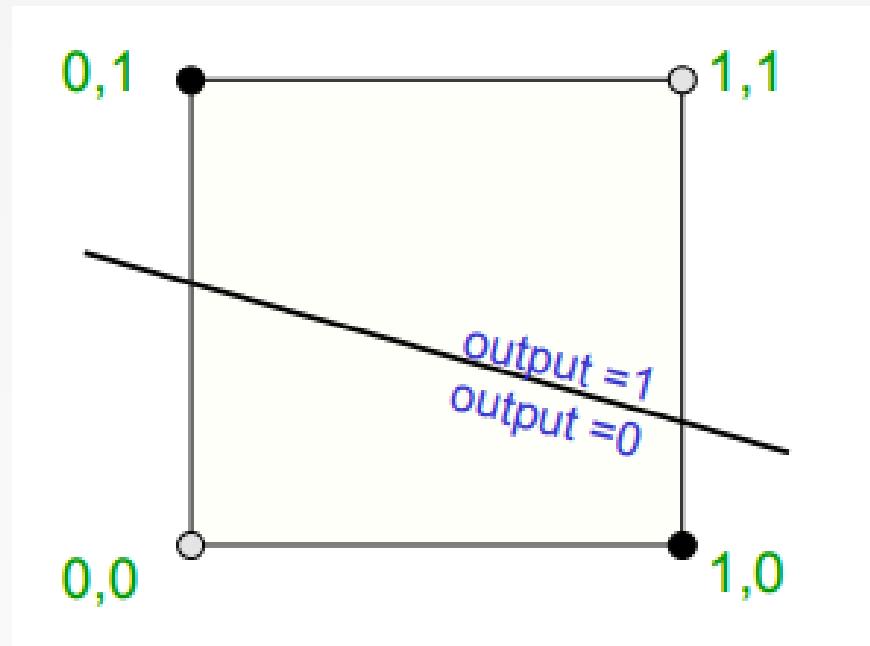
Agenda

- Neural Networks
- Backpropagation
- Deep Learning
- Python Practice
- Convolutional Neural Networks
- Recurrent Neural Networks

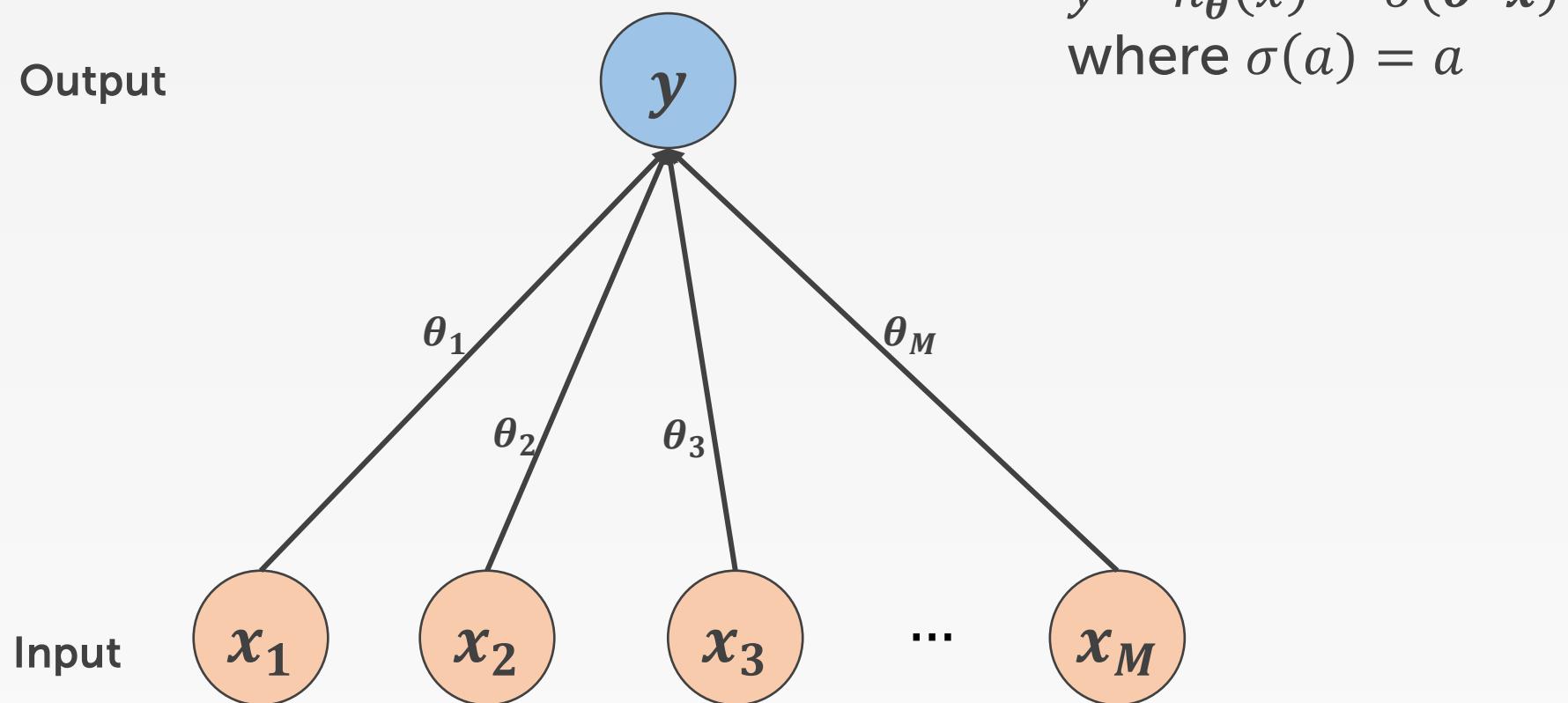
Neural Network

Limitation of Linear Classifier

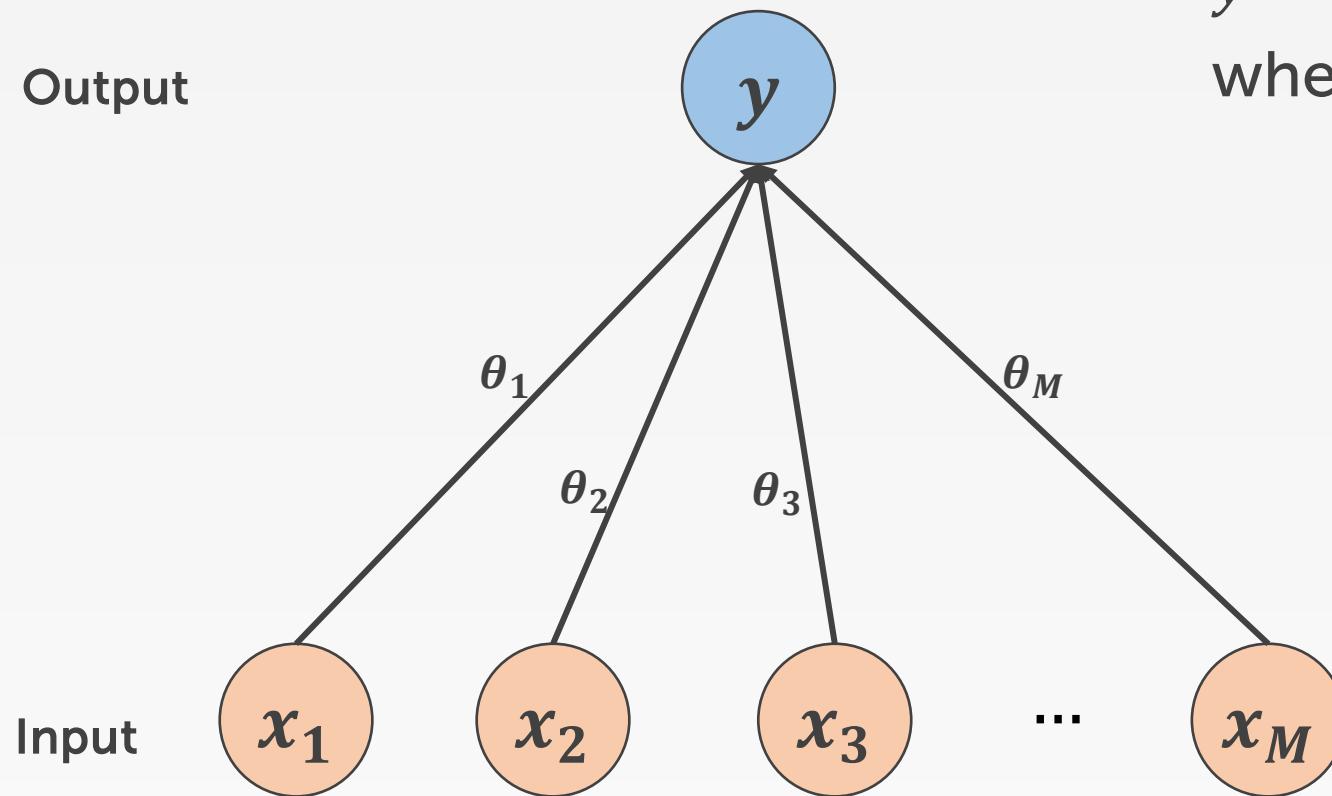
- Linear classifiers (e.g., logistic regression) classify inputs based on linear combinations of features x_i
- Many decisions involve non-linear functions of the input



Linear Regression

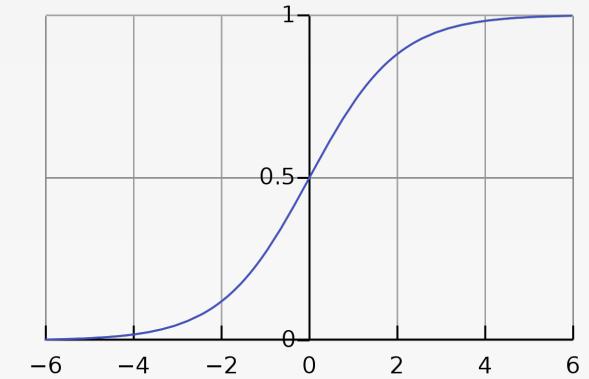


Logistic Regression

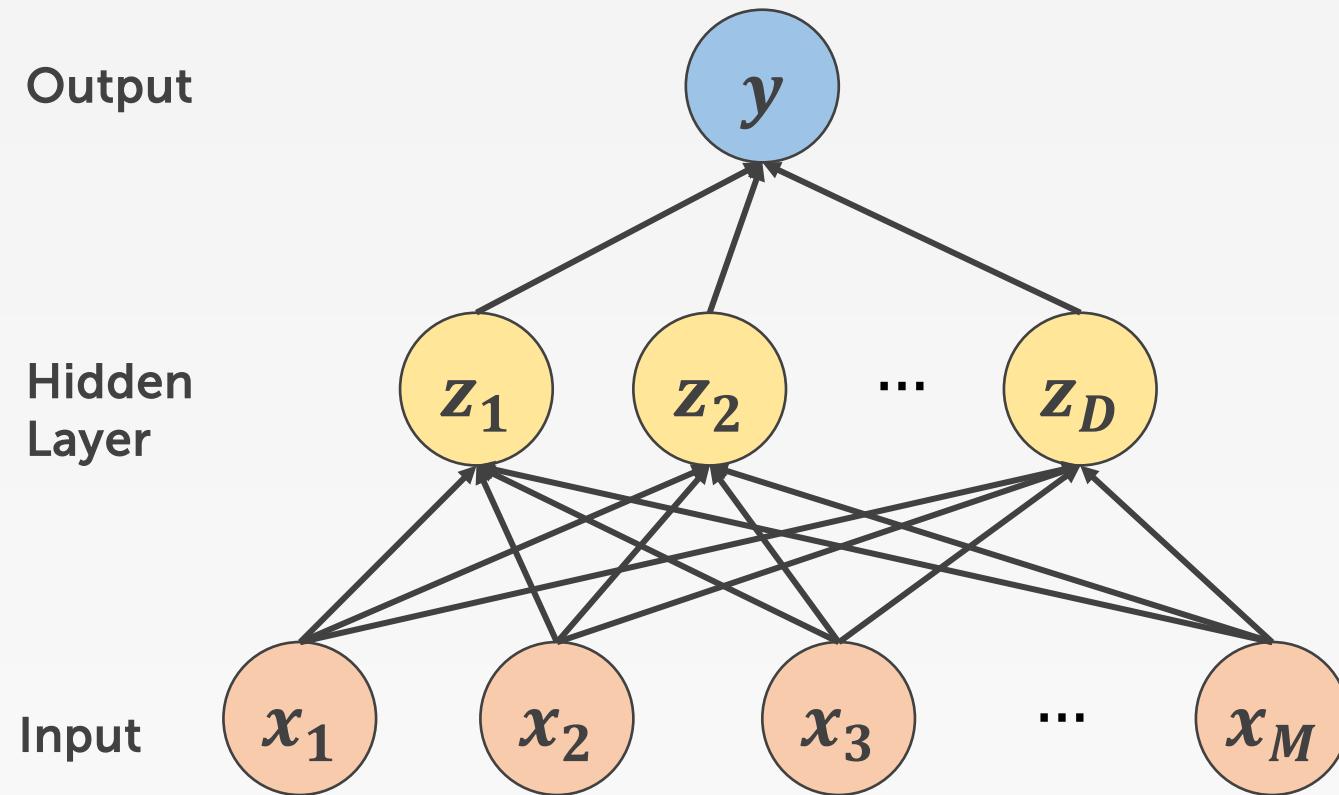


$$y = h_{\theta}(x) = \sigma(\theta^T x)$$

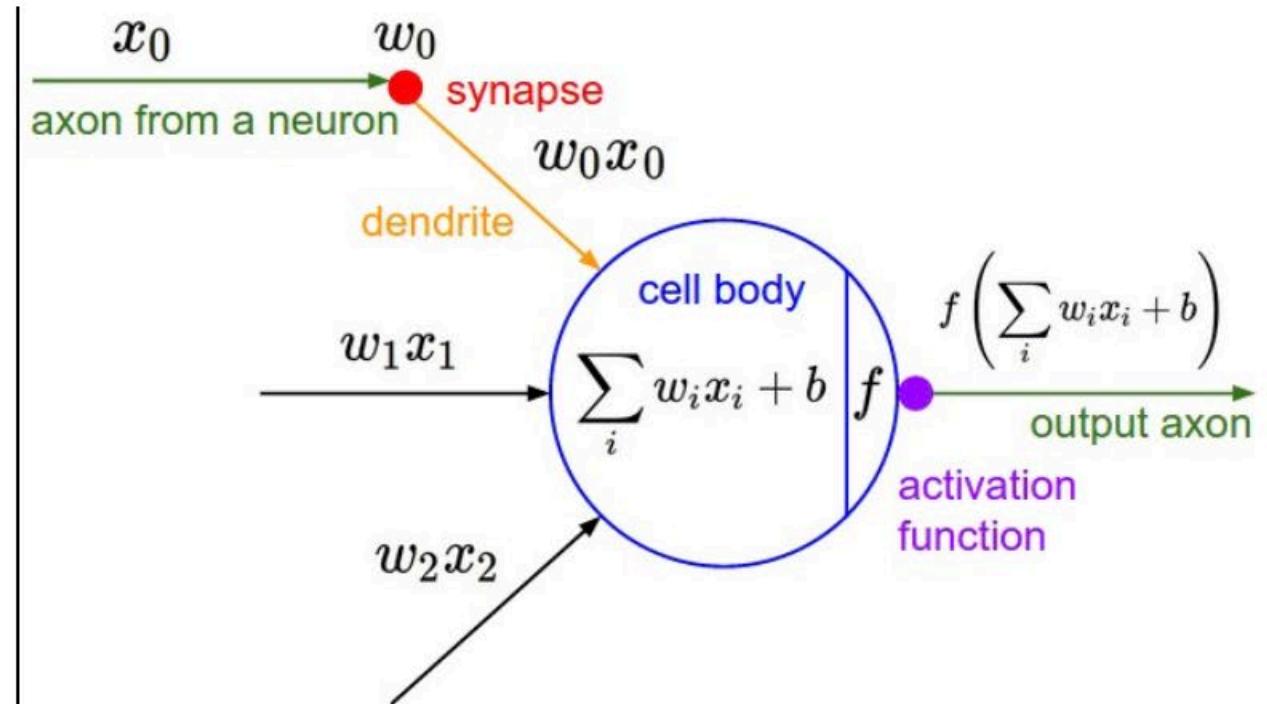
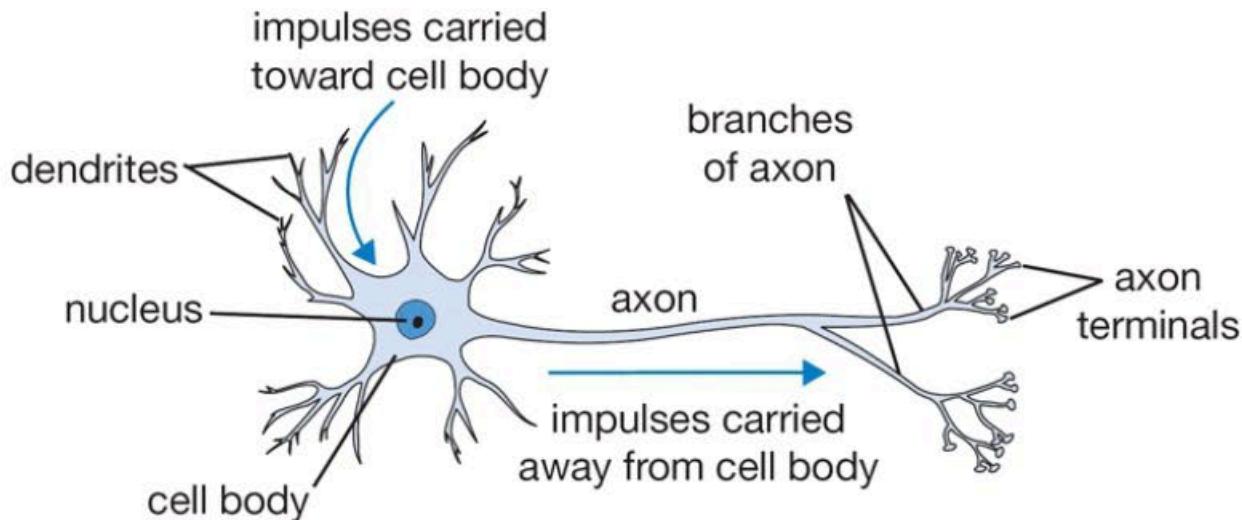
$$\text{where } \sigma(a) = \frac{1}{1+\exp(-a)}$$



Neural Network

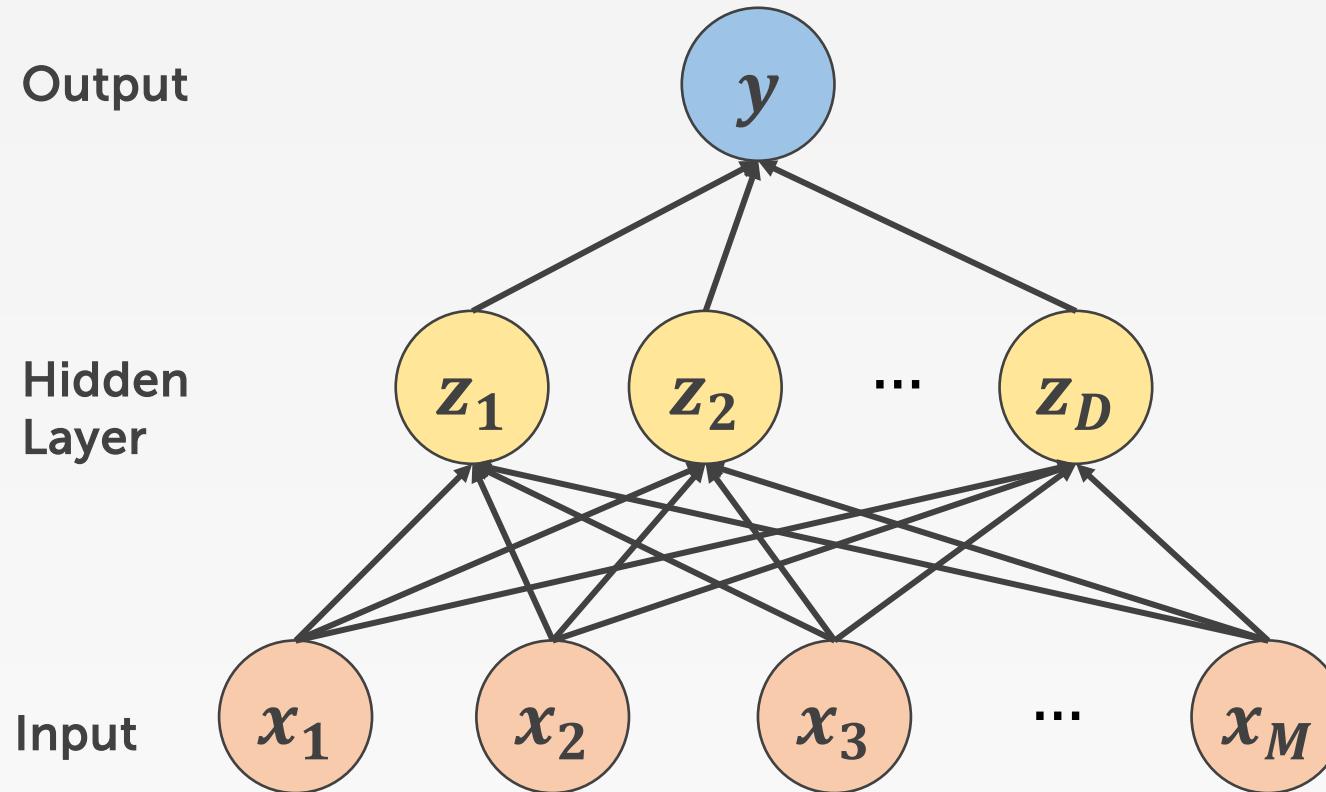


Inspiration: The Brain



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Neural Network



(E) Output (sigmoid)

$$y = \frac{1}{1+\exp(-b)}$$

(D) Output (linear)

$$b = \sum_{j=1}^D \beta_j z_j$$

(C) Hidden (sigmoid)

$$z_j = \frac{1}{1+\exp(-a_j)}, \forall j$$

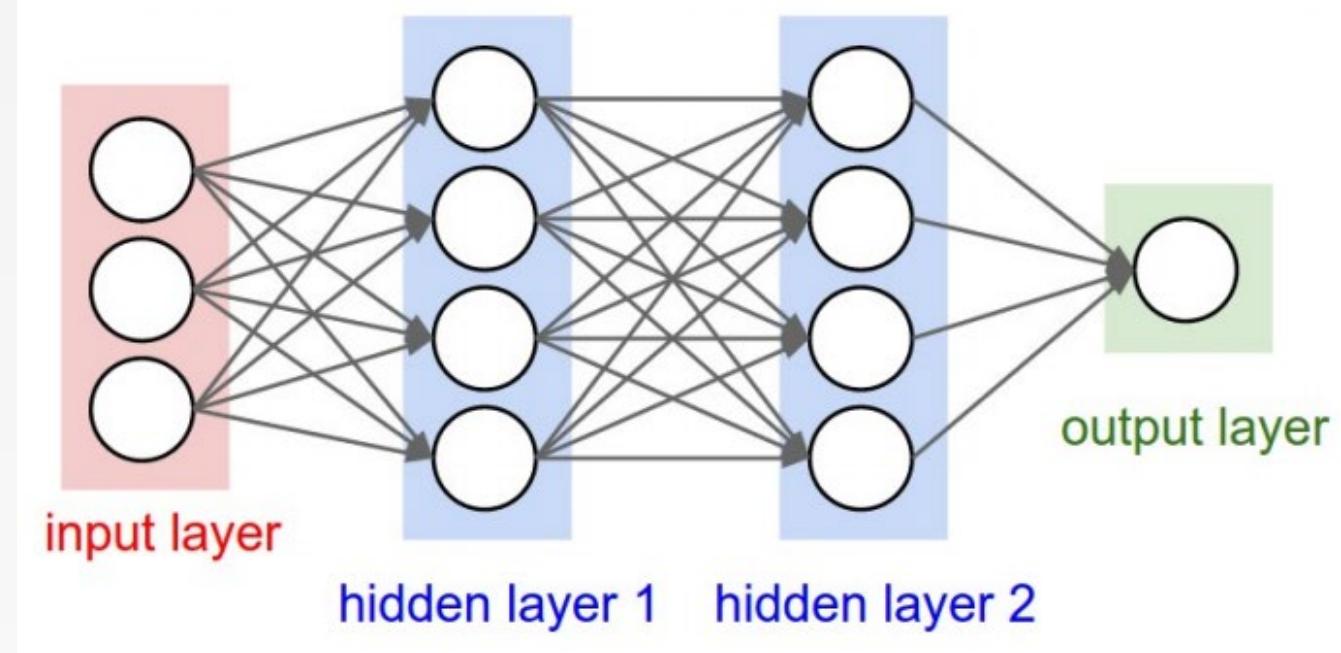
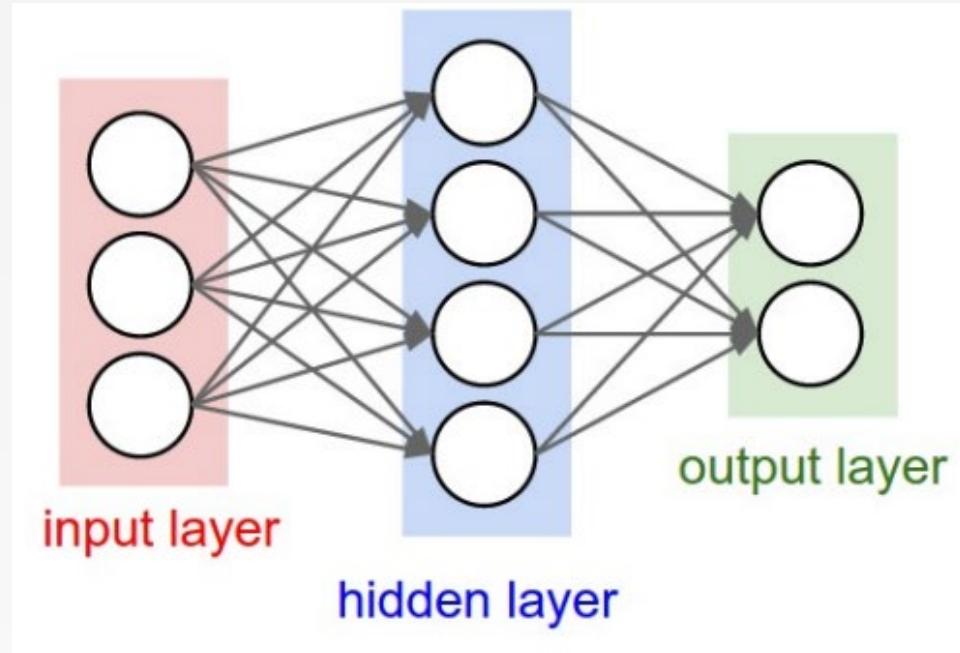
(B) Hidden (linear)

$$a_j = \sum_{i=1}^M \alpha_{ji} x_i, \forall j$$

(A) Input

Given $x_i, \forall i$

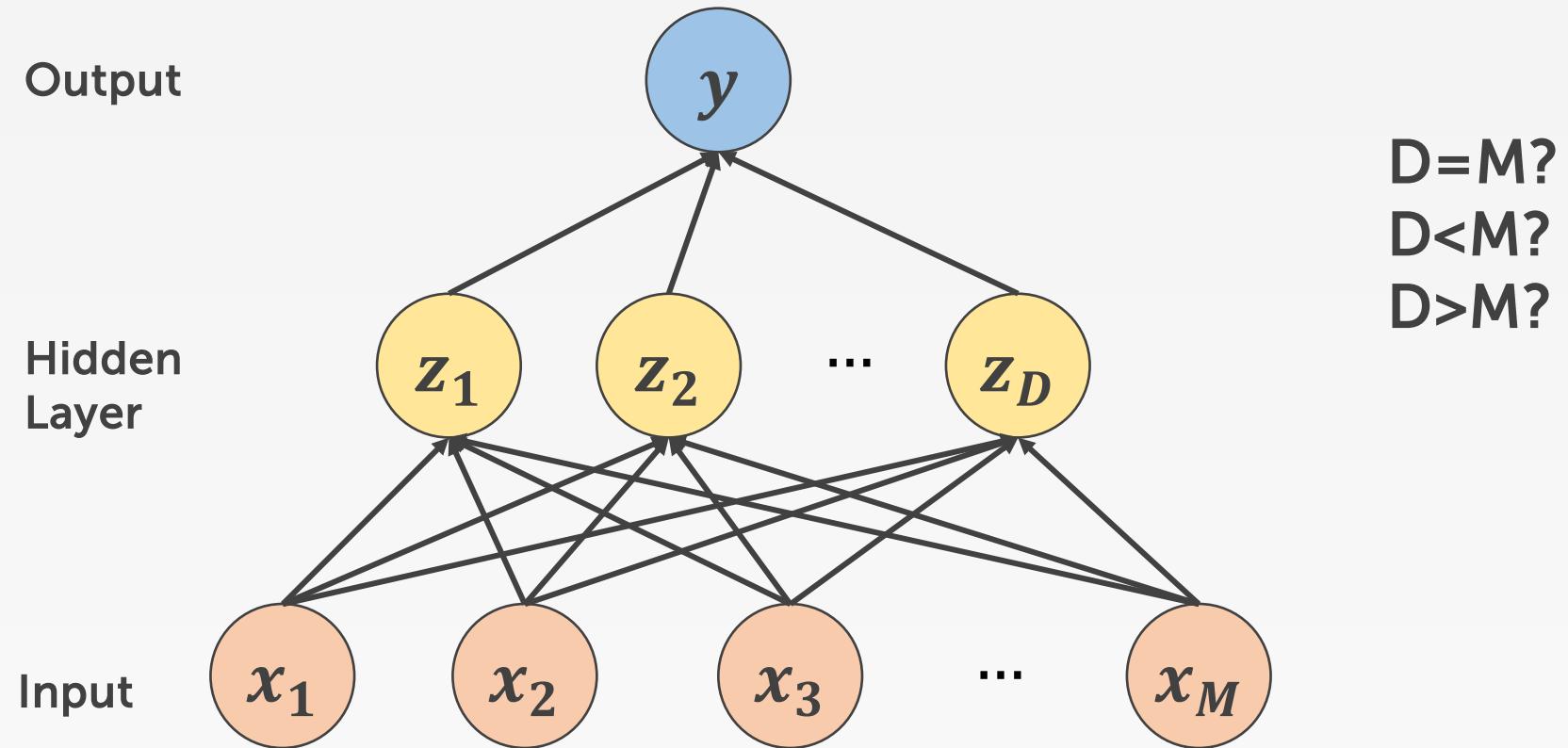
Multi-Layer Neural Network



Architecture

- # of hidden layers (depth)
- # of units per hidden layer (width)
- Type of activation function (nonlinearity)
- Form of objective function

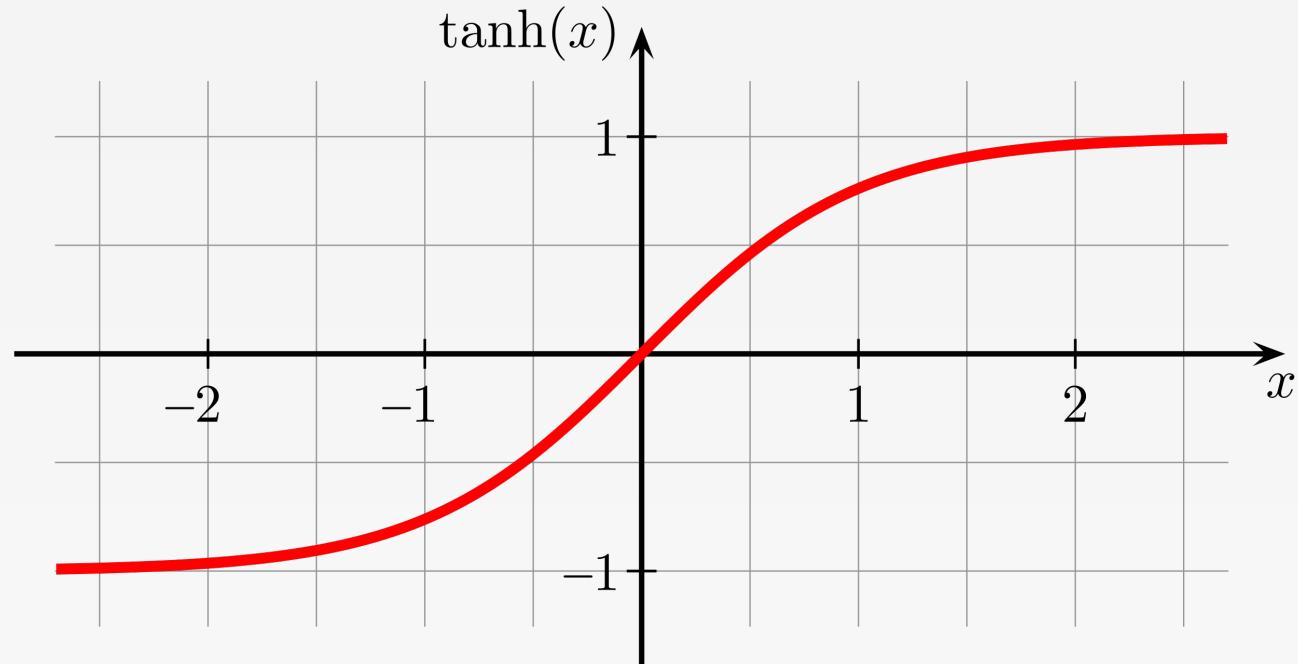
How many hidden units?



How many layers should we use?

- Theoretical answer:
 - Cybenko (1989): For any continuous function $g(x)$, there exists a 1-hidden layer neural network $h_\theta(x)$
s.t., $|h_\theta(x) - g(x)| < \epsilon$ for all x , assuming sigmoid activation functions.
- Empirical answer:
 - Before 2006: “Deep networks (e.g., 3 or more hidden layers) are too hard to train”
 - After 2006: “Deep networks are easier to train than shallow networks for many problems”

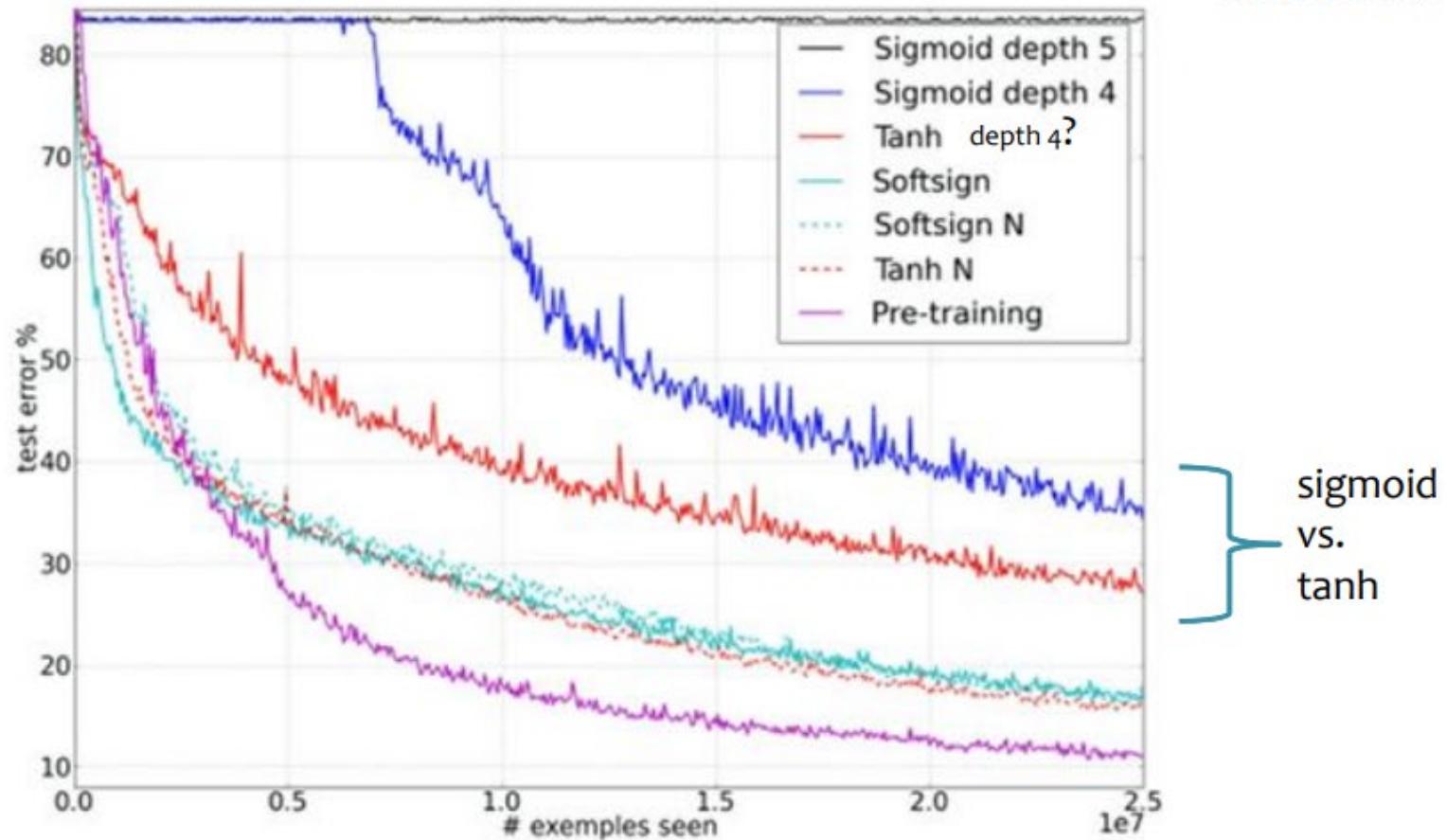
Alternative 1: tanh



$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Understanding the difficulty of training deep feedforward neural networks

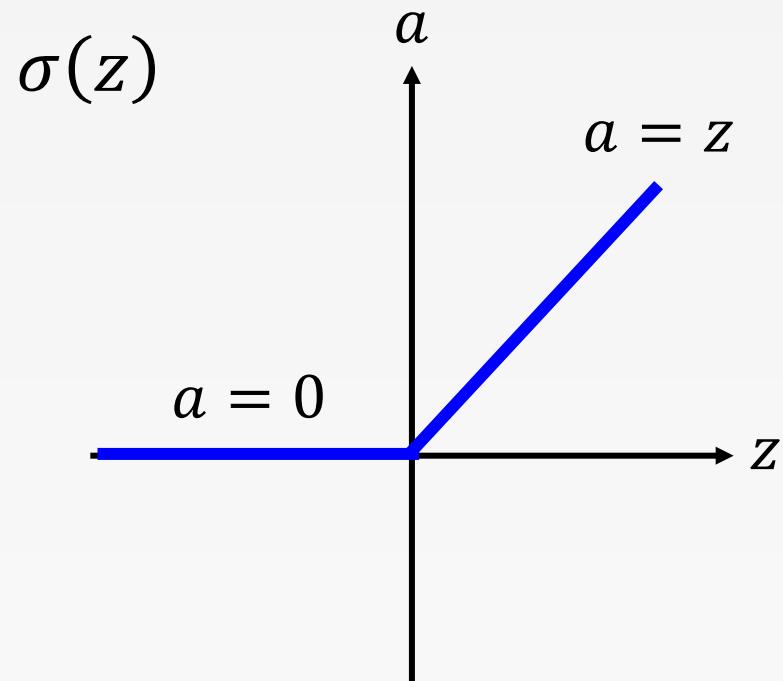
AI Stats 2010



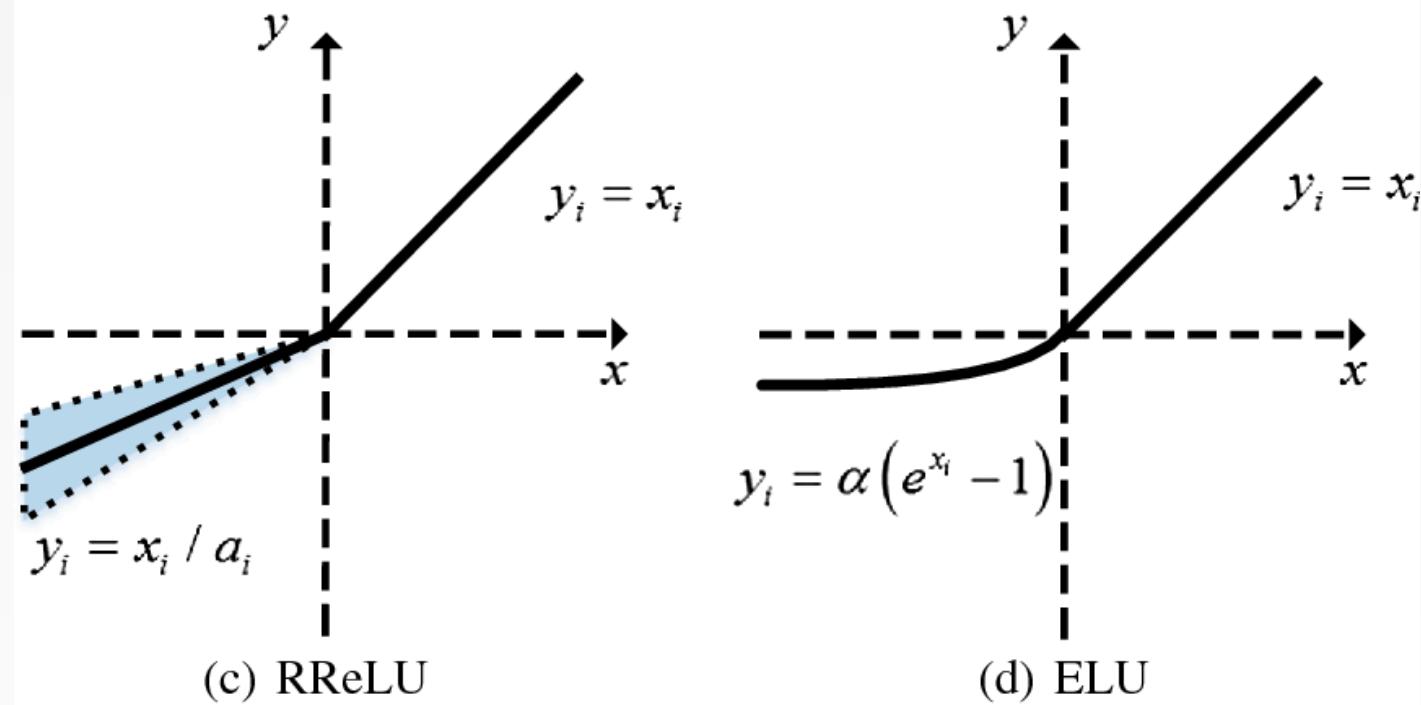
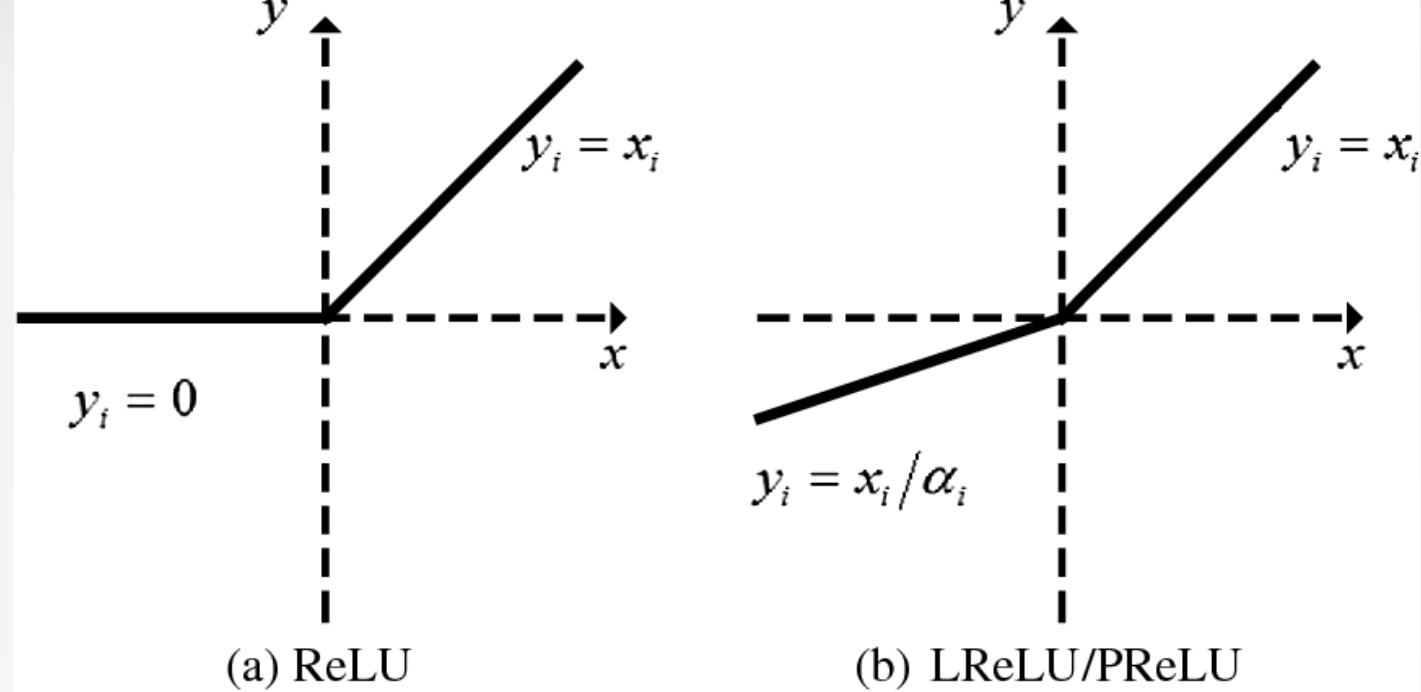
sigmoid
vs.
tanh

Figure from Glorot & Bentio (2010)

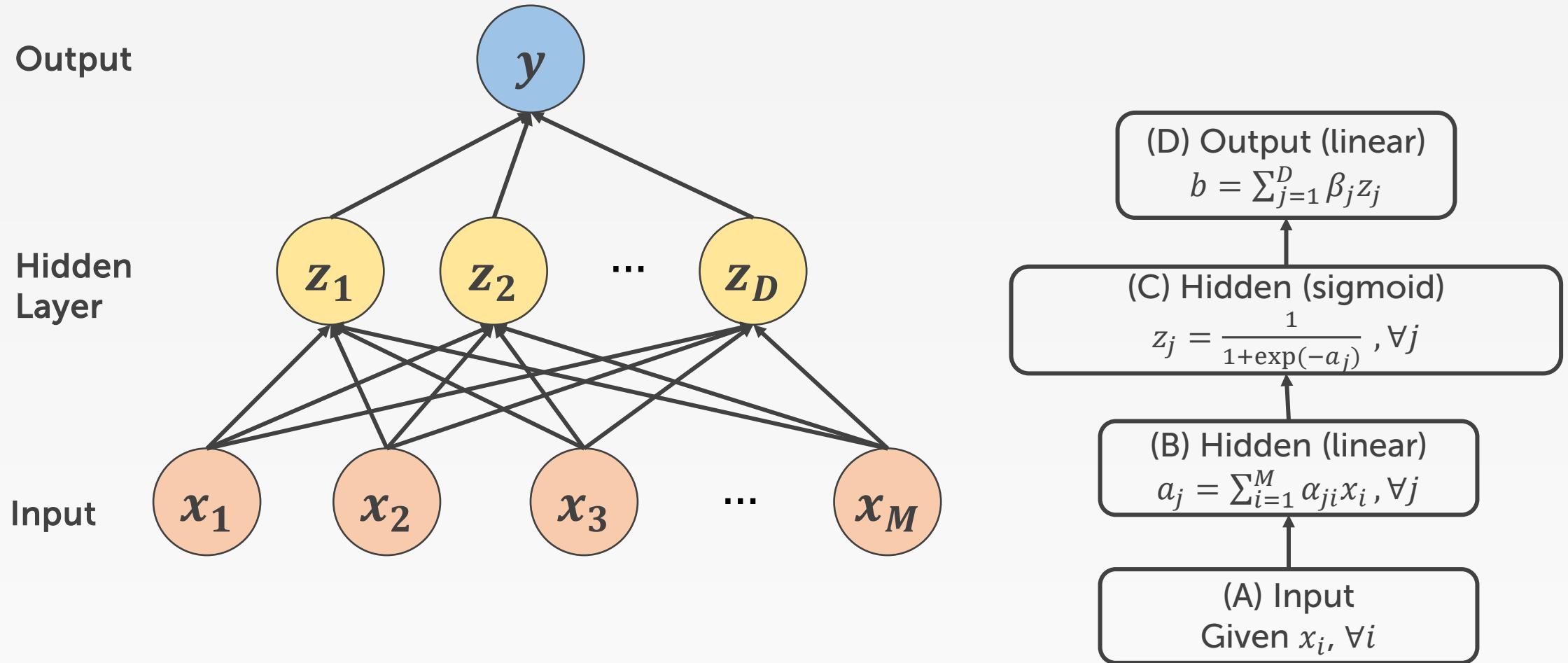
Alternative 2: ReLU



- Rectified Linear Unit (ReLU)



Neural Network for Regression



Objective Functions for NNs

- Quadratic Loss:
 - the same objective as Linear Regression
 - i.e. mean squared error
- Cross-Entropy:
 - the same objective as Logistic Regression
 - i.e. negative log likelihood

Forward

$$\text{Quadratic: } J = \frac{1}{2}(y - y^*)^2$$

$$\text{Cross Entropy: } J = y^* \log(y) + (1 - y^*) \log(1 - y)$$

Backward

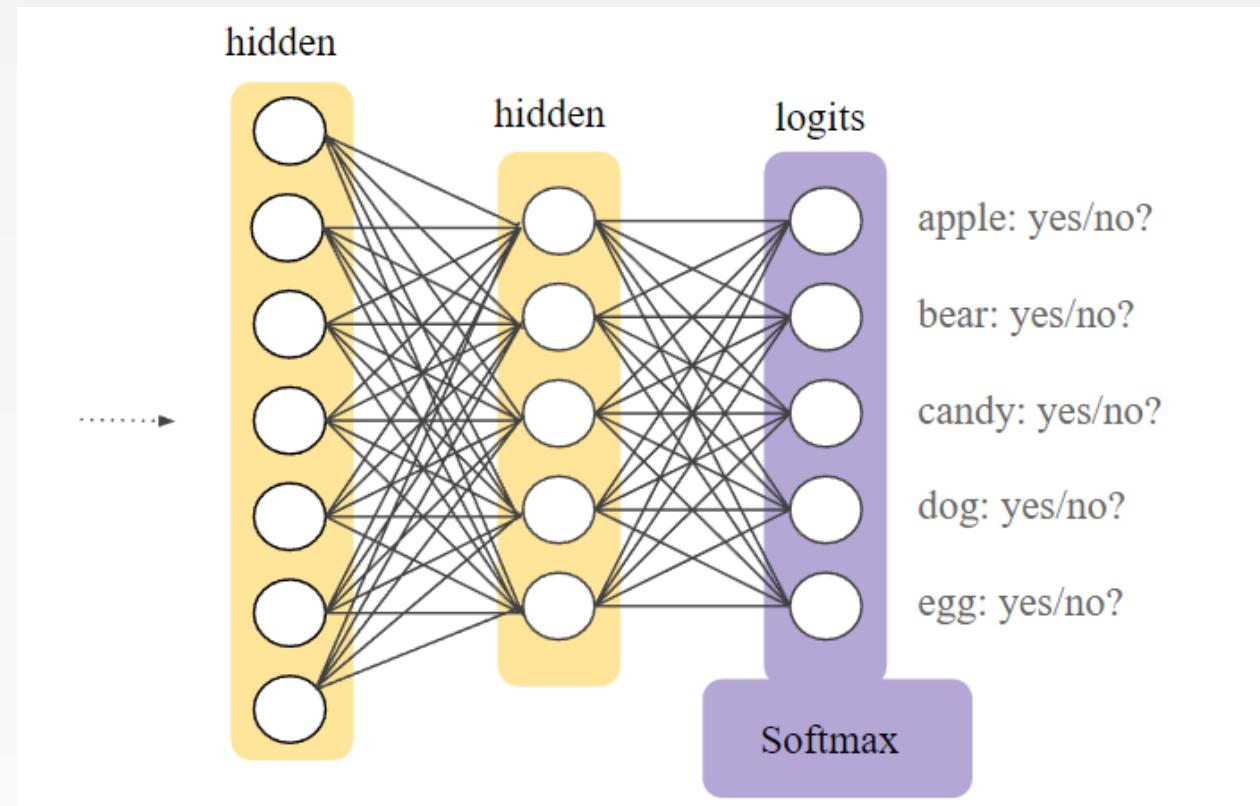
$$\frac{dJ}{dy} = y - y^*$$

$$\frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*) \frac{1}{1 - y}$$



Softmax

- Softmax (multinomial choice model)
- $y_1 = e^{z_1} / \sum_j e^{z_j}$



Backpropagation Algorithm

Recall: SGD

- Given training data:

$$\{\mathbf{x}_i, y_i\}_{i=1}^N$$

- Loss function: $\ell(\hat{y}, y_i) \in \mathbb{R}$, where $\hat{y} = f_{\theta}(\mathbf{x}_i)$
- Define goal:

$$\theta^* = \operatorname{argmin}_{\theta} \sum_{i=1}^N \ell(f_{\theta}(\mathbf{x}_i), y_i)$$

- Train with SGD:

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\theta}(\mathbf{x}_i), y_i)$$



Chain Rule

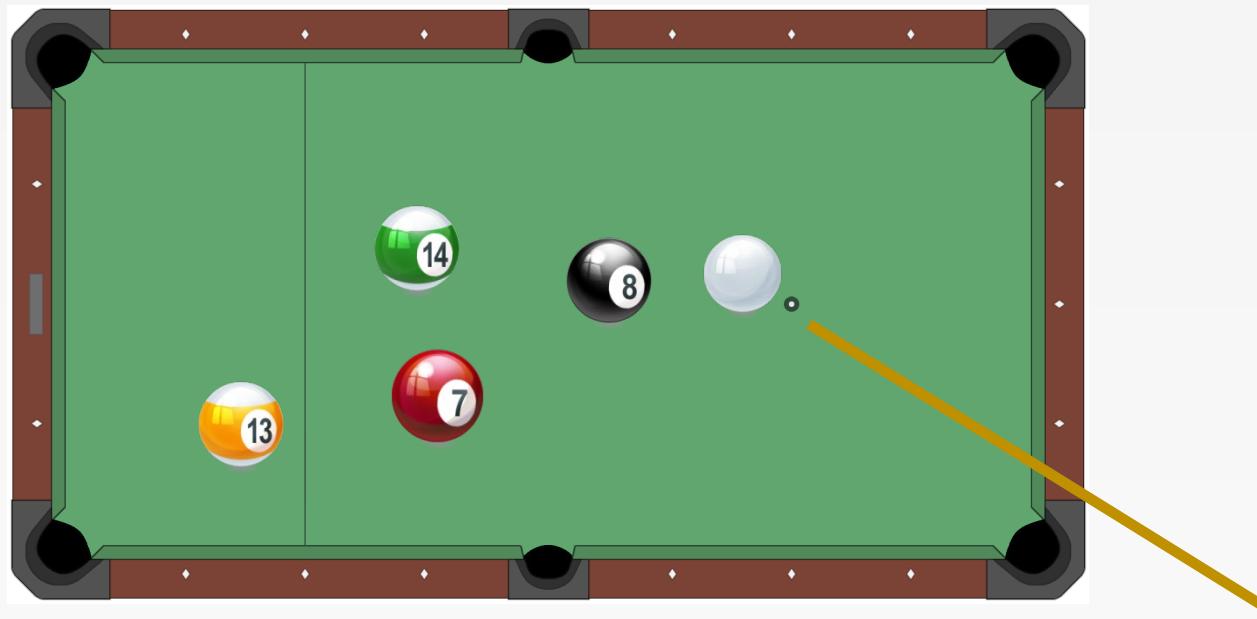
Given: $y = g(u)$ and $u = h(x)$

Chain Rule:

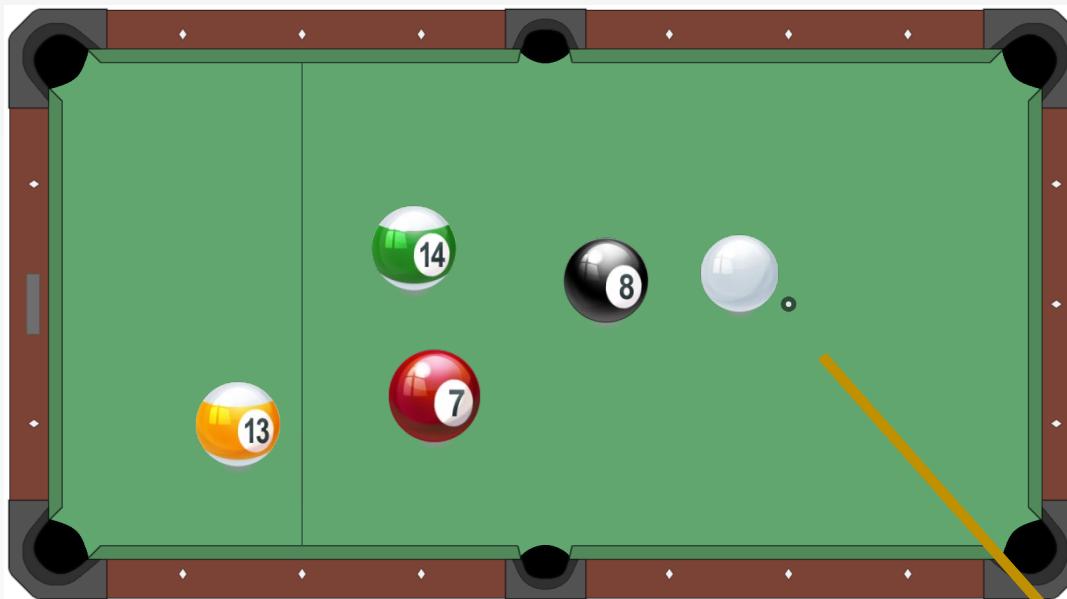
$$\frac{dy_i}{dx_k} = \sum_{j=1}^J \frac{dy_i}{du_j} \frac{du_j}{dx_k}, \forall i, k$$

- ❖ Backpropagation is just repeated application of the chain rule from Calculus course.

