



University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 1 – Introduction

Ying Weng
2024 Autumn

COMP3055 Machine Learning

Ying WENG

Teaching for 6 weeks

Office: PMB-341

Office hour: Monday 14:00 – 16:00

Email: ying.weng@nottingham.edu.cn

Zheng LU

Teaching for 6 weeks (Module Convenor)

Office: PMB-426

Office hour: TBC

Email: zheng.lu@nottingham.edu.cn

- Lecture
- Lab
 - Tuesday 16:00-18:00, IAMET-406
 - **For the 1st Teaching Week, there is no lab on Tuesday 24th Sep**

Lecture Schedule

- Plan ahead
- All material on Moodle
- Regular update

Lab Schedule

- Lab sessions will provide you hands on experiences what you learned during the lecture sessions
- Lab materials will be published on Moodle

Summary of Content

- Provide you with an introduction to **machine learning**, pattern **recognition**, and **data mining** techniques
- Enable you to consider both systems which are able to develop their own rules from trial-and-error experience to solve problems
- Find patterns in data without any supervision. Data mining techniques will make generation of new knowledge possible, including big datasets. This is now fashionably termed 'Big Data' science
- Cover a range of topics including: machine learning foundations; pattern recognition foundations; applications of machine learning; data mining techniques and evaluating hypotheses; artificial neural networks; deep learning
- You'll spend **six hours** each week (on average) **in lectures and computer practice** for this module

Assessments

❖ Exam: 70%

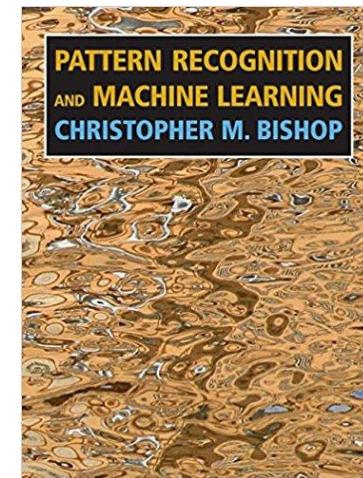
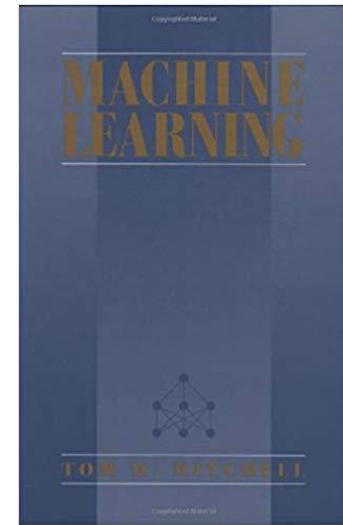
- 2 hour written examination

❖ Coursework: 30%

- 1 piece of individual programming assignment

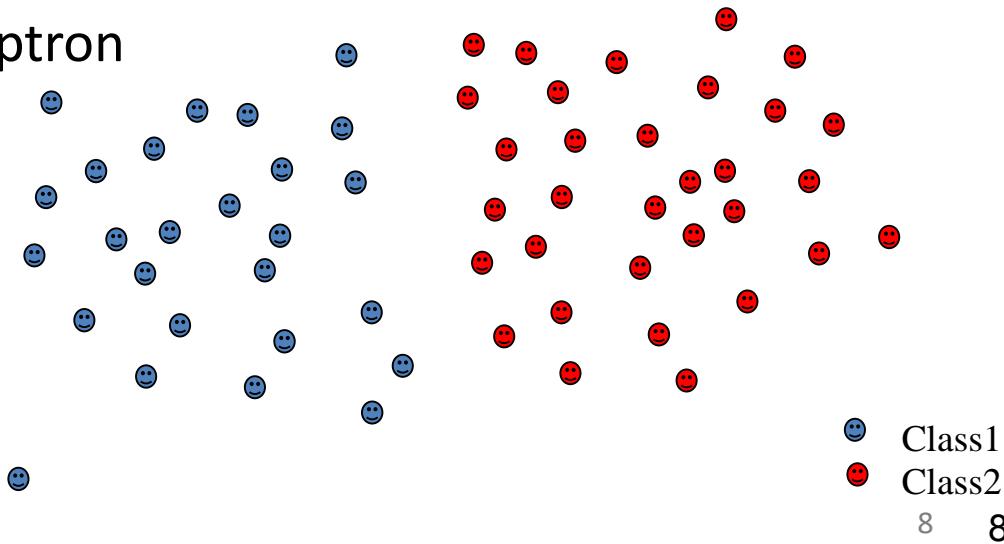
Textbooks

- Tom M. Mitchell, Machine Learning, McGraw-Hill
- Chris Bishop, Pattern Recognition and Machine Learning, Springer
- Some of recent technical papers