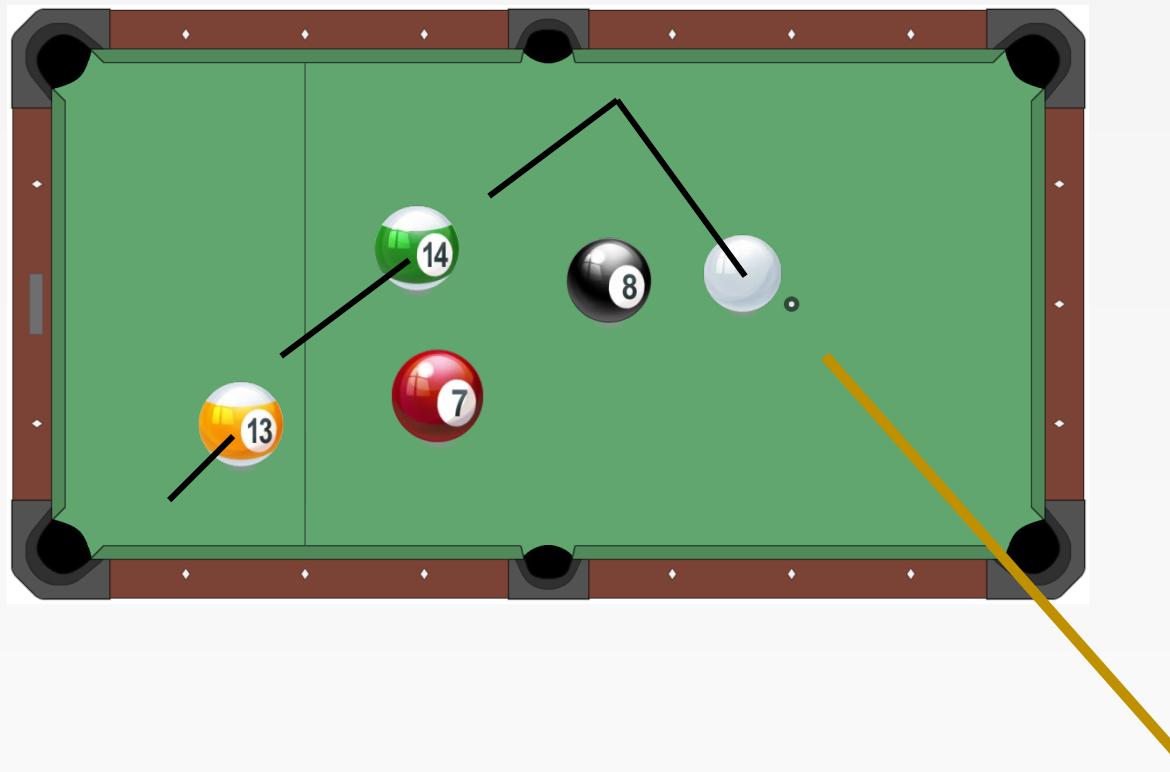
Error Back-Propagation



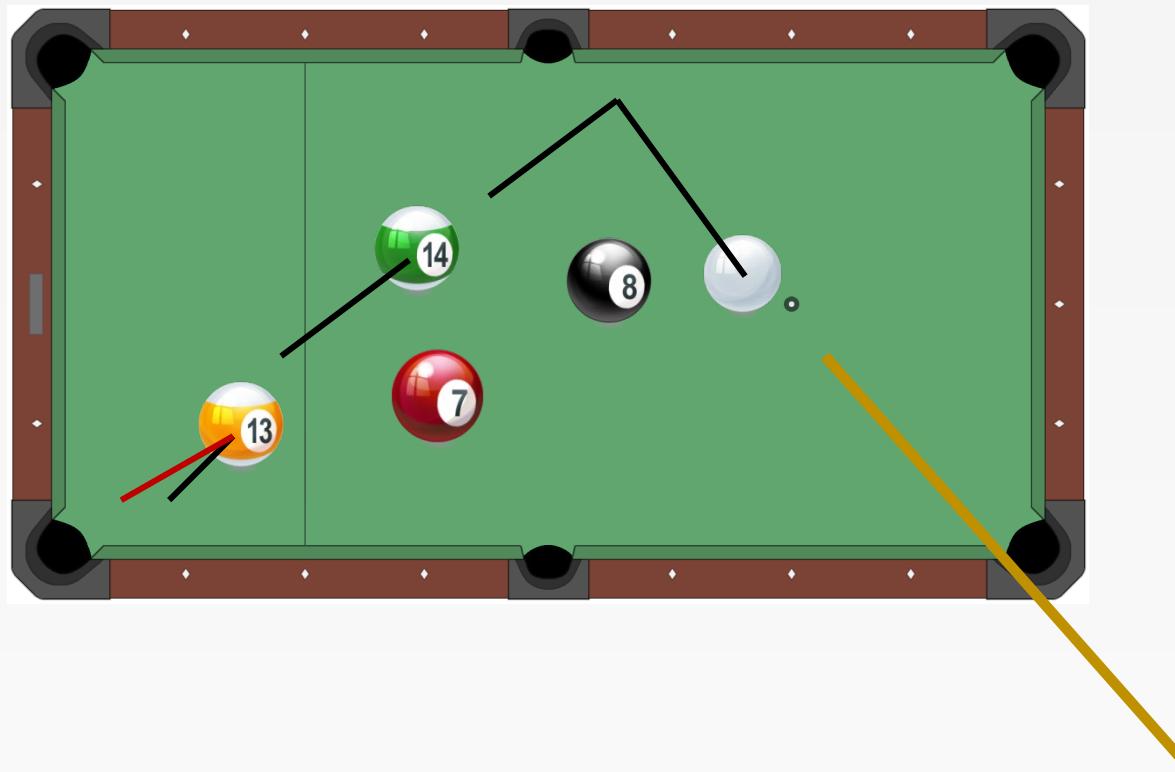
Error Back-Propagation



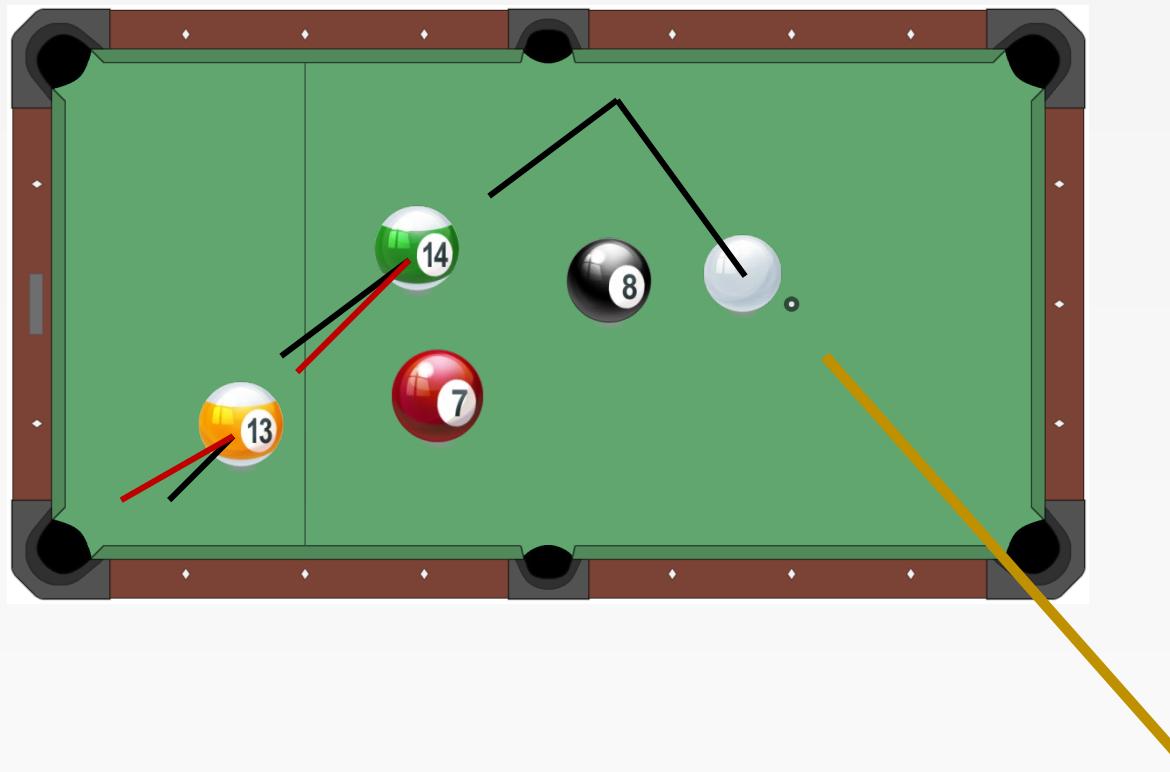
Error Back-Propagation



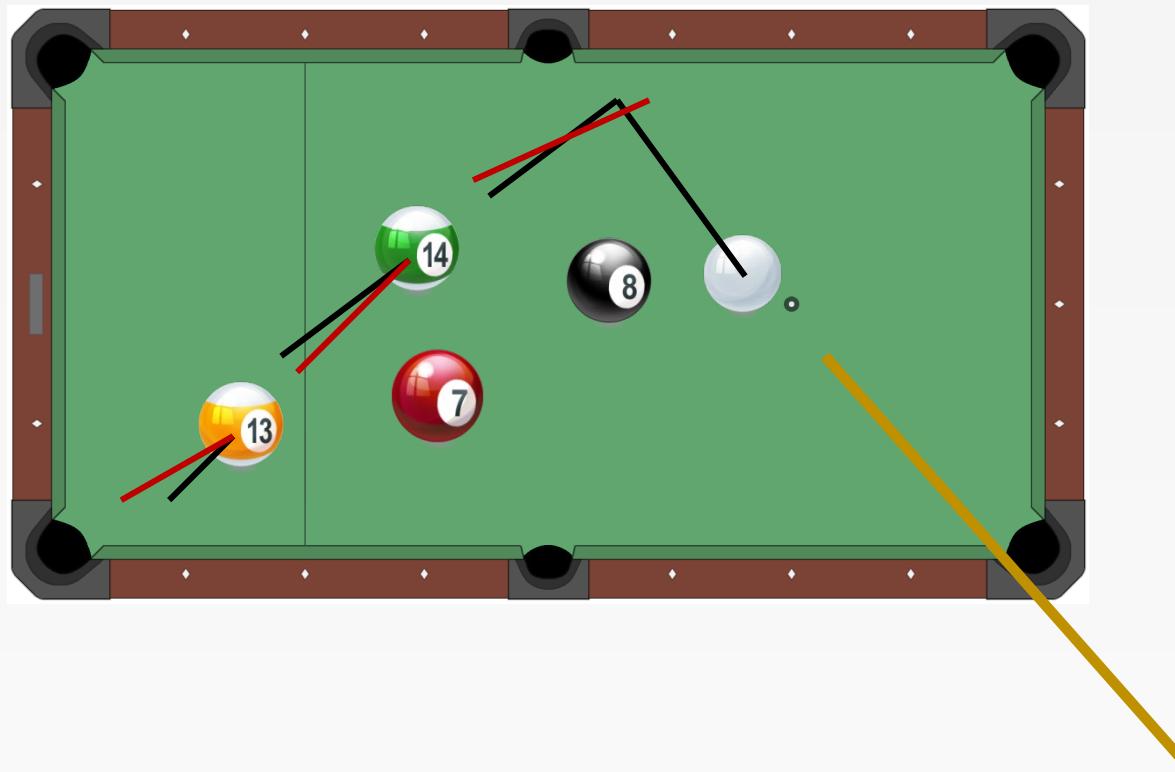
Error Back-Propagation



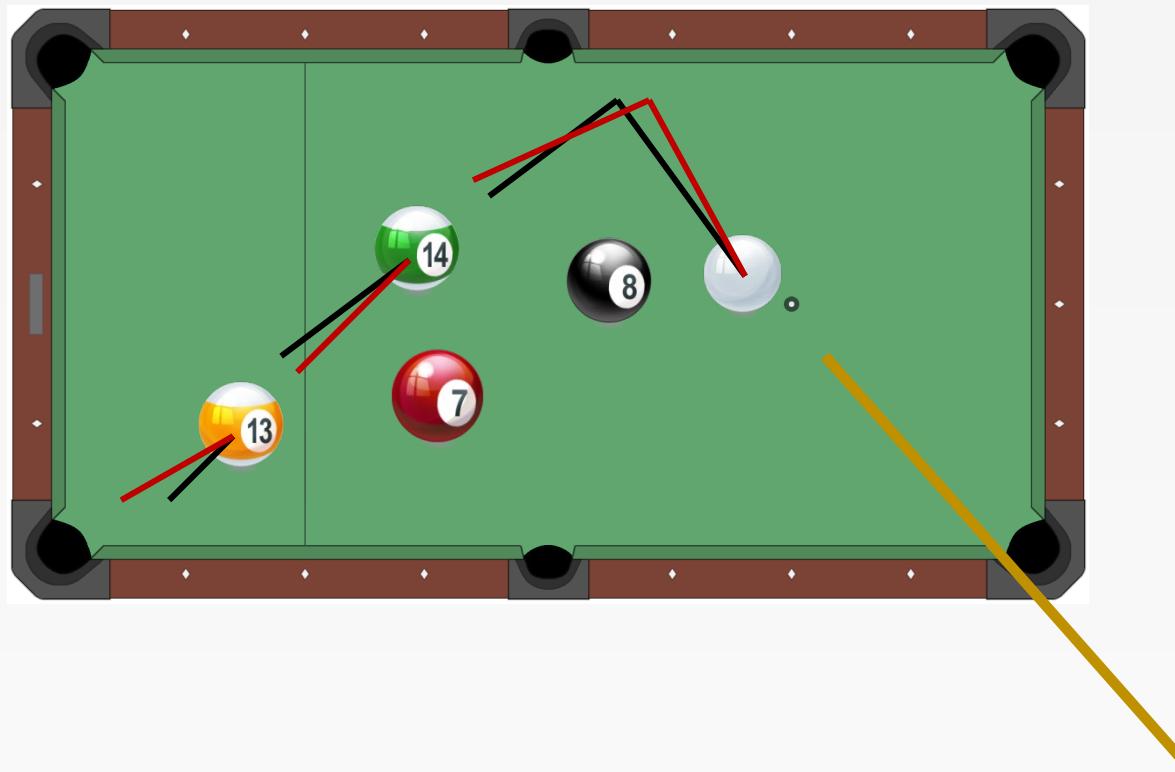
Error Back-Propagation



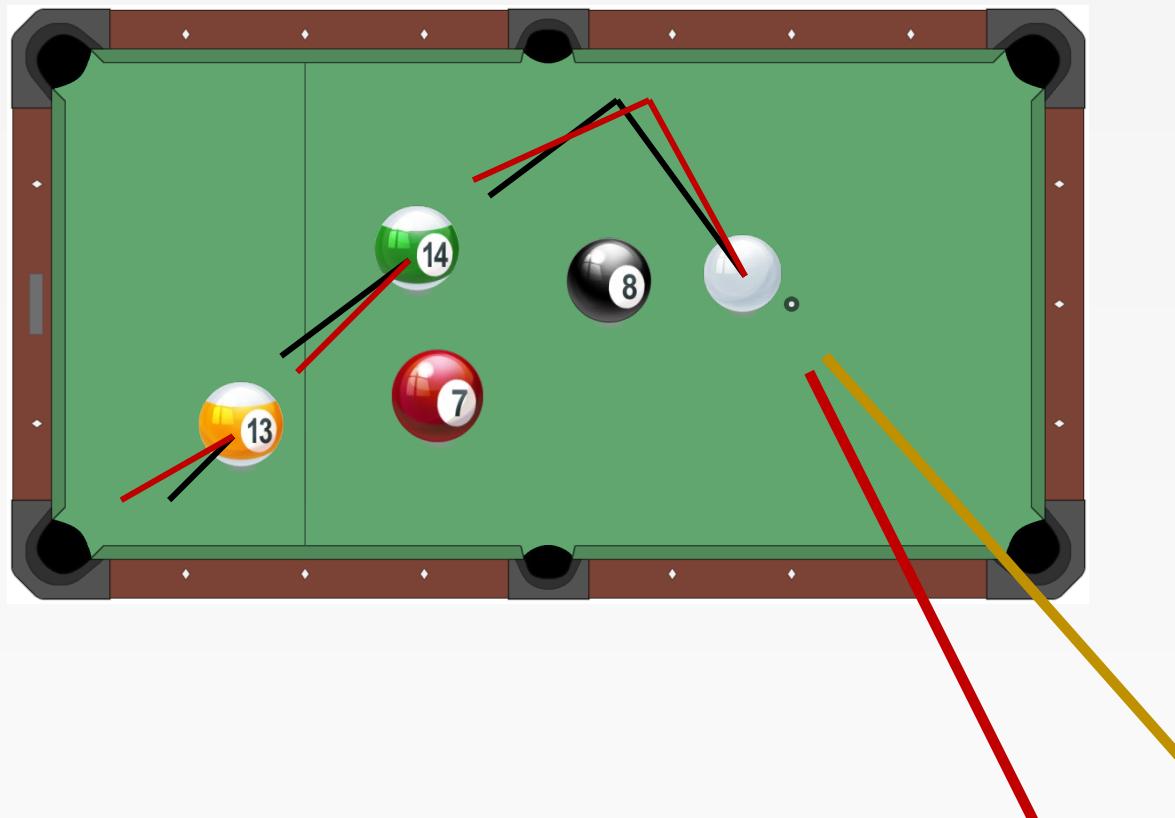
Error Back-Propagation



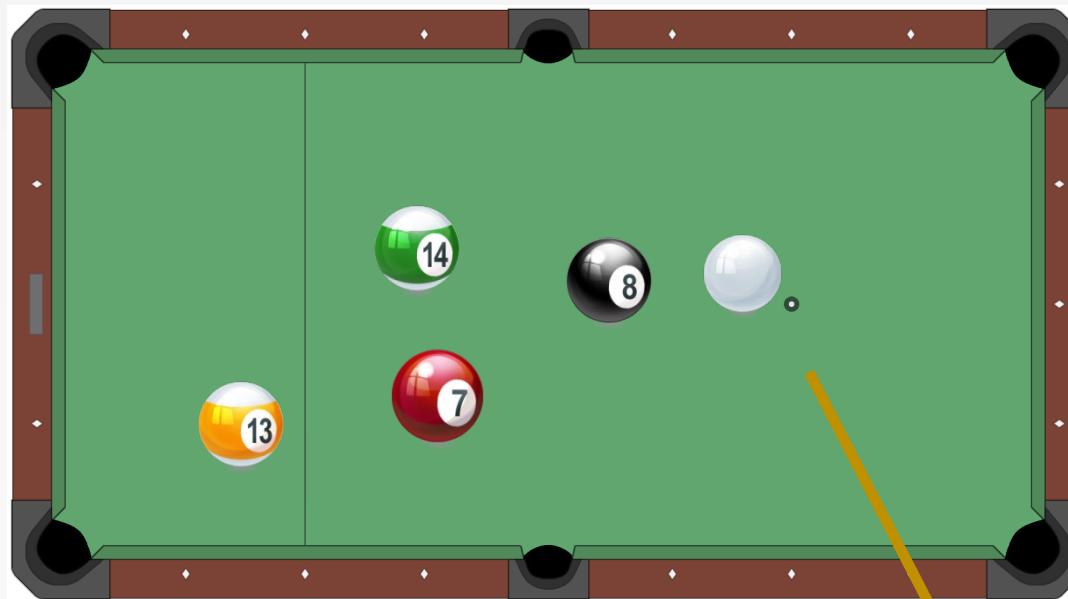
Error Back-Propagation



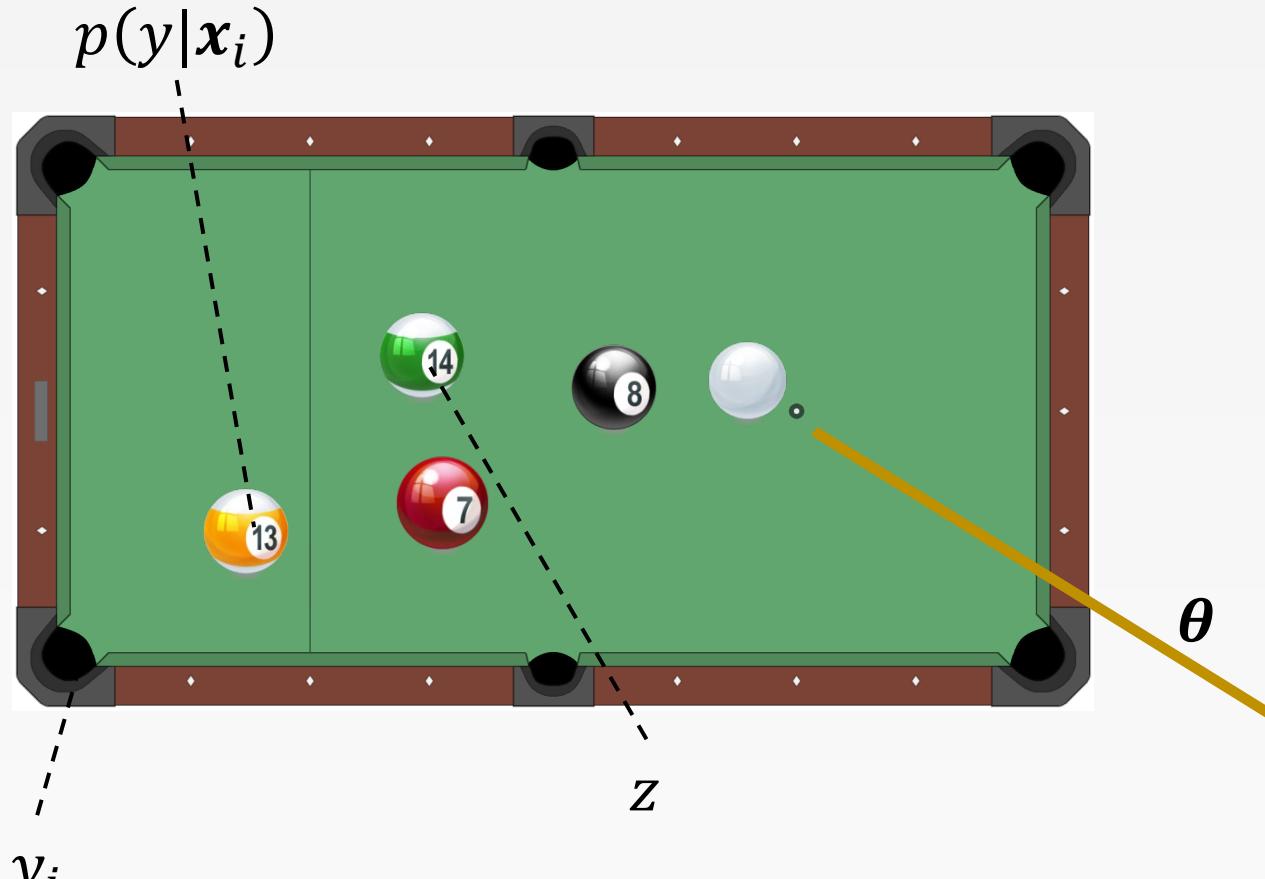
Error Back-Propagation



Error Back-Propagation



Error Back-Propagation



Simple Idea

For each training instance:

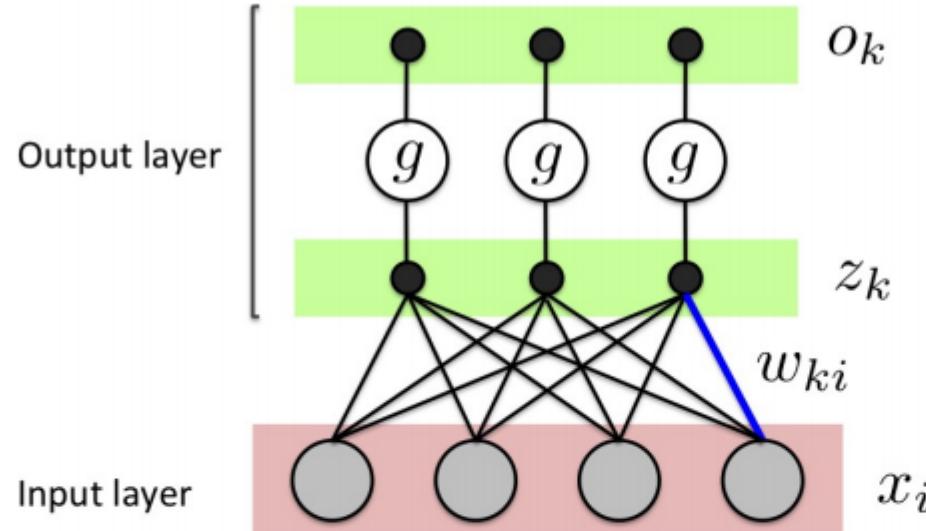
- Make a prediction (forward pass)

- Measure the error

- Go through each layer in reverse to measure the error contribution from each connection (reverse pass)

- Slightly tweaks the connection weights to reduce error (Gradient Descent)

Compute Gradient



- Error gradients for single layer network:
$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial z_k} \frac{\partial z_k}{\partial w_{ki}}$$
- Error gradient is computable for any continuous activation function $g()$, and any continuous error function

An Example

Forward

Backward

Loss

$$J = y^* \log y + (1 - y^*) \log(1 - y)$$

$$\frac{dJ}{dy} = \frac{y^*}{y} + \frac{(1 - y^*)}{1 - y}$$

Sigmoid

$$y = \frac{1}{1 + \exp(-b)}$$

$$\frac{dJ}{db} = \frac{dJ}{dy} \frac{dy}{db}, \quad \frac{dy}{db} = \frac{\exp(-b)}{(\exp(-b) + 1)^2}$$

Linear

$$b = \sum_{j=0}^D \beta_j z_j$$

$$\frac{dJ}{d\beta_j} = \frac{dJ}{db} \frac{db}{d\beta_j}, \quad \frac{db}{d\beta_j} = z_j$$

$$\frac{dJ}{dz_j} = \frac{dJ}{db} \frac{db}{dz_j}, \quad \frac{db}{dz_j} = \beta_j$$

Sigmoid

$$z_j = \frac{1}{1 + \exp(-a_j)}$$

$$\frac{dJ}{da_j} = \frac{dJ}{dz_j} \frac{dz_j}{da_j}, \quad \frac{dz_j}{da_j} = \frac{\exp(-a_j)}{(\exp(-a_j) + 1)^2}$$

Linear

$$a_j = \sum_{i=0}^M \alpha_{ji} x_i$$

$$\frac{dJ}{d\alpha_{ji}} = \frac{dJ}{da_j} \frac{da_j}{d\alpha_{ji}}, \quad \frac{da_j}{d\alpha_{ji}} = x_i$$

$$\frac{dJ}{dx_i} = \frac{dJ}{da_j} \frac{da_j}{dx_i}, \quad \frac{da_j}{dx_i} = \sum_{j=0}^D \alpha_{ji}$$



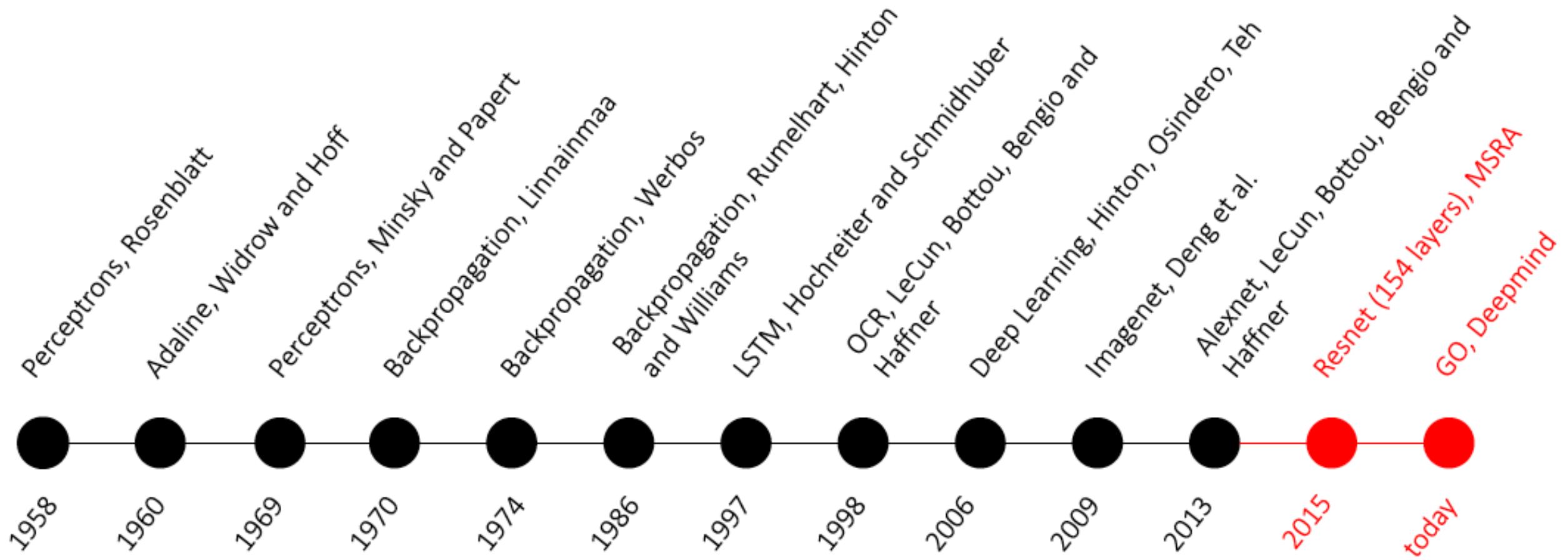
Backpropagation

- Training neural nets:
- Loop until convergence:
 - for each example n
 - Given input, propagate activity forward ($x \rightarrow h \rightarrow \ell$) (**forward pass**)
 - Propagate gradients backward (**backward pass**)
 - Update each weight (via **gradient descent**)

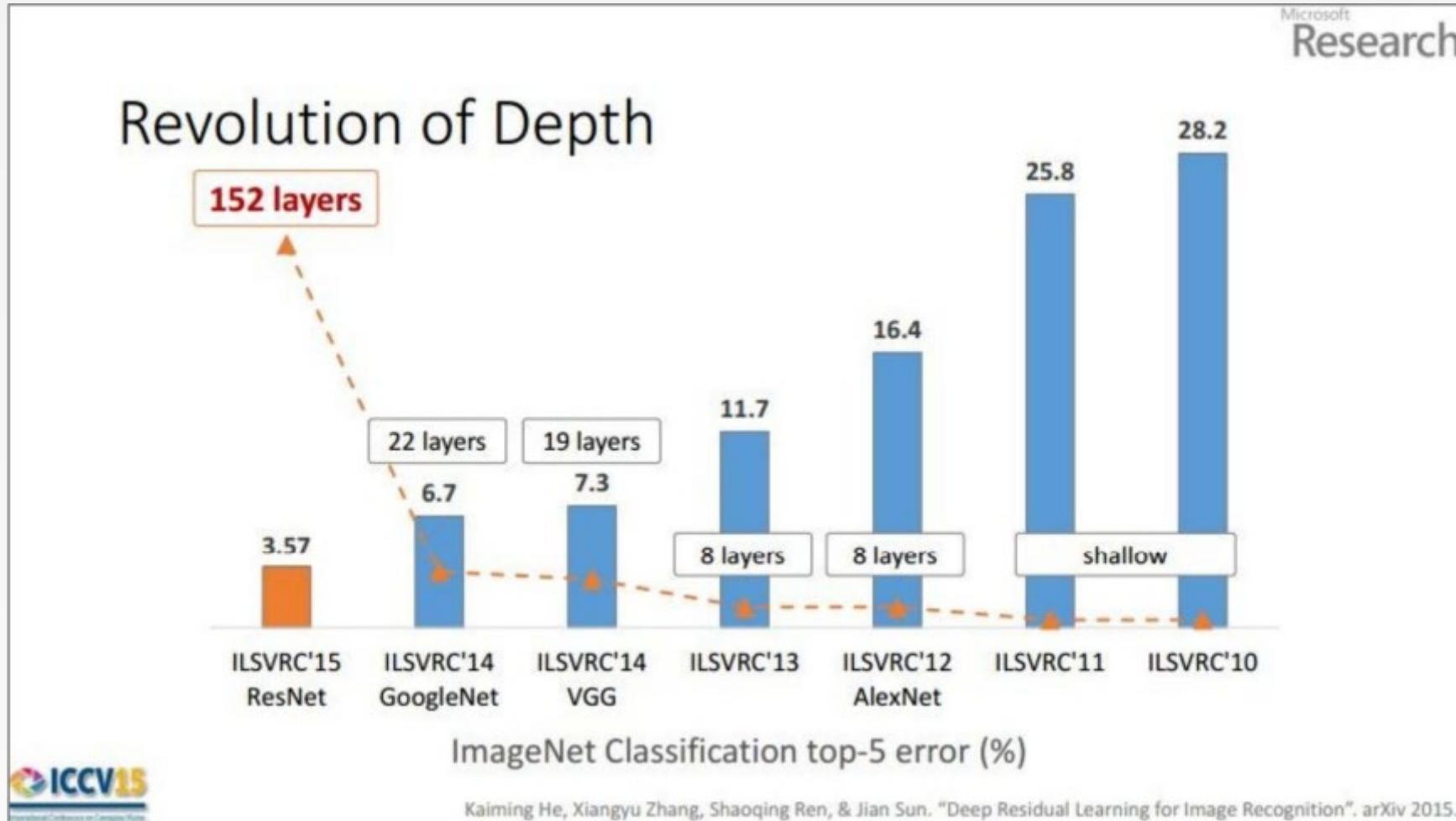
Deep Learning



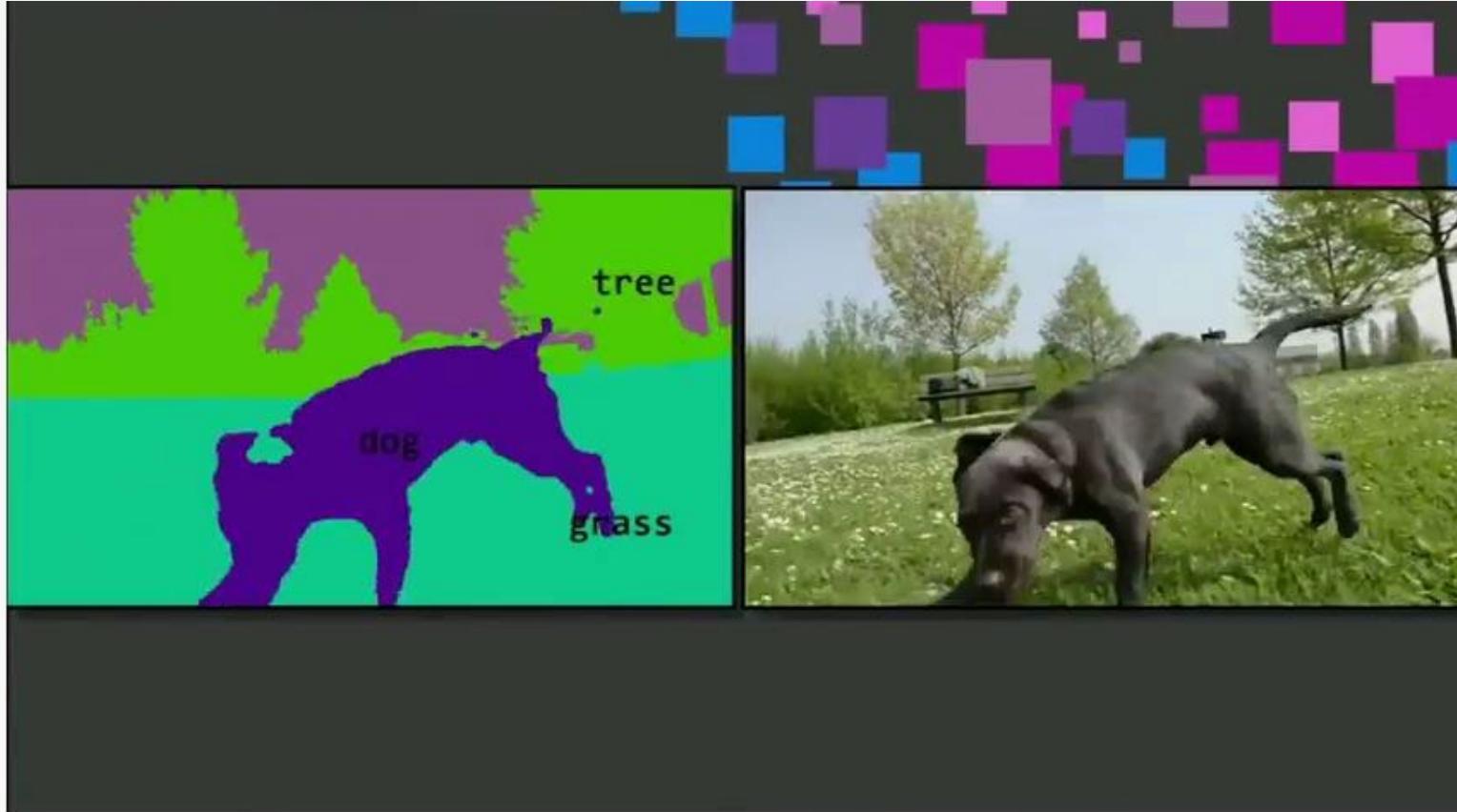
Why is everyone talking about Deep Learning?



CNN for Image Recognition



Object Detection



Microsoft Deep Learning Semantic Image Segmentation

Captioning

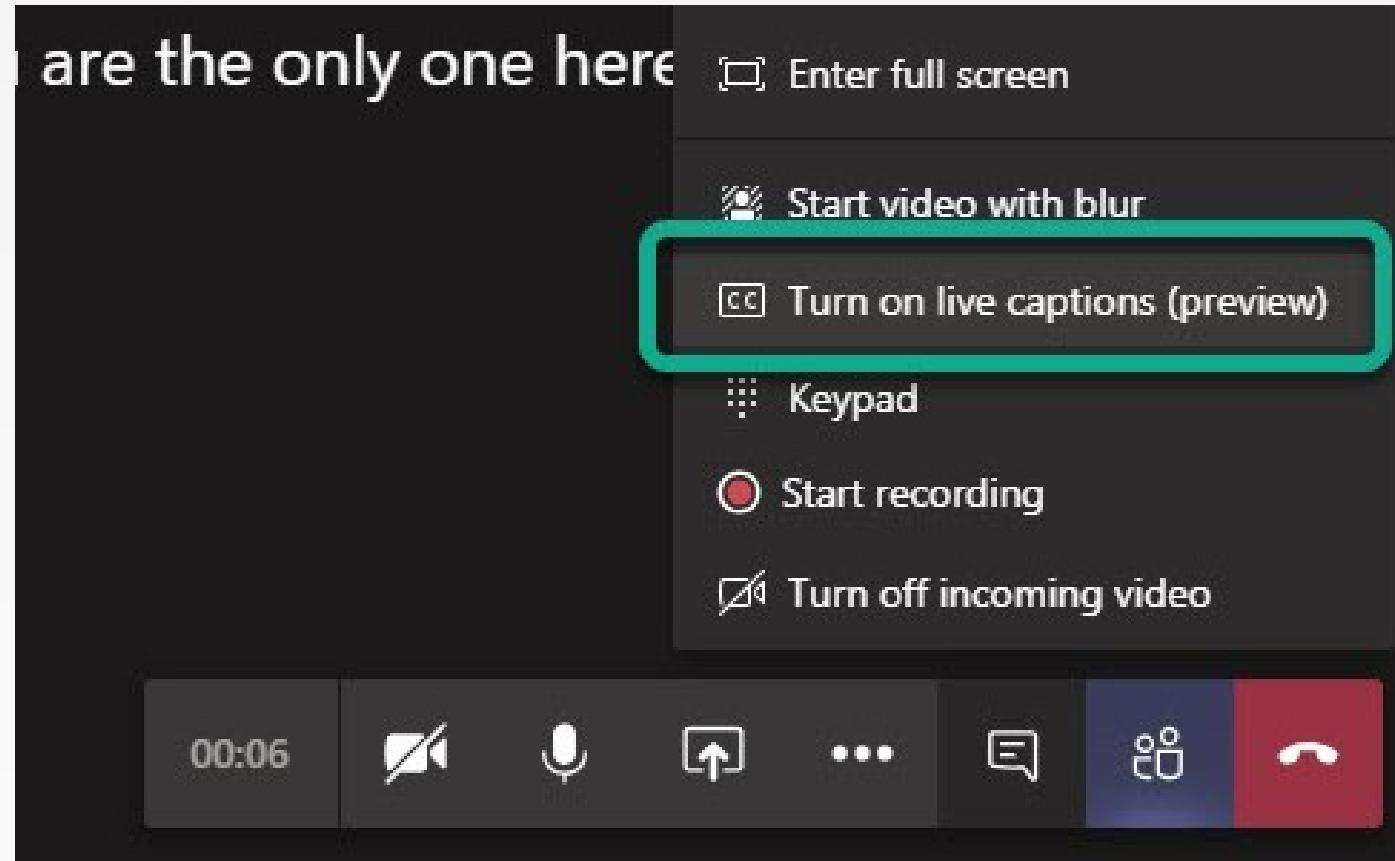
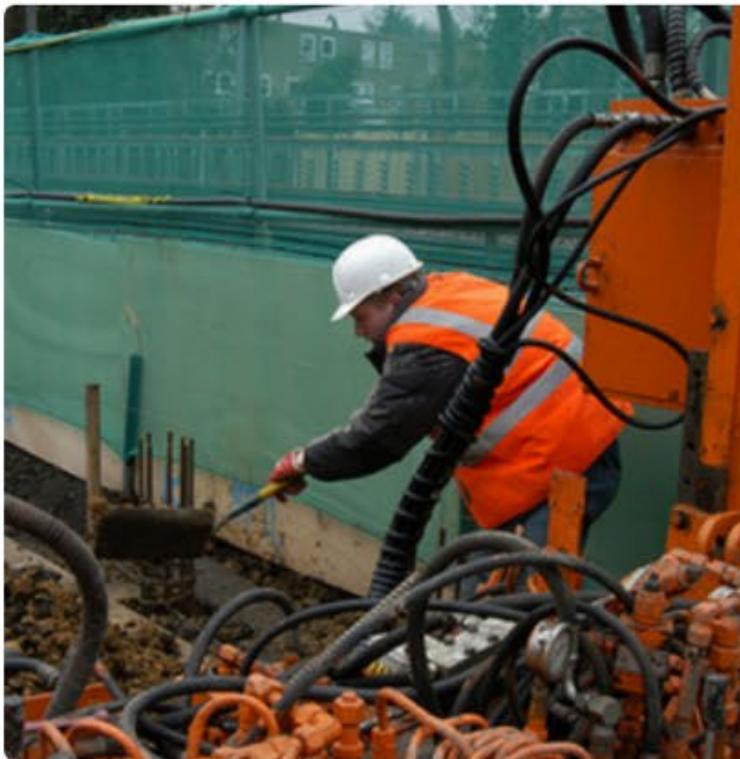


Image Captioning



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



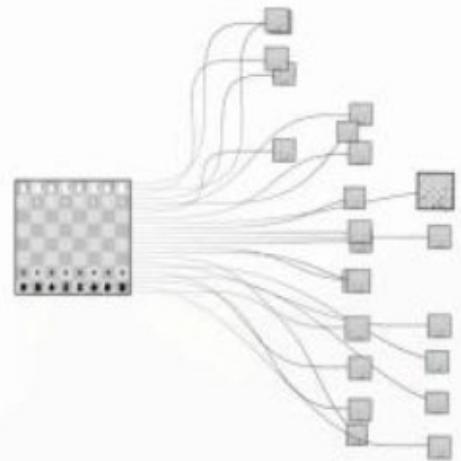
"two young girls are playing with lego toy."

Self-driving Cars

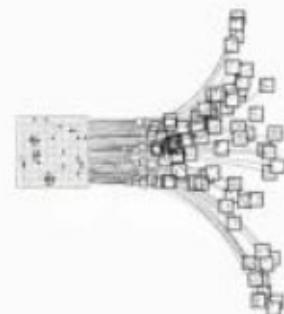


Self Driving Cars HD

Game AI



Chess: 10^{47}
Deep Blue, Feb 10, 1996



Go: 10^{170}
AlphaGo, March, 2016

Let's see how it works!

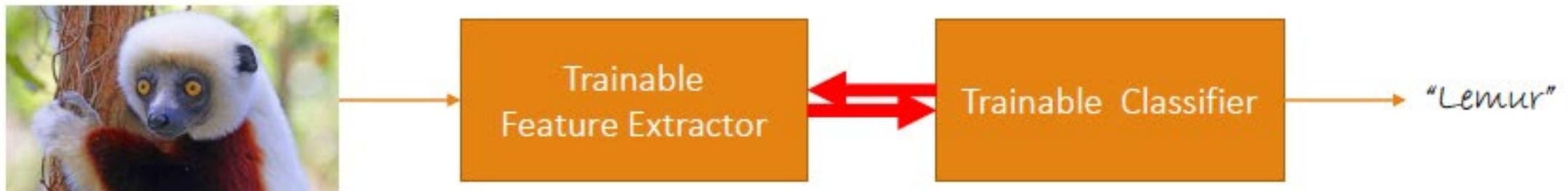


Comparison

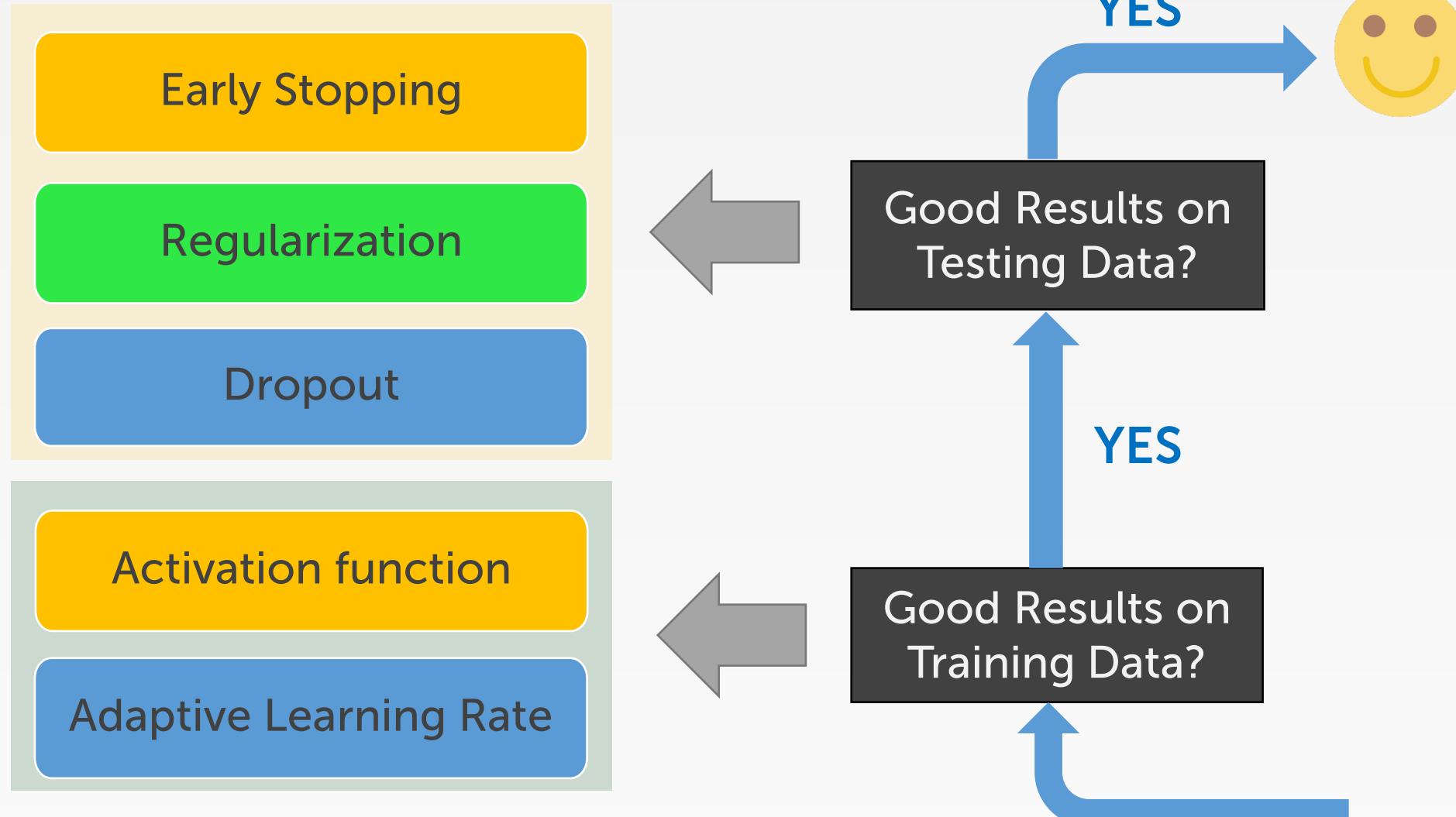
- Traditional pattern recognition



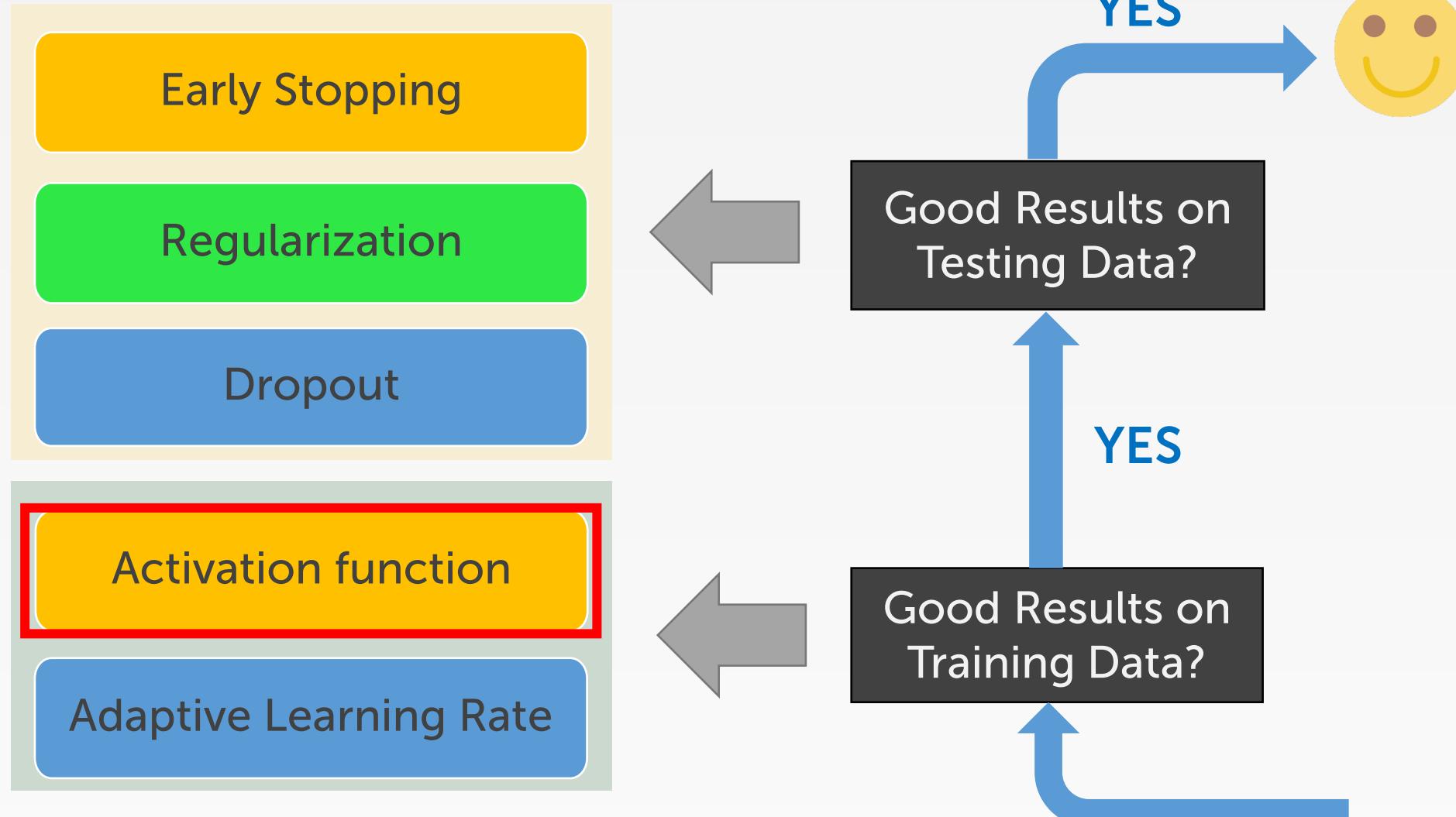
- End-to-end learning → Features are also learned from data



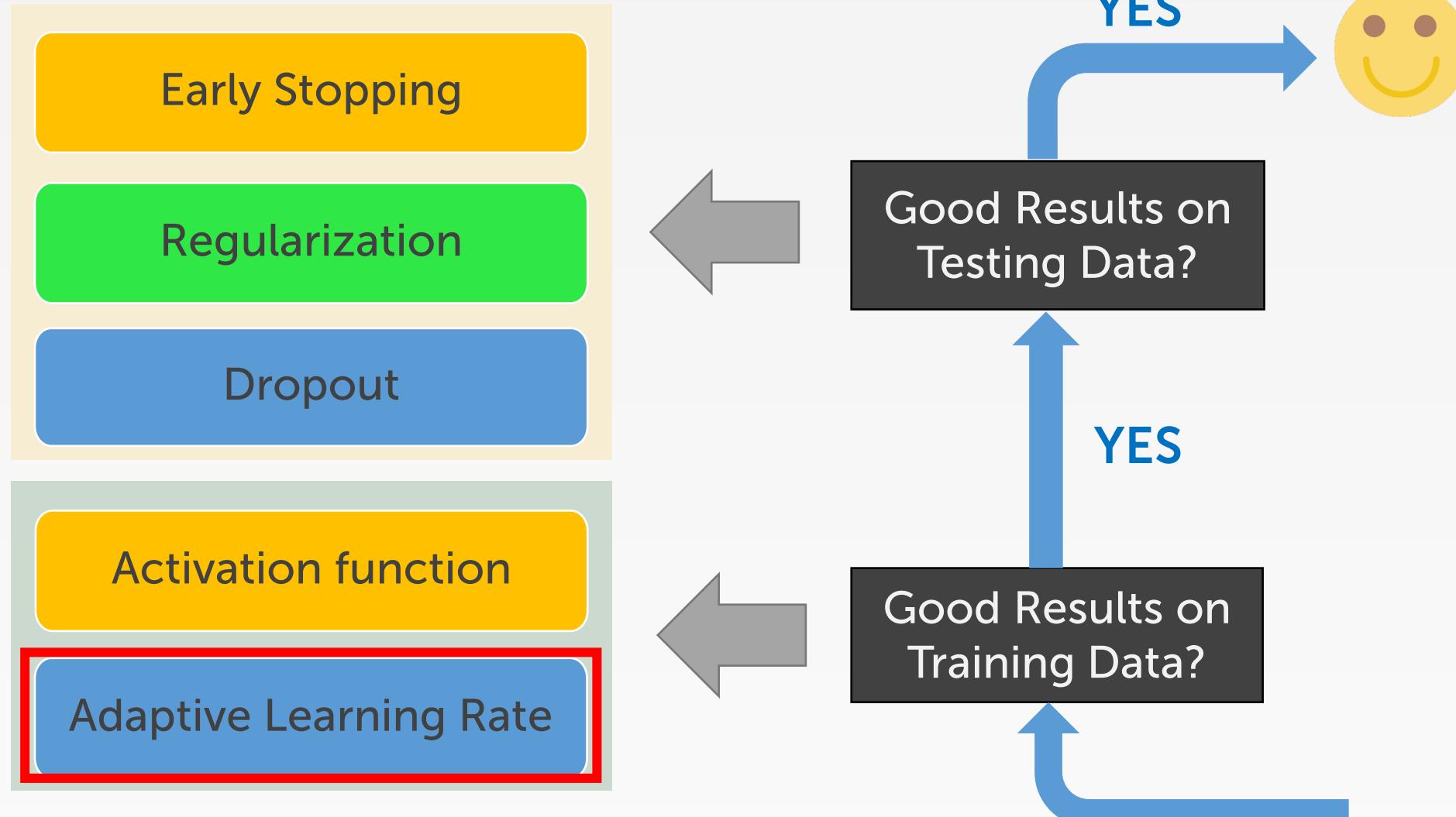
Recipe of Deep Learning



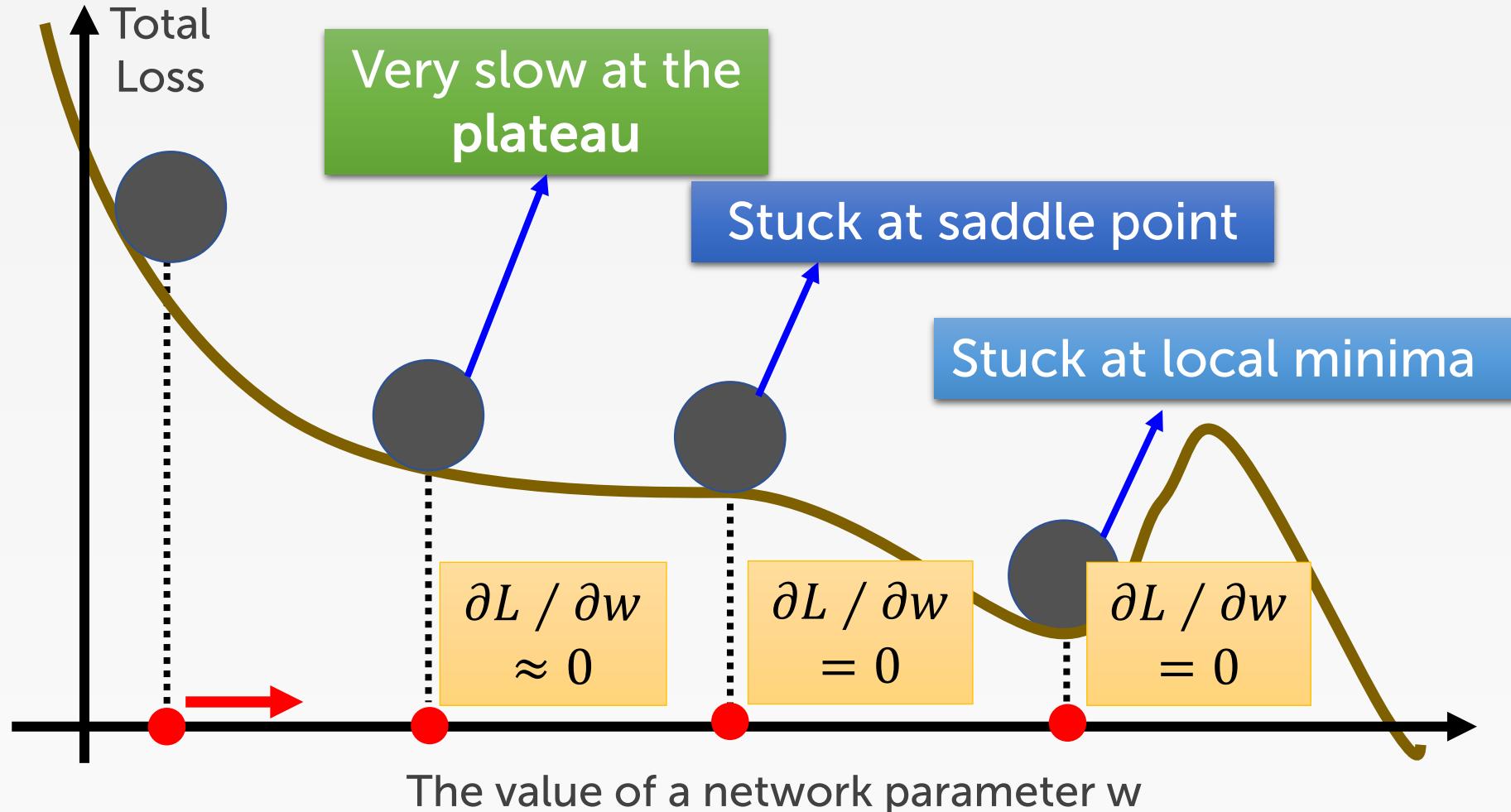
Recipe of Deep Learning



Recipe of Deep Learning



Hard to find optimal network parameters



Learning Rate Scheduling

- Predetermined piecewise constant learning rate

For example, set the learning rate to $\eta_0 = 0.1$ at first, then to $\eta_1 = 0.001$ after 50 epochs. Although this solution can work very well, it often requires fiddling around to figure out the right learning rates and when to use them.

- Performance scheduling

Measure the validation error every N steps (just like for early stopping) and reduce the learning rate by a factor of λ when the error stops dropping.

- Exponential scheduling

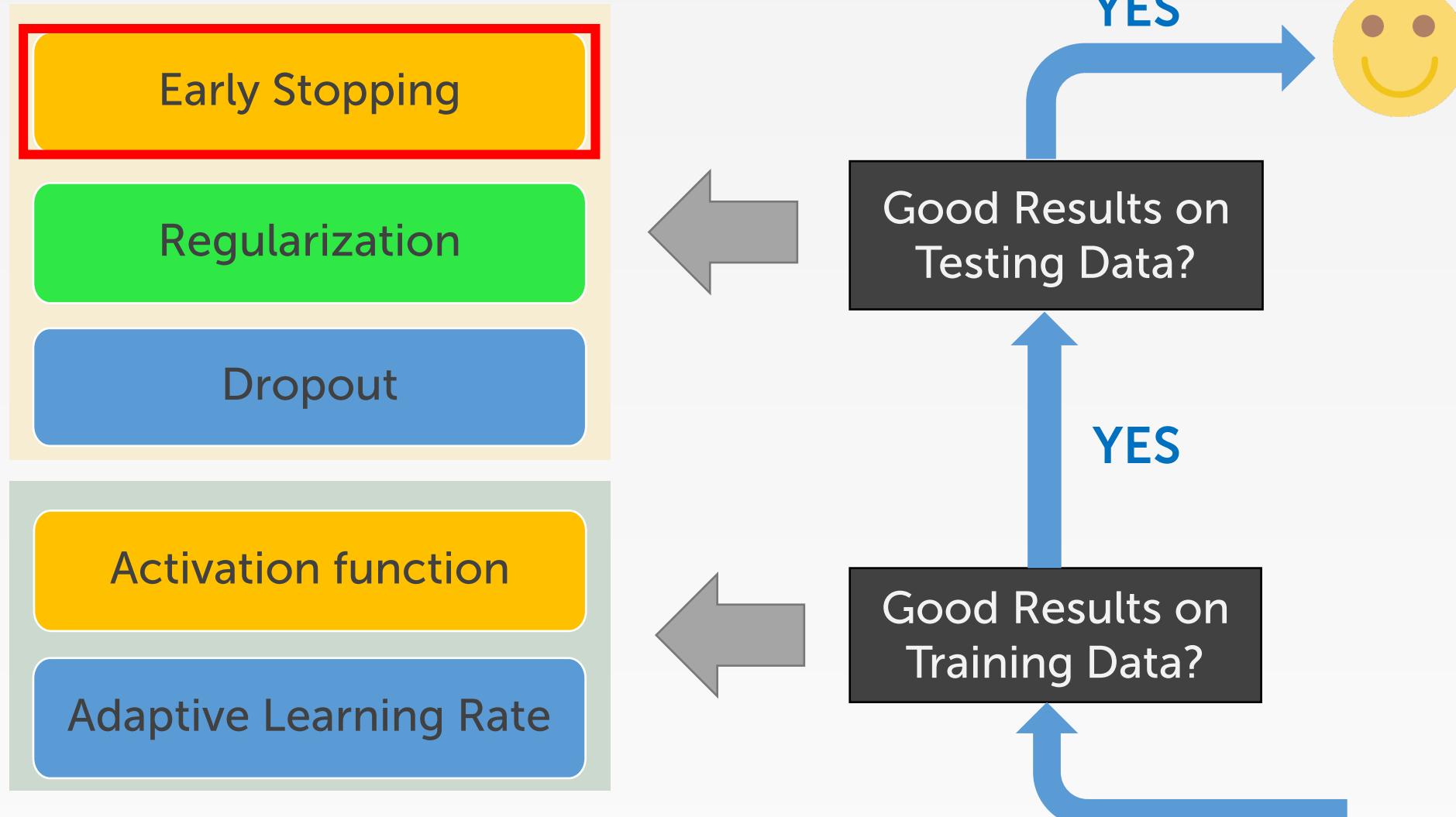
Set the learning rate to a function of the iteration number t: $\eta(t) = \eta_0 10^{-t/r}$. This works great, but it requires tuning η_0 and r. The learning rate will drop by a factor of 10 every r steps.

- Power scheduling

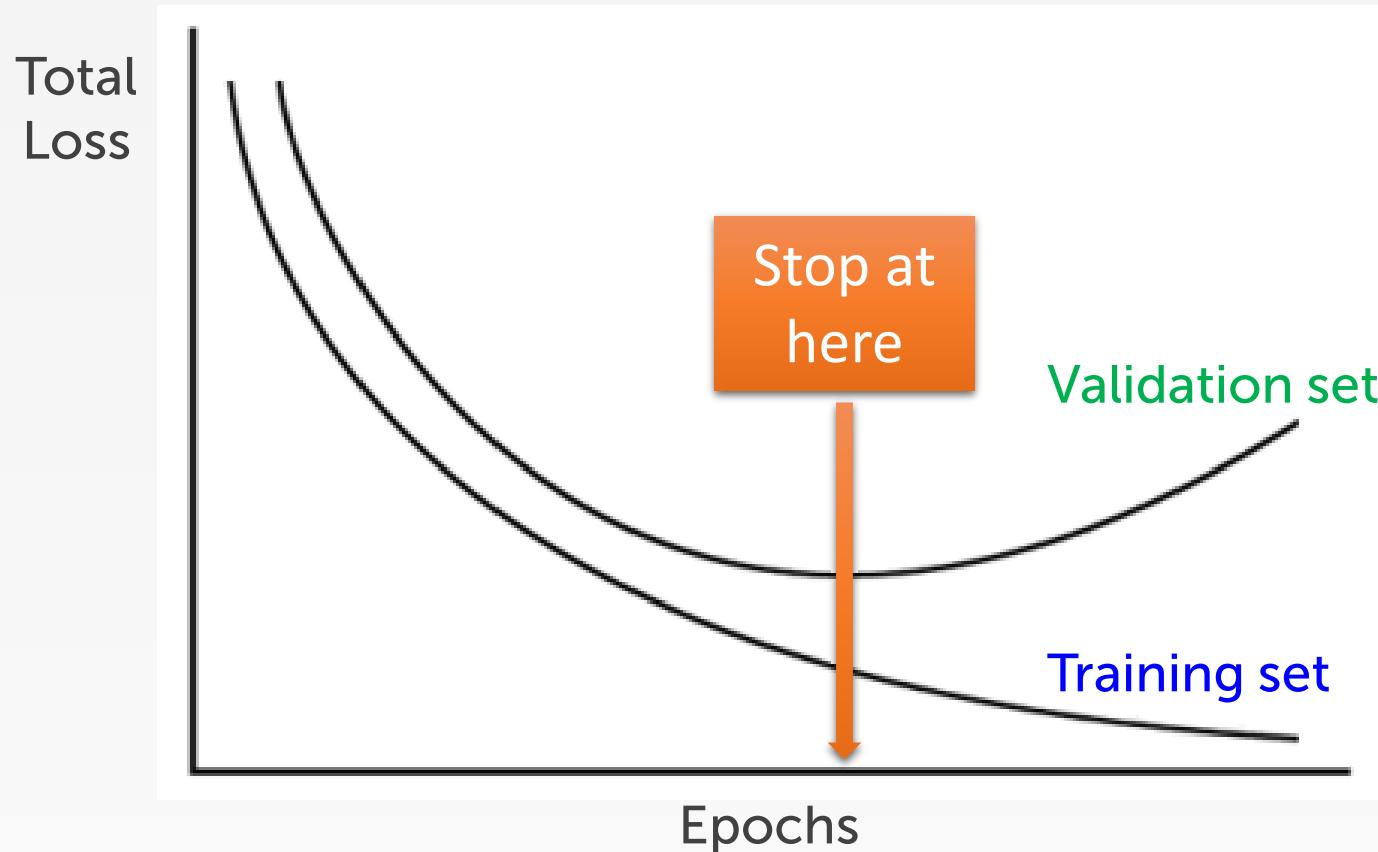
Set the learning rate to $\eta(t) = \eta_0 \left(1 + \frac{t}{r}\right)^{-c}$. The hyperparameter c is typically set to 1. This is similar to exponential scheduling, but the learning rate drops much more slowly.

Avoid Overfitting

Recipe of Deep Learning



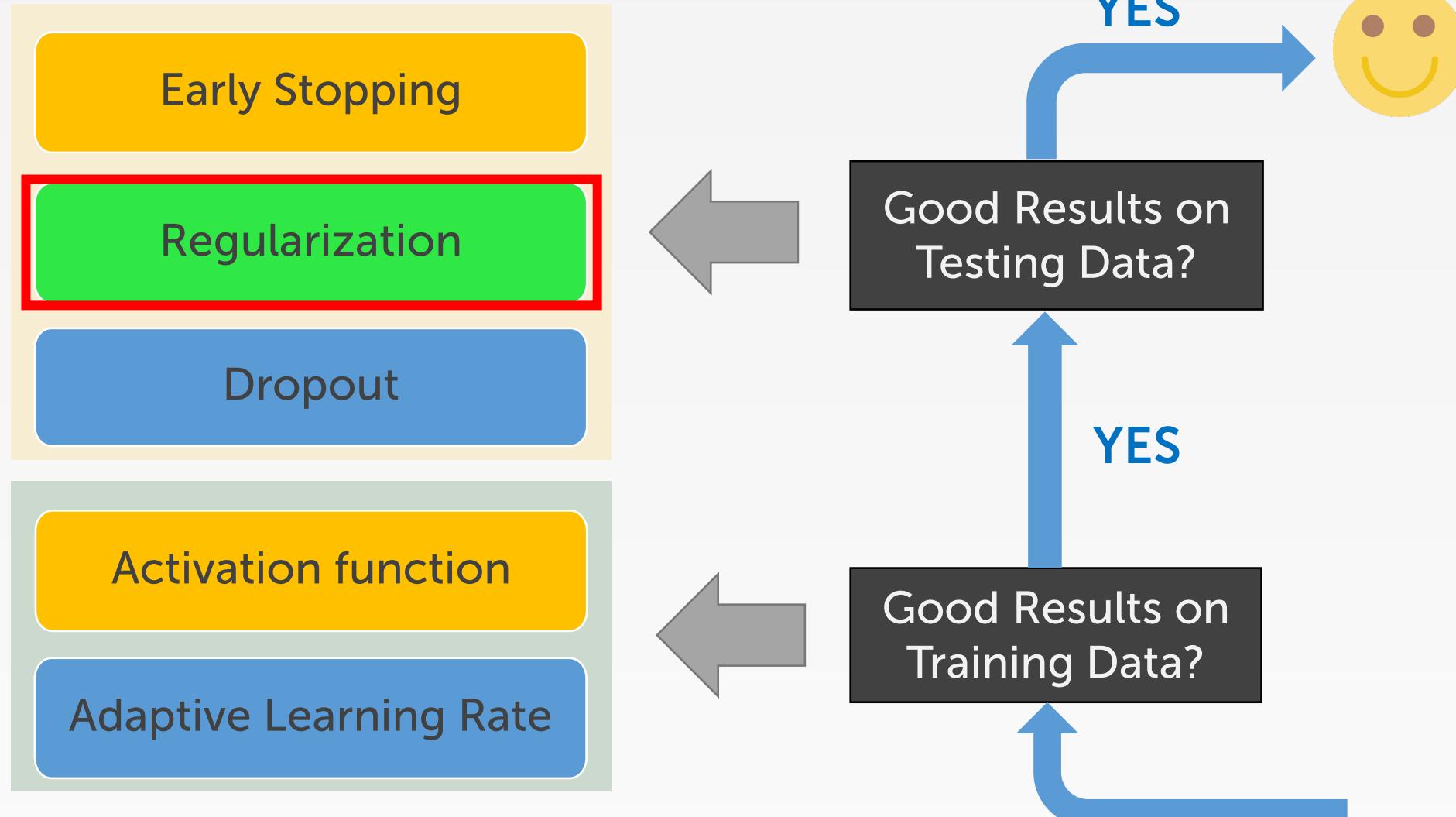
Early Stopping



Example:

- evaluate the model on a validation set at regular intervals (e.g., every 50 steps)
- save a “winner” snapshot if it outperforms previous “winner” snapshots
- Count the number of steps since the last “winner” snapshot was saved, and interrupt training when this number reaches some limit (e.g., 2,000 steps)
- restore the last “winner” snapshot.

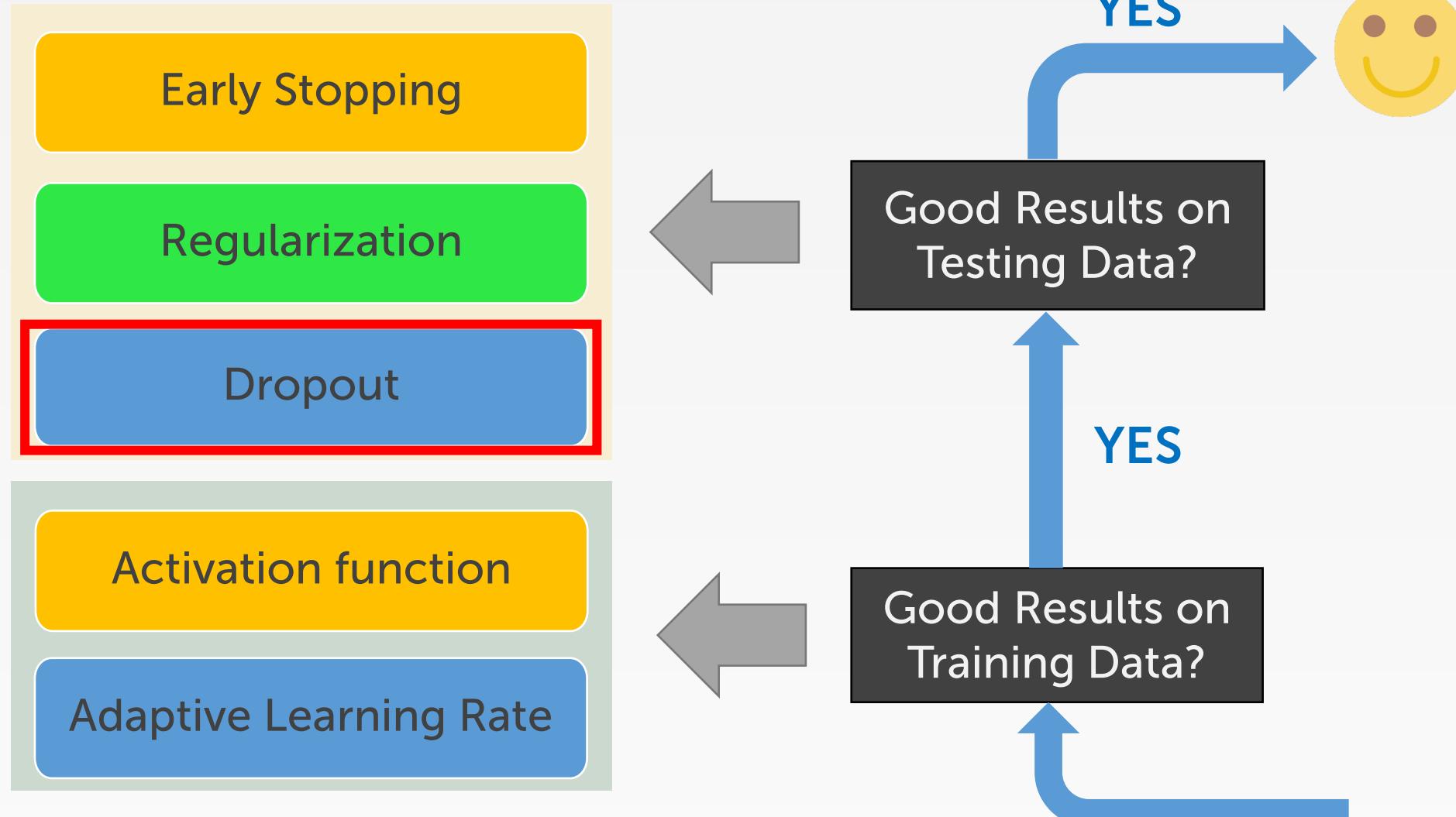
Recipe of Deep Learning



Regulation

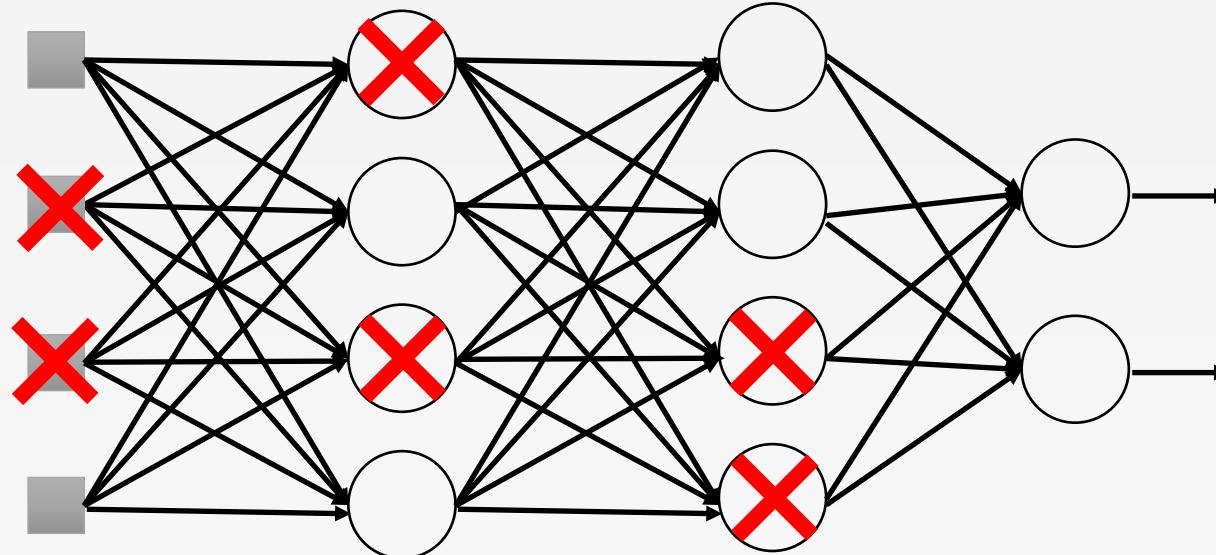
- New loss function:
 - $L'(\theta) = L(\theta) + \lambda \frac{1}{2} \|\theta\|_2$
- $l2$ normalization:
 - $\|\theta\|_2 = (\omega_1)^2 + (\omega_2)^2 + \dots$
- $l1$ normalization:
 - $\|\theta\|_2 = |\omega_1| + |\omega_2| + \dots$

Recipe of Deep Learning



Dropout

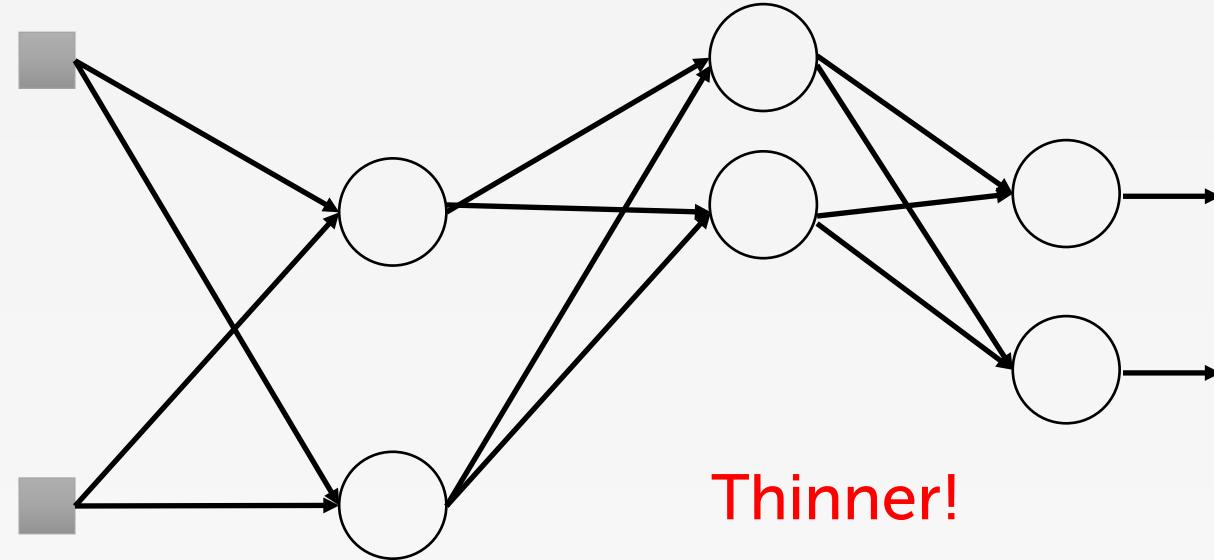
Training:



- **Each time before updating the parameters**
 - Each neuron has $p\%$ to dropout

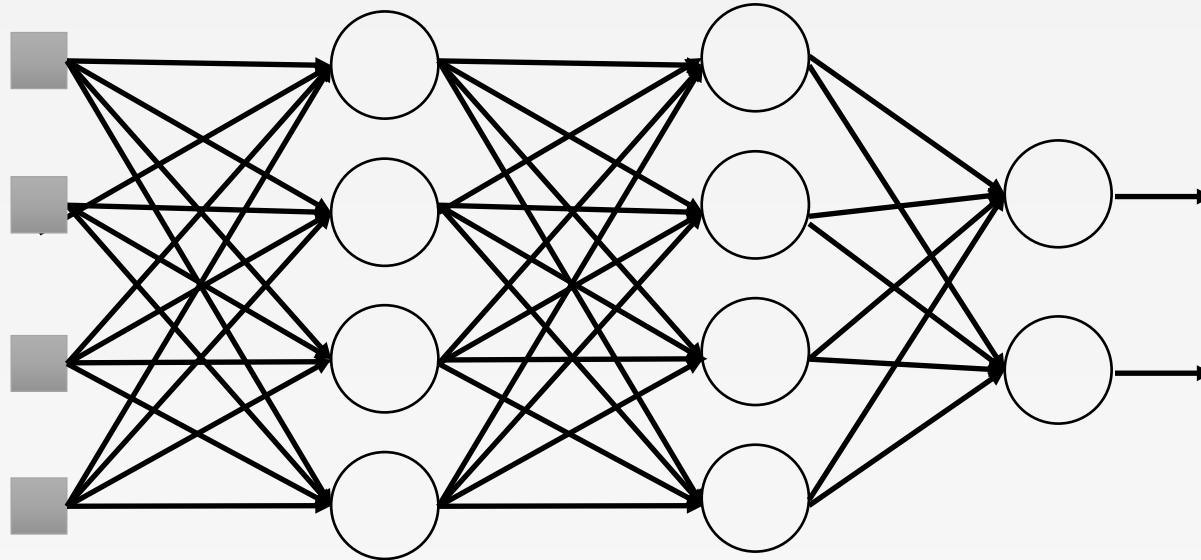
Dropout

Training:



Dropout

Testing:



➤ No dropout

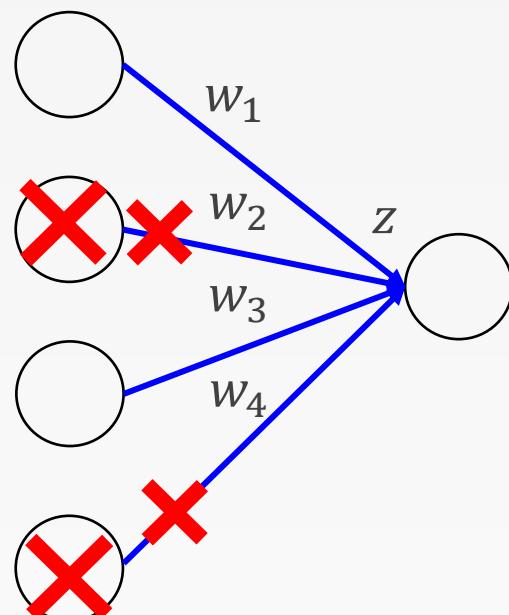
- If the dropout rate at training is $p\%$, all the weights times $1-p\%$
- Assume that the dropout rate is 50%.
If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout - Intuitive Reason

- Why the weights should multiply $(1-p)\%$ (dropout rate) when testing?

Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout

Weights from training

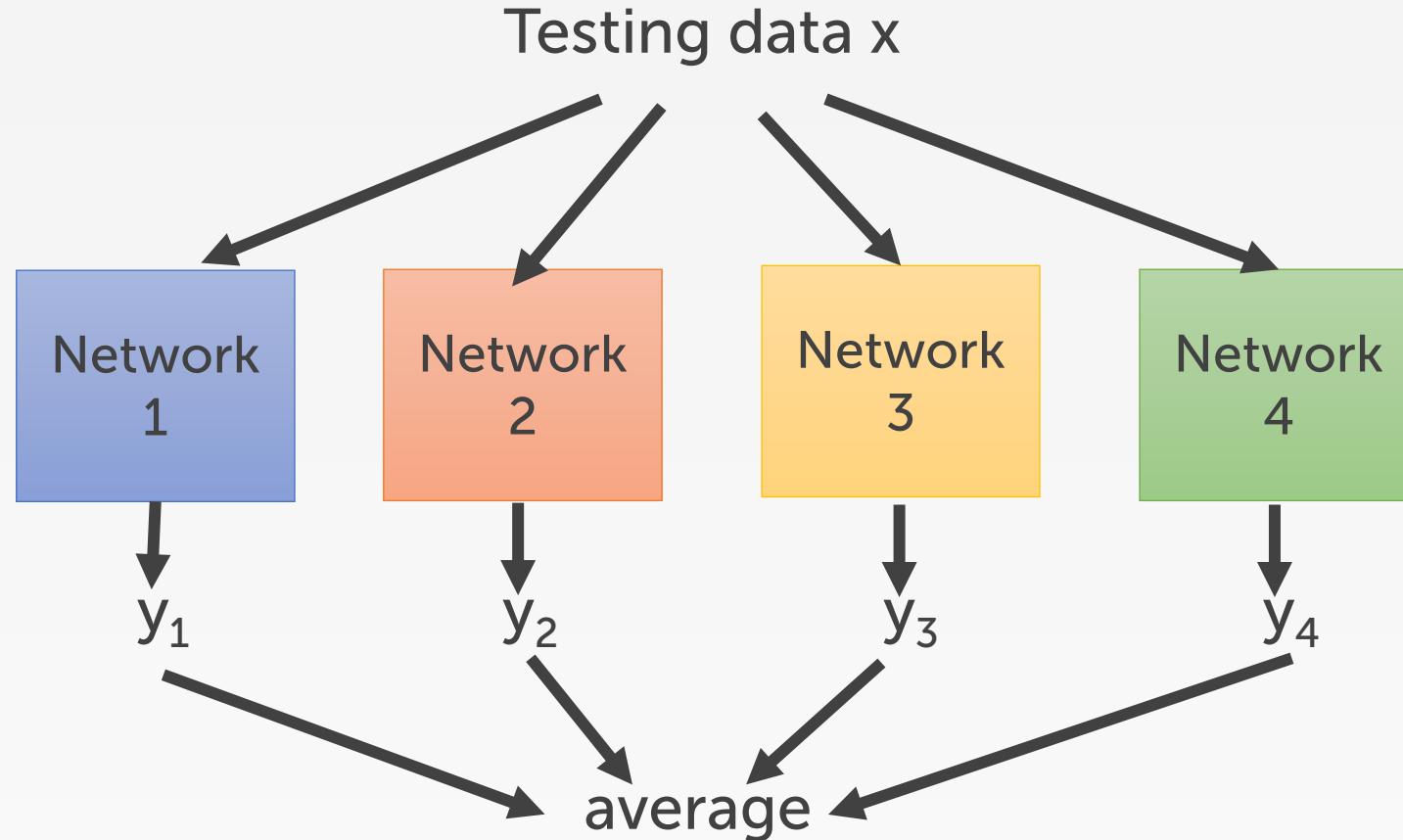
$$\begin{matrix} 0.5 \times \\ \text{---} \\ w_1 \end{matrix} \quad \longrightarrow \quad z' \approx 2z$$

$$\begin{matrix} 0.5 \times \\ \text{---} \\ w_2 \end{matrix} \quad \begin{matrix} 0.5 \times \\ \text{---} \\ w_3 \end{matrix} \quad \begin{matrix} 0.5 \times \\ \text{---} \\ w_4 \end{matrix}$$

Weights multiply $1-p\%$

$$\longrightarrow \quad z' \approx z$$

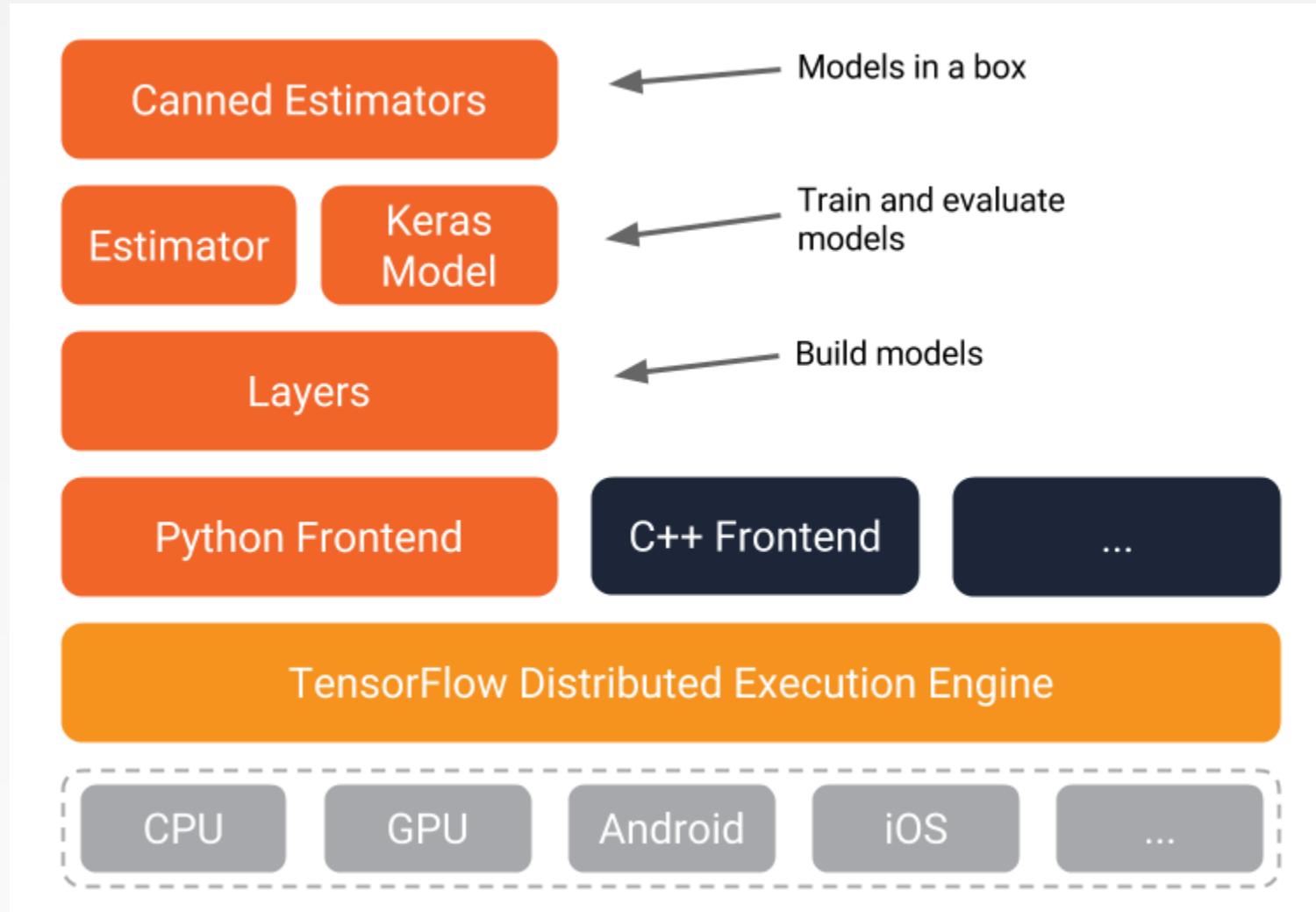
Dropout is a kind of ensemble.