Topics

- Design learning systems
- Data collection
- Learning theory practice
- Instance based learning, KNN etc.
- Bayesian learning
- Data processing representation
- Unsupervised learning, clustering, K-Means etc.
- Decision tree, random forest
- Perceptron, multilayer perceptron
- Support vector machine
- Deep learning, CNN etc.
- ...
- (to be continued)



How to Get 70+

- **Studying...**
 - You are recommended to study the relevant notes before attending the lecture or lab.
 - Review as soon as possible to maximize retention.
- **Practice...**
 - **Do the lab exercise yourself** and repeat the practice for better learning.
 - If you get help on the labs, don't just blindly accept it, but try to understand what each part of the code is doing.
 - Do the Math in the lecture for better understanding.
- **Assignments...**
 - Start work on the assignment **when they are released**, and come up with a good plan to finish it.
 - It will take **longer** than you expect to fix problems in your program, so make sure you have plenty of time to complete.



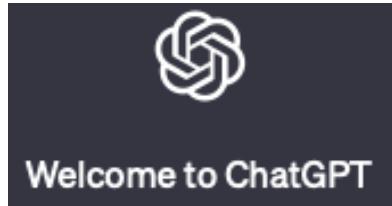
Do the thing right, Do the right thing



A Few Quotes

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)
- “Machine learning is the next Internet”
(Tony Tether, Director, DARPA)
- Machine learning is the hot new thing”
(John Hennessy, President, Stanford)
- “Machine learning is going to result in a real revolution” (Greg Papadopoulos, CTO, Sun)
- “Machine learning is today’s discontinuity”
(Jerry Yang, CEO, Yahoo)

ChatGPT

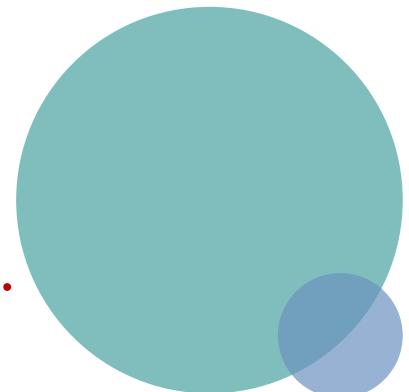


- ❖ An artificial intelligence (AI) chatbot developed by OpenAI and released in November 2022
- ❖ Built on top of GPT-4 foundational large language model (LLM)
- ❖ Fine-tuned to transfer learning

incredible, powerful, helpful

ChatGPT has rapidly immersed itself into our lives.
I think everyone has read articles related to it.

ChatGPT is a milestone in the era of large models.



What is Machine Learning?

- It is very hard to write programs that solve problems like recognizing a face.
 - We don't know what program to write because we don't know how our brain does it.
 - Even if we had a good idea about how to do it, the program might be horrendously complicated.
- Instead of writing a program by hand, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
 - The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
 - If we do it right, the program works for new cases as well as the ones we trained it on.

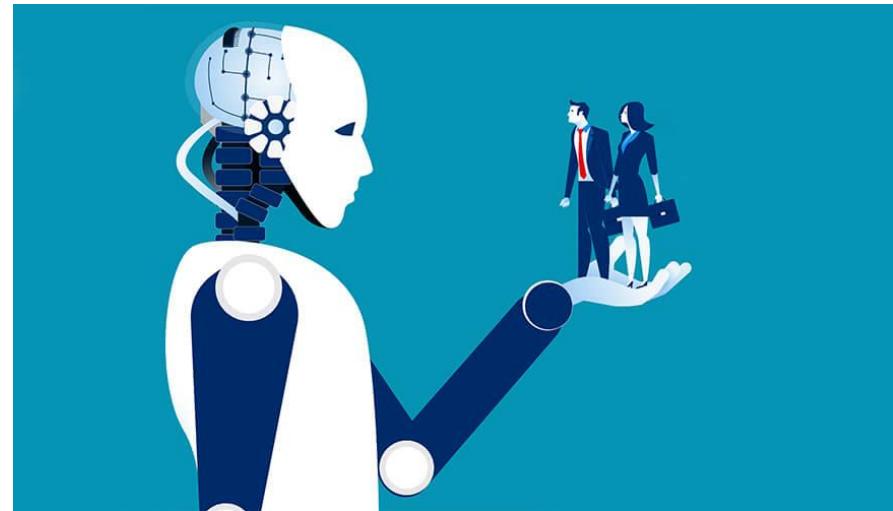
What is Machine Learning?