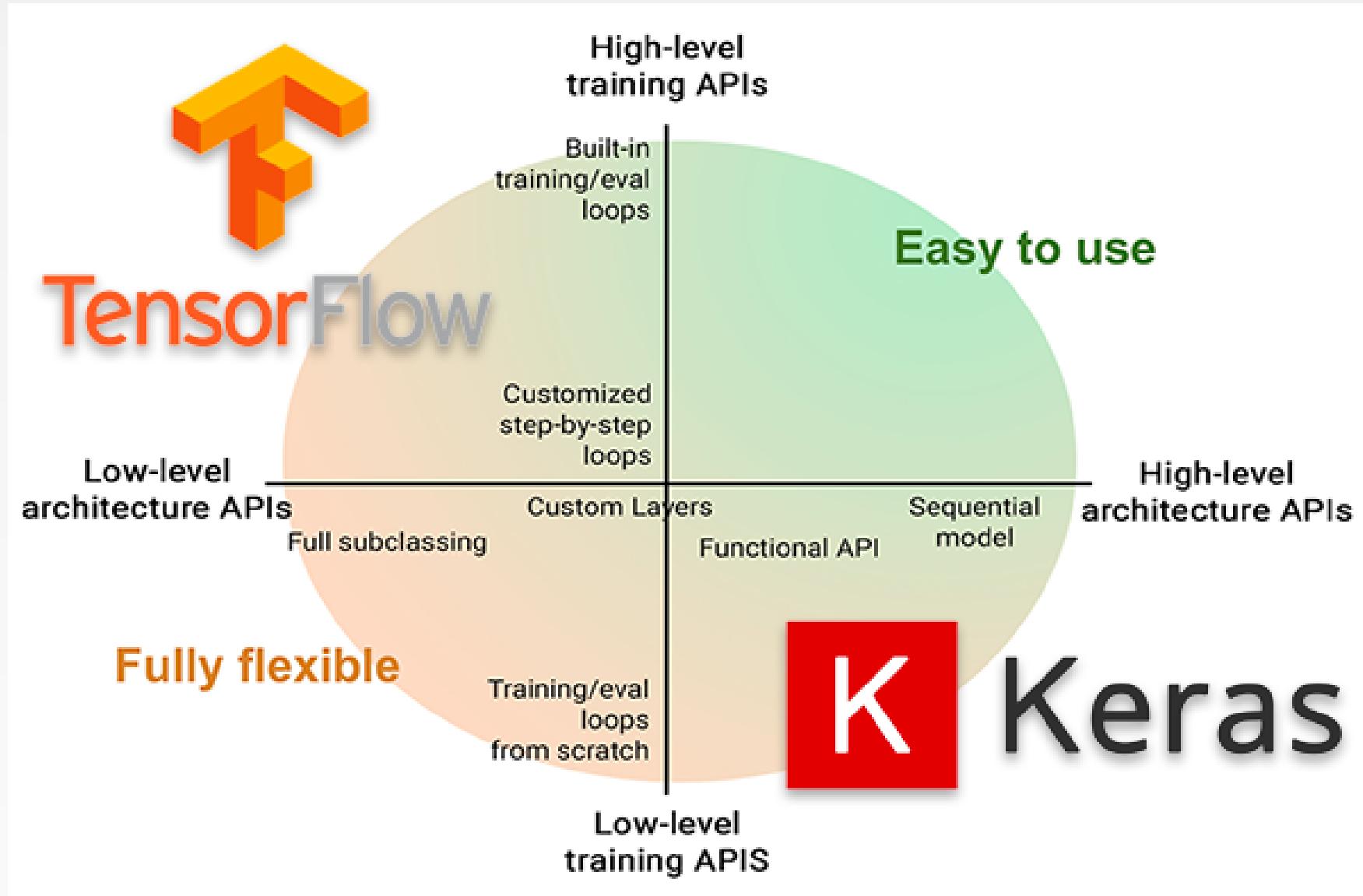




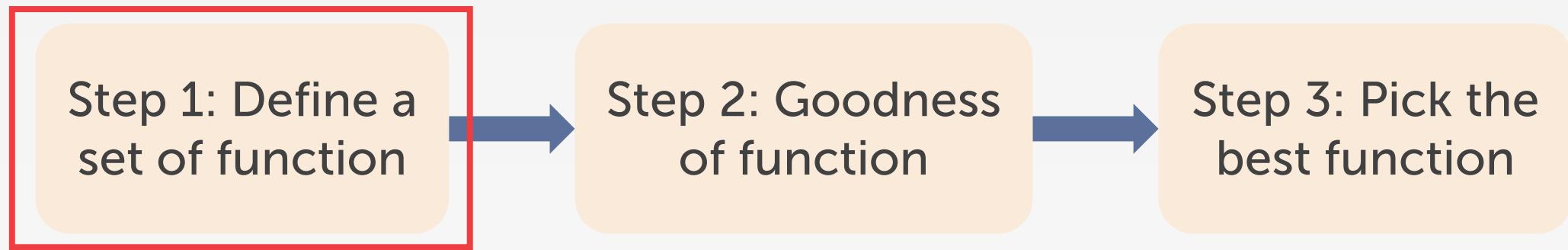
“Hello World”

TensorFlow

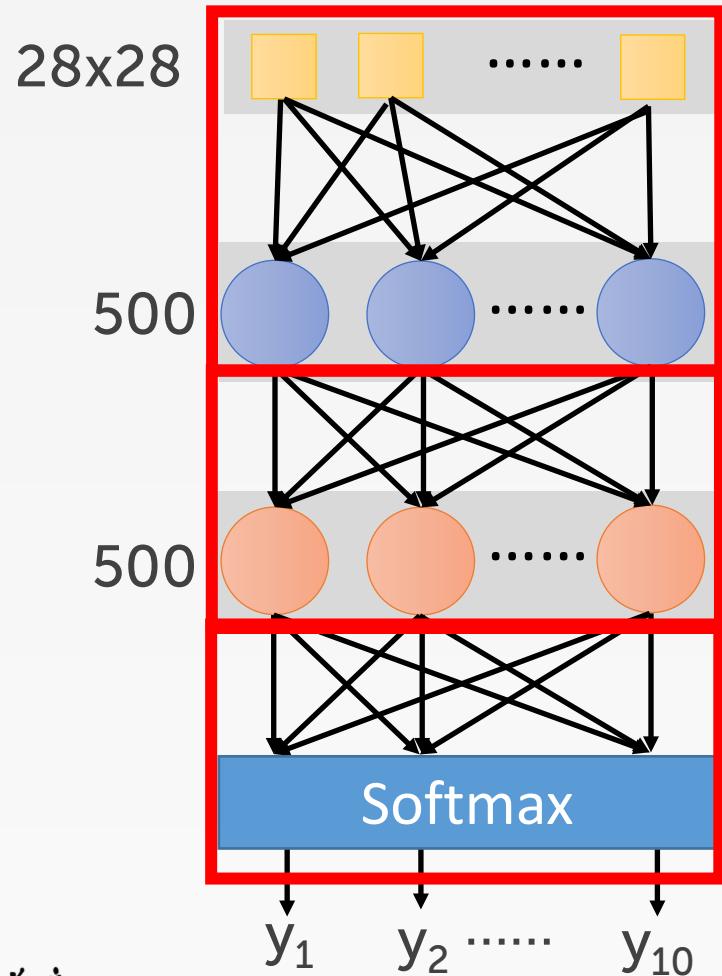




Keras



Keras: Define a Function



```
model = Sequential()
```

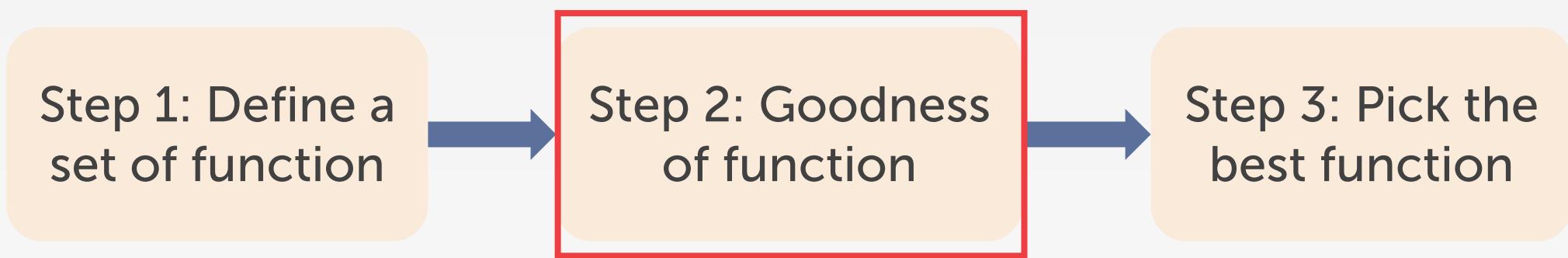
```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ) )
model.add( Activation('sigmoid') )
```

softplus, softsign, relu, tanh, hard_sigmoid, linear

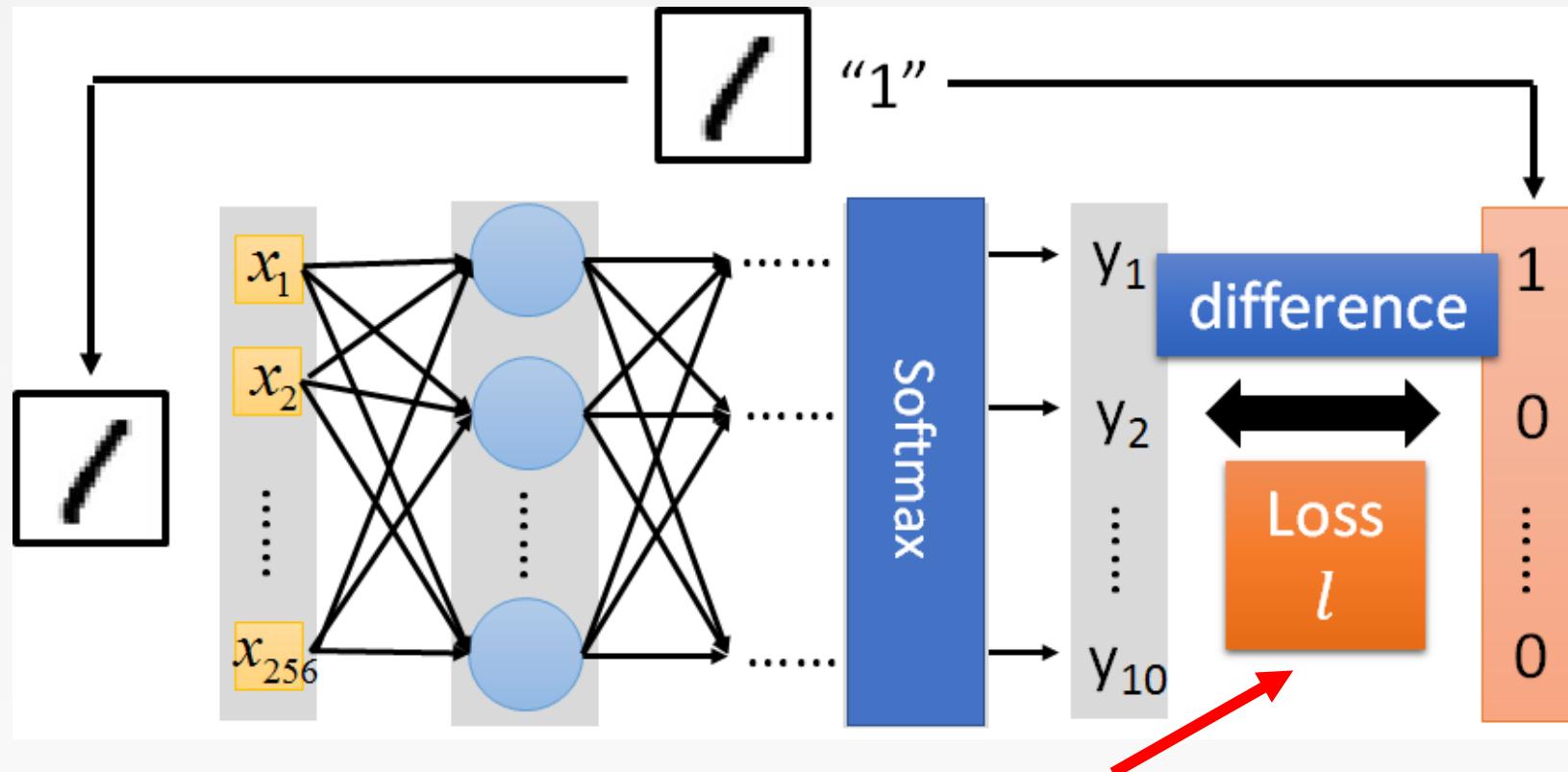
```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

```
model.add( Dense( output_dim=10 ) )
model.add( Activation('softmax') )
```

Keras



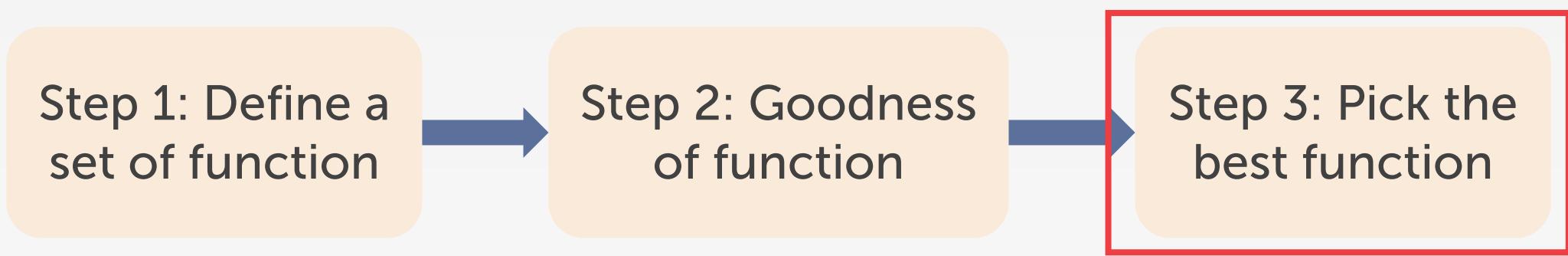
Keras: Goodness of Function



```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Several alternatives: <https://keras.io/objectives/>

Keras



Keras: Pick the Best

Step 3.1: Configuration

```
model.compile(loss='categorical_crossentropy',  
              optimizer='adam',  
              metrics=['accuracy'])
```

SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam

Step 3.2: Find the optimal network parameters

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

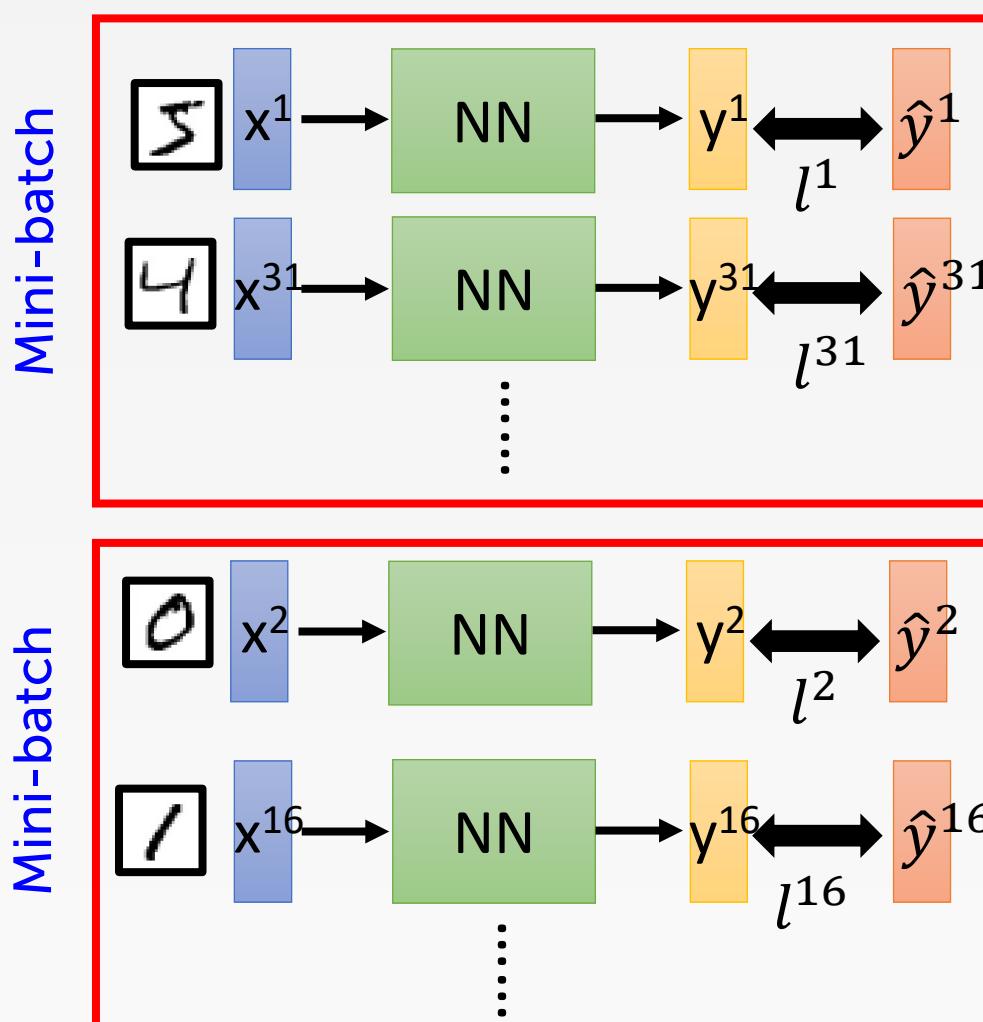
Training data
(Images)

Labels

In the following slides



Mini-batch

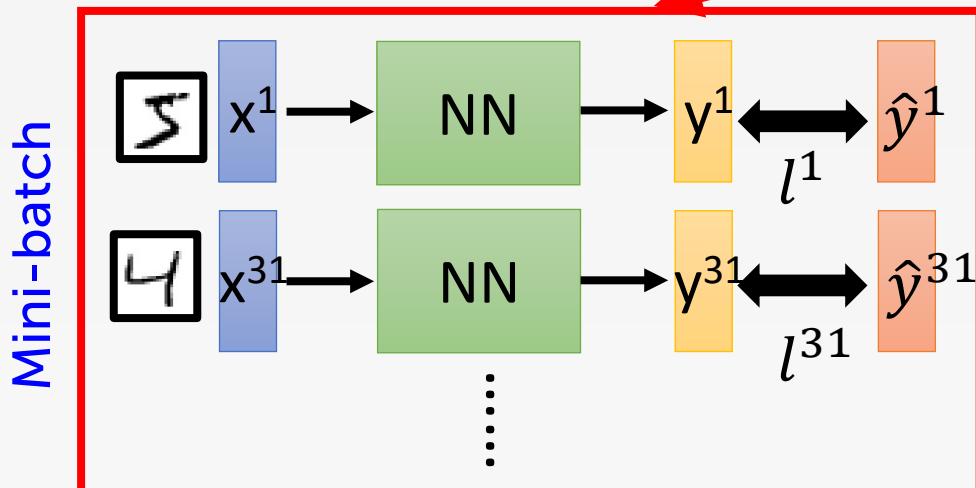


- Randomly initialize network parameters
 - Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
 Update parameters once
 - Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
 Update parameters once
 ...
 - Until all mini-batches have been picked
- One epoch

Repeat the above process

Mini-batch

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

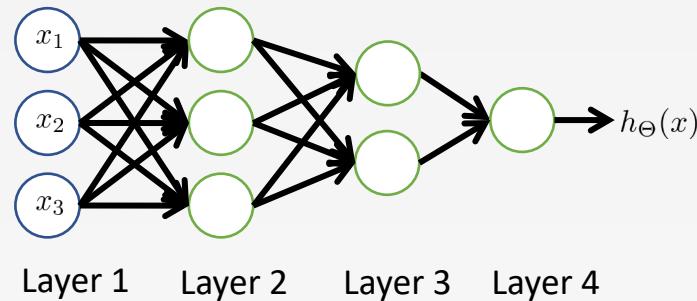


100 examples in a mini-batch

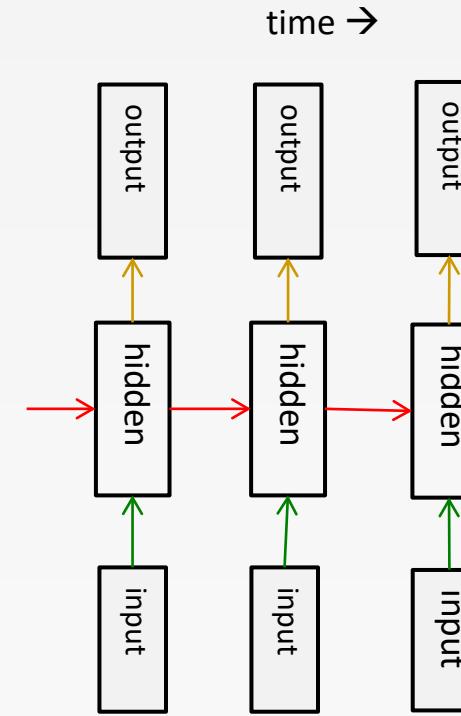
- Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
 Update parameters once
- Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
 Update parameters once
- ⋮
- Until all mini-batches have been picked

Network Architectures

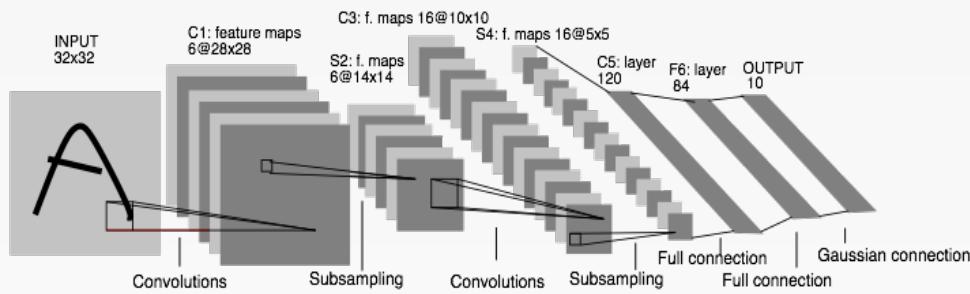
Fully connected



Recurrent

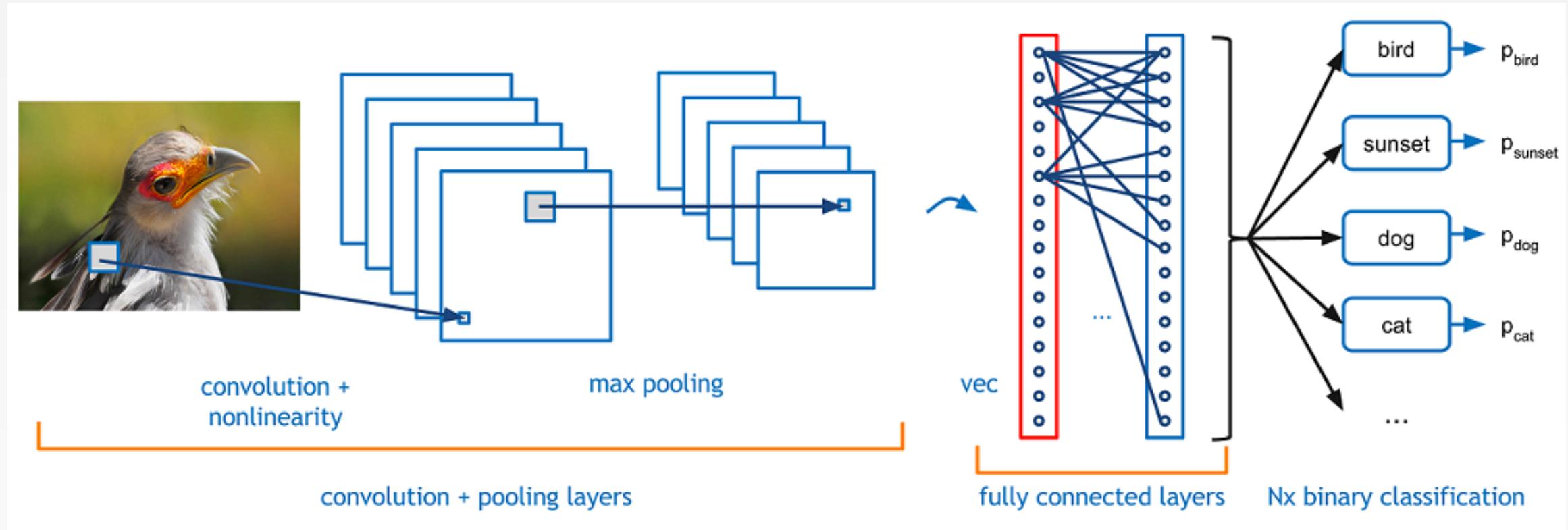


Convolutional



Convolutional Neural Network

CNN Architecture



Key Ideas

- Replace matrix multiplication in neural nets with convolution
- Parameter Sharing
 - Parameter sharing scheme is used in Convolutional Layers to control the number of parameters
- Image processing:
 - Not connected to every single pixel in the input image, but only to pixels in their receptive fields.

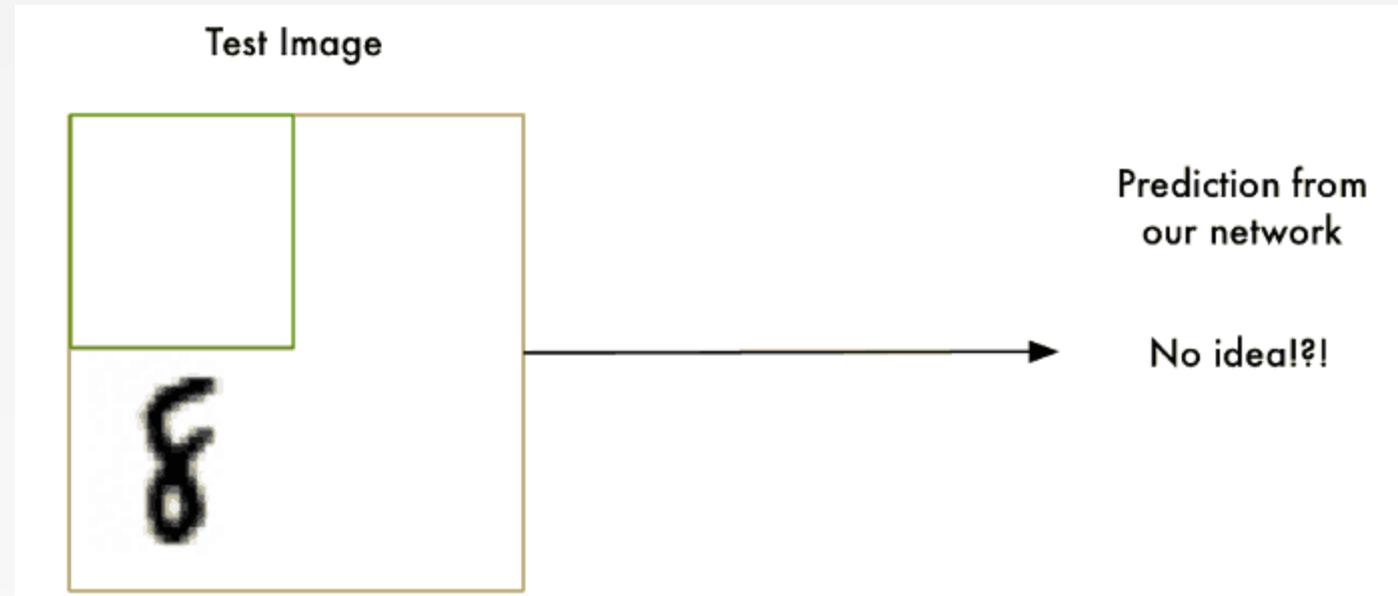


0	2	15	0	0	11	10	0	0	0	9	9	0	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	0	252	255	248	144	6	0
0	13	111	255	255	245	255	182	181	248	252	242	208	30	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	194	9	0	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	2	62	255	250	125	3	0
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	249	255	240	255	125	0	5	0
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

0	2	15	0	0	11	10	0	0	0	9	9	0	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	0	252	255	248	144	6	0
0	13	111	255	255	245	255	182	181	248	252	242	208	30	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	194	9	0	0
0	0	111	255	242	255	158	24	0	0	6	39	255	232	230	56
0	218	251	250	137	7	11	0	0	2	62	255	250	125	3	0
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	249	255	240	255	125	0	5	0
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

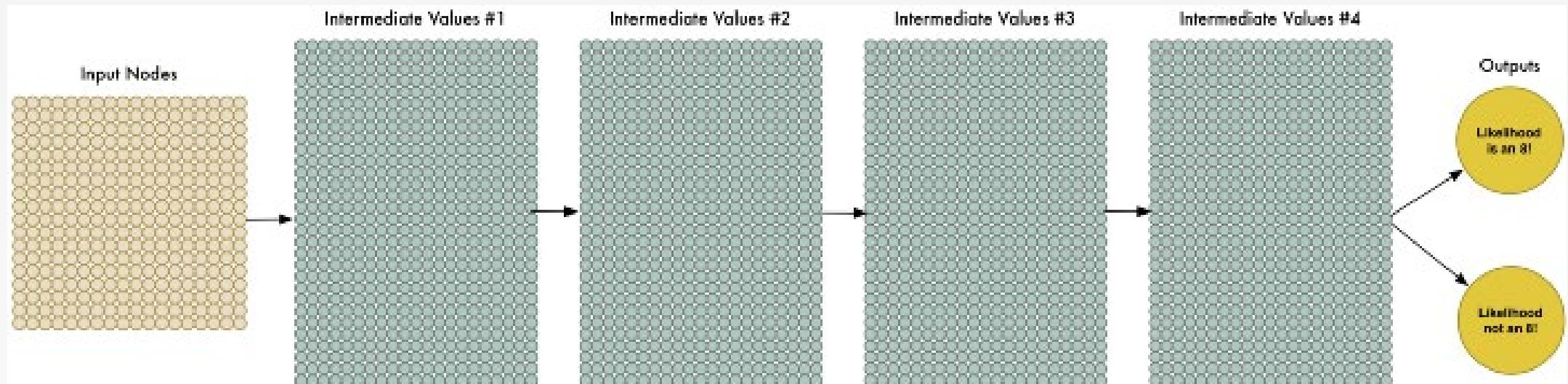
Preprocessing of pixel values

Search with a Sliding Window

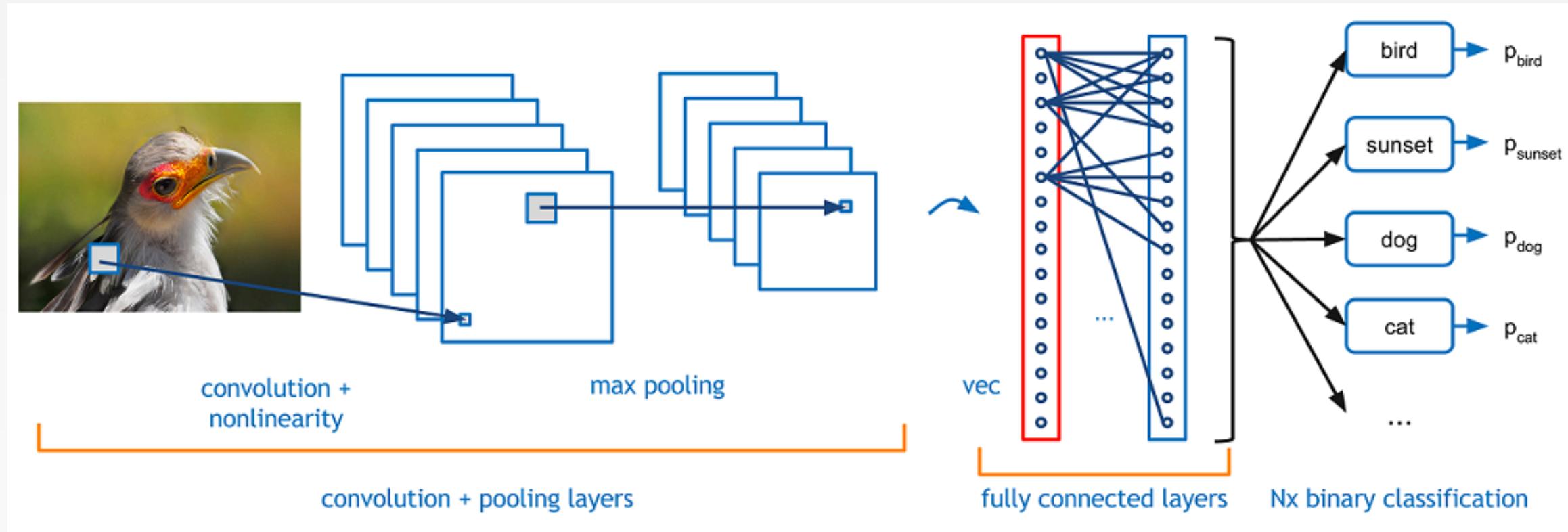




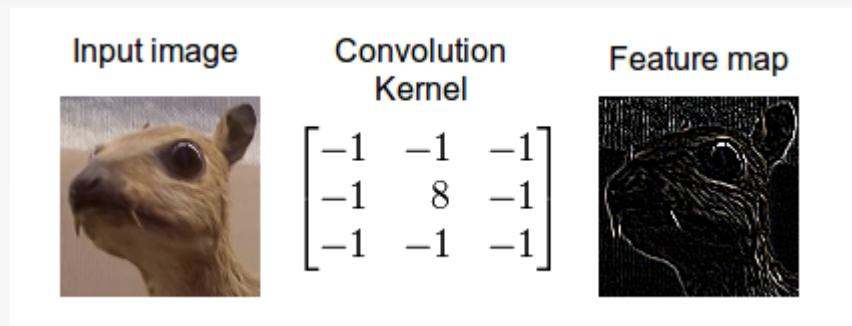
A Deep Neural Net?

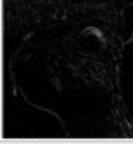


CNN Architecture



Convolution Layer



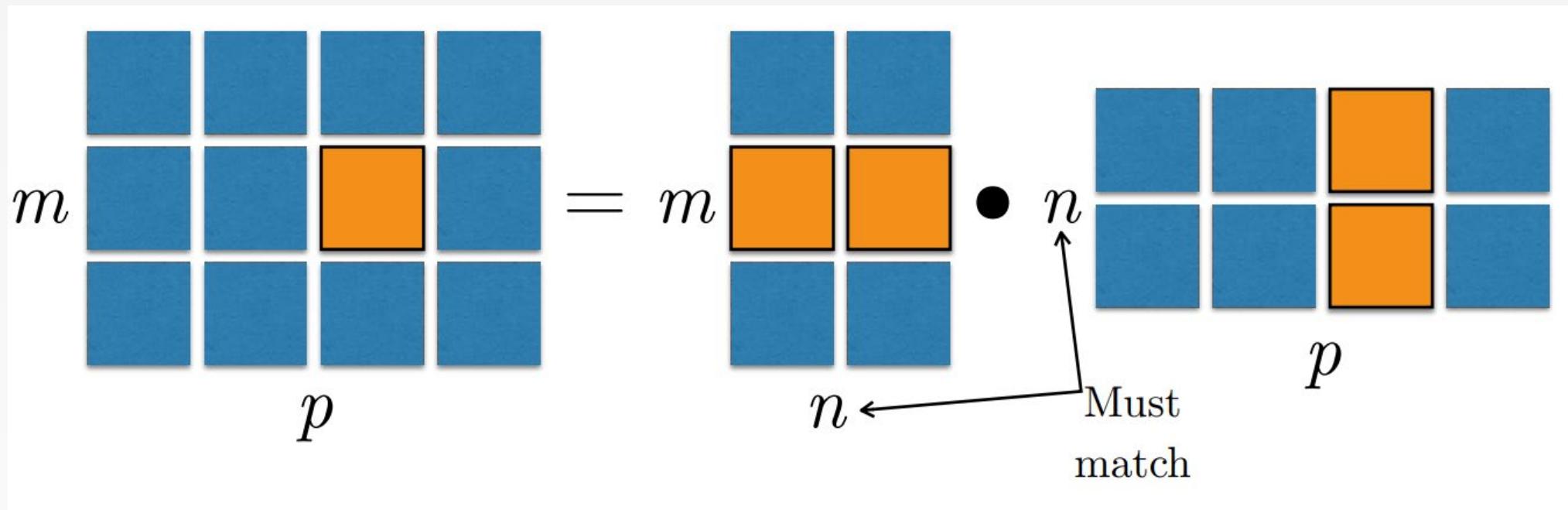
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

[CNN Explainer](#)

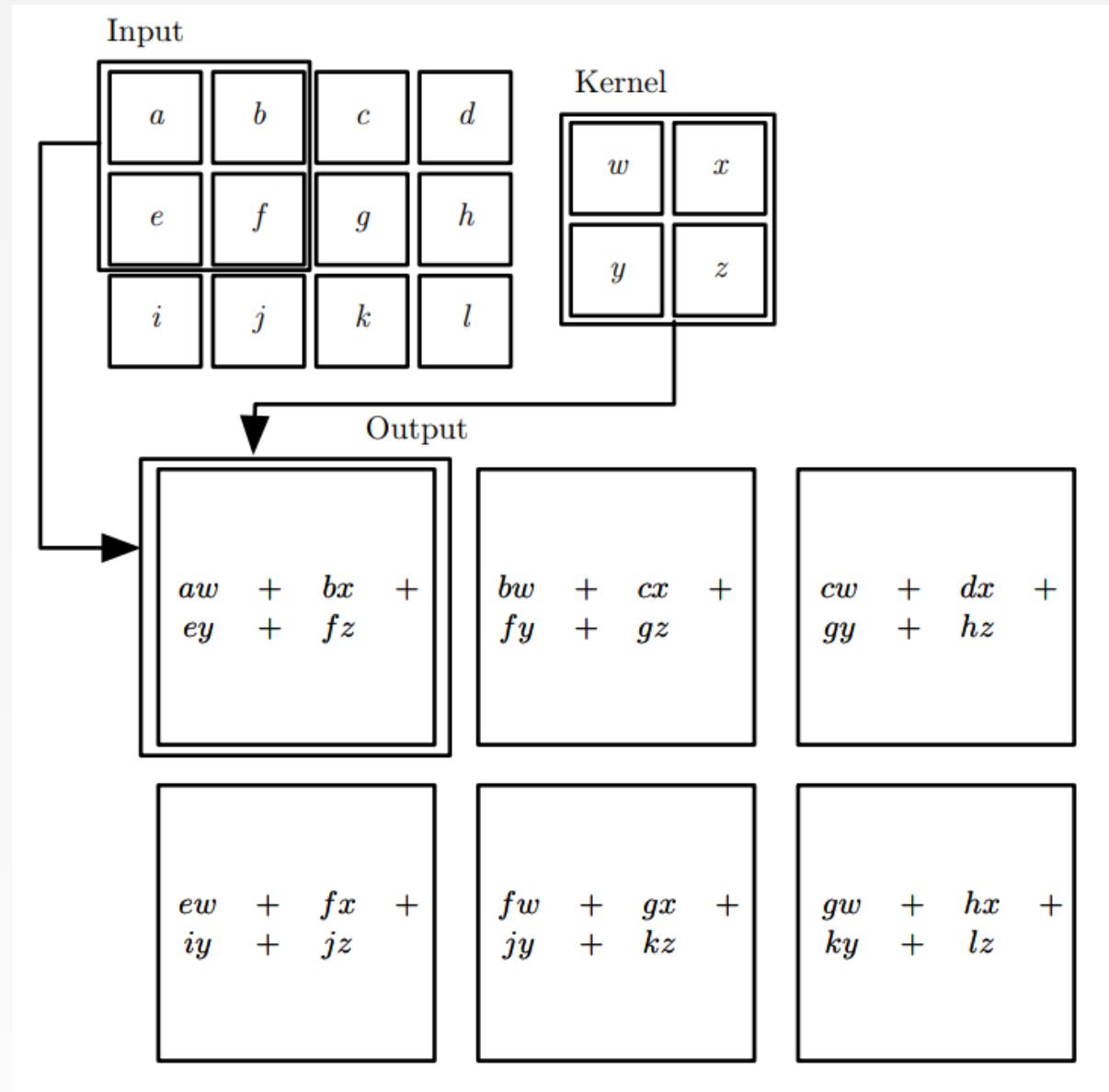
Matrix Product

$$C = AB$$

$$C_{ij} = \sum_k A_{i,k}B_{k,j}$$



2D Convolution



Convolutional Layer

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1
Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2
Matrix

: :

Those are the network parameters to be learned.

Convolutional Layer

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Convolutional Layer

stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

We set stride=1 below

Convolutional Layer

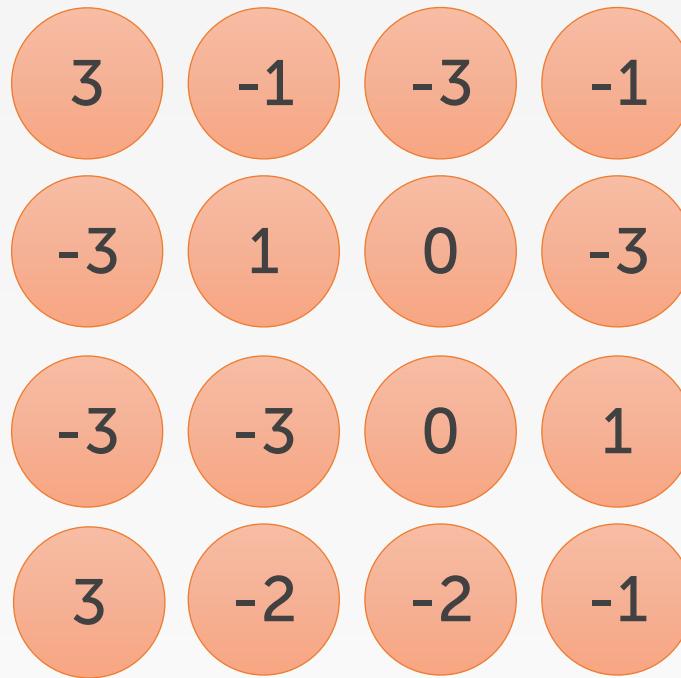
stride=1

1	0	0	0	0	0	1
0	1	0	0	1	0	
0	0	1	1	0	0	
1	0	0	0	1	0	
0	1	0	0	1	0	
0	0	1	0	1	0	

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1



Convolutional Layer

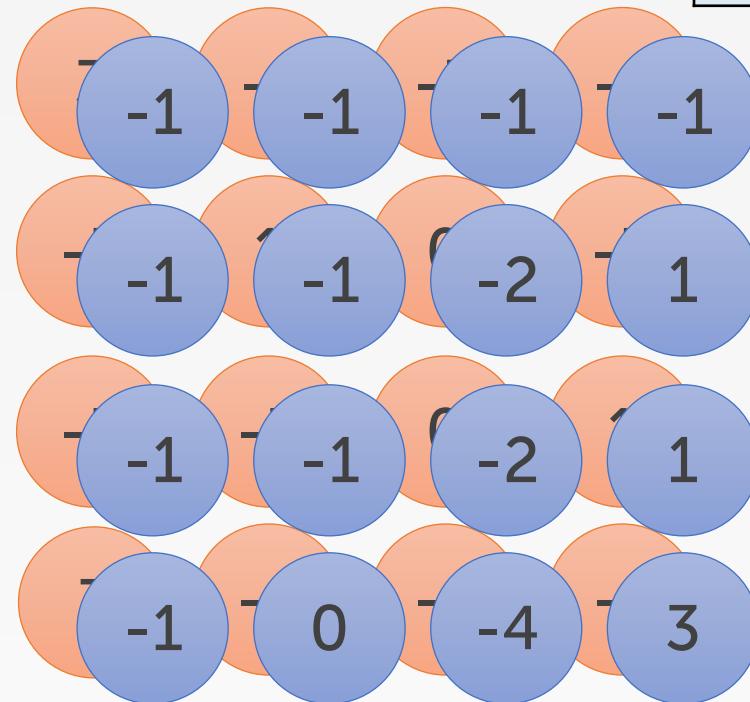
stride=1

1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	0	1	1	0	0	0
1	0	0	0	0	1	0
0	1	0	0	0	1	0
0	0	1	0	0	1	0

6 x 6 image

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



Padding

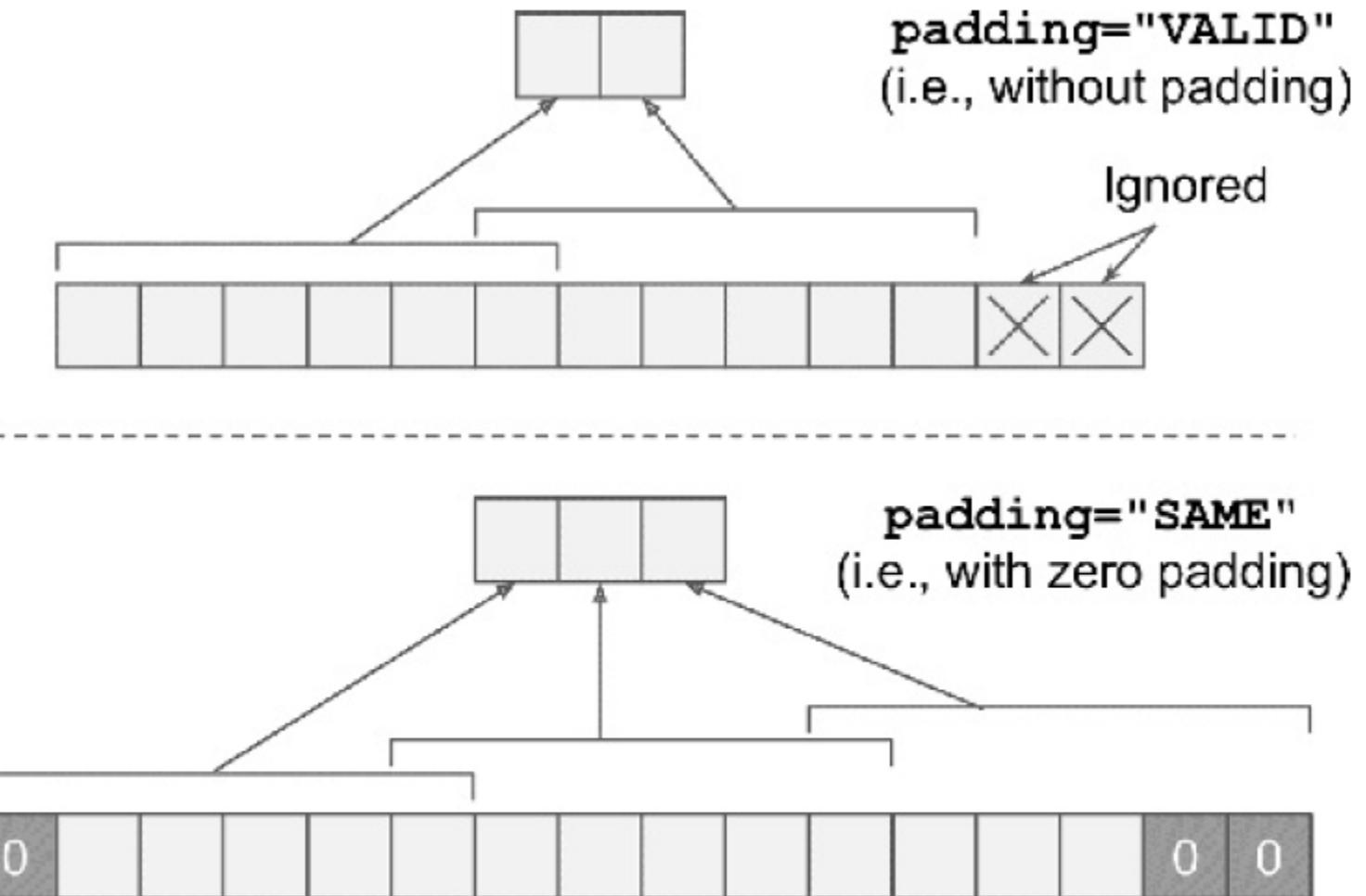
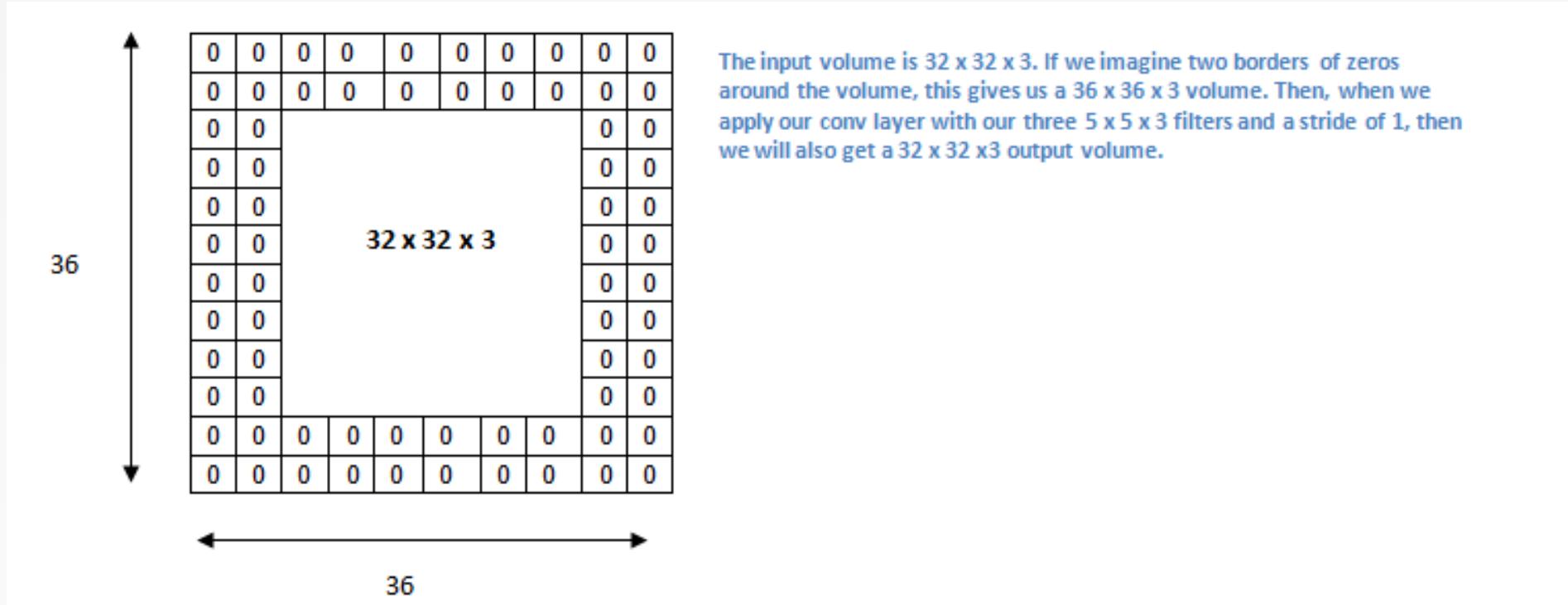


Figure 13-7. Padding options — input width: 13, filter width: 6, stride: 5

https://keras.io/api/layers/convolution_layers/convolution2d/

Padding

- If you have a stride of 1 and if you set the size of zero padding to $\frac{K-1}{2}$, where K is the filter size, then the input and output volume will always have the same spatial dimensions.



Padding

$$O = \frac{W-K+2P}{S} + 1$$

O : output height/length

W : input height/length

K : filter size

P : padding

S : stride

Real-world Example

- Input image: $227 \times 227 \times 3$
- Filter size: $11 \times 11 \times 3$; Stride = 4; no zero padding
- Number of filters = 96
- Output size: $55 \times 55 \times 96$

Pooling Layer

- It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture.
- Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.
 - **max-pooling**: chose the max in a window
 - **mean-pooling**: take the average

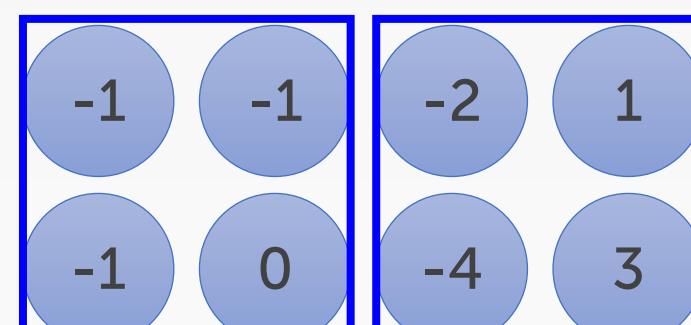
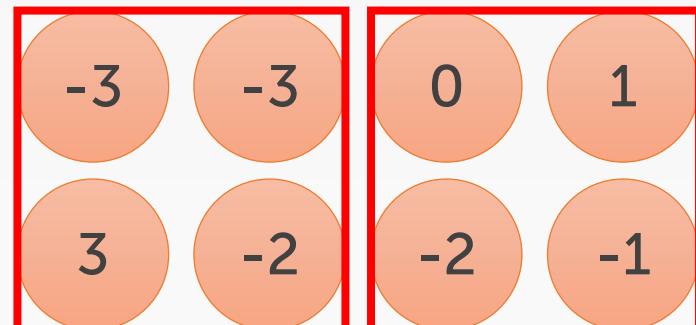
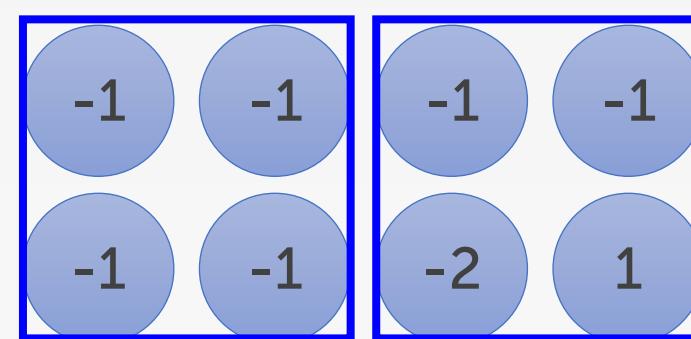
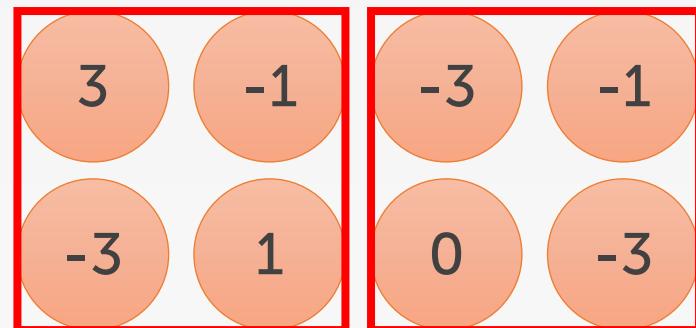
Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

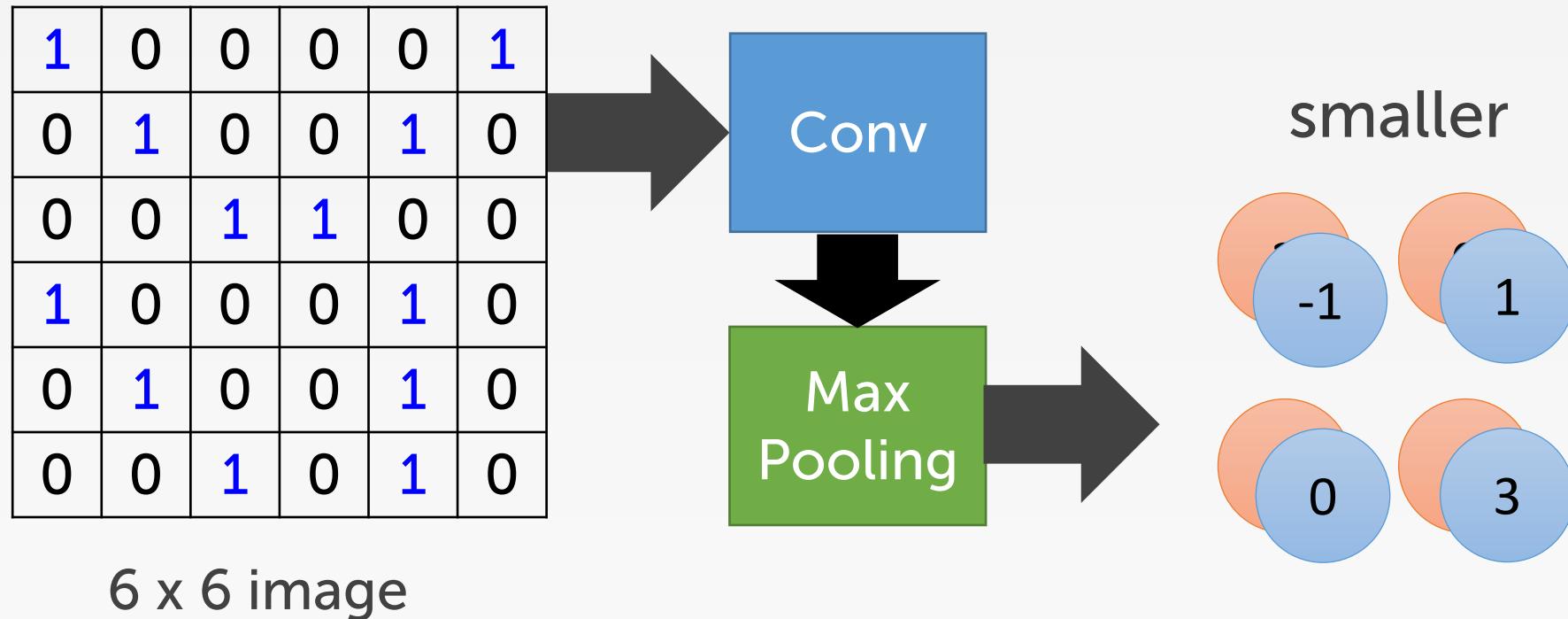
Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

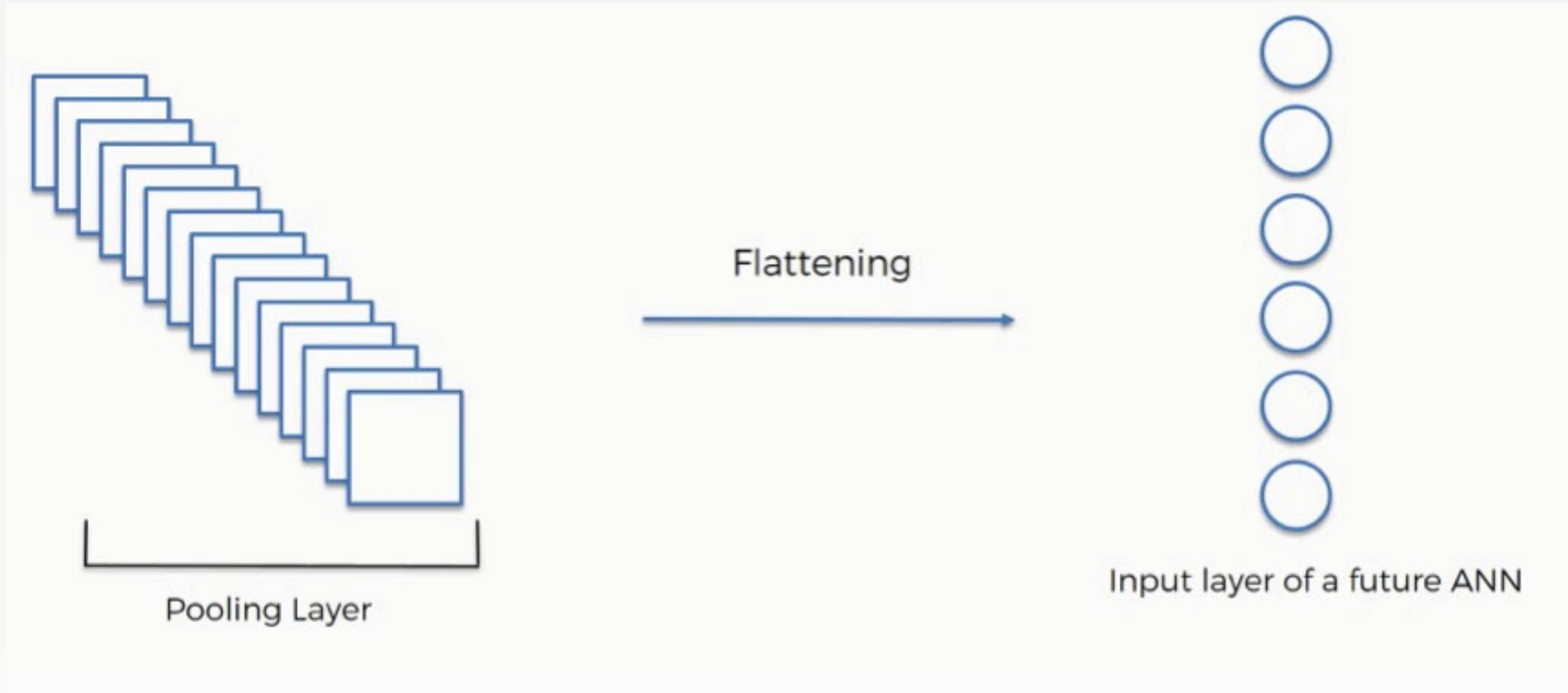
Filter 2



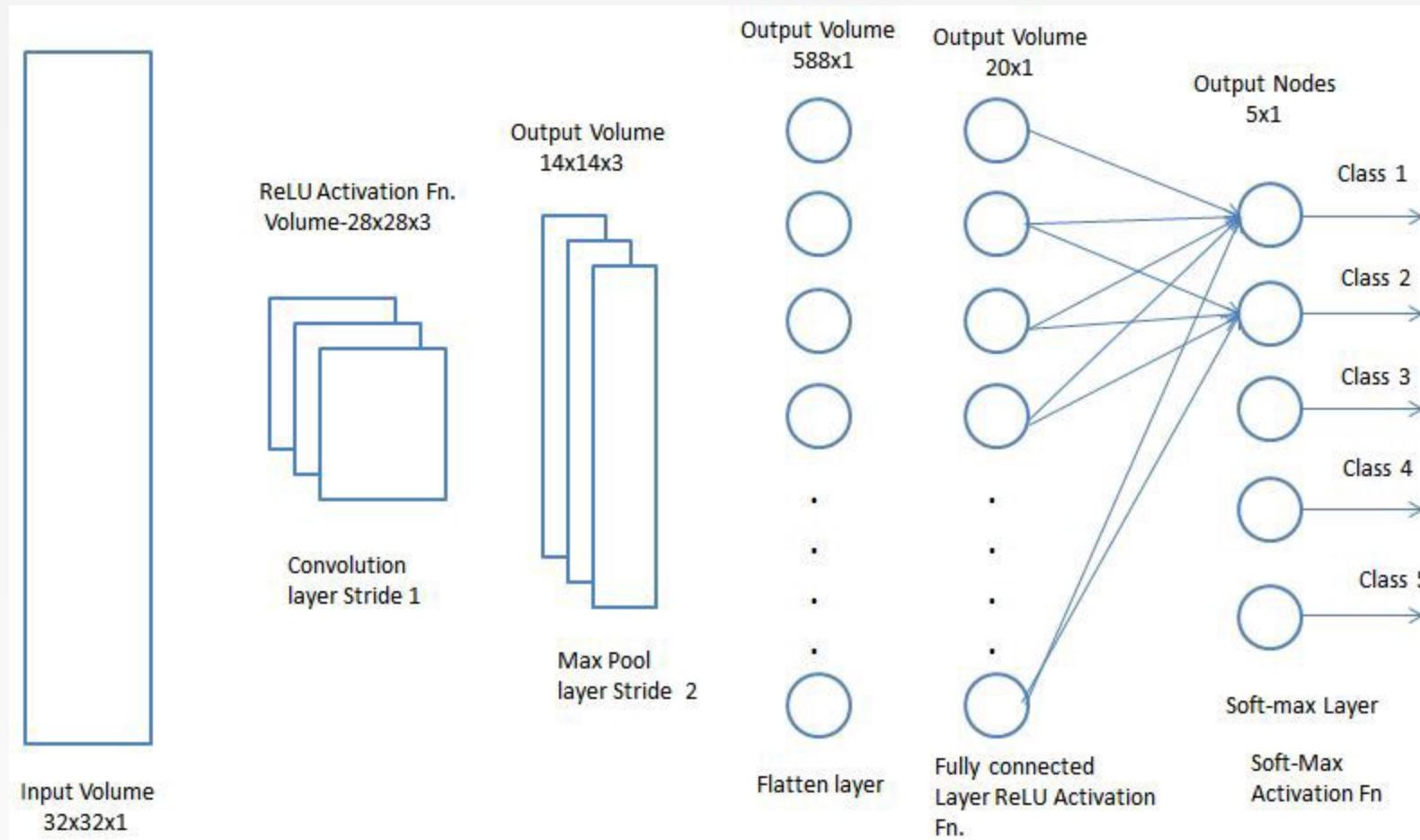
Max Pooling



Fully Connected Layer



Fully Connected Layer



Popular CNN Architectures

- LeNet
- AlexNet
- GoogLeNet
- ResNet

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

LeNet-5 Architecture

LeNet-5

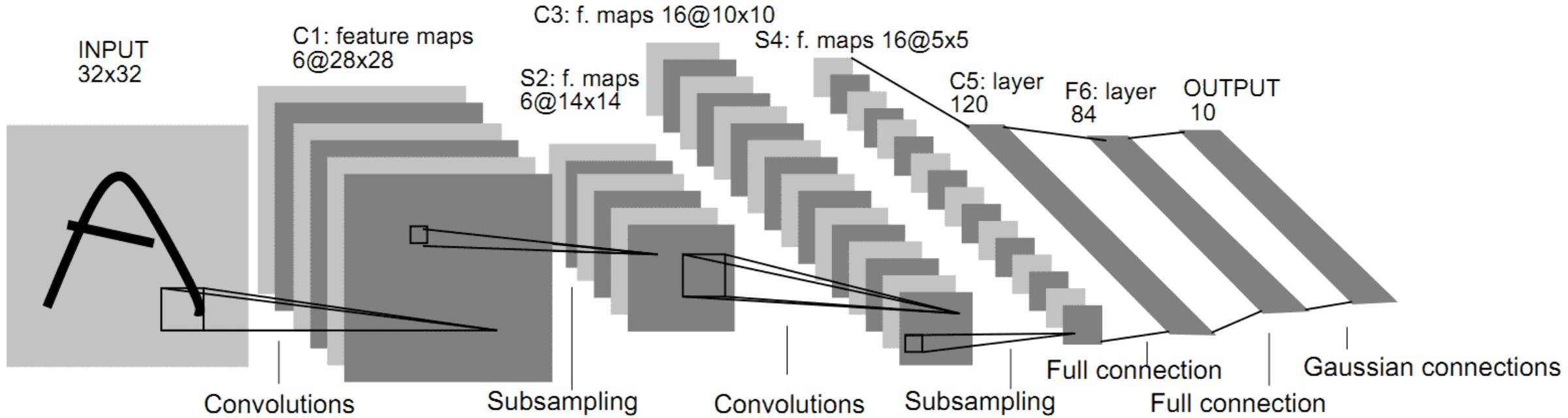


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Implementation

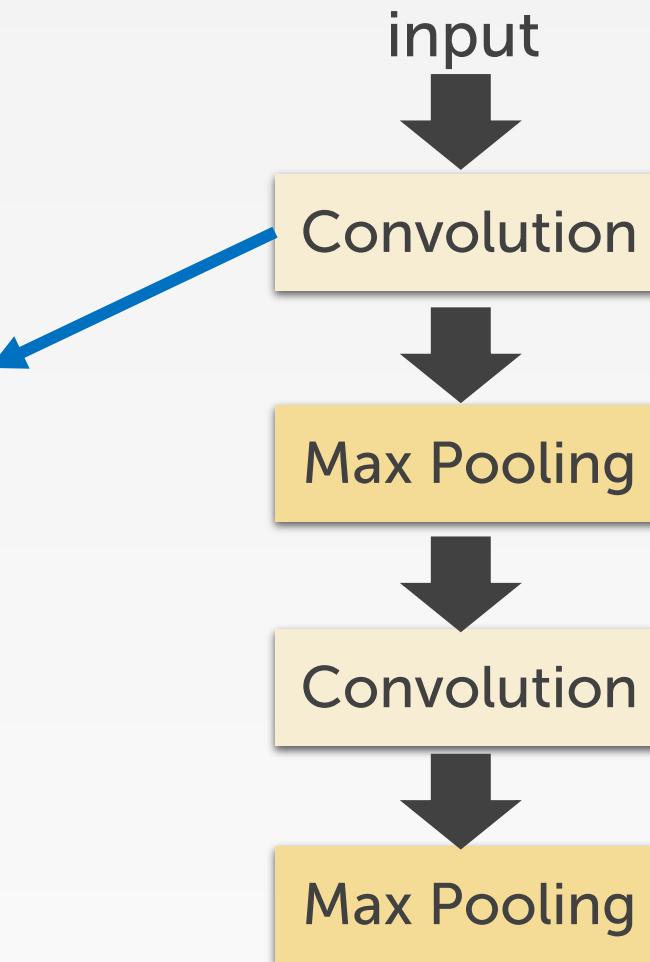
```
keras.Input(shape=(28,28,1)),  
layers.Conv2D(25, kernel_size=(3, 3), activation="relu")
```

Input_shape = (28, 28, 1)
 28 x 28 pixels 1: black/white, 3: RGB

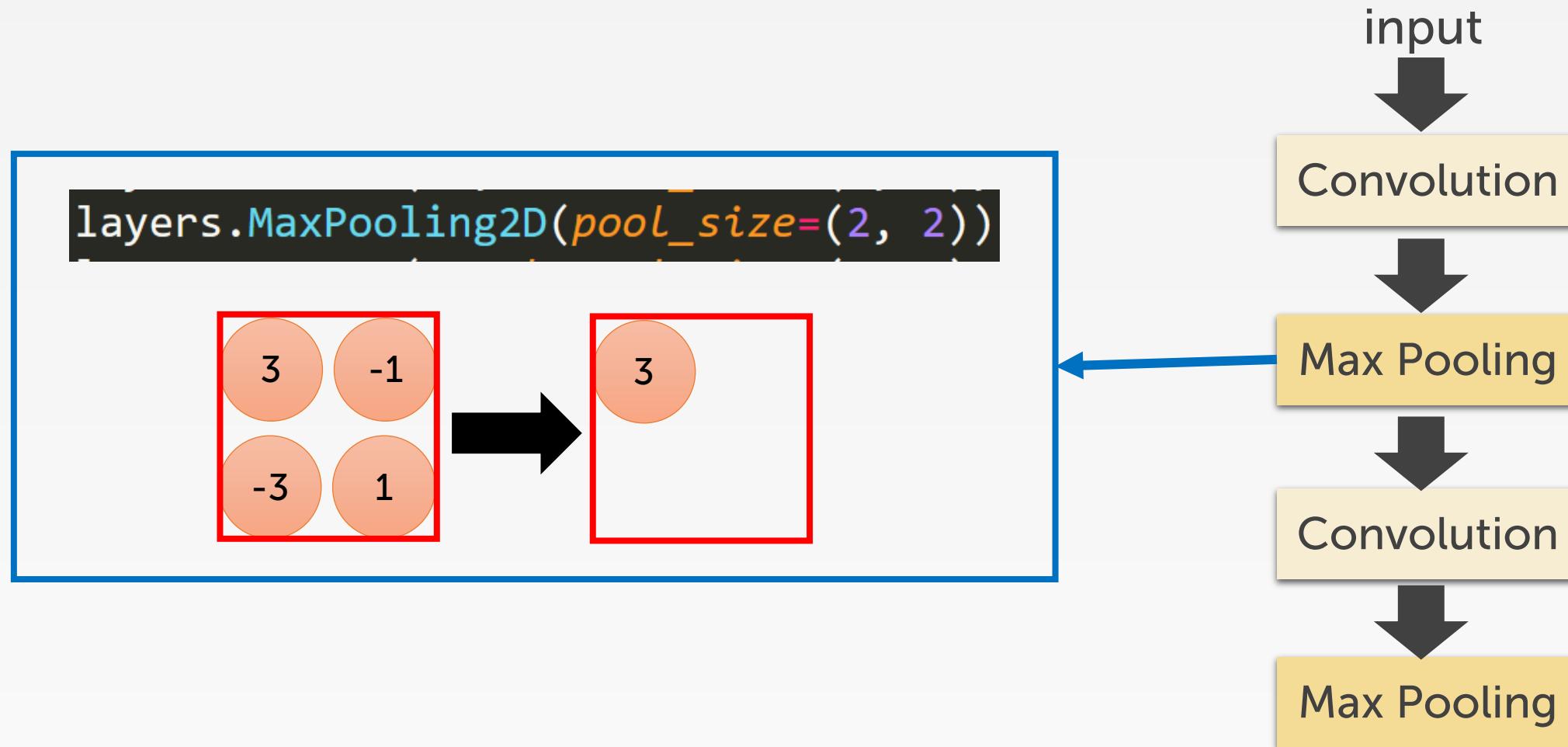
1	-1	1			
-1	1		-1	1	-1
-1		-1	1	-1	
	-1	1	1	-1	

... ...

There are 25
3x3 filters.



Implementation



Implementation



Implementation

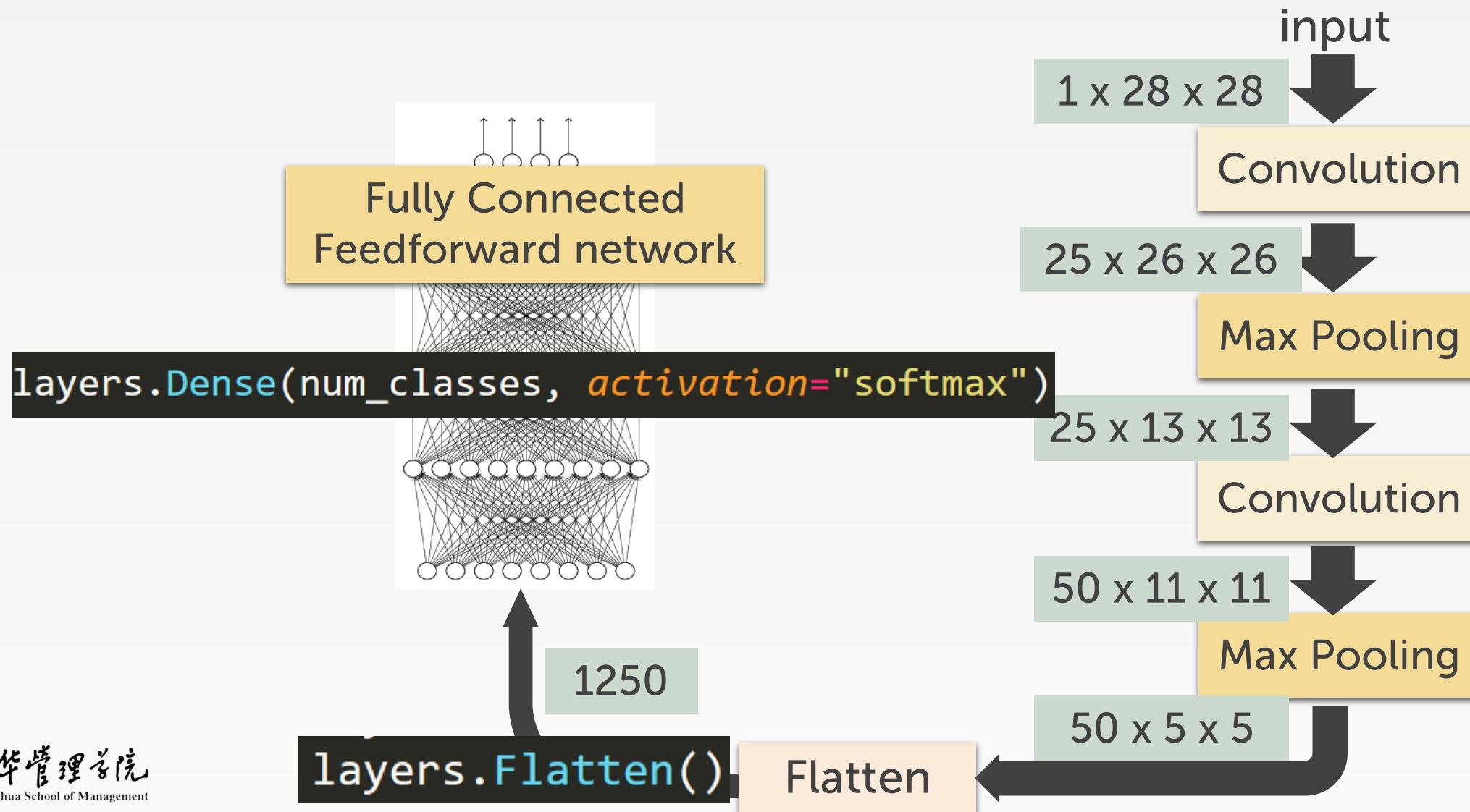


Figure 1. Comparison of Unverified and Verified Photos

Unverified



Verified



Zhang, S., Lee, D., Singh, P.V. and Srinivasan, K.. How much is an image worth? Airbnb property demand estimation leveraging large scale image analytics. *Management Science* (2021)

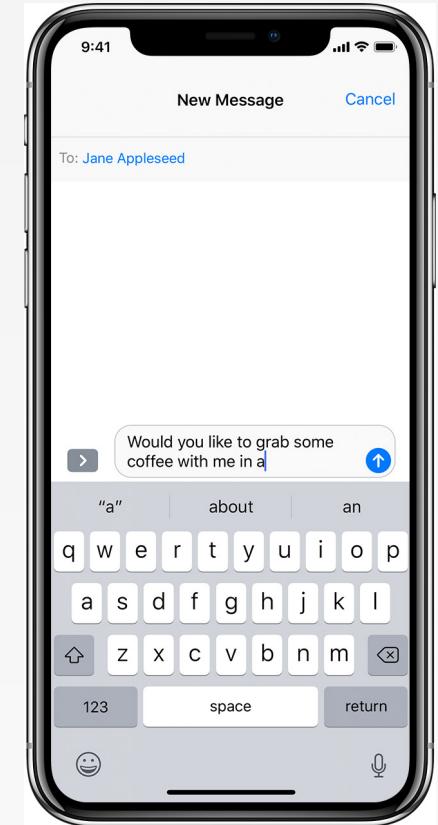
12 Human-Interpretable Image Attributes				
Composition	1	<i>DIAGONAL_DOMINANCE</i>	2.516**	0.945
	2	<i>VISUAL_BALANCE_INTENSITY</i>	4.618***	1.350
	3	<i>VISUAL_BALANCE_COLOR</i>	8.869***	2.143
	4	<i>RULE_OF_THIRDS</i>	3.537**	1.106
Color	5	<i>WARM_HUE</i>	4.715*	2.363
	6	<i>SATURATION</i>	3.920*	1.927
	7	<i>BRIGHTNESS</i>	3.434*	1.679
	8	<i>CONTRAST_OF_BRIGHTNESS</i>	-4.897*	2.411
Figure-Ground Relationship	9	<i>IMAGE_CLARITY</i>	6.212**	2.175
	10	<i>SIZE_DIFFERENCE</i>	3.807*	1.541
	11	<i>COLOR_DIFFERENCE</i>	2.728*	1.372
	12	<i>TEXTURE_DIFFERENCE</i>	2.313*	1.090
Fixed Effect			Property	
Seasonality			City-Year-Month	
Num. Observations			76,901	
R-squared			0.6670	
Note: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.				



Recurrent Neural Network

Sequence of Data

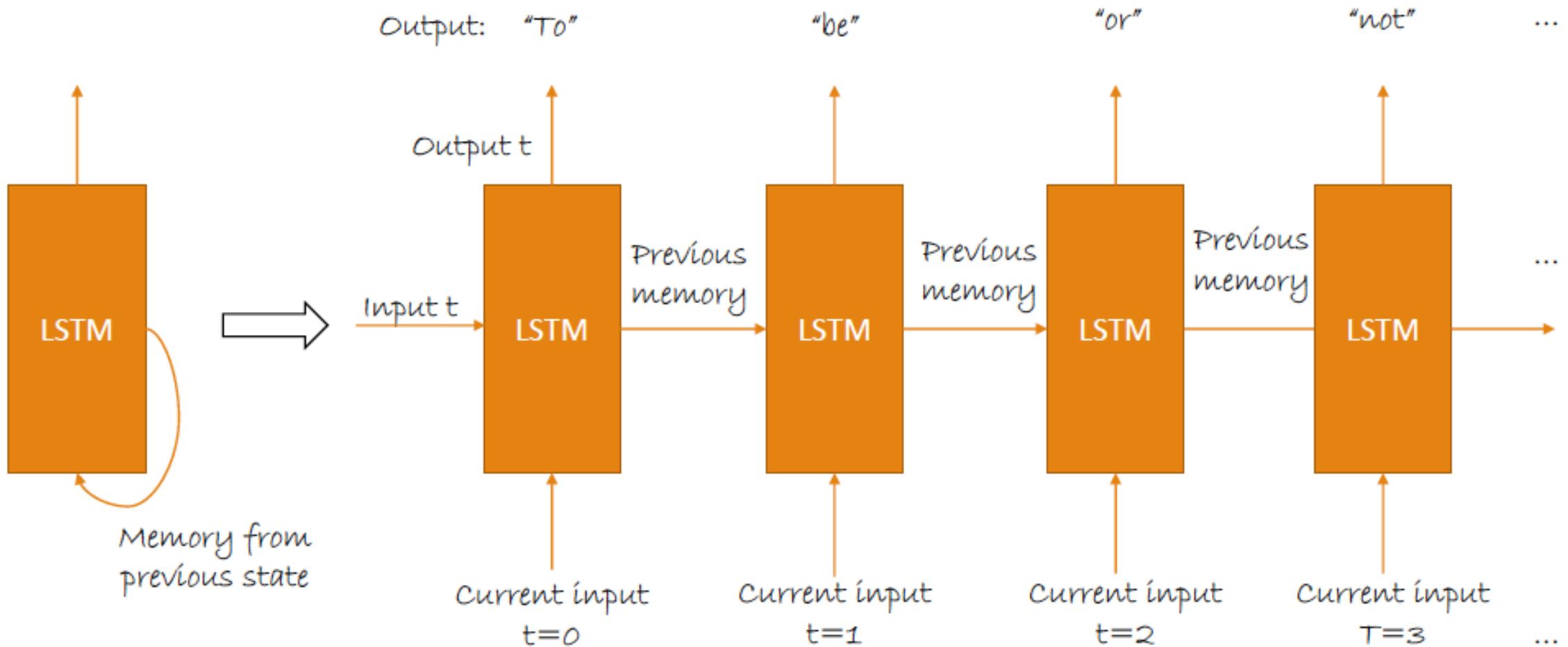
- Sequences in our world:
 - Audio
 - Text
 - Video
 - Weather
 - Stock market
- Sequential data is why we build RNN architectures
- RNNs are tools for making predictions about sequences



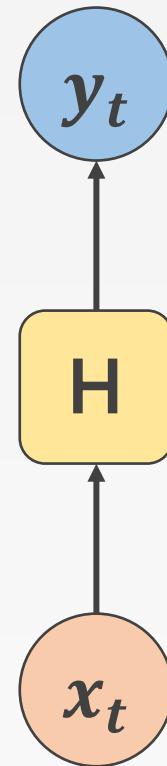
Limitations of Previous Networks

- Fixed Length:
 - Inputs and outputs are of fixed lengths
- Independence
 - Data (e.g., images) are independent of one another

Nutshell: RNN



Neural Network



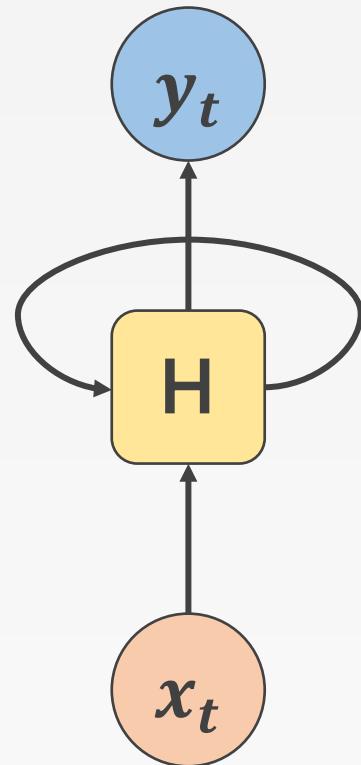
x_t : Input

y_t : Output/Prediction

H: chunk of NN

Every input is treated independently

RNN

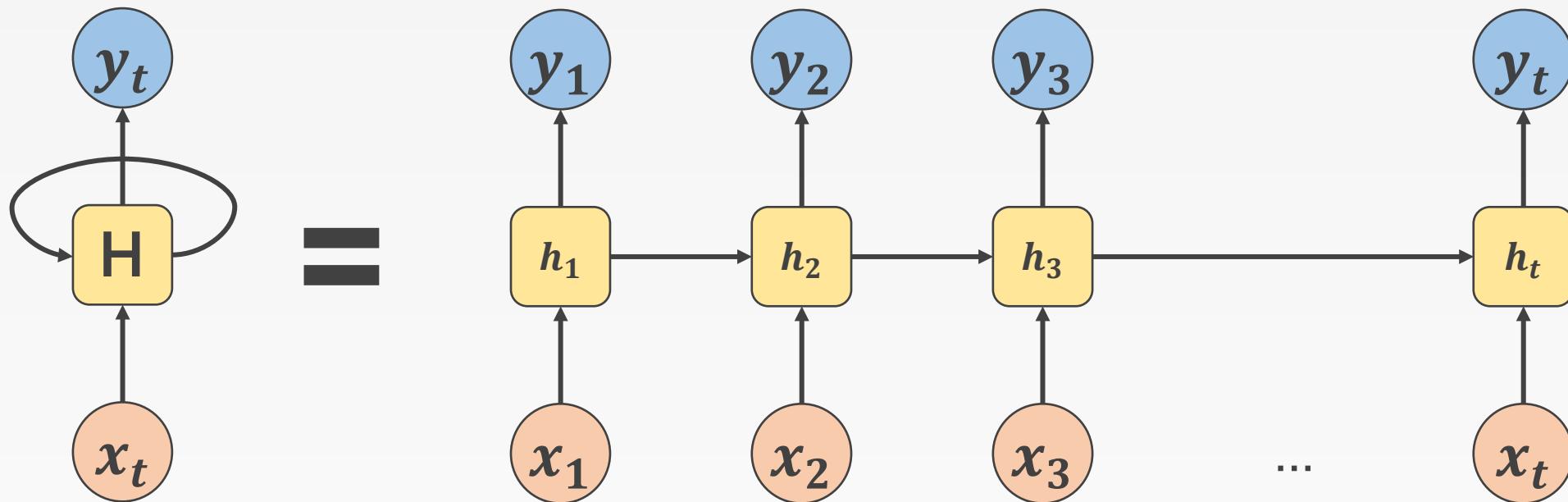


The loop allows information to be passed from one time step to the next.

Now we are modeling the dynamics

RNN

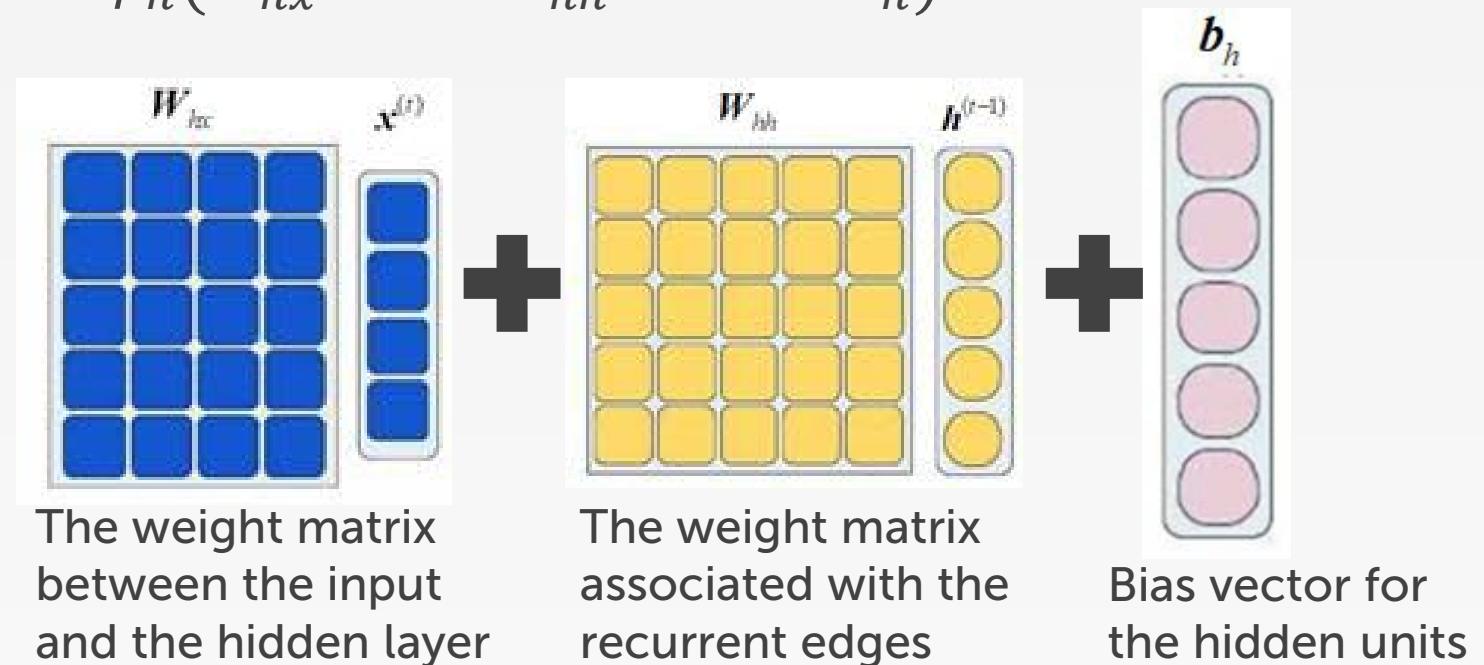
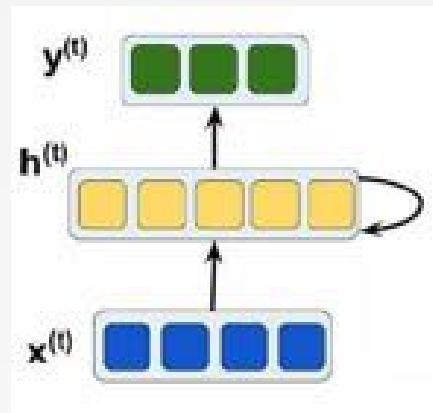
- A recurrent neural network can be thought of as multiple copies of the network, each passing a message to a successor.



	x - input	y - output
Speech recognition		"Fuzzy Wuzzy was a bear. Fuzzy Wuzzy had no hair."
Music generation	\emptyset	
Sentiment classification	"Decent effort. The plot could have been better."	
DNA sequence analysis	ACTGTACCCATGTGACTGCC	ACT TACCCATGTGACTGC CC
Machine translation	"El que no arriesga, no gana."	"If you don't take risks, you cannot win."
Video activity recognition		Running
Name entity recognition	"Ygritte says Jon Snow knows nothing."	" Ygritte says Jon Snow knows nothing."

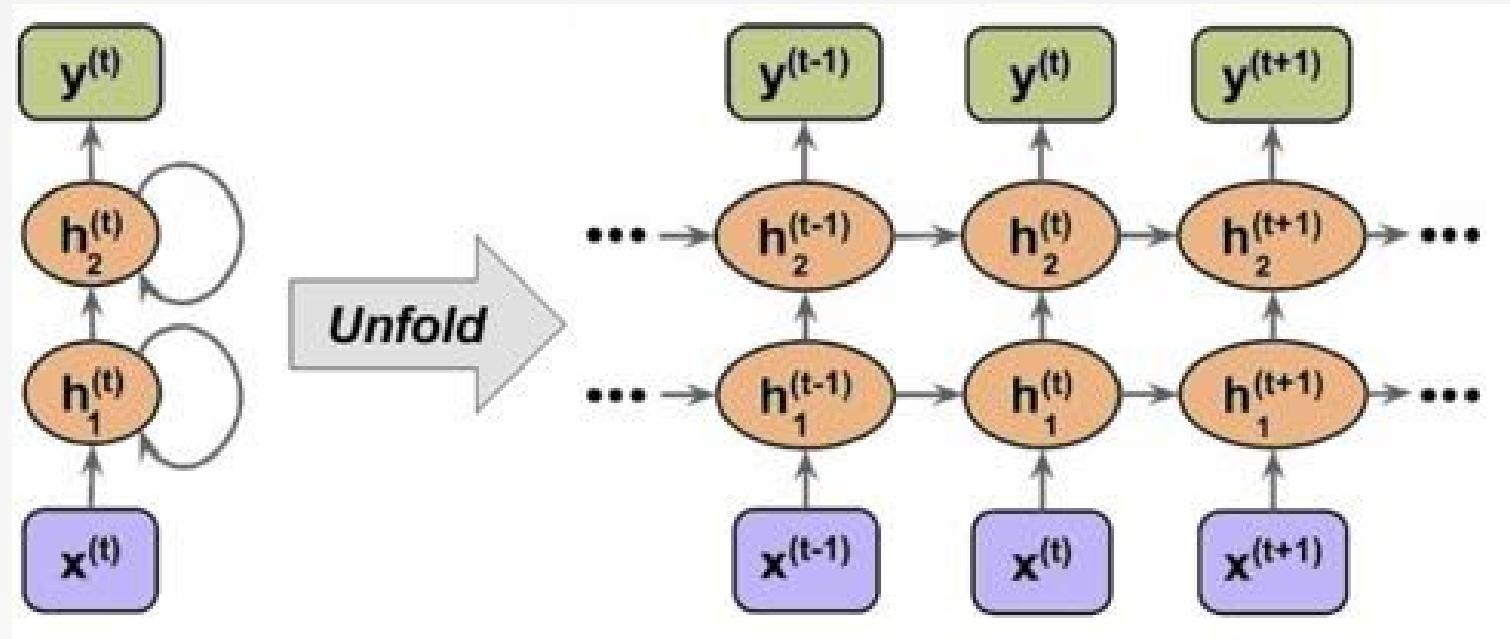
RNN Structure

- New hidden state: $h^{(t)} = \phi_h(W_{hx}x^{(t)} + W_{hh}h^{(t-1)} + b_h)$



- Output: $y^{(t)} = \phi_y(W_{yh}h^{(t)} + b_y)$

Multilayer RNN



Deep RNNs

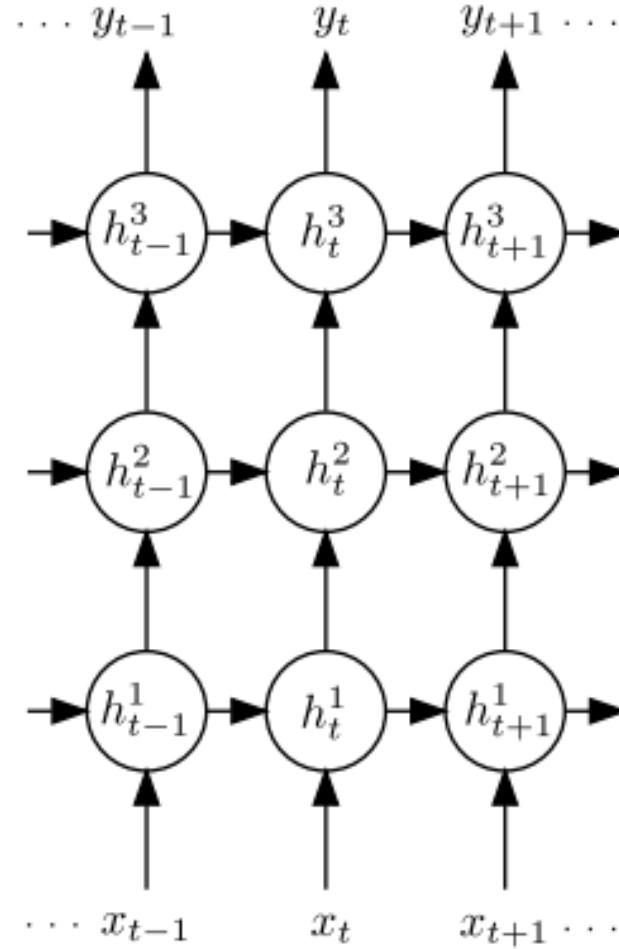


Figure from (Graves et al., 2013)

Long Short-Time Memory

- Motivation:
 - Standard RNNs have trouble learning long distance dependencies

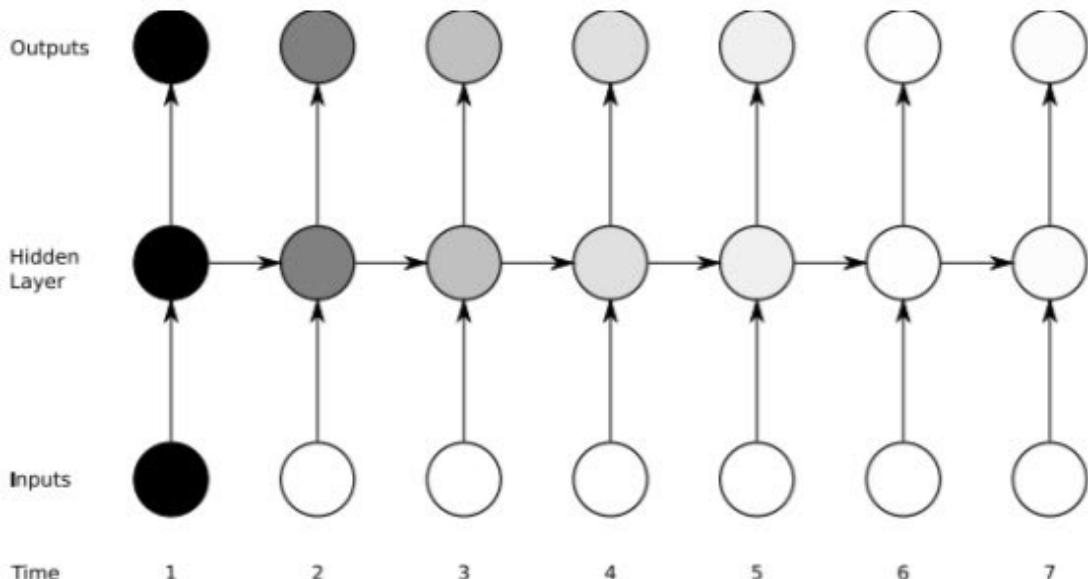
EX #1: The cat, which already ate a bunch of food, was full.

EX #2: The cats, which already ate a bunch of food, were full.

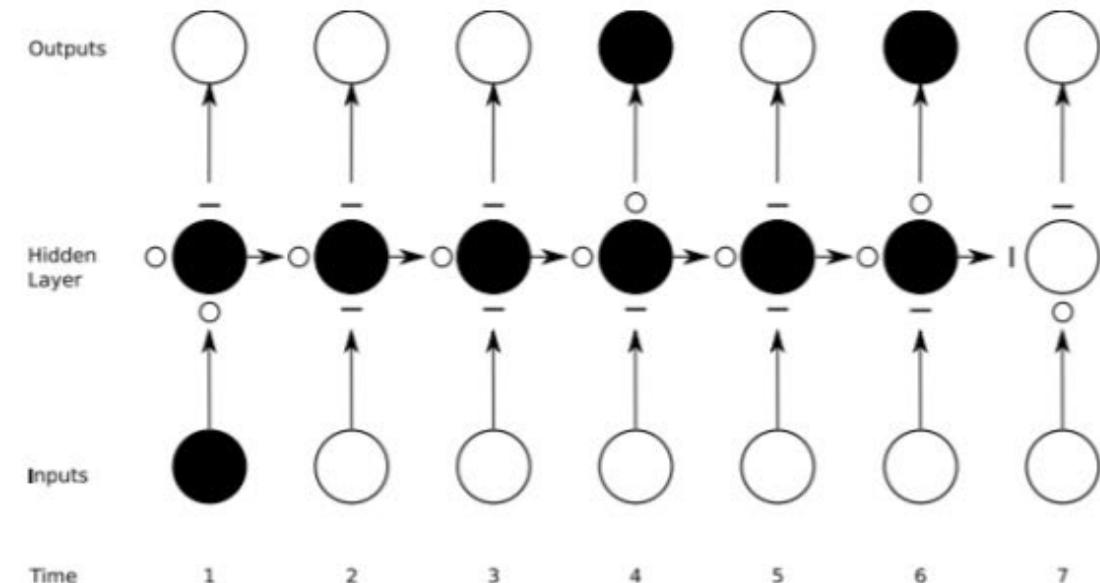
- Motivation:
 - Vanishing gradient problem for standard RNNs

LSTMs reduce vanishing gradient problem

Standard Recurrent Network



LSTM Network



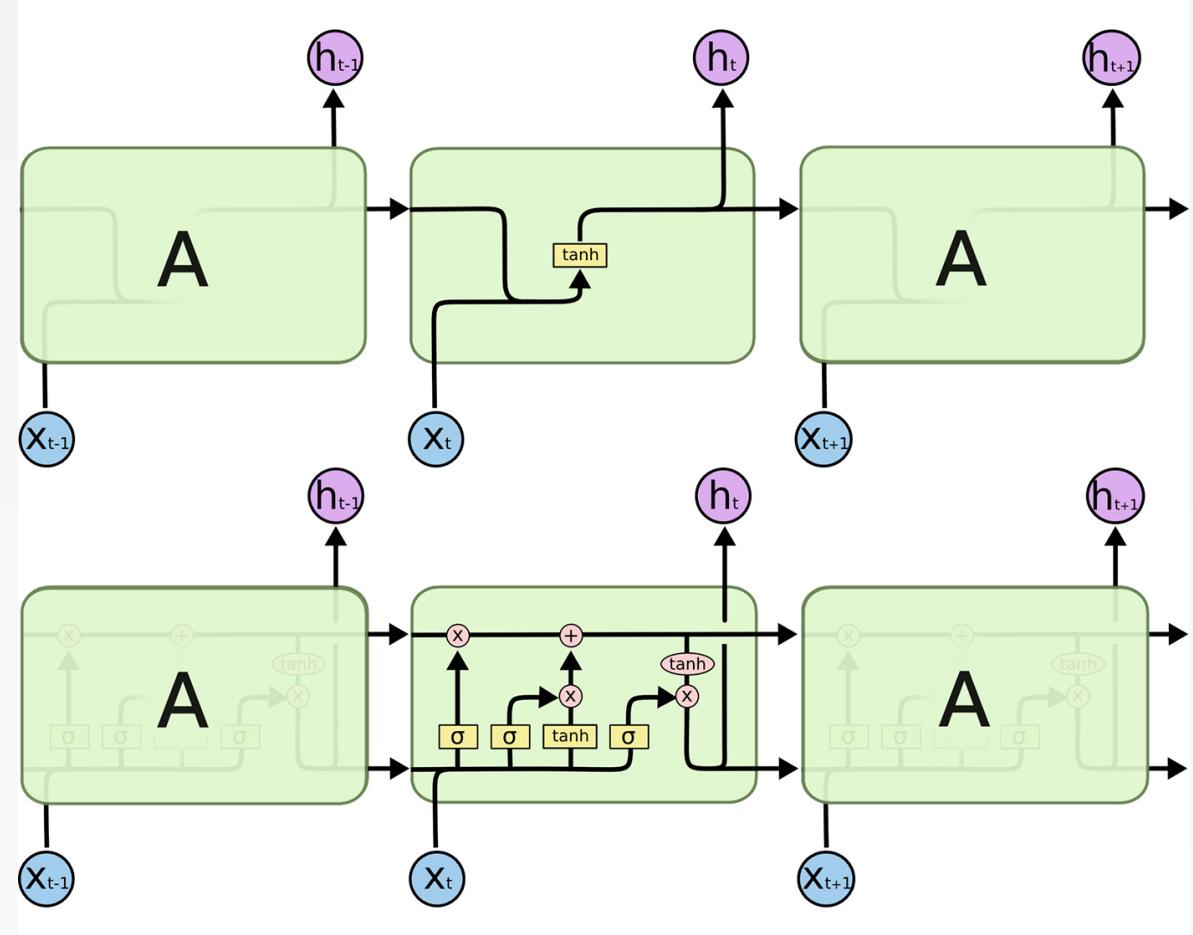
Graves et al 2013

- The darker the shade, the greater the sensitivity

The various “gates” determine the propagation of information and can choose to “remember” or “forget” information

LSTM (Long Short Term Memory)

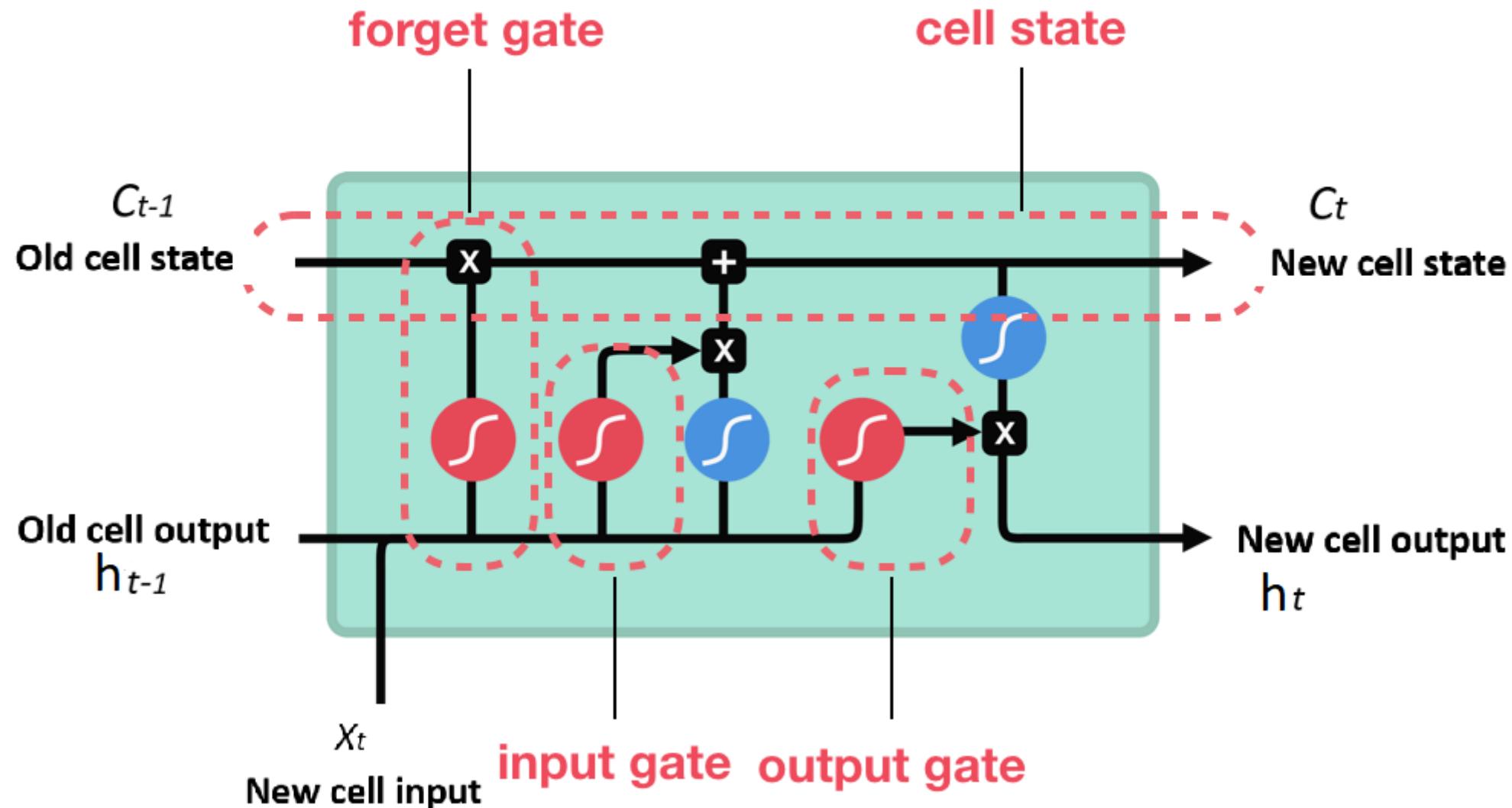
- LSTM is a RNN architecture
- Suggested in 1997 by Hochreiter and Schmidhuber as a solution to the vanishing gradient problem
- An LSTM cell stores a value (state) for either long or short time periods



Gate

- Sigmoid: outputs numbers between:
 - Zero “let nothing through” and
 - One “let everything through!”