- Automating automation
- Getting computers to program themselves
- Writing software is the bottleneck
- Let the data do the work instead!

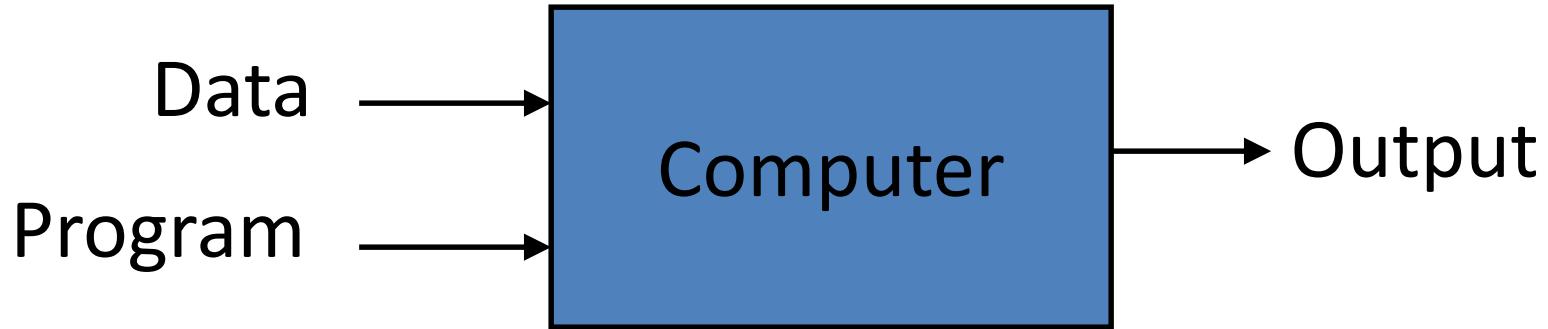


What is Machine Learning?

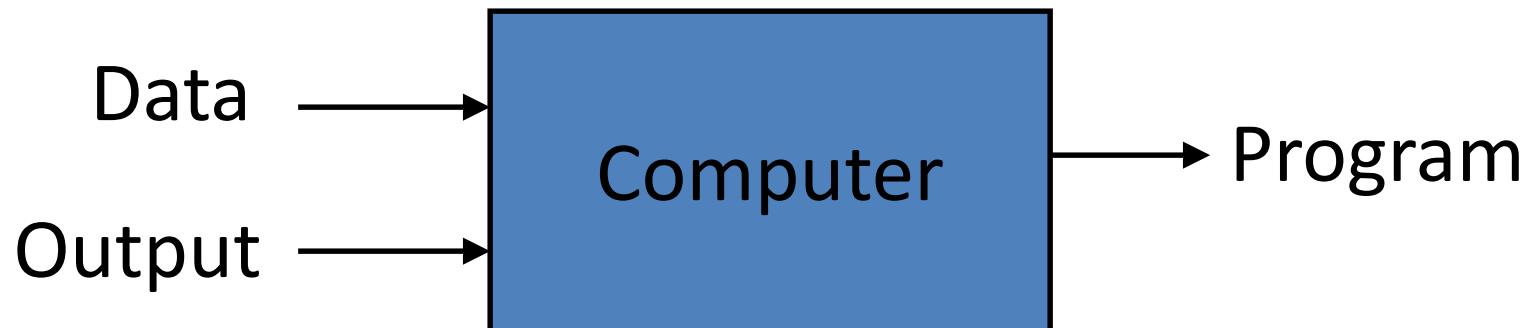
- Machine Learning is making great strides
 - Large, good data sets
 - Computer power
 - Progress in algorithms
- Many interesting applications
 - Commercial
 - Scientific



Traditional Programming



Machine Learning



Magic?