LSTM



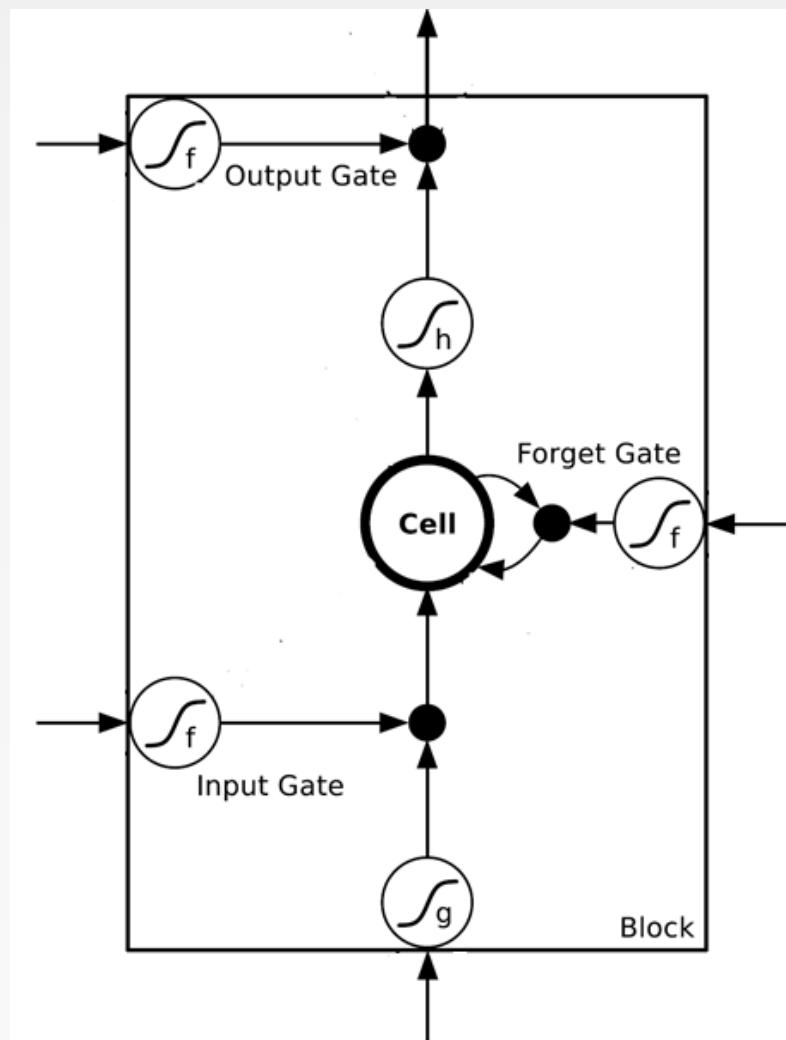
LSTM - Example

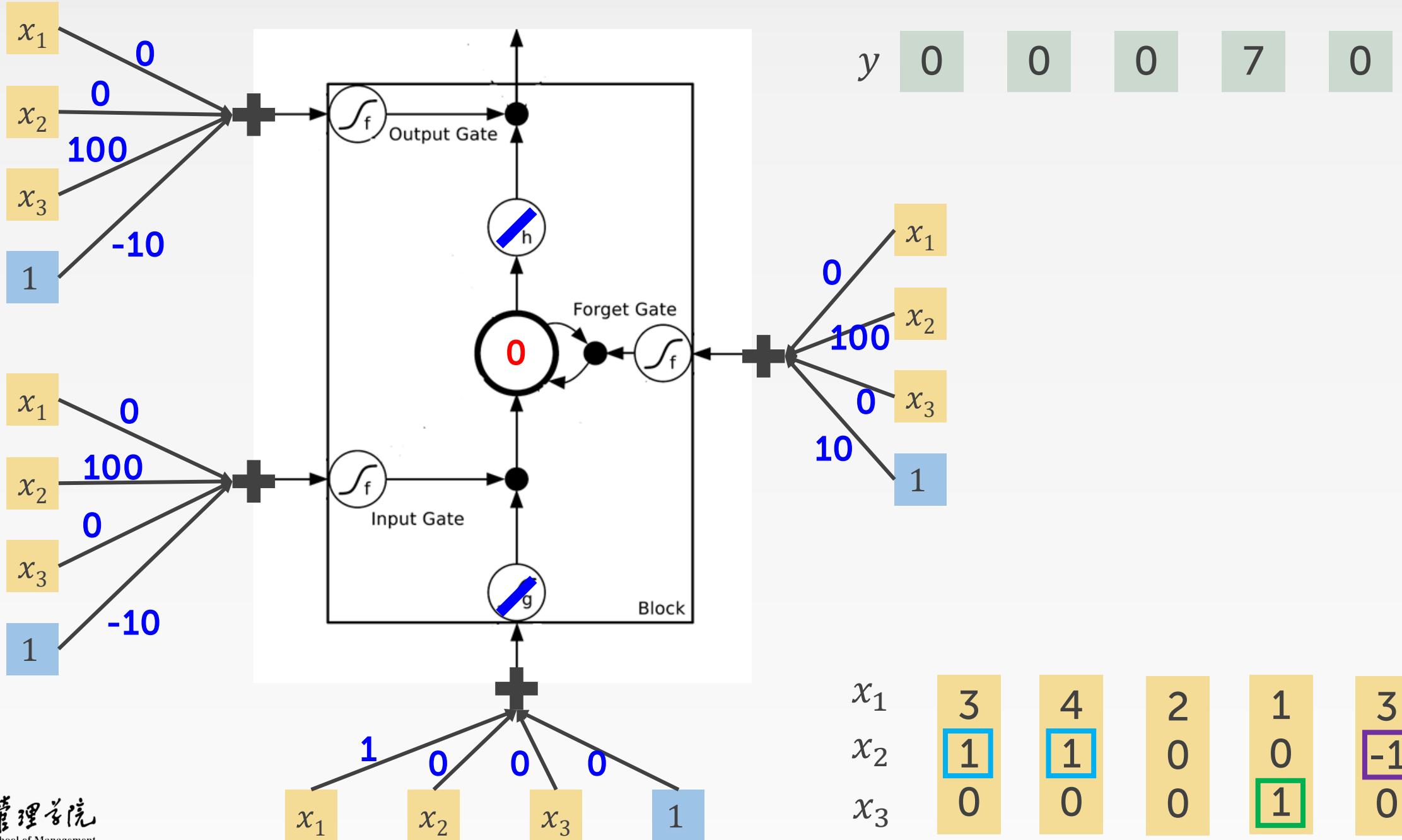
	0	0	3	3	7	7	7	0	6
x_1	1	3	2	4	2	1	3	6	1
x_2	0	1	0	1	0	0	-1	1	0
x_3	0	0	0	0	0	1	0	0	1
y	0	0	0	0	0	7	0	0	6

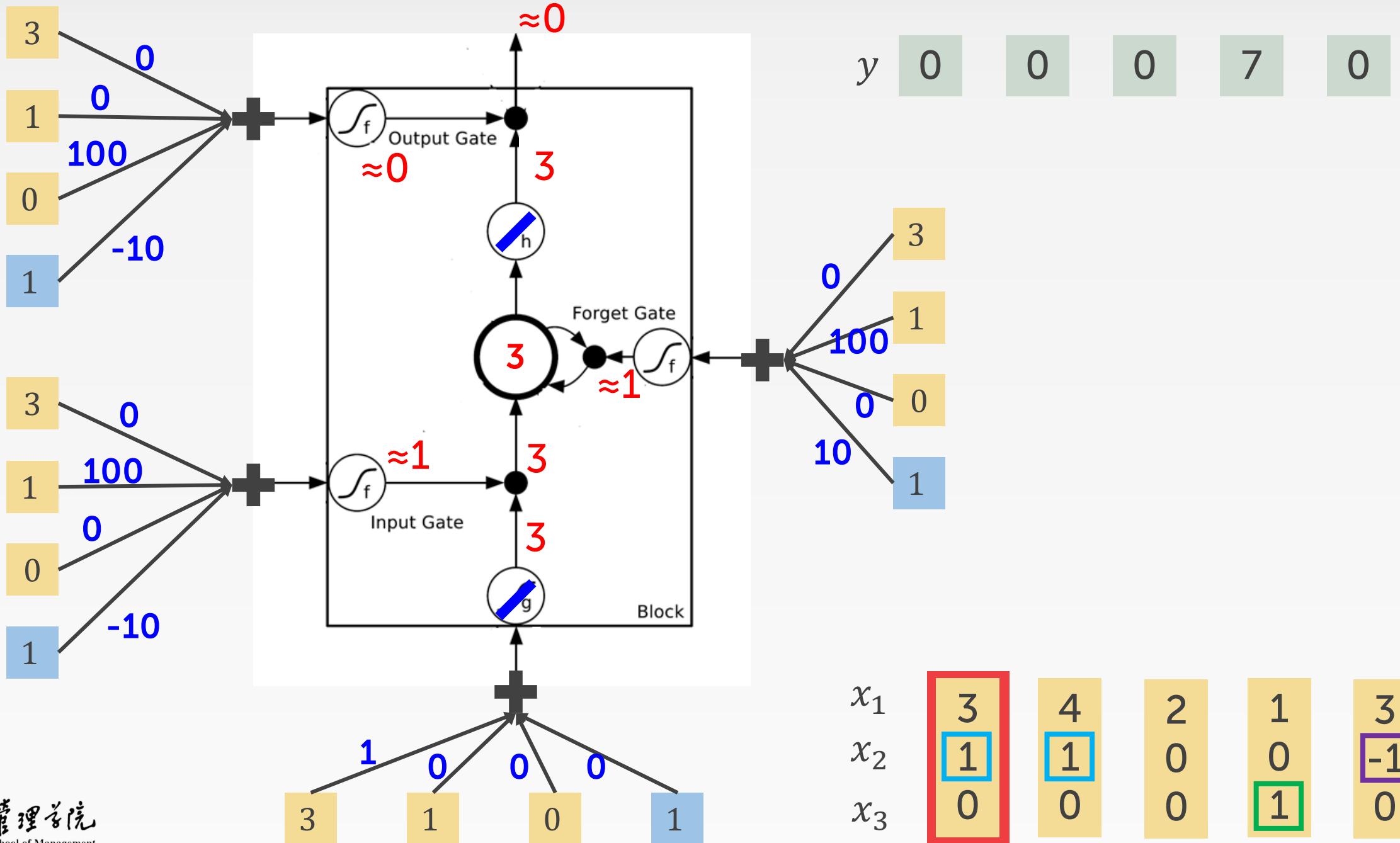
When $x_2 = 1$, add the numbers of x_1 into the memory

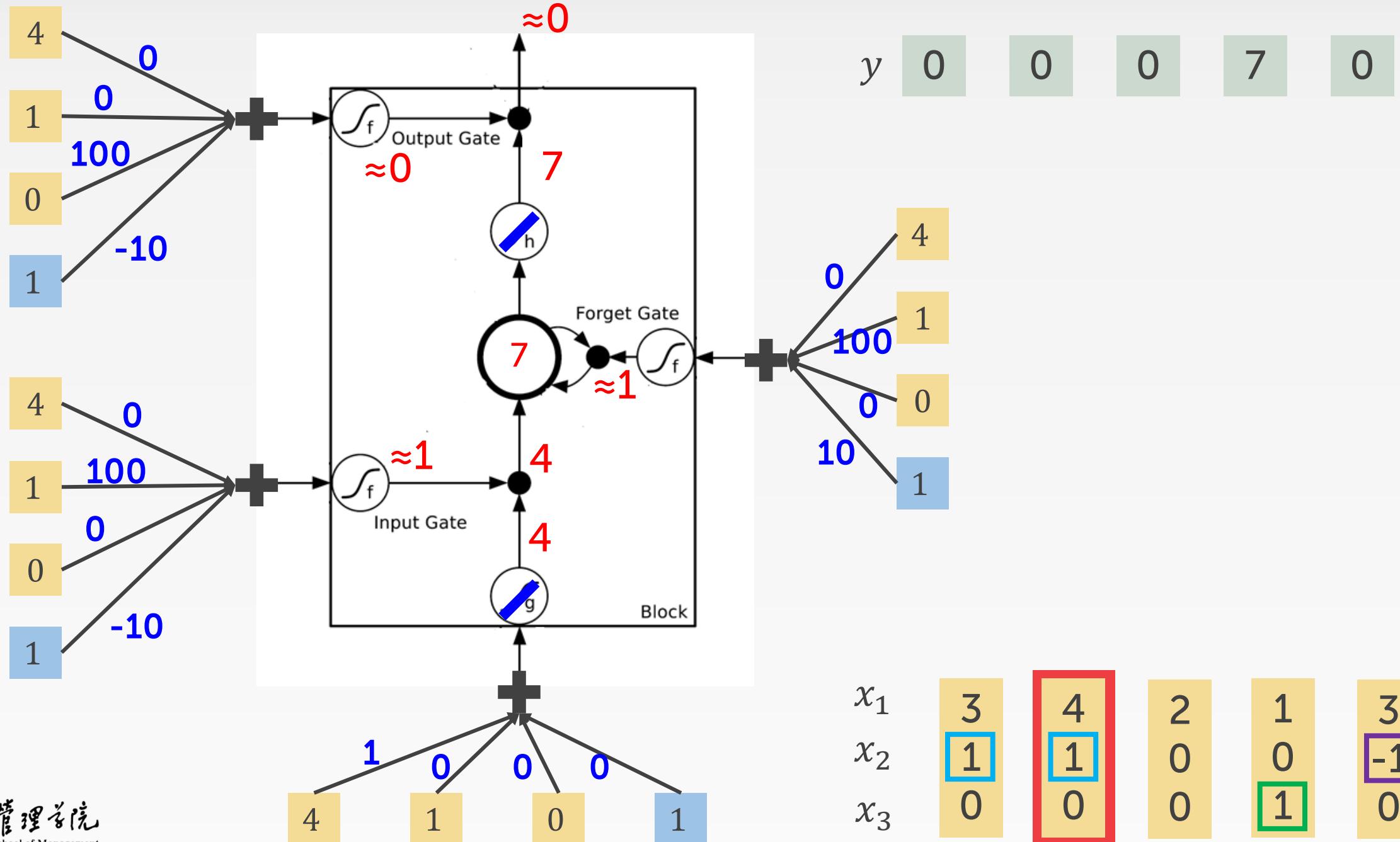
When $x_2 = -1$, reset the memory

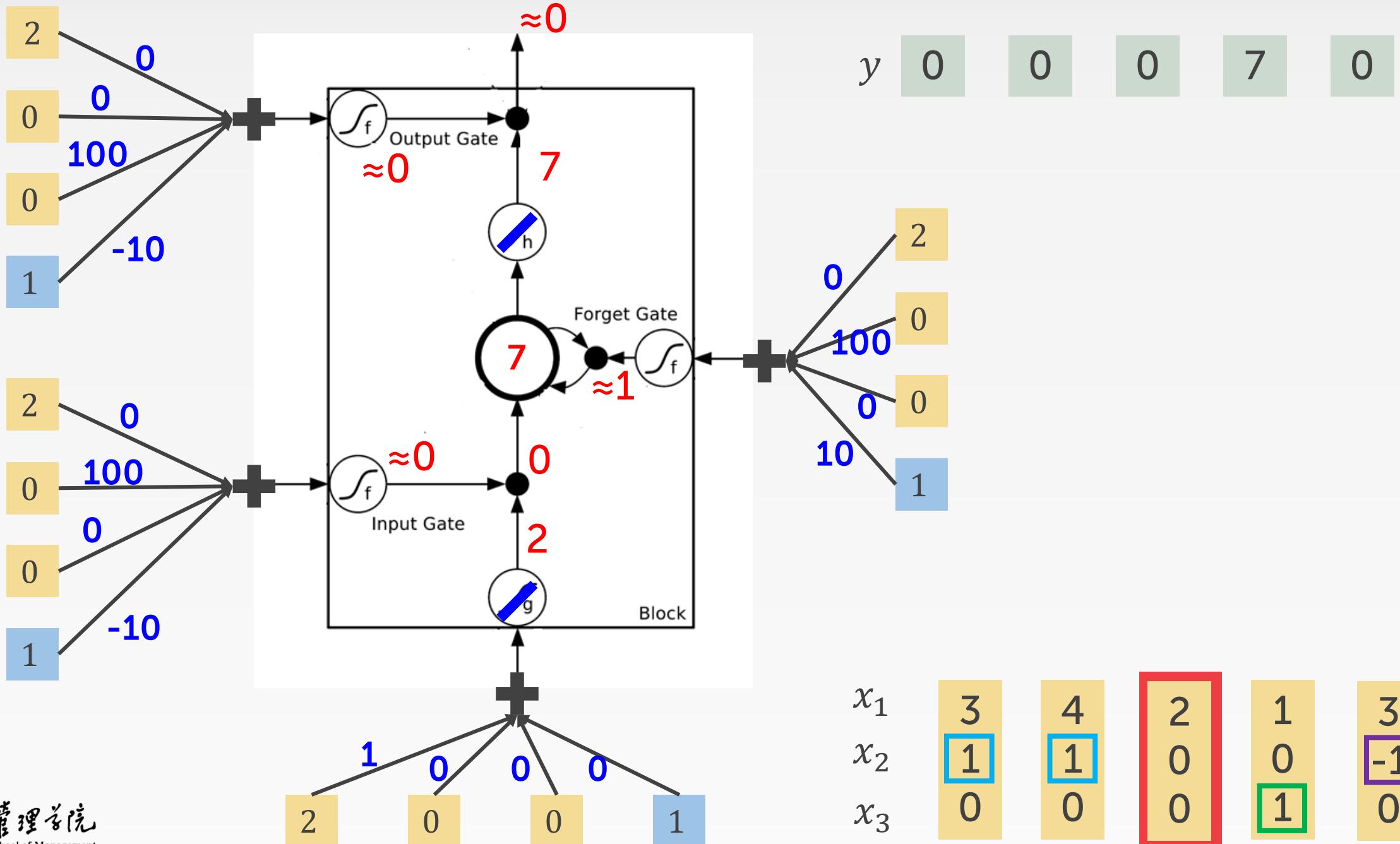
When $x_3 = 1$, output the number in the memory.

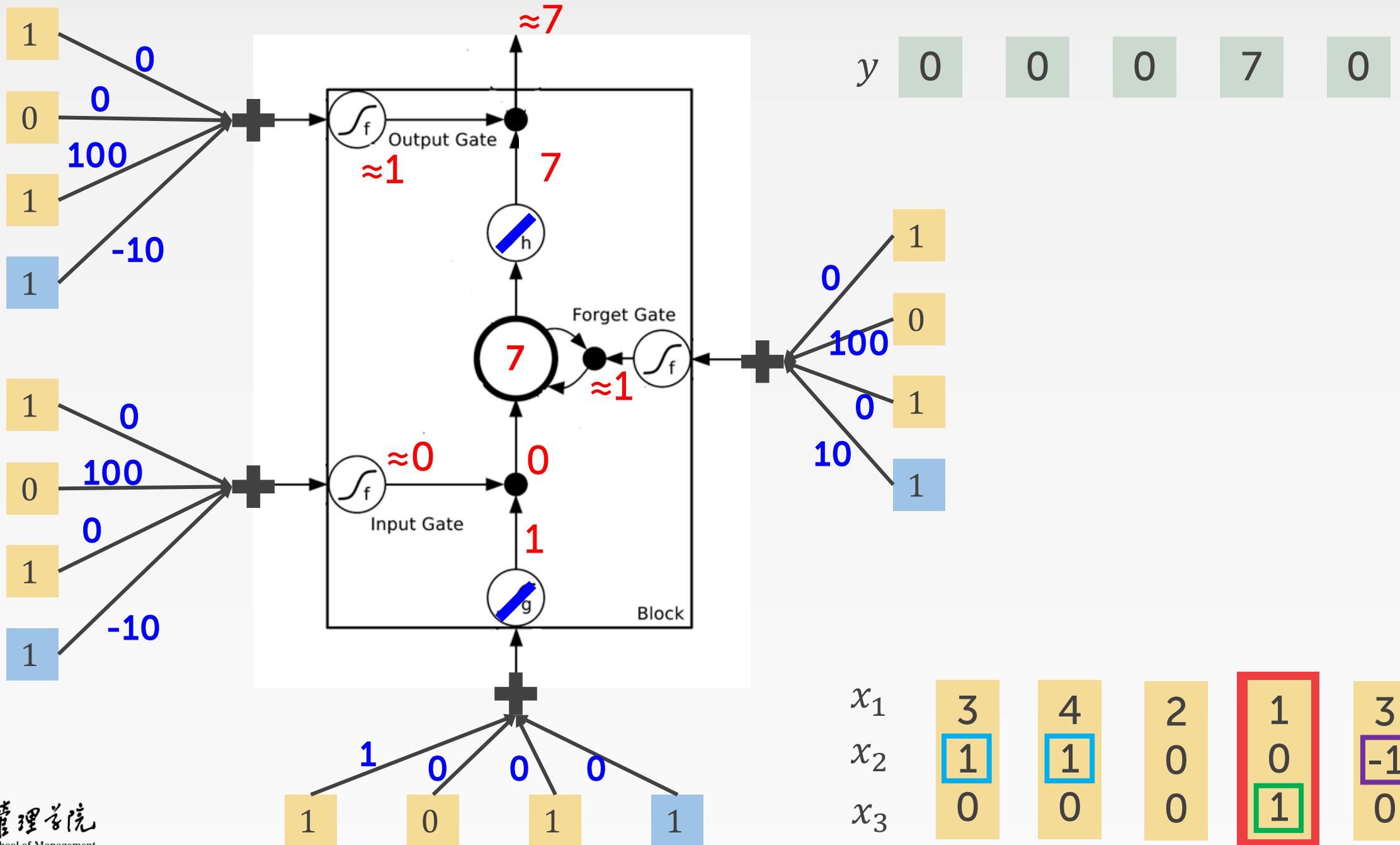


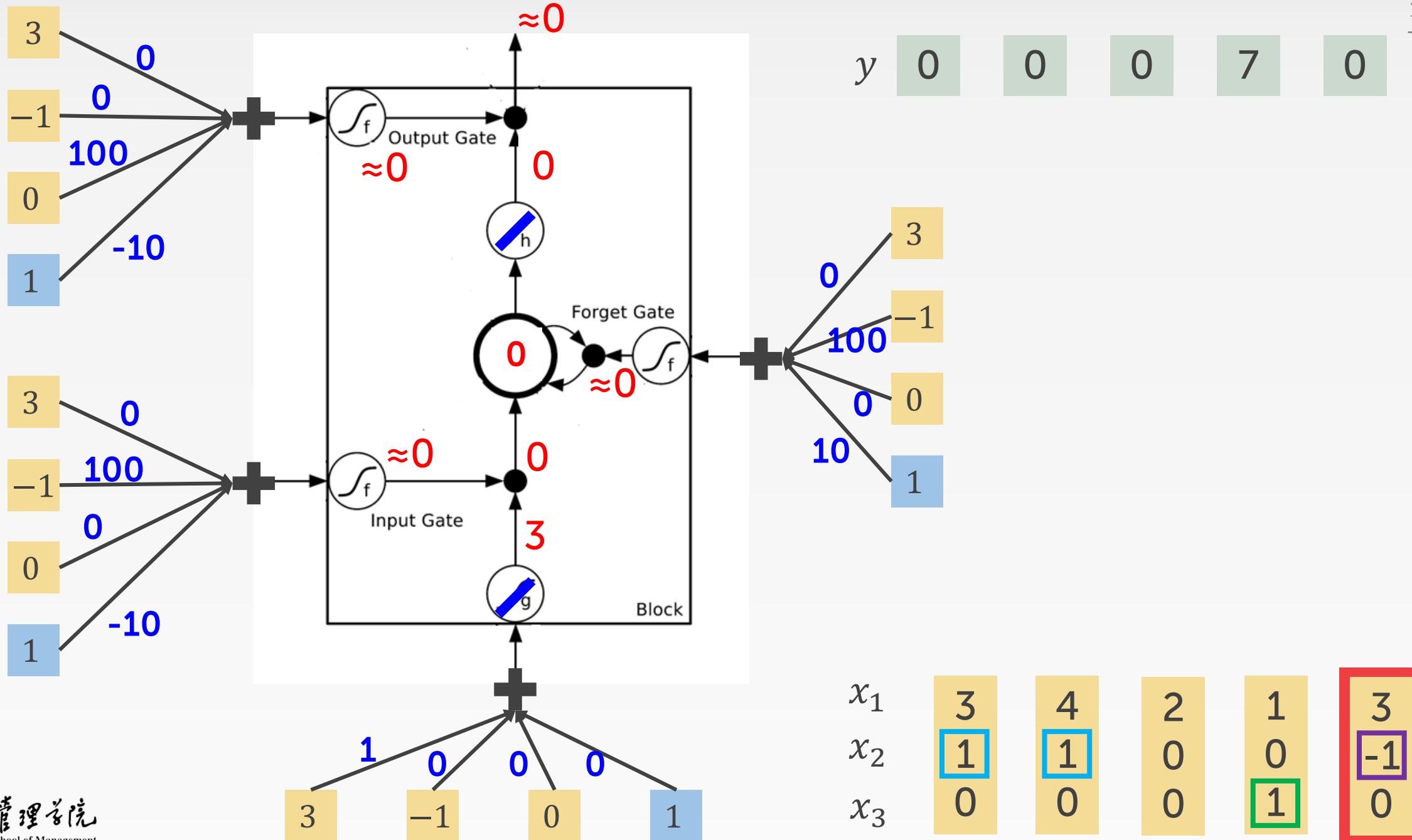








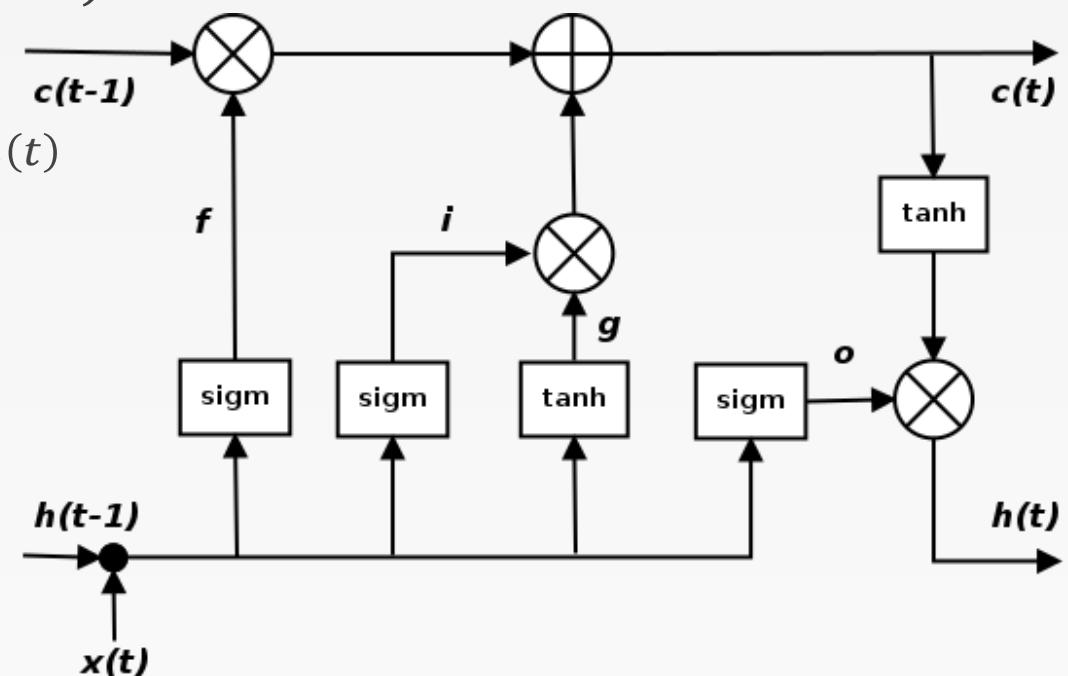




LSTM Computations

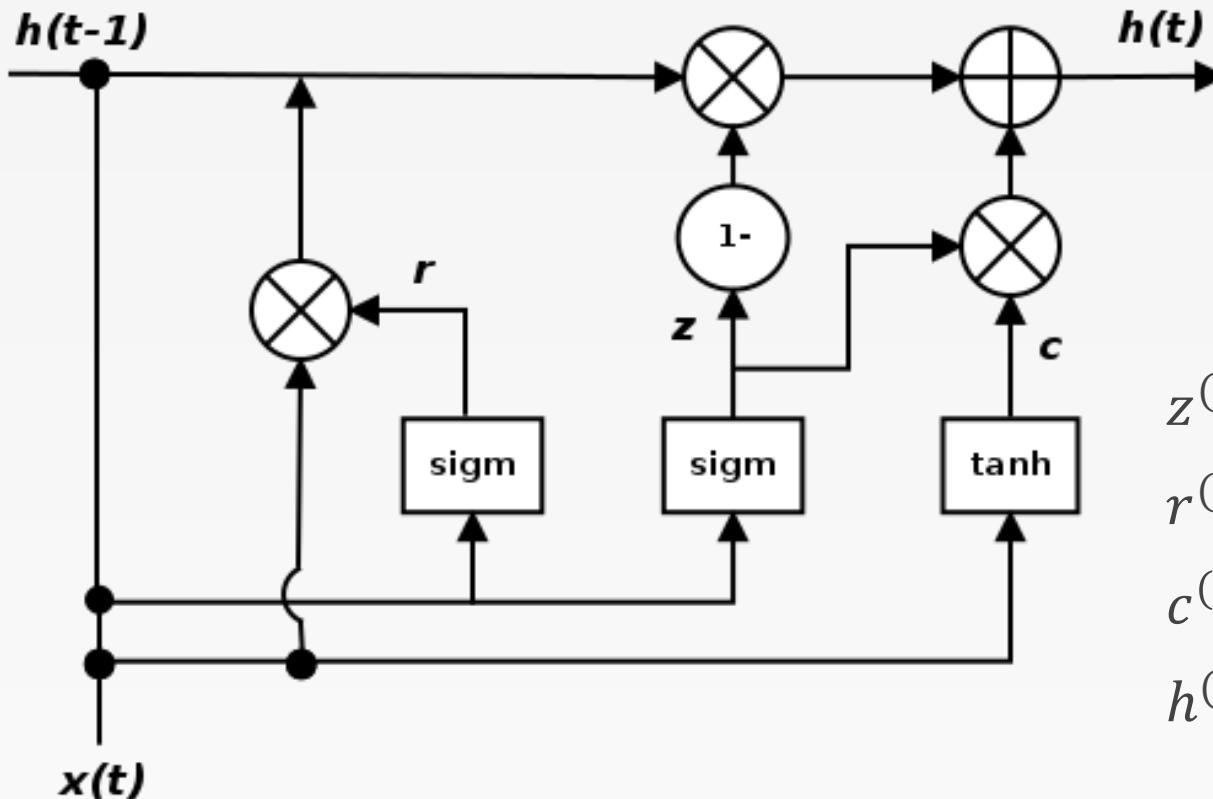
- Forget gate: $f^{(t)} = \sigma(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + b_f)$
- Input gate: $i^{(t)} = \sigma(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + b_i)$
- Output gate: $o^{(t)} = \sigma(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + b_o)$
- $g^{(t)} = \tanh(W_{gx}x^{(t)} + W_{gh}h^{(t-1)} + b_g)$
- New cell gate: $c^{(t)} = f^{(t)} \otimes c^{(t-1)} + i^{(t)} \otimes g^{(t)}$
- $y^{(t)} = h^{(t)} = o^{(t)} \otimes \tanh(c^{(t)})$

σ is the sigmoid function
 \otimes refers to element-wise product



Gated Recurrent Unit (GRU)

- Similar performance as LSTM with less computation
- They have fewer parameters than LSTM, as they lack an output gate



$$z^{(t)} = \sigma(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + b_z)$$

$$r^{(t)} = \sigma(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + b_r)$$

$$c^{(t)} = \tanh(W_{cx}x^{(t)} + W_{ch}(r^{(t)} \otimes h^{(t-1)}) + b_c)$$

$$h^{(t)} = z^{(t)} \otimes h^{(t-1)} + (1 - z^{(t)}) \otimes c^{(t)}$$

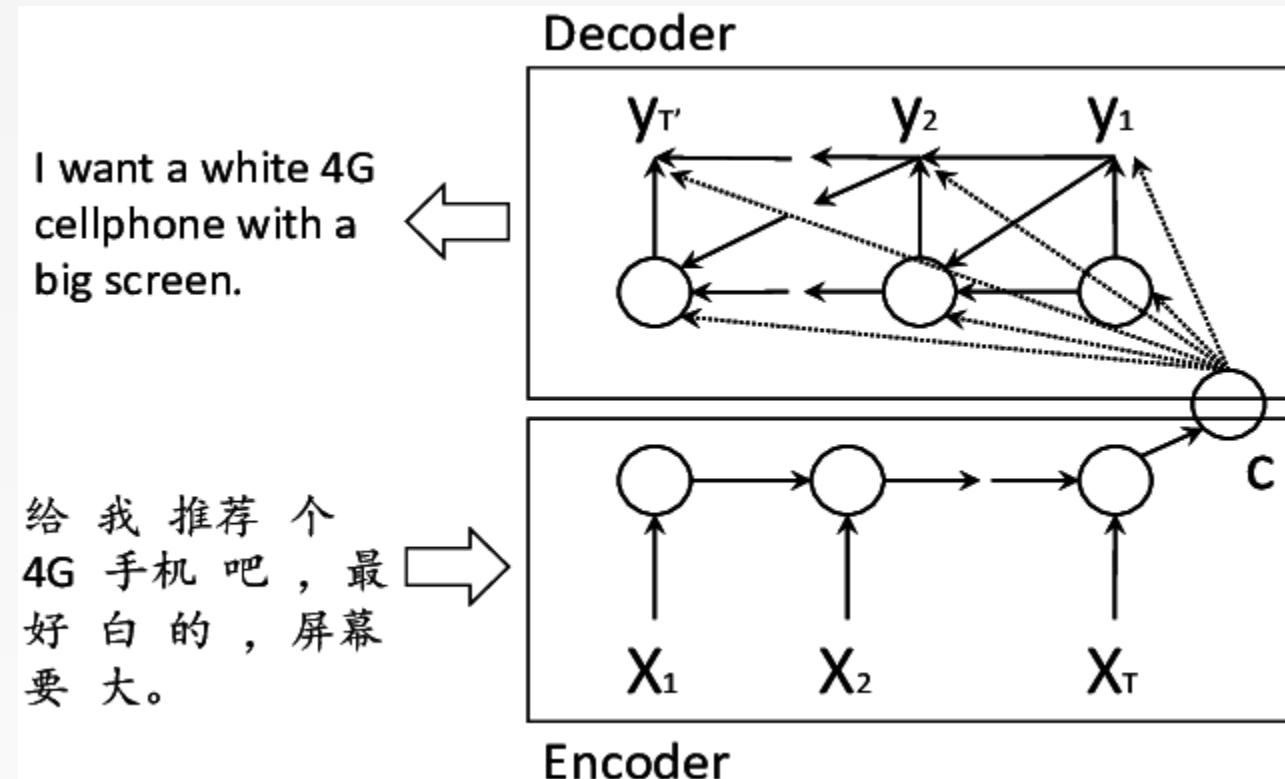
RNN Applications

- Stock Price Prediction using LSTM
 - [Project](#)
 - [Python Demo](#)
- Machine Translation
- Visual Question Answering



Machine Translation

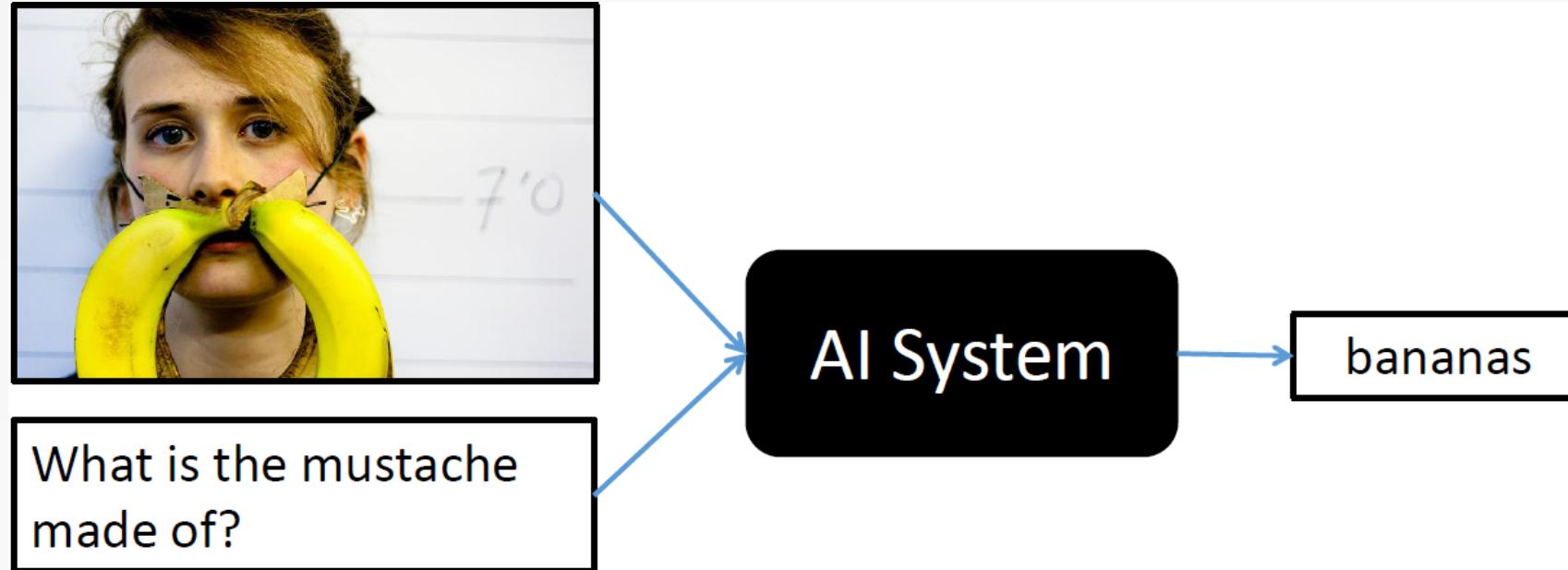
- In machine translation, the input is a sequence of words in source language, and the output is a sequence of words in target language.

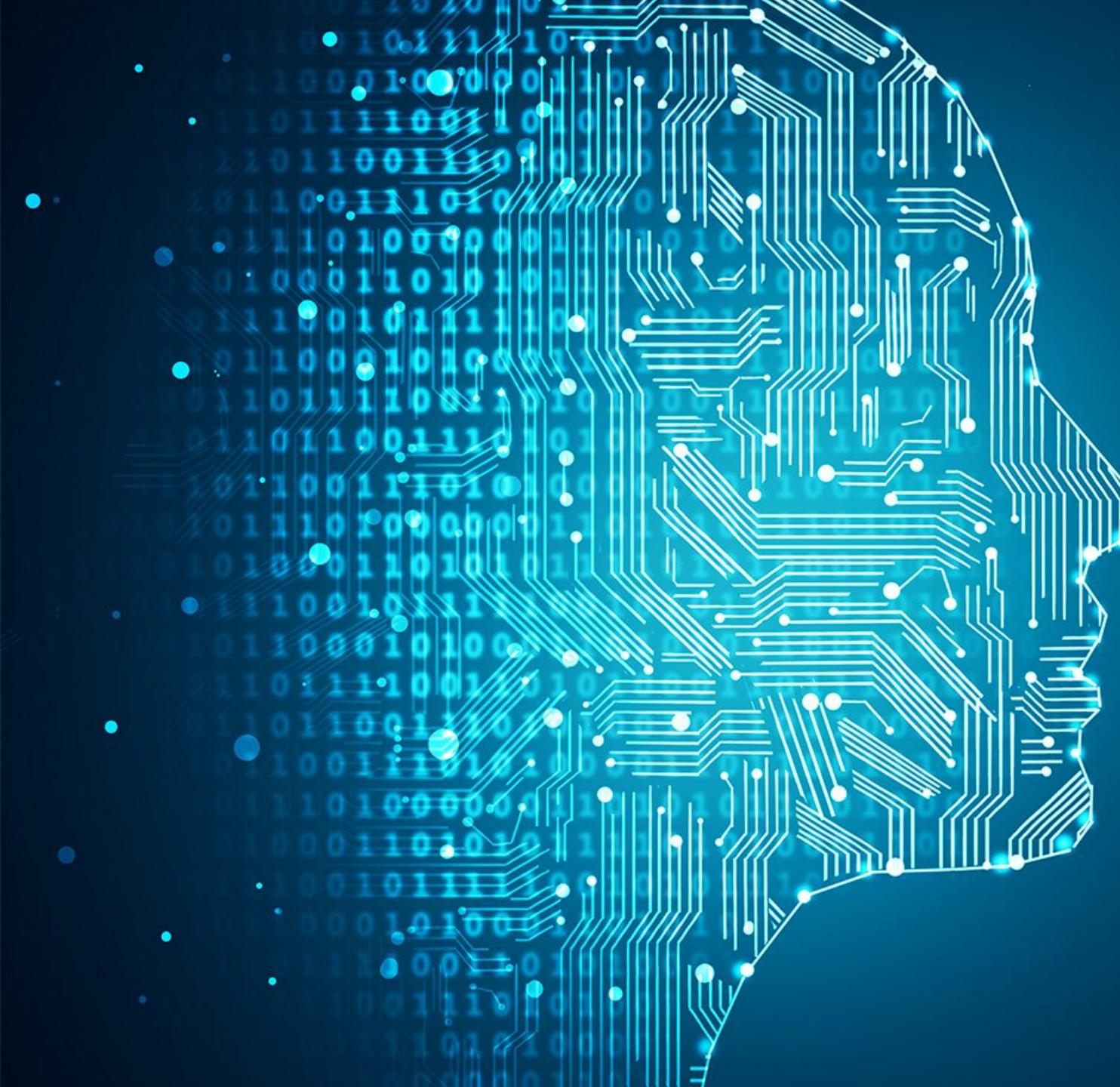


Visual Question Answering

- VQA: Given an image and a natural language question about the image, the task is to provide an accurate natural language answer

DEMO



机器学习与人工智能 Machine Learning and Artificial Intelligence

Lecture 10 Reinforcement Learning

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

Agenda

- Overview
- Markov Decision Process
- Q-Learning

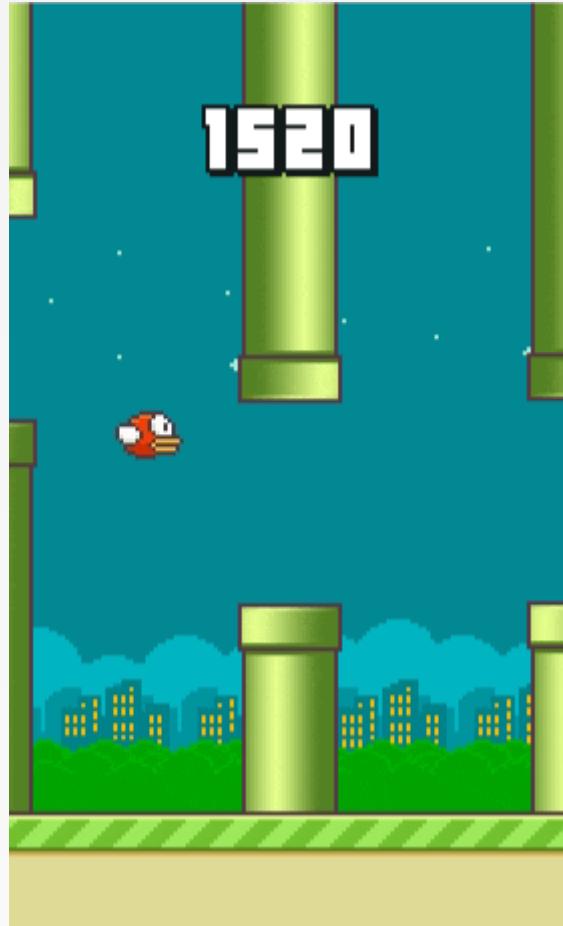
RL and ML

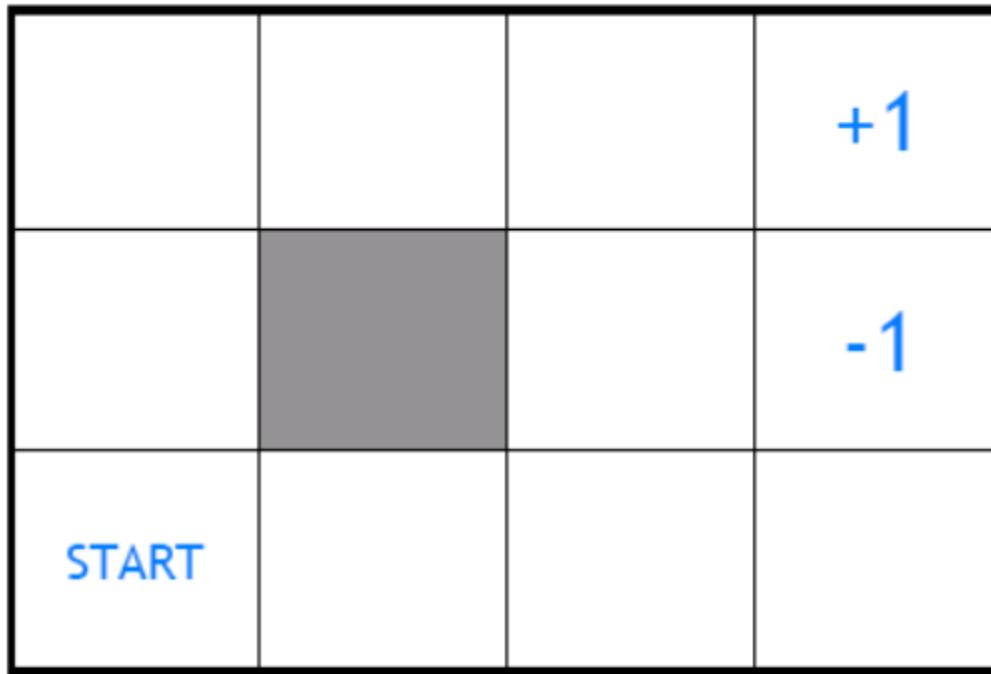
- 1. Supervised Learning (error correction)
 - learning approaches to regression & classification
 - learning from examples, learning from a teacher
- 2. Unsupervised Learning
 - learning approaches to dimensionality reduction, density estimation, recoding data based on some principle, etc.
- 3. Reinforcement Learning
 - learning approaches to sequential decision making
 - learning from a critic, learning from delayed reward

(Partial) List of Applications

- Robotics:
 - Navigation, walking, juggling, ...
- Games:
 - Backgammon, Chess, ...
- Operation Research:
 - Warehousing, transportation, scheduling, ...
- Control:
 - Helicopters, elevators, admission control in telecom, ...

Flappy Birds





actions: UP, DOWN, LEFT, RIGHT

UP

80%	move UP
10%	move LEFT
10%	move RIGHT



- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- what's the strategy to achieve max reward?

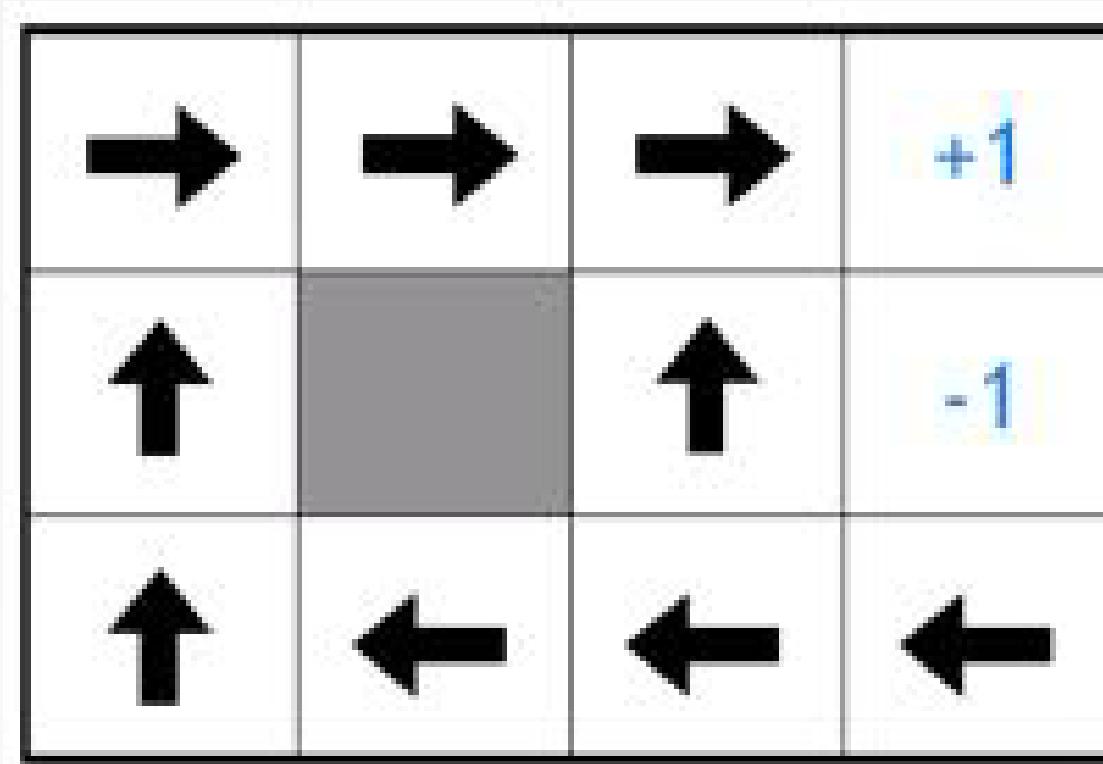
Characteristics of RL

- RL is learning how to map states to actions, so as to **maximize** a numerical **reward** over time.
- Unlike other forms of learning, it is a **multistage** decision-making process (often Markovian).
- Actions may affect not only the immediate reward but also subsequent rewards (**Delayed effect**)

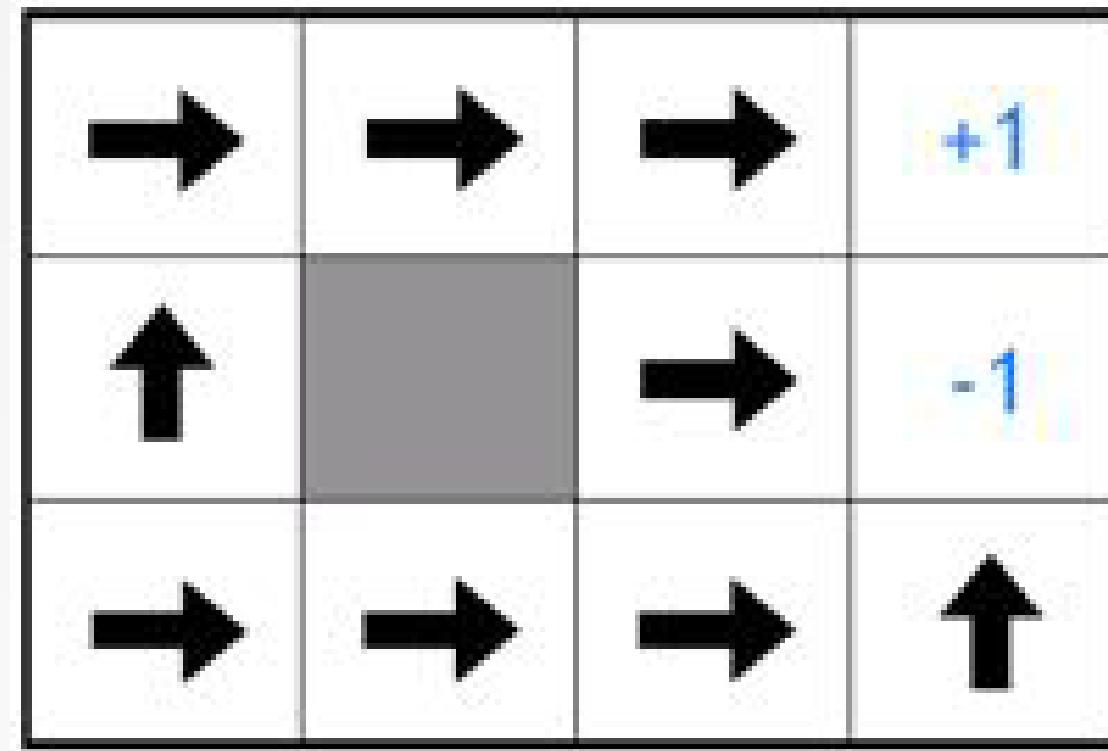
Elements of RL

- Environment:
 - Physical world in which the agent operates
- State:
 - Current situation of the agent
- A policy
 - A map from state space to action space.
 - May be stochastic.
- A reward function
 - It maps each state (or, state-action pair) to a real number, called reward.
- A value function
 - Value of a state (or, state-action pair) is the total expected reward, starting from that state (or, state-action pair).

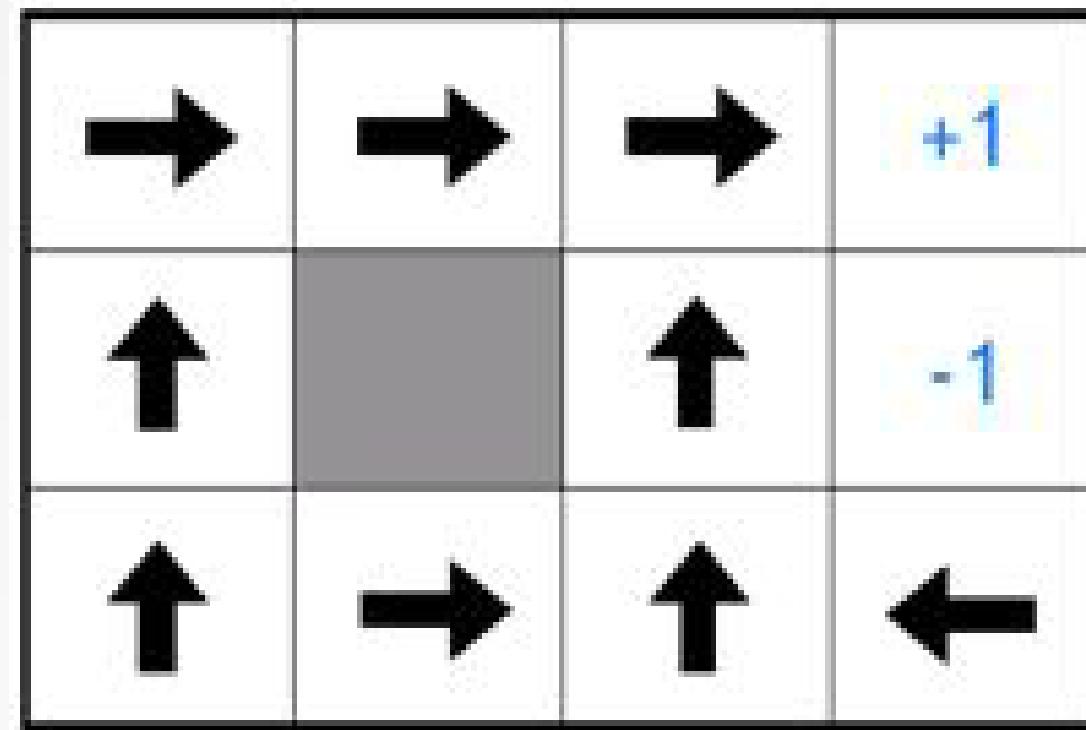
Is this policy optimal?



Reward for each step - 2



Reward for each step -0.1



The Precise Goal

- To find a **policy** that maximizes the **Value function**.
 - There are different approaches to achieve this goal in various situations.
-
- **Markov Decision Process** (value/policy iteration):
 - Require state transition and reward function
 - **Q-Learning**:
 - Unknown reward and transition function

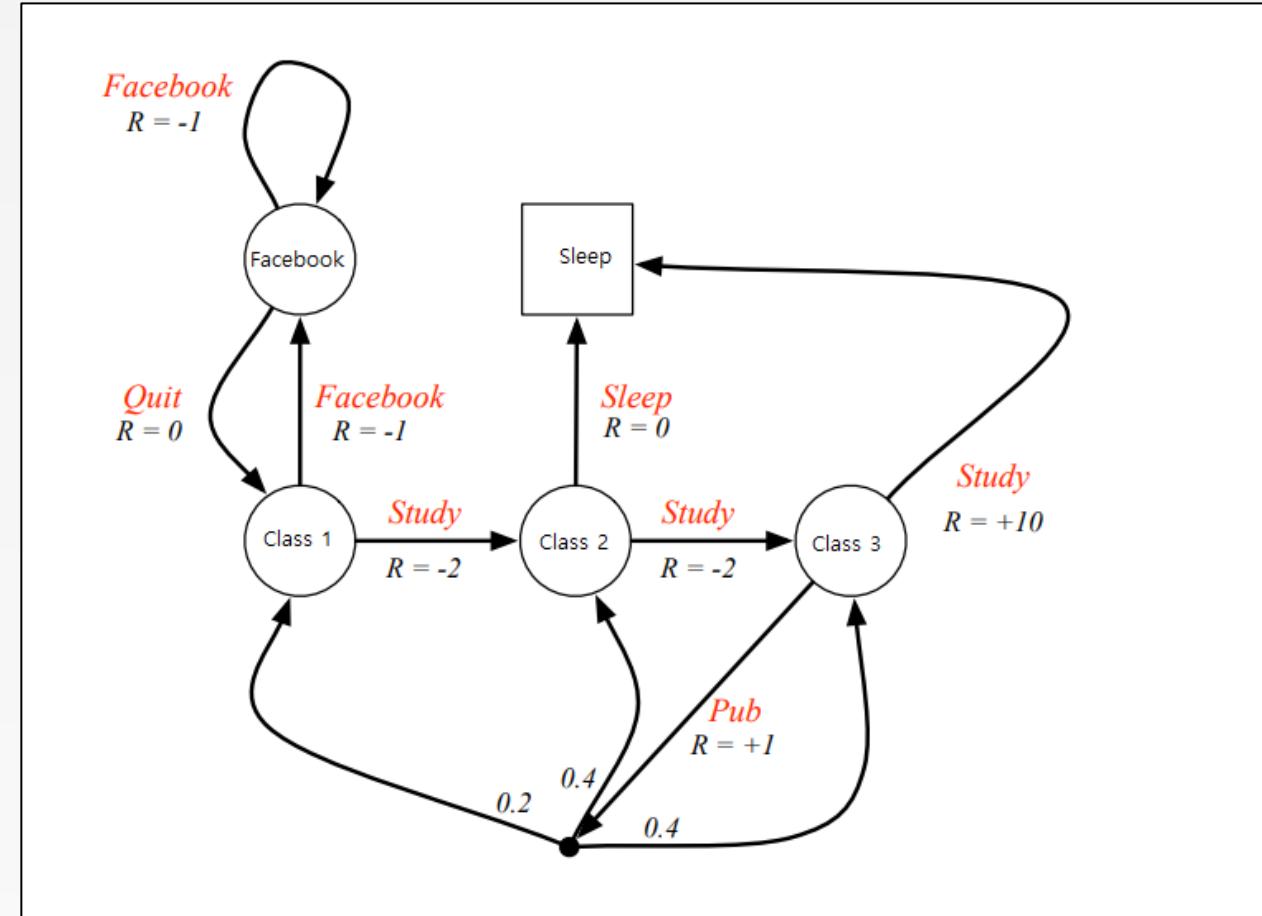
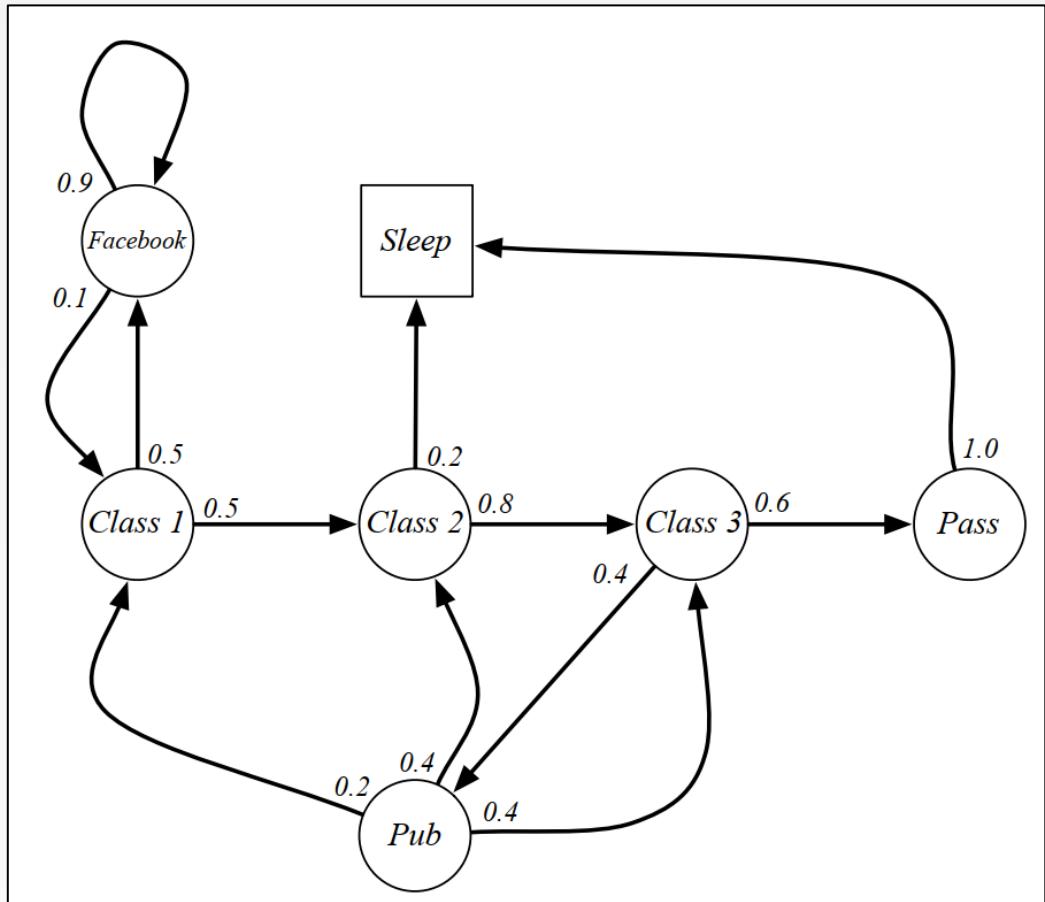
Markov Decision Process (MDP)

Components

- S : set of possible states
- A : set of possible actions
- $R(s, a)$: reward function
- $P(s'|s, a) = P(S_{t+1} = s' | S_t = s, A_t = a)$: state transition probabilities
- Markovian Assumption

$$P(S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \dots, S_1, A_1) = P(S_{t+1}|S_t, A_t)$$

Student MDP



Model for our Data

- Start at state $s_0 \in S$
- At time t , agent observes $s_t \in S$,
then chooses $a_t \in A$, $a_t = \Pi(s_t)$
then receives $r_t \in R = R(s_t, a_t)$,
and changes to $s_{t+1} \in S \sim P(\cdot | s_t, a_t)$
- Total payoff is: (γ is the discounted factor, $0 < \gamma < 1$)
$$r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \dots$$

Goal

- Learn a policy $\Pi: S \rightarrow A$ for choosing actions to maximize the expected total payoff: $E[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots] = \sum_{t=0}^{\infty} \gamma^t E[r_t]$
- $E[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots]$: infinite-horizon discounted reward

Fixed Point Iteration for Optimization

$$J(\boldsymbol{\theta})$$

$$\frac{dJ(\boldsymbol{\theta})}{d\theta_i} = 0 = f(\boldsymbol{\theta})$$

$$0 = f(\boldsymbol{\theta}) \rightarrow \theta_i = g(\boldsymbol{\theta})$$

$$\theta_i^{(t+1)} = g(\boldsymbol{\theta}^{(t)})$$

$$J(x) = \frac{x^3}{3} - 1.5x^2 + 2x$$

$$\frac{dJ(x)}{dx} = 0 = x^2 - 3x + 2 = f(x)$$

$$x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$



Fixed Point Iteration for Optimization

$$J(x) = \frac{x^3}{3} - 1.5x^2 + 2x$$

$$\frac{dJ(x)}{dx} = 0 = x^2 - 3x + 2 = f(x)$$

$$x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$

```

1  def f(x):
2      return x**2 - 3.*x + 2
3
4  def g(x):
5      return (x**2 + 2)/3
6
7
8  def solveJ(x0,f,g,n):
9      x = x0
10     for i in range(n + 1):
11         print("i = %2d x = %.4f f(x) = %.4f" %(i,x,f(x)))
12         x = g(x)
13
14 if __name__ == "__main__":
15     x = solveJ(0,f,g,20)

```



Fixed Point Iteration for Optimization

```

1  def f(x):
2      return x**2 - 3.*x + 2
3
4  def g(x):
5      return (x**2 + 2)/3
6
7
8  def solveJ(x0,f,g,n):
9      x = x0
10     for i in range(n + 1):
11         print("i = %2d x = %.4f f(x) = %.4f" %(i,x,f(x)))
12         x = g(x)
13
14 if __name__ == "__main__":
15     x = solveJ(0,f,g,20)

```

```

i = 0 x = 0.0000 f(x) = 2.0000
i = 1 x = 0.6667 f(x) = 0.4444
i = 2 x = 0.8148 f(x) = 0.2195
i = 3 x = 0.8880 f(x) = 0.1246
i = 4 x = 0.9295 f(x) = 0.0755
i = 5 x = 0.9547 f(x) = 0.0474
i = 6 x = 0.9705 f(x) = 0.0304
i = 7 x = 0.9806 f(x) = 0.0198
i = 8 x = 0.9872 f(x) = 0.0130
i = 9 x = 0.9915 f(x) = 0.0086
i = 10 x = 0.9944 f(x) = 0.0057
i = 11 x = 0.9963 f(x) = 0.0038
i = 12 x = 0.9975 f(x) = 0.0025
i = 13 x = 0.9983 f(x) = 0.0017
i = 14 x = 0.9989 f(x) = 0.0011
i = 15 x = 0.9993 f(x) = 0.0007
i = 16 x = 0.9995 f(x) = 0.0005
i = 17 x = 0.9997 f(x) = 0.0003
i = 18 x = 0.9998 f(x) = 0.0002
i = 19 x = 0.9999 f(x) = 0.0001
i = 20 x = 0.9999 f(x) = 0.0001

```



Value Function

- Bellman Equation

$$V^\Pi(s) = R(s, a) + \gamma \sum_{s_1 \in S} P(s_1 | s, a) V^\Pi(s_1)$$

Value Function



$R(s, a) = -100$ if falling

$R(s, a) = 100$ if succeed

$R(s, a) = 0$ otherwise

$$\gamma = 0.9$$

Value Iteration

Initialize $V(s) = 0$ or randomly

while not converged:

for $s \in S$:

$$V(s) = \max_a R(s, a) + \gamma \sum_{s'} p(s'|s, a)V(s')$$

Initialize $Q(s, a) = 0$, $V(s) = 0$ or randomly

while not converged:

for $s \in S$

for $a \in A$

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} p(s'|s, a)V(s')$$

$$V(s) = \max_a Q(s, a)$$

Return $\Pi(s) = \operatorname{argmax}_a Q(s, a), \forall s$



Policy Iteration

Algorithm 1 Policy Iteration

1: **procedure** POLICYITERATION($R(s, a)$,
transition probabilities)

2: Initialize policy π randomly

3: **while** not converged **do**

4: Solve Bellman equations for fixed policy π

Compute value
function for fixed
policy is easy

$\gamma, p(\cdot|s, a)$
System of $|S|$
equations and $|S|$
variables

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s))V^\pi(s'), \quad \forall s$$

5: Improve policy π using new value function

$$\pi(s) = \operatorname{argmax}_a R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a)V^\pi(s')$$

Greedy policy
w.r.t. current
value function

Greedy policy might **remain the
same** for a particular state if there is
no better action

6: **return** π

Q-Learning

Exploration and Exploitation

- Exploration: find more information
- Exploitation: maximize the reward by exploiting already known information



K-Armed Bandits Problem

Motivation

Initialize $Q(s, a) = 0, V(s) = 0$ or randomly

while not converged:

for $s \in S$

for $a \in A$

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} p(s'|s, a)V(s')$$

$$V(s) = \max_a Q(s, a)$$

What if we don't know

$R(s, a)$ or $p(s'|s, a)$?



Motivation

- Let $Q^*(s, a)$ be the (true) expected discounted reward for taking a in s
- $V^*(s) = \max_a Q^*(s, a)$
- $Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) \max_{a'} Q^*(s', a')$
- $\Pi^*(s) = \text{argmax}_a Q^*(s, a)$
- Idea of Q-Learning: If we can learn Q^* , then we can define Π^* without $R(s, a)$ and $p(s'|s, a)$

Algorithm

1. Initialize $Q(s, a) = 0$
2. Do forever:
 - a. Select action a and execute it
 - b. Receive reward r from environment
 - c. Observe new state s'
 - d. Update $Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$, When the transition probability is *deterministic*.
or update $Q(s, a) \leftarrow (1 - \alpha_n)Q(s, a) + \alpha_n \left(r + \gamma \max_{a'} Q(s', a') \right)$, when
the transition probability is *stochastic*. $\alpha_n = \frac{1}{1 + visit_n(s, a)}$

ϵ -greedy Variant

- Choose hyperparameter ϵ
- New step 2a.
 - With prob. $(1 - \epsilon)$: select action $a = \max_{a'} Q(s, a')$
 - With prob. ϵ : select random action $a \in A$

Deep Q-Learning

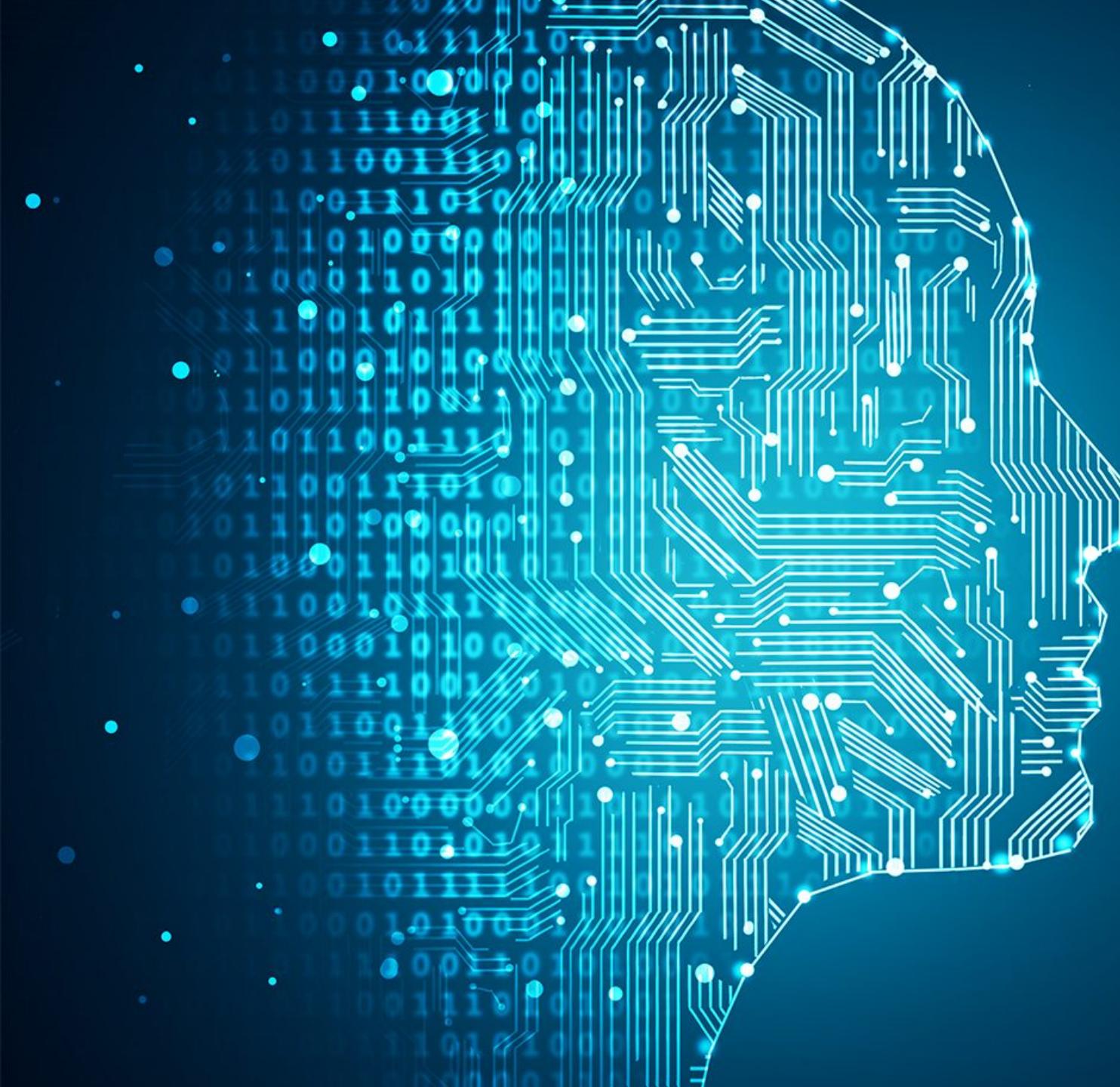
Question: What if our state space S is too large to represent with a table?

Examples:

- s_t = pixels of a video game
- s_t = continuous values of sensors in a manufacturing robot
- s_t = sensor output from a self-driving car

Answer: Use a parametric function to approximate the table entries





机器学习与人工智能 Machine Learning and Artificial Intelligence

Lecture 11 Review

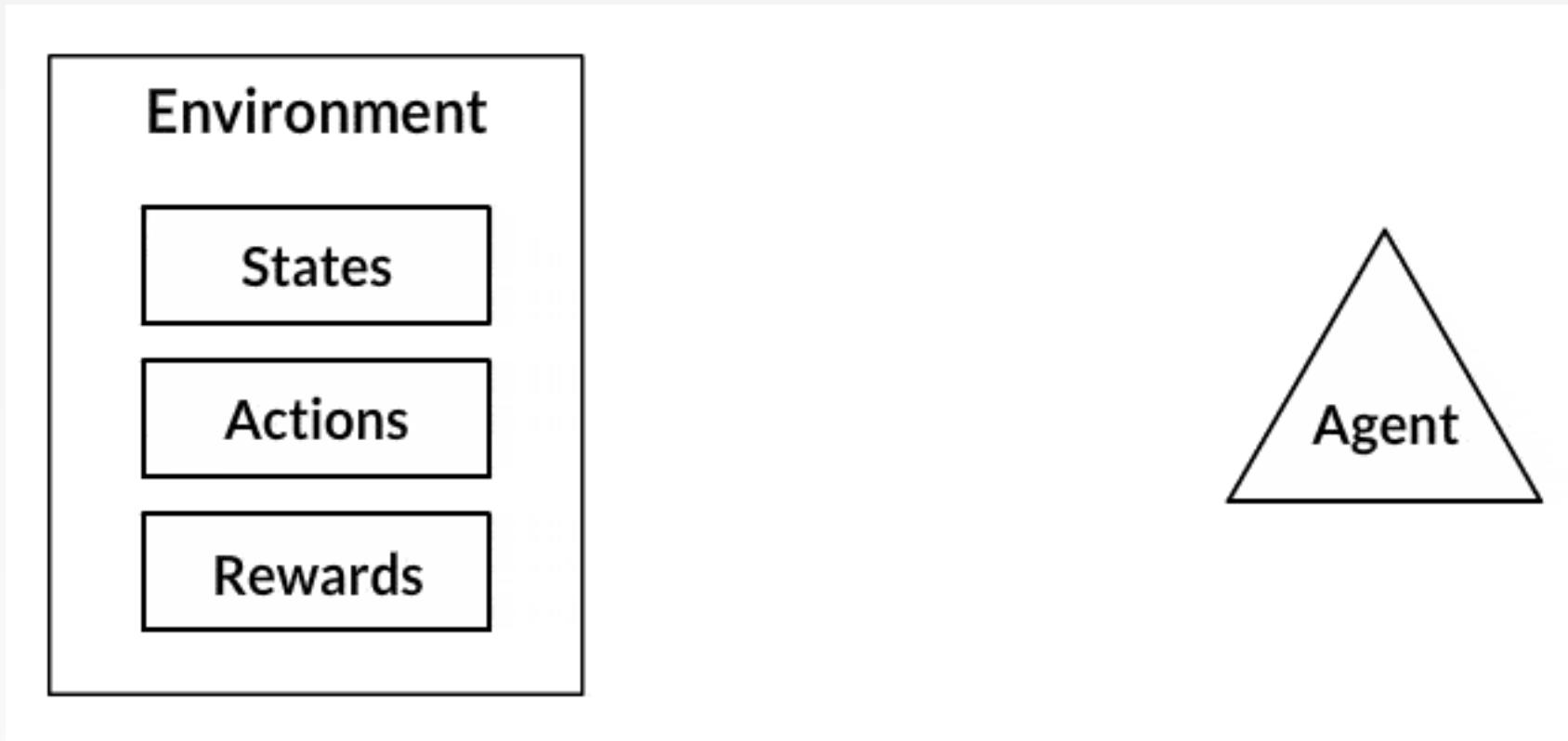
Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2021 Fall

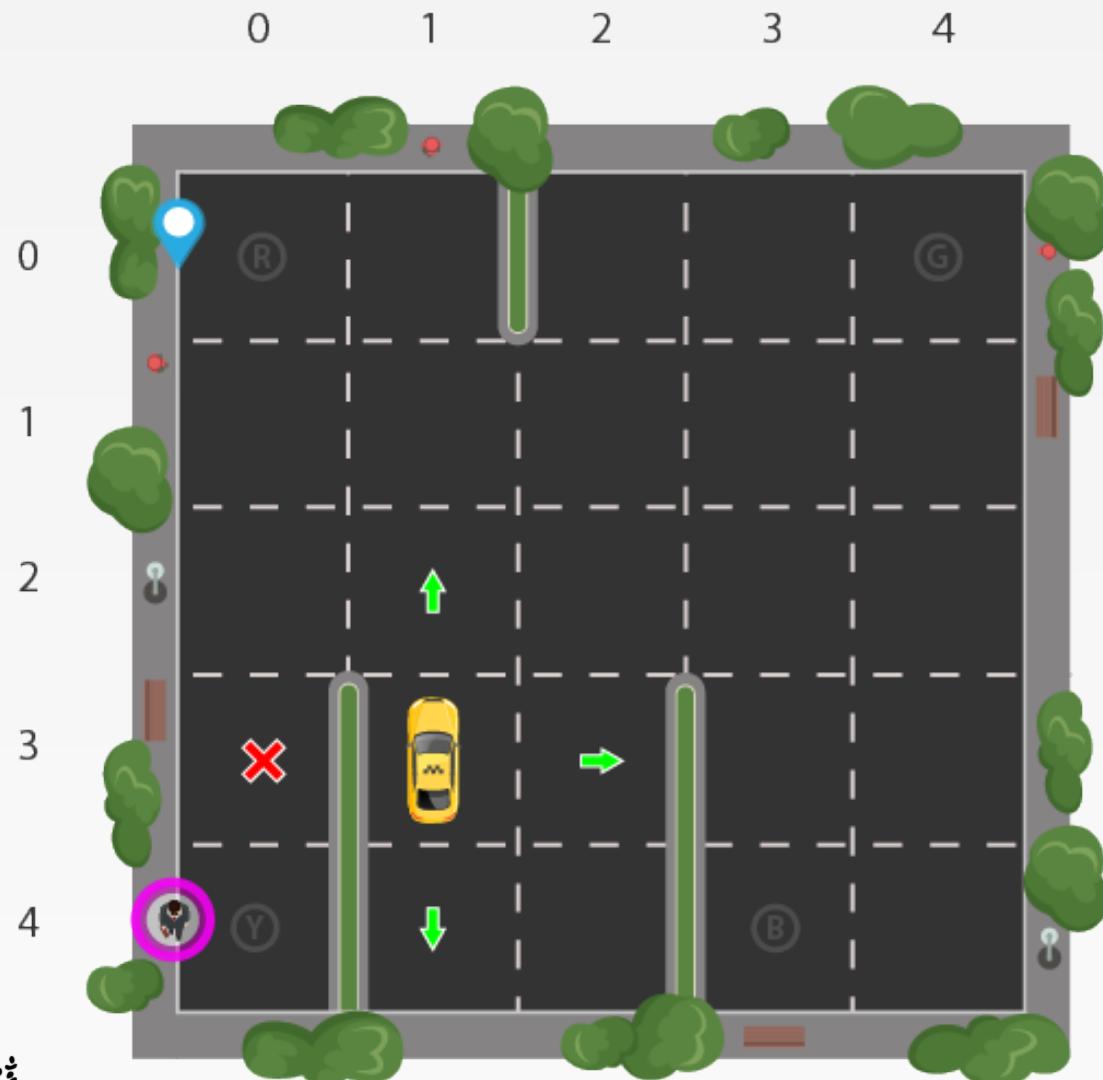
Reinforcement Learning



Open AI Gym

- <https://gym.openai.com/>
- “Open source interface to reinforcement learning tasks. The gym library provides an easy-to-use suite of reinforcement learning tasks.”
- “We provide the environment; you provide the algorithm.”

Taxi-v3



+	-	-	-	-	+
	R:		:	:G	
	:		:	:	
	:		:	:	
	Y	:		B:	
+	-	-	-	-	+

Basic Setup

```
import gym

env = gym.make("Taxi-v3").env
env.render()

env.reset() # reset environment to
env.render()

print("Action Space {}".format(env.
print("State Space {}".format(env.
```

+-----+
R: : : G
: : :
: : :
: : :
Y : B:
+-----+

Action Space Discrete(6)
State Space Discrete(500)



Basic Setup

```
state = env.encode(3, 1, 2, 0) #  
print("State:", state)  
  
env.s = state  
env.render()  
  
env.P[328]  
  
{0: [(1.0, 428, -1, False)],  
 1: [(1.0, 228, -1, False)],  
 2: [(1.0, 348, -1, False)],  
 3: [(1.0, 328, -1, False)],  
 4: [(1.0, 328, -10, False)],  
 5: [(1.0, 328, -10, False)]}
```

State: 328

R:	:	:	G
:	:	:	:
:	:	:	:
	█	:	:
Y	:	B:	

Solve the problem without RL

```

env.s = 328 # set environment to illustration's state

epochs = 0
penalties, reward = 0, 0

frames = [] # for animation

done = False

while not done:
    action = env.action_space.sample()
    state, reward, done, info = env.step(action)

    if reward == -10:
        penalties += 1

    # Put each rendered frame into dict for animation
    frames.append({
        'frame': env.render(mode='ansi'),
        'state': state,
        'action': action,
        'reward': reward
    })

    epochs += 1

```

R:	:	:	G
:	:	:	:
:	:	:	:
	Y	:	B:

(Dropoff)

Timestep: 1

State: 328

Action: 5

Reward: -10



Q-Learning (Training)

```
import numpy as np
q_table = np.zeros([env.observation_space.n, env.action_space.n])

import random
from IPython.display import clear_output

# Hyperparameters
alpha = 0.1
gamma = 0.6
epsilon = 0.1

# For plotting metrics
all_epochs = []
all_penalties = []
```



Q-Table

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0

	327	0	0	0	0	0	0

	499	0	0	0	0	0	0



Q-Table

Training
↓

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	·	·	·	·	·	·	·
	·	·	·	·	·	·	·
	·	·	·	·	·	·	·
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603



Q-Learning (Training)

```
for i in range(1, 100001):
    state = env.reset()

    epochs, penalties, reward, = 0, 0, 0
    done = False

    while not done:
        if random.uniform(0, 1) < epsilon:
            action = env.action_space.sample() # Explore action space
        else:
            action = np.argmax(q_table[state]) # Exploit learned values

        next_state, reward, done, info = env.step(action)

        old_value = q_table[state, action]
        next_max = np.max(q_table[next_state])

        new_value = (1 - alpha) * old_value + alpha * (reward + gamma * next_max)
        q_table[state, action] = new_value

        if reward == -10:
            penalties += 1

        state = next_state
        epochs += 1
```

Q-Learning (Evaluation)

```
total_epochs, total_penalties = 0, 0
episodes = 100

for _ in range(episodes):
    state = env.reset()
    epochs, penalties, reward = 0, 0, 0

    done = False

    while not done:
        action = np.argmax(q_table[state])
        state, reward, done, info = env.step(action)

        if reward == -10:
            penalties += 1

        epochs += 1

    total_penalties += penalties
    total_epochs += epochs
```



Comparison

Measure	Random agent's performance	Q-learning agent's performance
Average rewards per move	-3.9012092102214075	0.6962843295638126
Average number of penalties per episode	920.45	0.0
Average number of timesteps per trip	2848.14	12.38



Exam Logistics



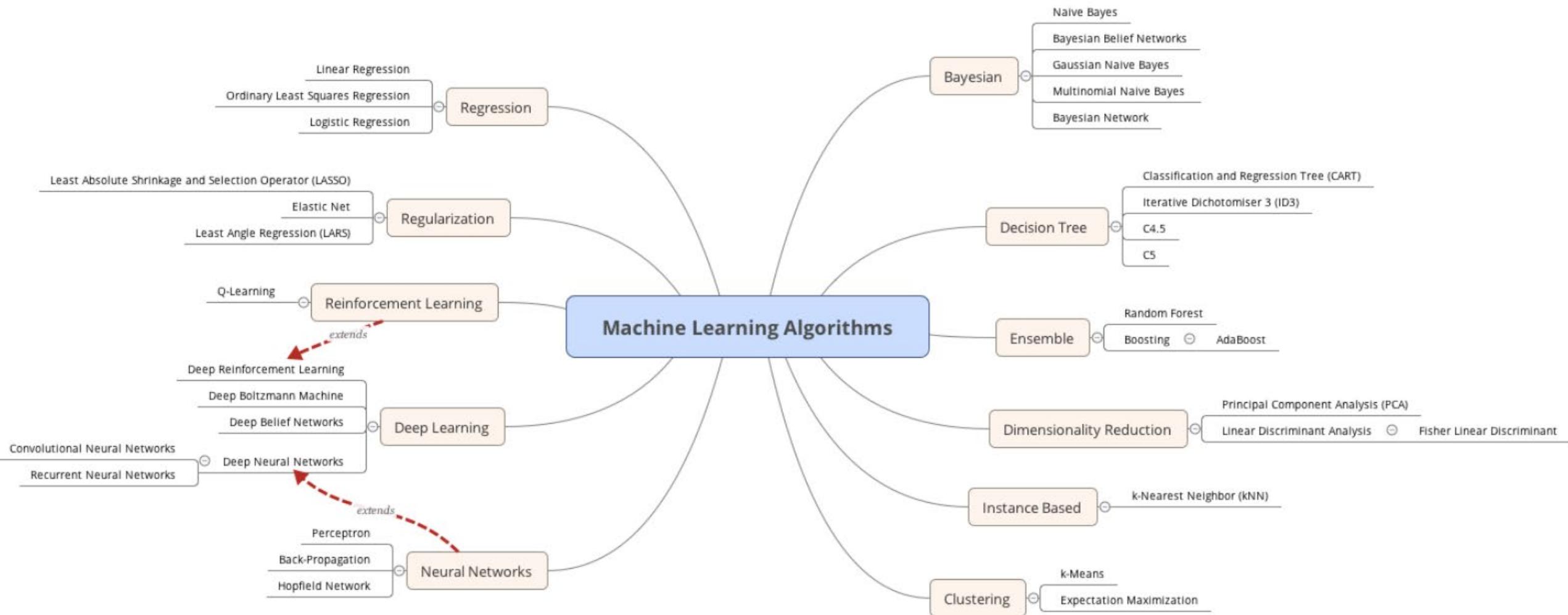
Final Exam

- December 2, 2021 **3:10-5:10 pm**
- Location: **光华101教室**
 - Online (4 students, Teams)
- Closed-book
- One-page (A4 size) cheating paper
- A calculator is allowed



Final Exam

- Total points: 100 + 5 bonus
- Formats of questions:
 - True/false
 - Multiple choice questions
 - Short answer questions
 - Interpreting figures
- Either Chinese or English is OK.



Topics

- Important concepts:
 - Supervised vs. Unsupervised
 - Generative vs. Discriminative
 - Optimization
 - Gradient Descent
 - Model selection and evaluation
 - Overfitting
 - Regularization
 - Applications

Types of ML Systems

Criteria

Whether or not they are trained with human supervision

Supervised Learning

Fraud detection
Prediction of stock markets

Unsupervised Learning

Customer segmentation
Recommendation

Semi-supervised Learning

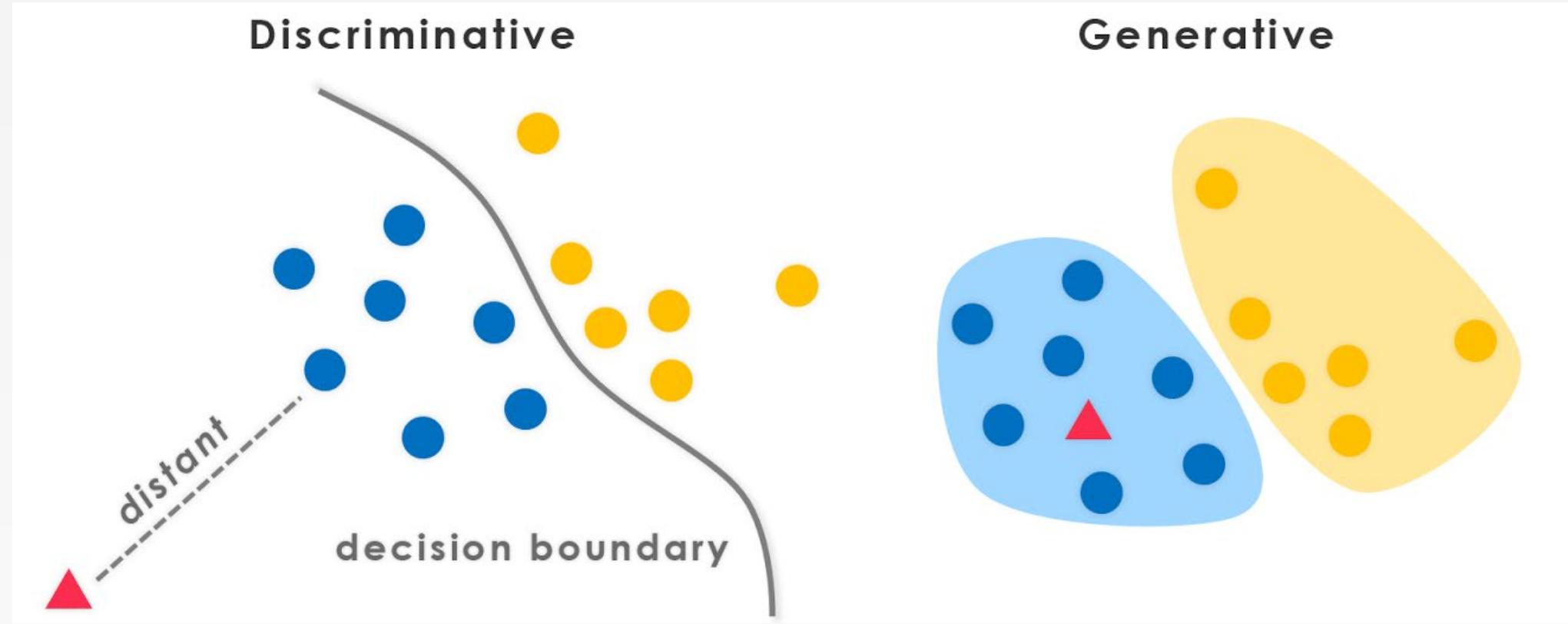
Photo-hosting service
Speech analysis
Web-content classification

Reinforcement Learning

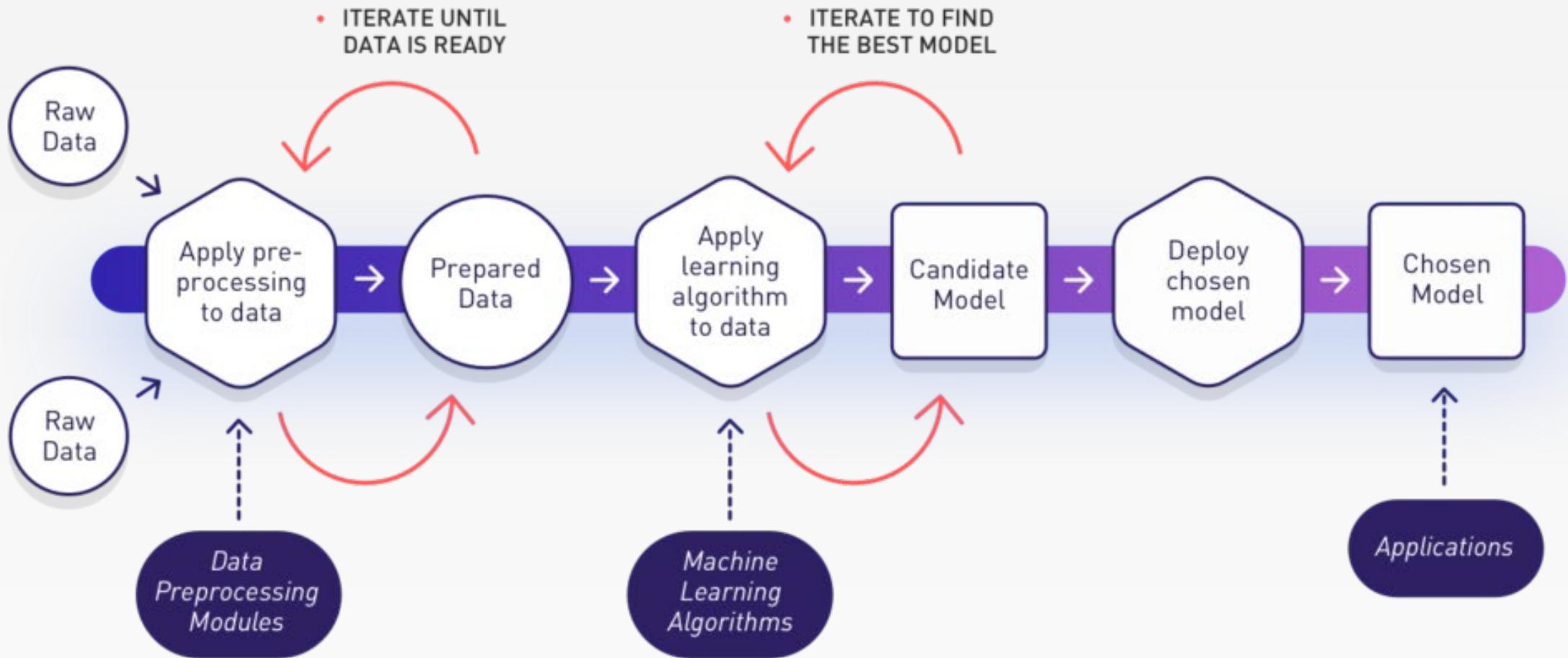
Robotics
Go games
Self-driving cars



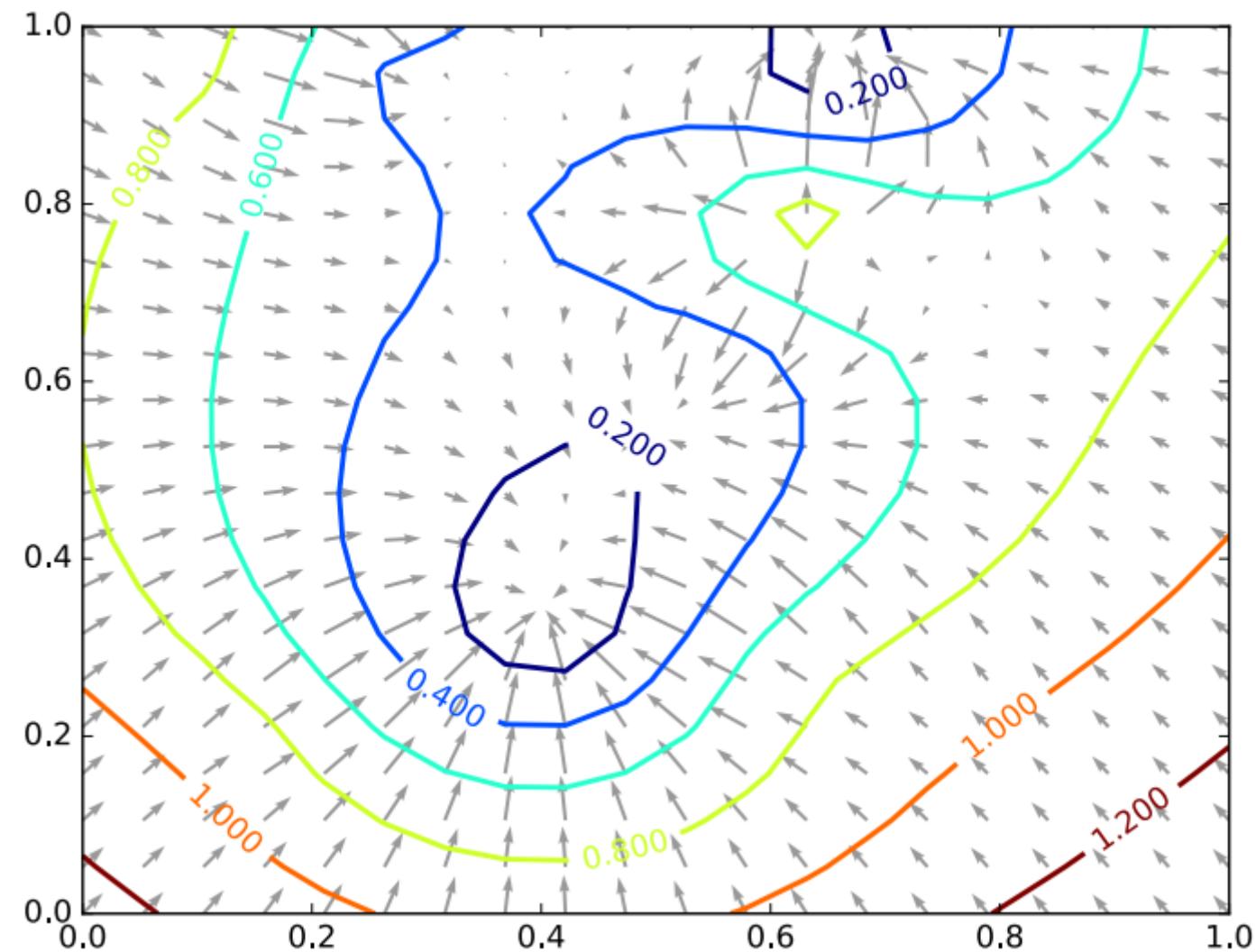
Generative vs. Discriminative



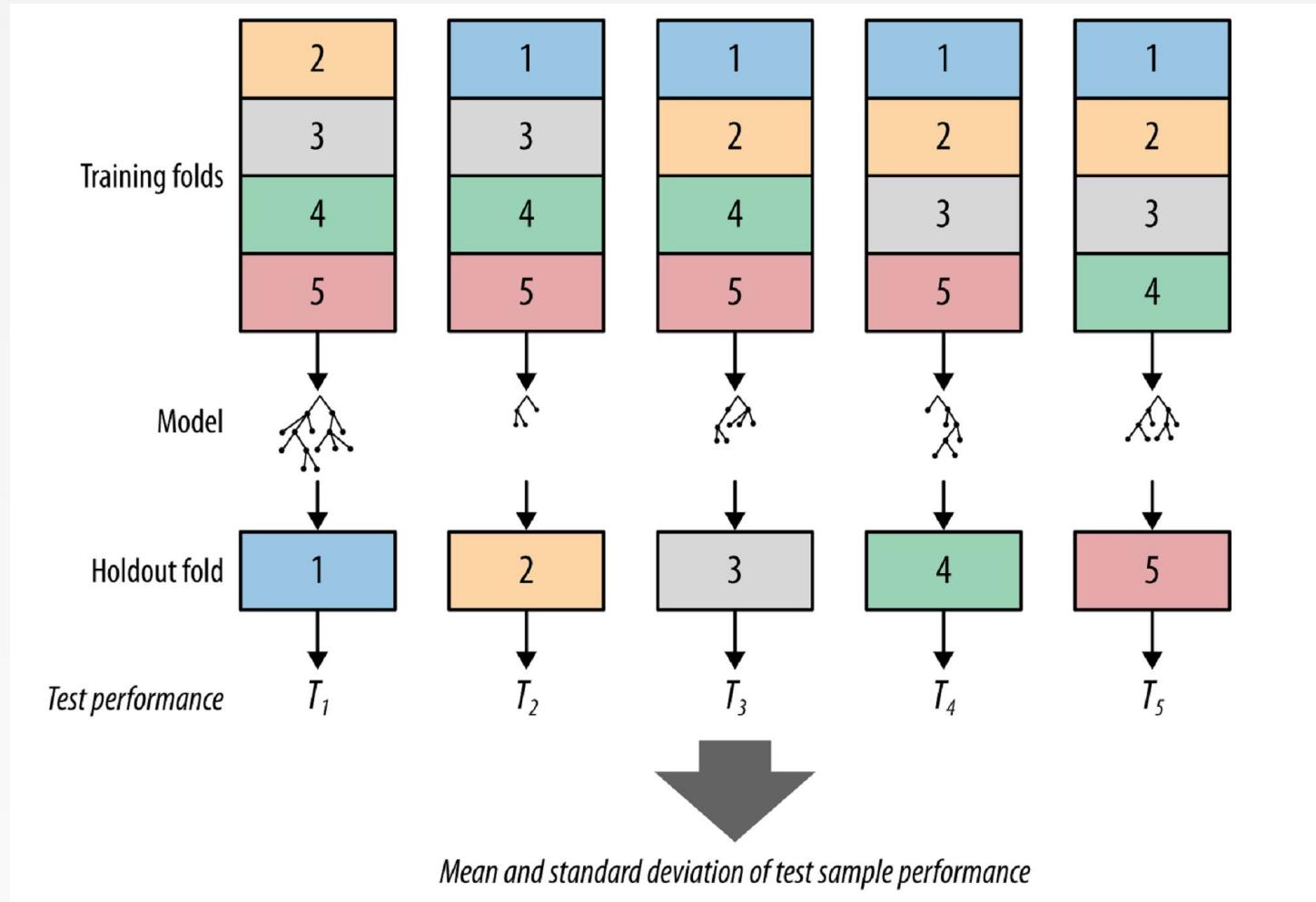
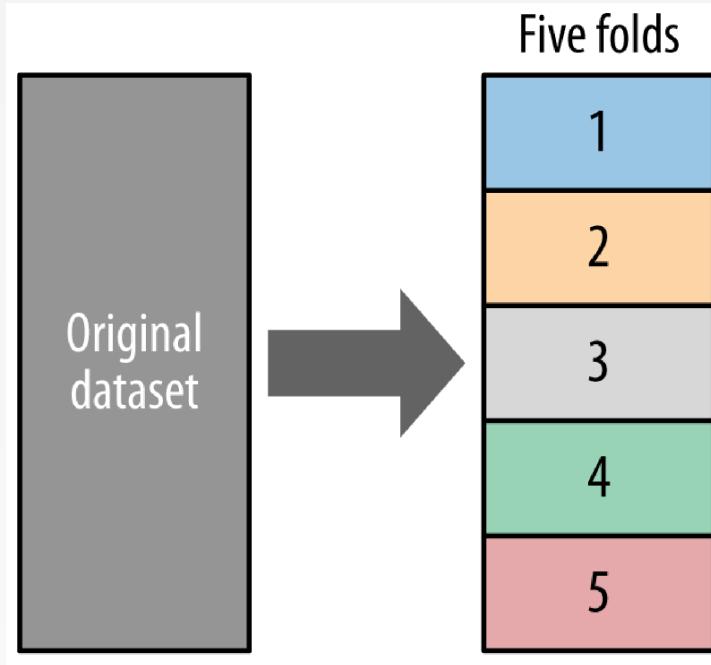
Machine Learning Workflow



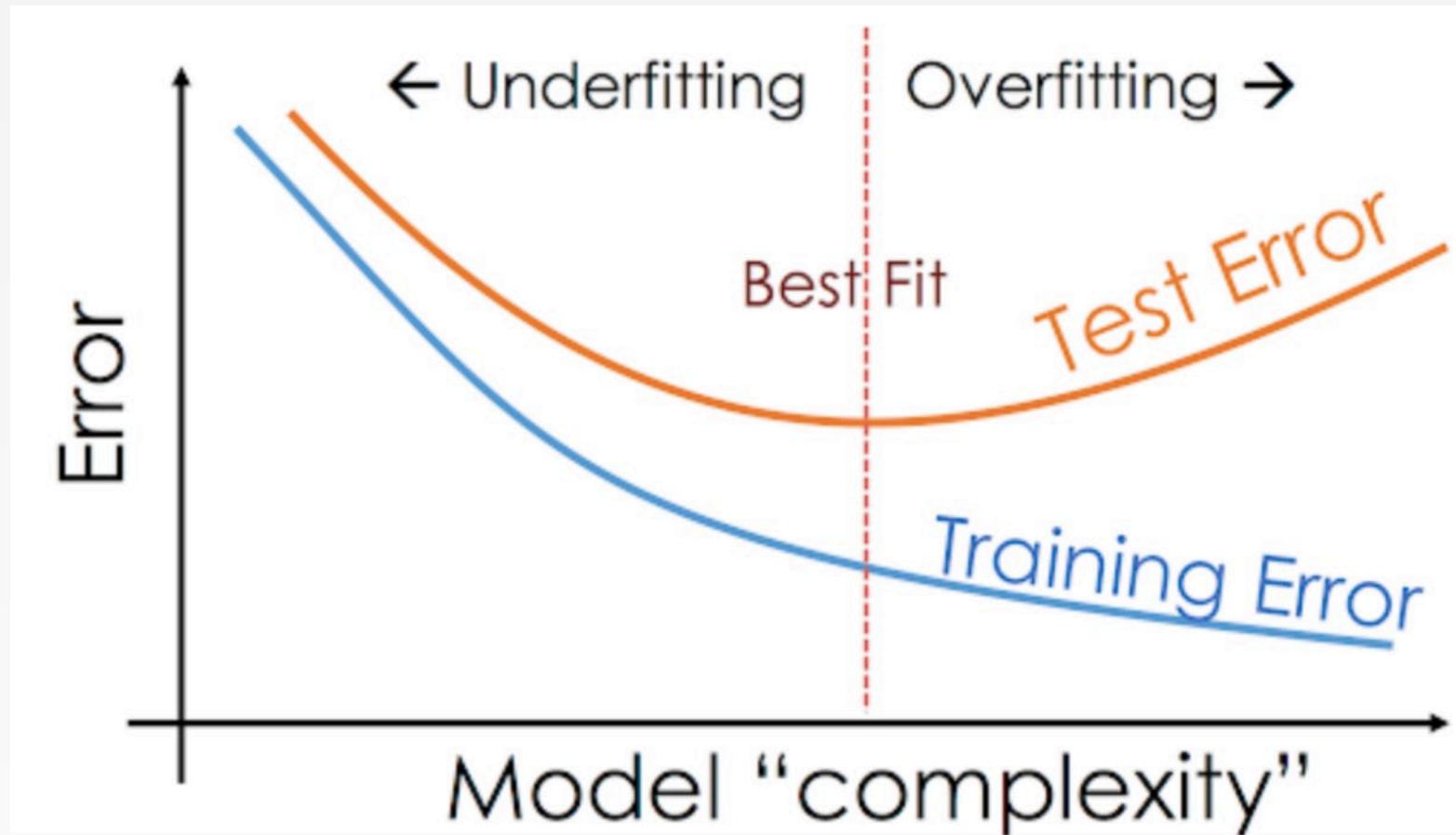
Gradient Descent



N-fold Cross Validation



Overfitting



Regularization

- Goal: optimize some combination of fit and simplicity
 - Penalize the magnitude of coefficients of features
 - Minimize the error between predicted and actual examples
- Ridge Regression:
 - L2-norm: adds penalty equivalent to square of the magnitude of coefficients
- Lasso Regression:
 - L1-norm: adds penalty equivalent to absolute value of the magnitude of coefficients



Topics

- Models:
 - KNN
 - Regressions
 - Linear; Logistic; Polynomial; Ridge; LASSO
 - Decision Trees
 - SVM
- Naïve Bayes
- Hidden Markov Model
- Ensemble Models

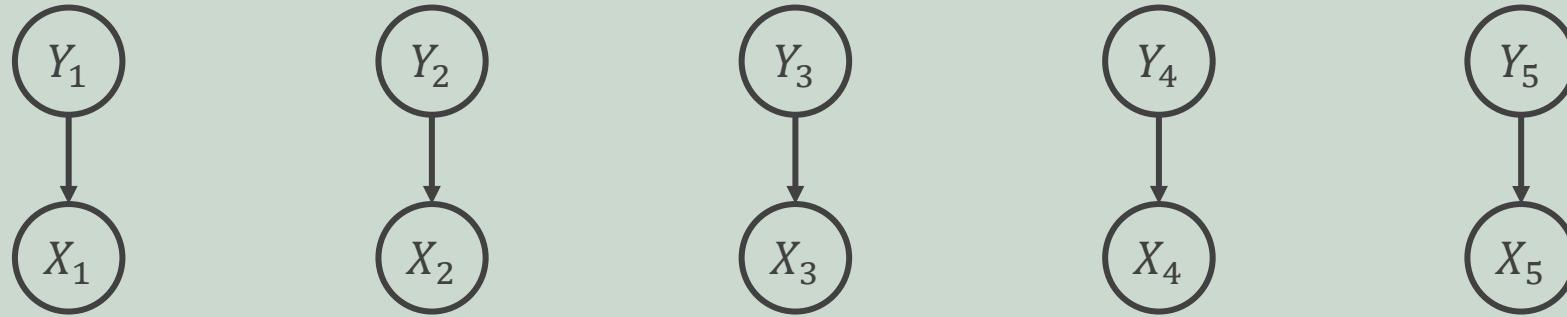


DT Comparison

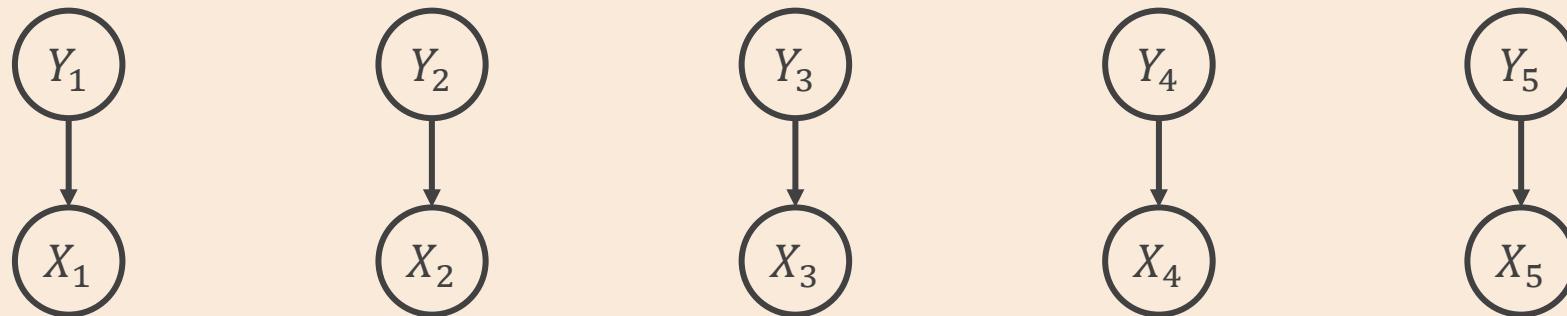
<i>Features</i>	<i>ID3</i>	<i>C4.5</i>	<i>CART</i>
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression



HMM



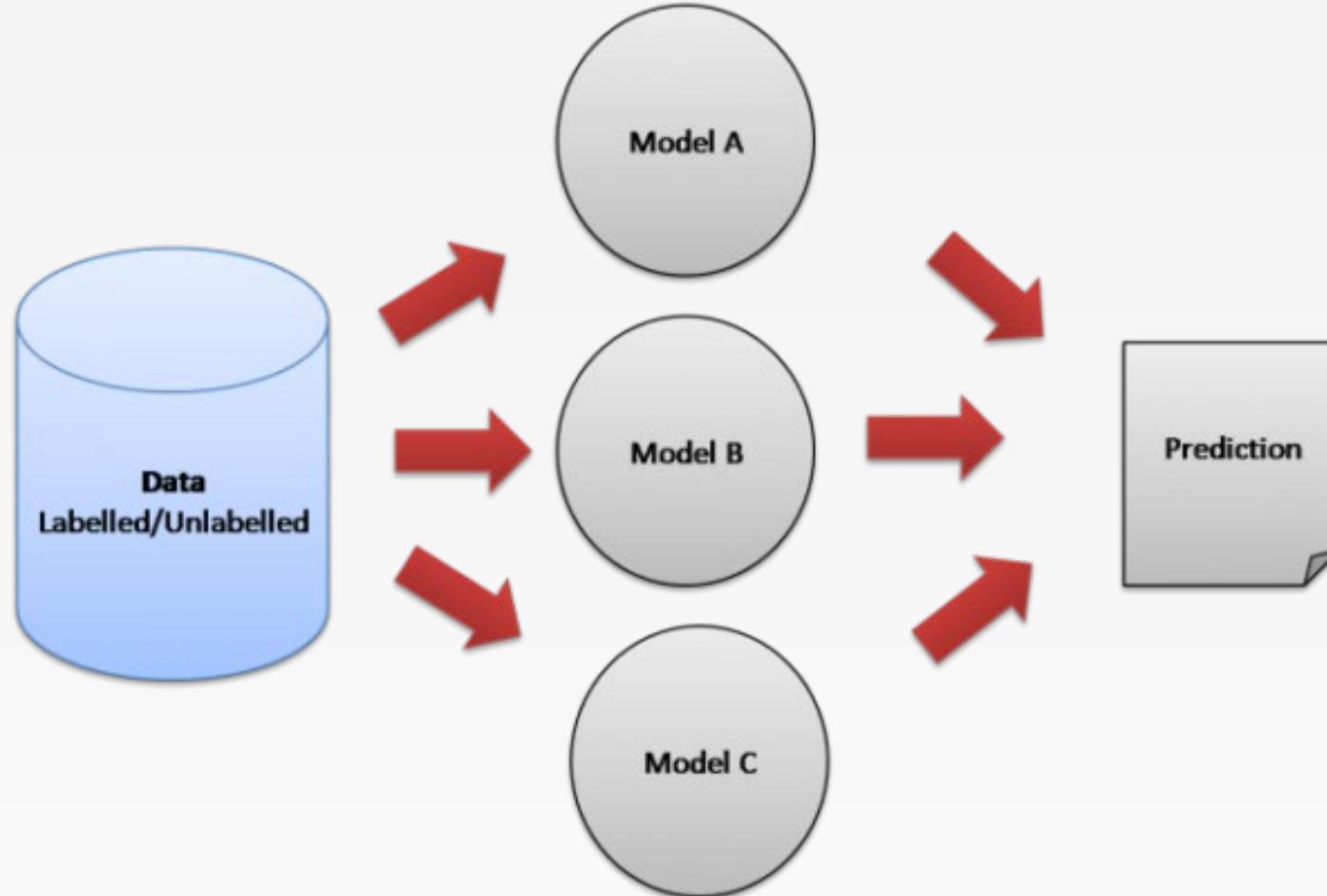
Naïve Bayes: $P(X, Y) = \prod_{t=1}^T P(X_t | Y_t)p(Y_t)$



HMM: $P(X, Y | Y_0) = \prod_{t=1}^T P(X_t | Y_t)p(Y_t | Y_{t-1})$



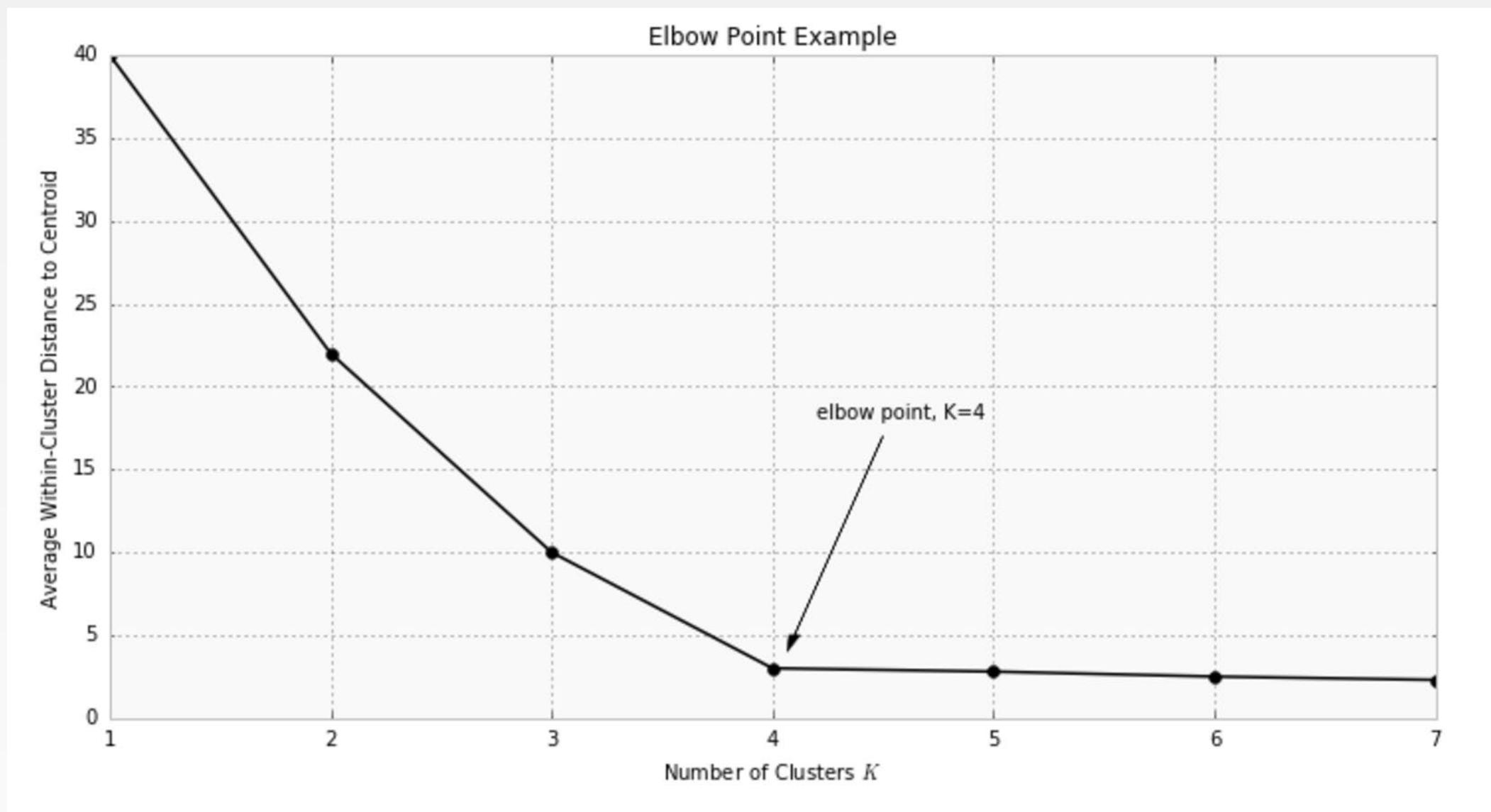
Wisdom of the Crowd



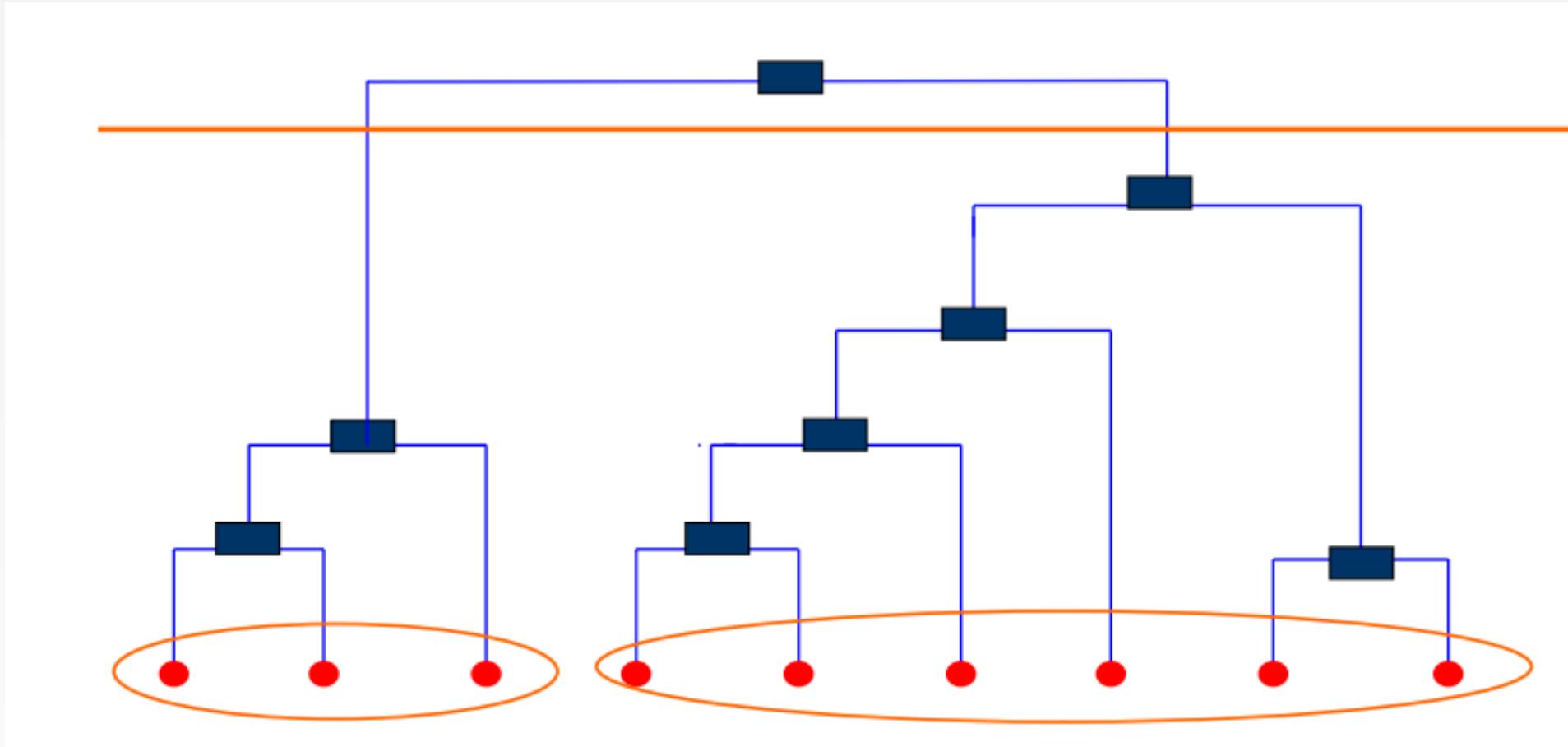
Topics

- Models:
 - Clustering
 - PCA



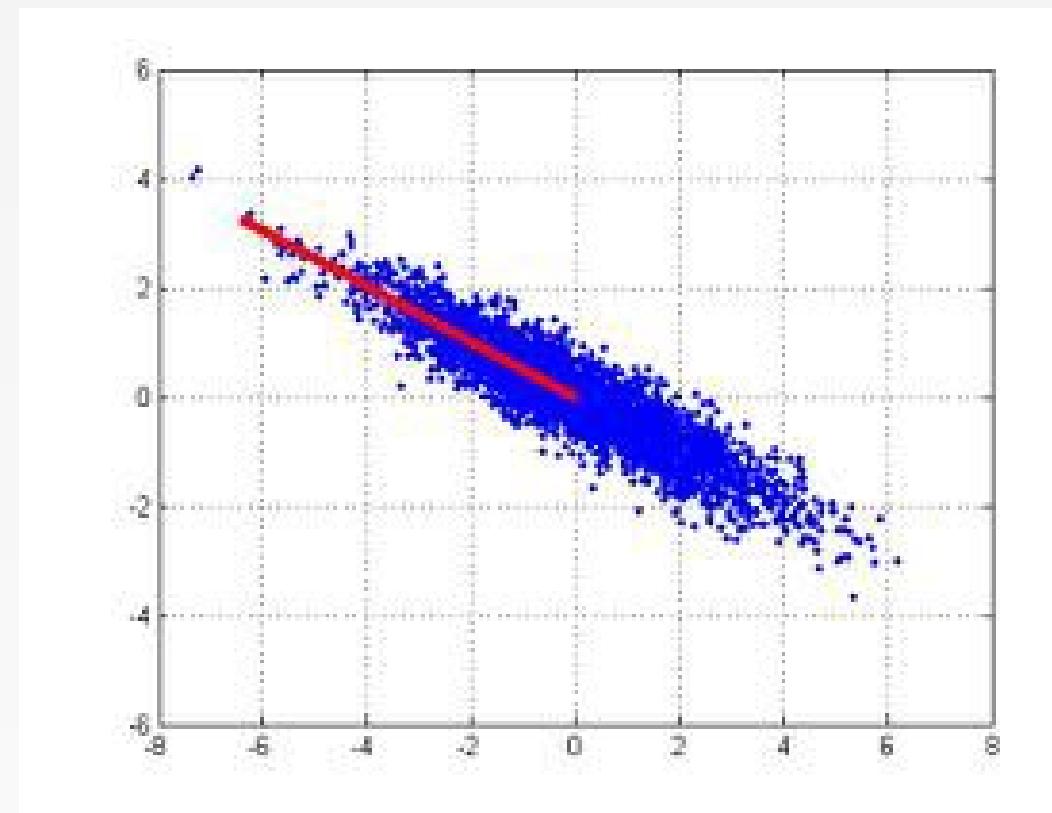


Dendrogram



PCA

First find the direction of maximum variance, labeled “Component 1”



Along this direction:

- Features are most correlated with each other
- Contains the most of the information

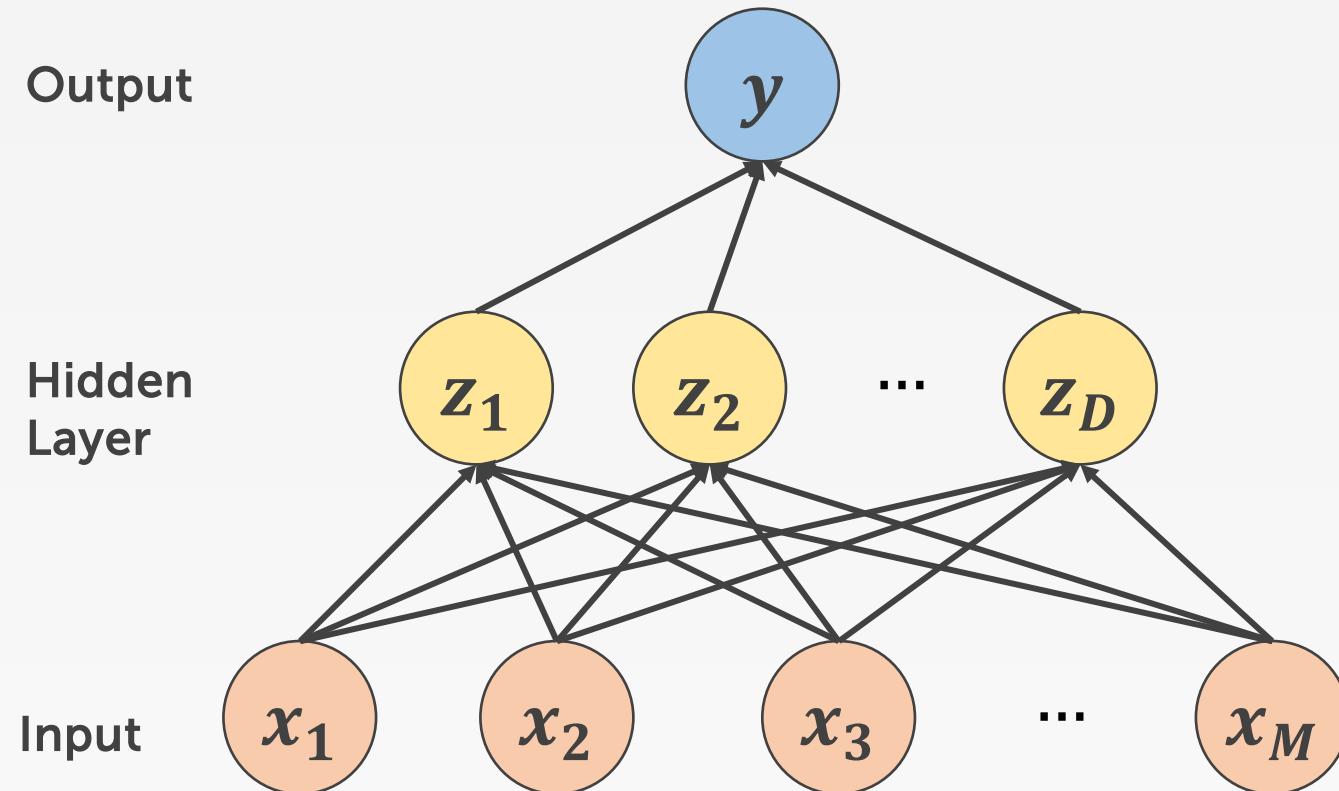


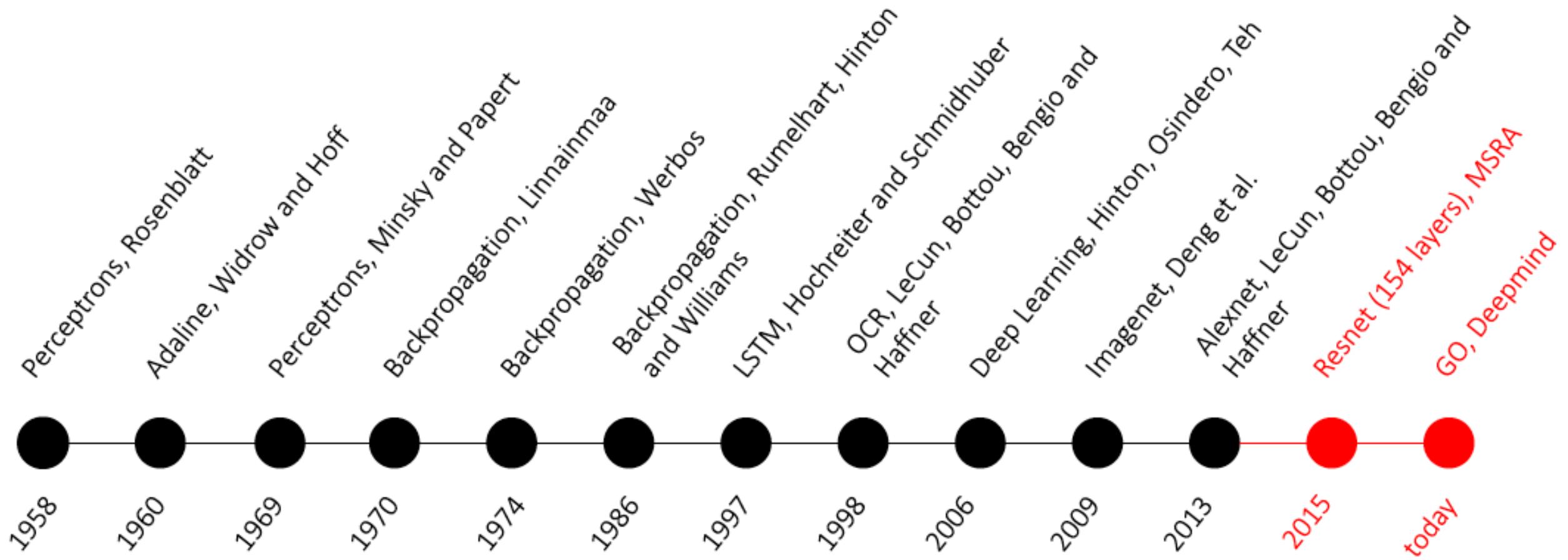
Topics

- Deep Learning Introduction
 - Neural Network
 - Backpropagation
 - Basic NN architectures
- Reinforcement Learning
 - Basic concepts
 - Value iteration vs. policy iteration
 - Q-learning

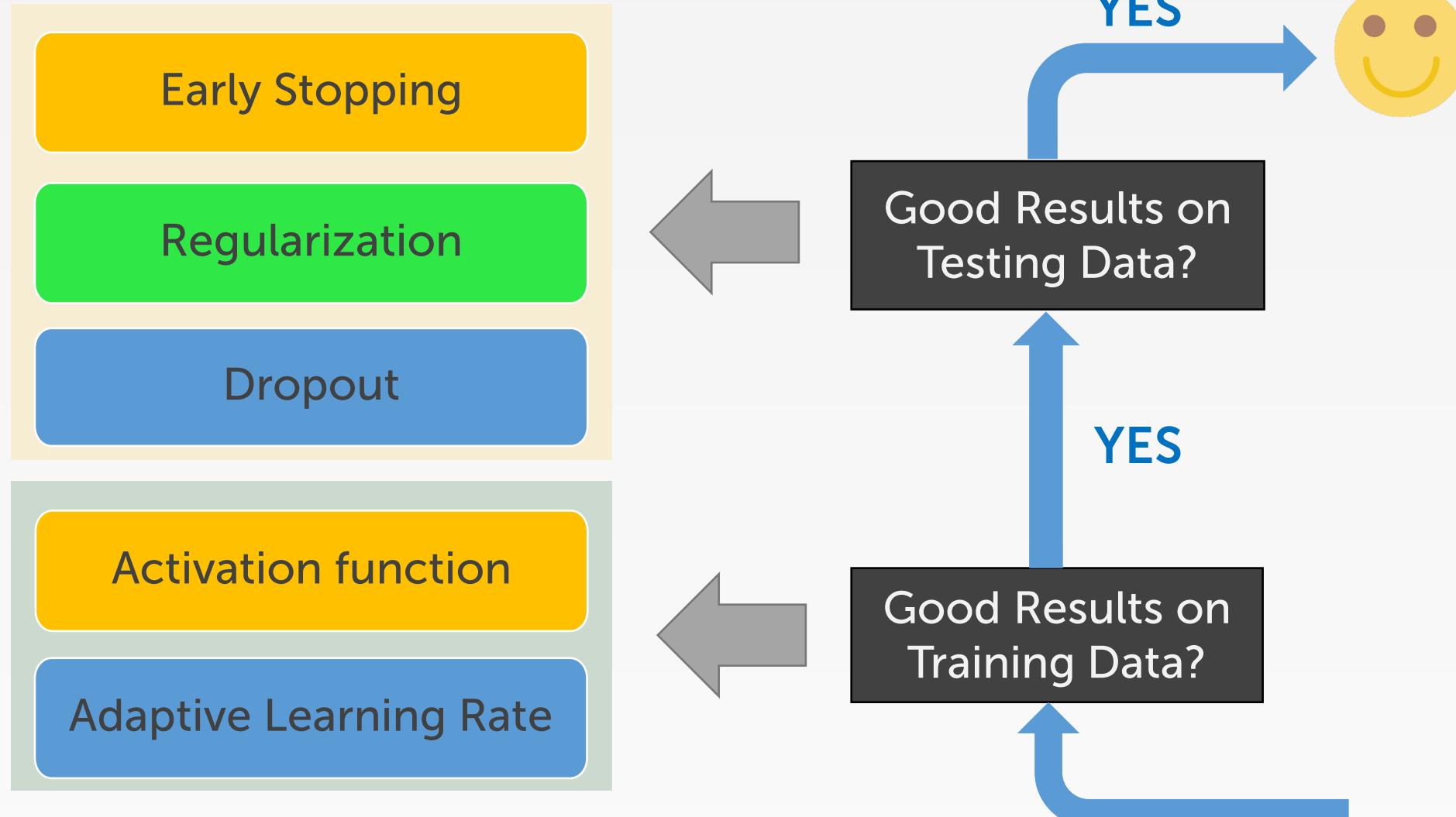


Neural Network

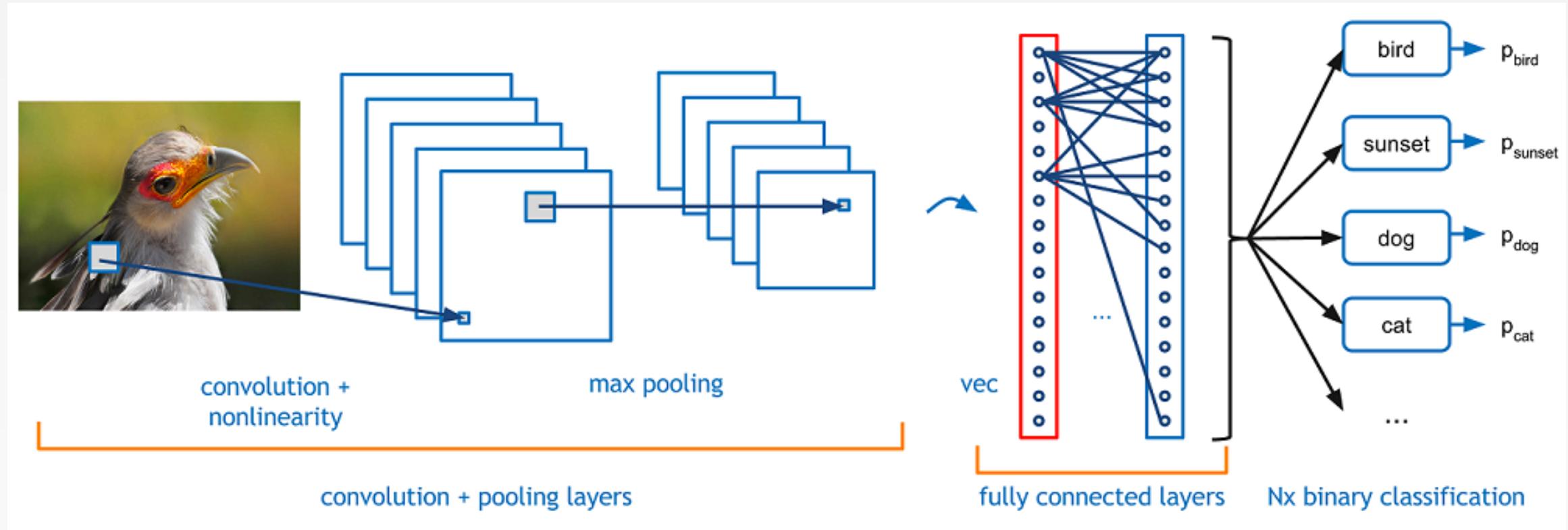




Recipe of Deep Learning



CNN Architecture



Elements of RL

- Environment:
 - Physical world in which the agent operates
- State:
 - Current situation of the agent
- A policy
 - A map from state space to action space.
 - May be stochastic.
- A reward function
 - It maps each state (or, state-action pair) to a real number, called reward.
- A value function
 - Value of a state (or, state-action pair) is the total expected reward, starting from that state (or, state-action pair).

Exploration and Exploitation

- Exploration: find more information
- Exploitation: maximize the reward by exploiting already known information



Sample Questions

Sample Question 1

True or False

A classifier that attains 100% accuracy on the training set and 70% accuracy on test set is better than a classifier that attains 70% accuracy on the training set and 75% accuracy on test set.

F

Sample Question 2

True or False

Each attribute can be used for a node split in a decision tree only once.

F



Sample Question 3

T or F

- Reinforcement learning differs from supervised learning because it has a temporal structure in the learning process, whereas, in supervised learning, the prediction of a data point does not affect the data you would see in the future

T

- Value iteration is better at balancing exploration and exploitation compared with policy iteration

F
(nothing to do with exploration)



Sample Question 4

Which technique(s) would be useful for the following business problem?
“Predict whether the loan applicants are likely to default”

- A. Linear Regression
- B. Decision Tree
- C. Unsupervised learning models
- D. Logistic Regression



Sample Question 5

Which of the following statement is true for k-NN classifiers?

- A. k-NN requires an explicit training step
- B. The classification accuracy is better with larger values of k
- C. k-NN is efficient in handling large-scale training data
- D. A small value of k might lead to the overfitting issue

n o such
→ direct
relation-
ship



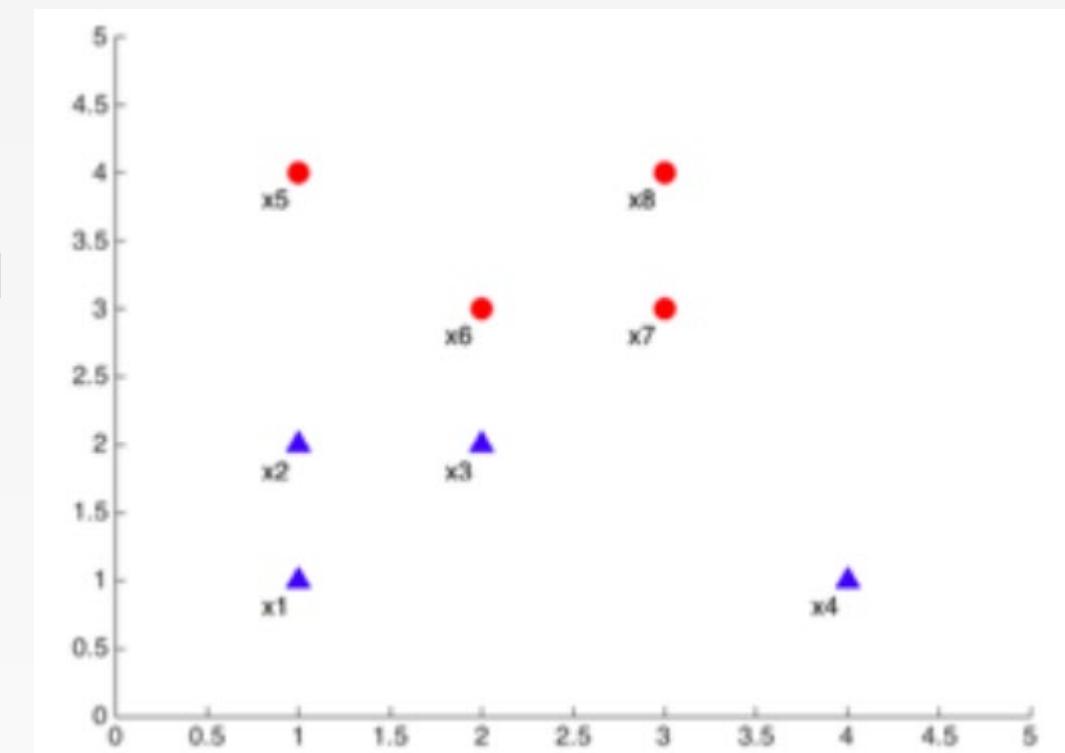
Sample Question 6

In the linearly separable case, which of the following may happen to the decision boundary of a SVM classifier if one of the training samples is removed.

\rightarrow if support vector

- A. Shifts toward the point removed
- B. Shifts away from the point removed
- C. Does not change

\rightarrow if not support vector



Sample Question 7

We are given n data points, x_1, \dots, x_n and asked to cluster them using K-means. If you choose the value for k to optimize the objective function, how many clusters will be used?

- A. 1
- B. 2
- C. n
- D. $\log(n)$

Sample Question 8

Consider a regression model where we want to predict variable y from a single feature x . Consider two possible models to be estimated:

$$y = \omega_0 + \omega_1 x \quad (\text{B.1})$$

$$y = \omega_0 + \omega_1 x + \omega_2 x^2 \quad (\text{B.2})$$

1. Which model is more likely to fit the training data better? Explain your reasoning.
2. Which model is more likely to fit the test data better? Explain your reasoning.

B.2]

→ CANNOT be said
if few obs, B.2 might overfit
and obs, might be better

$\omega_2 = 0$
then same
hence //



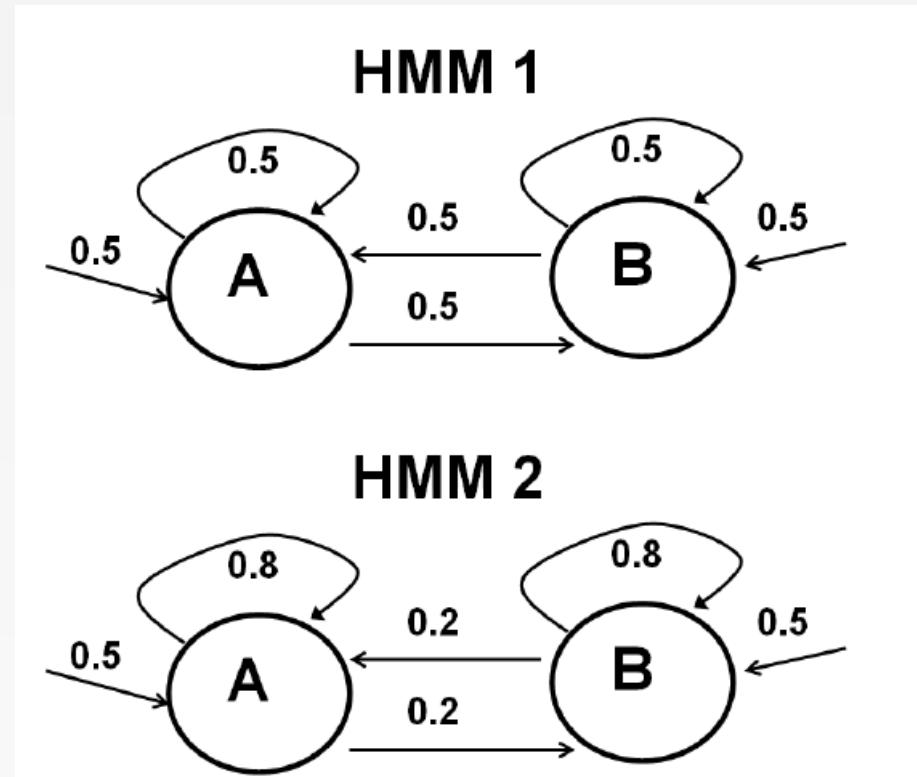
Sample Question 9

The figure above presents two HMMs. States are represented by circles and transitions by edges.

In both, emissions are deterministic and listed inside the states.

Transition probabilities and starting probabilities are listed next to the relevant edges.

For example, in HMM 1 we have a probability of 0.5 to start with the state that emits A and a probability of 0.5 to transition to the state that emits B if we are now in the state that emits A.



Sample Question 9 (cont.)

Let P_1 be: $P_1 = P(O_{100} = A, O_{101} = B, O_{102} = A, O_{103} = B)$ for HMM1 and let P_2 be: $P_2 = P(O_{100} = A, O_{101} = B, O_{102} = A, O_{103} = B)$ for HMM2. Choose the correct answer from the choices below and briefly explain.

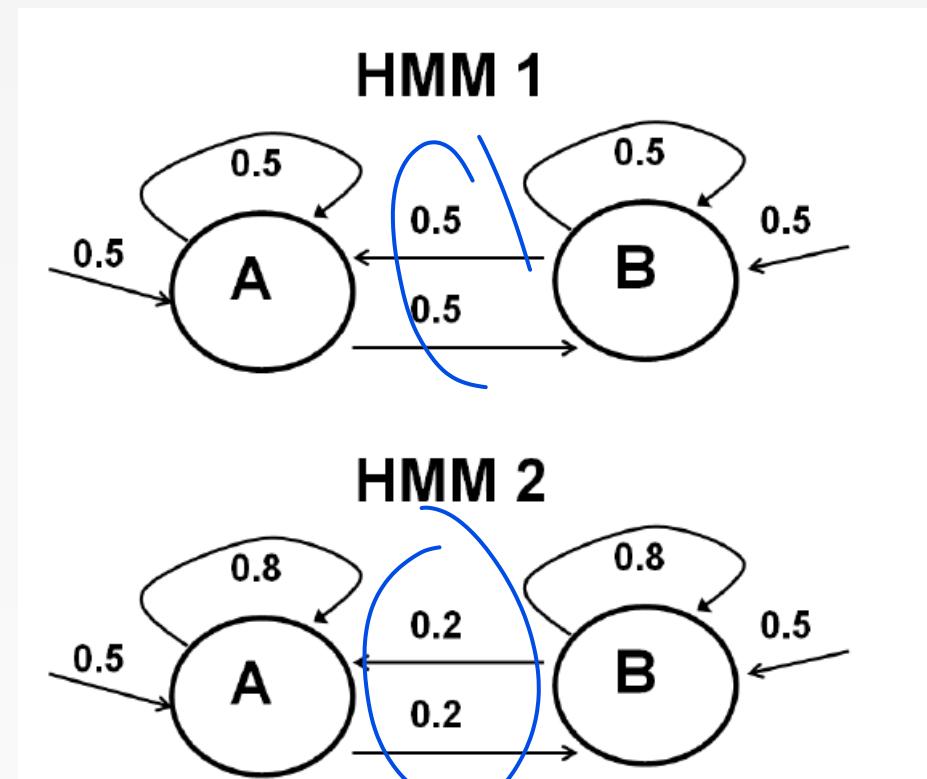
1. $P_1 > P_2$

$$0.5^4 > 0.5 \cdot 0.2^2$$

2. $P_1 < P_2$

3. $P_1 = P_2$

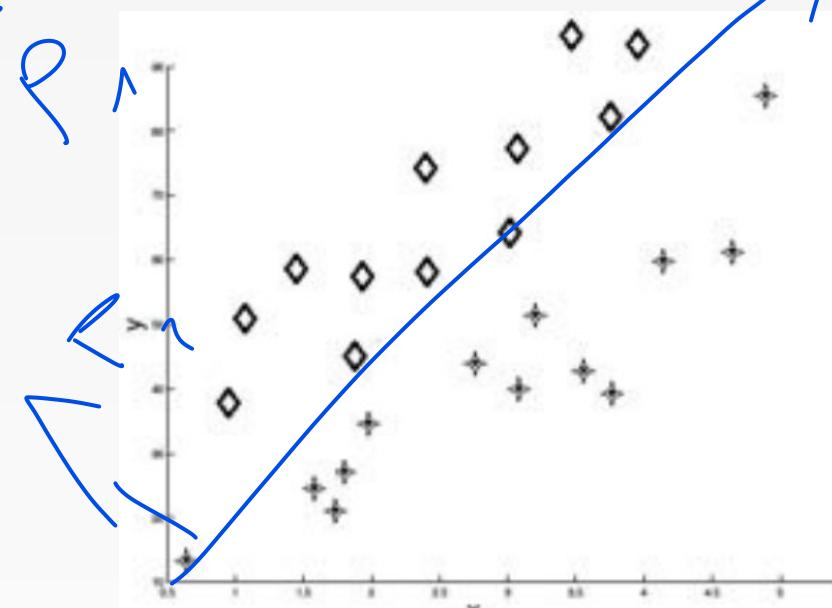
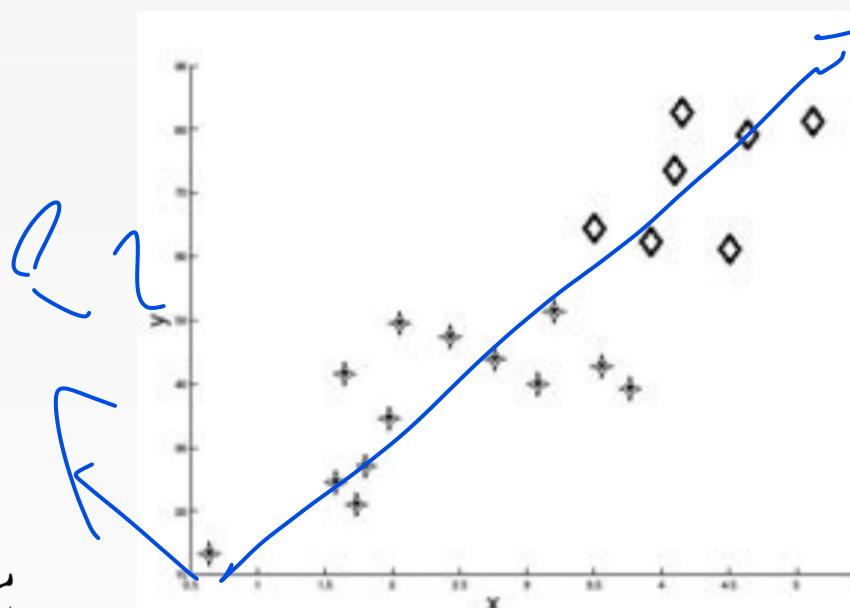
4. Impossible to tell



Sample Question 10

In the following plots, a train set of data points X belonging to two classes on R^2 are given, where the original features are the coordinates (x, y) . For each, answer the following questions:

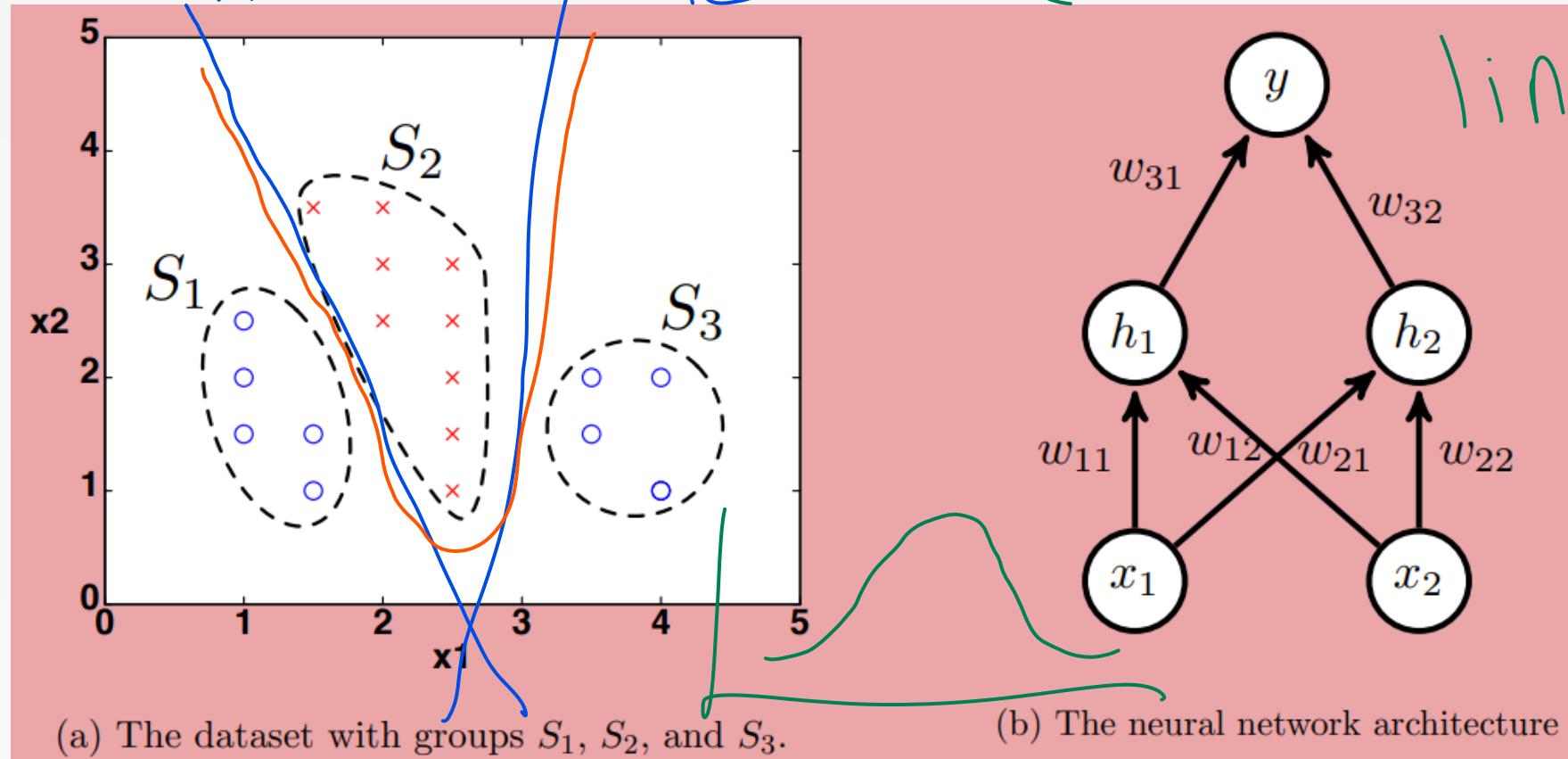
1. Draw all the principal components.
2. Can we correctly classify this dataset by using a threshold function after projecting onto one of the principal components?



(PCFs
are vectors
so there is
a direction)

Sample Question 11

Can the neural network in Figure (b) correctly classify the dataset given in Figure (a)?



Sample Question 12

You are given a data set of 10,000 students with their sex, height, and hair color. You are trying to build a classifier to predict the sex of a student, so you randomly split the data into a training set and a test set.

- Sex $\in \{\text{male, female}\}$
- Height $\in [0,300]$ centimeters
- Hair $\in \{\text{brown, black, blond, red, green}\}$
- 3240 men in the data set
- 6760 women in the data set

Under the assumption necessary for NB, answer the following questions

1. T or F: As height is a continuous valued variable, Naive Bayes is not appropriate since it cannot handle continuous valued variables
F \rightarrow Gaussian NB
2. T or F: $P(\text{height} | \text{sex, hair}) = P(\text{height} | \text{sex})$.
T



Good Luck!