No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs



Task: Handwriting Recognition

It is very hard to say what makes a 2

0 0 0 1 1 (1 1 1 2

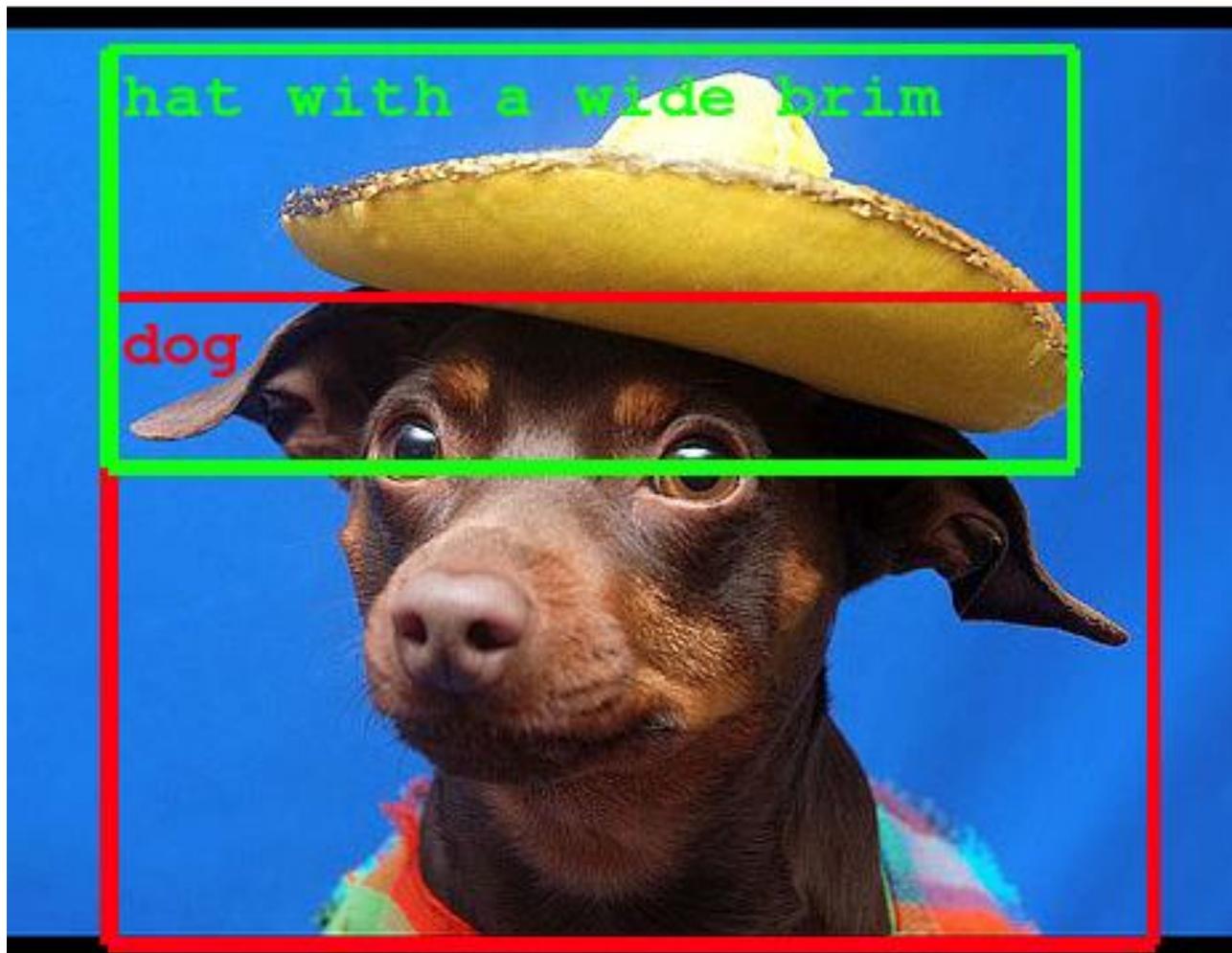
2 2 2 2 2 2 2 3 3 3

3 4 4 4 4 4 5 5 5 5

6 6 7 7 7 7 7 8 8 8

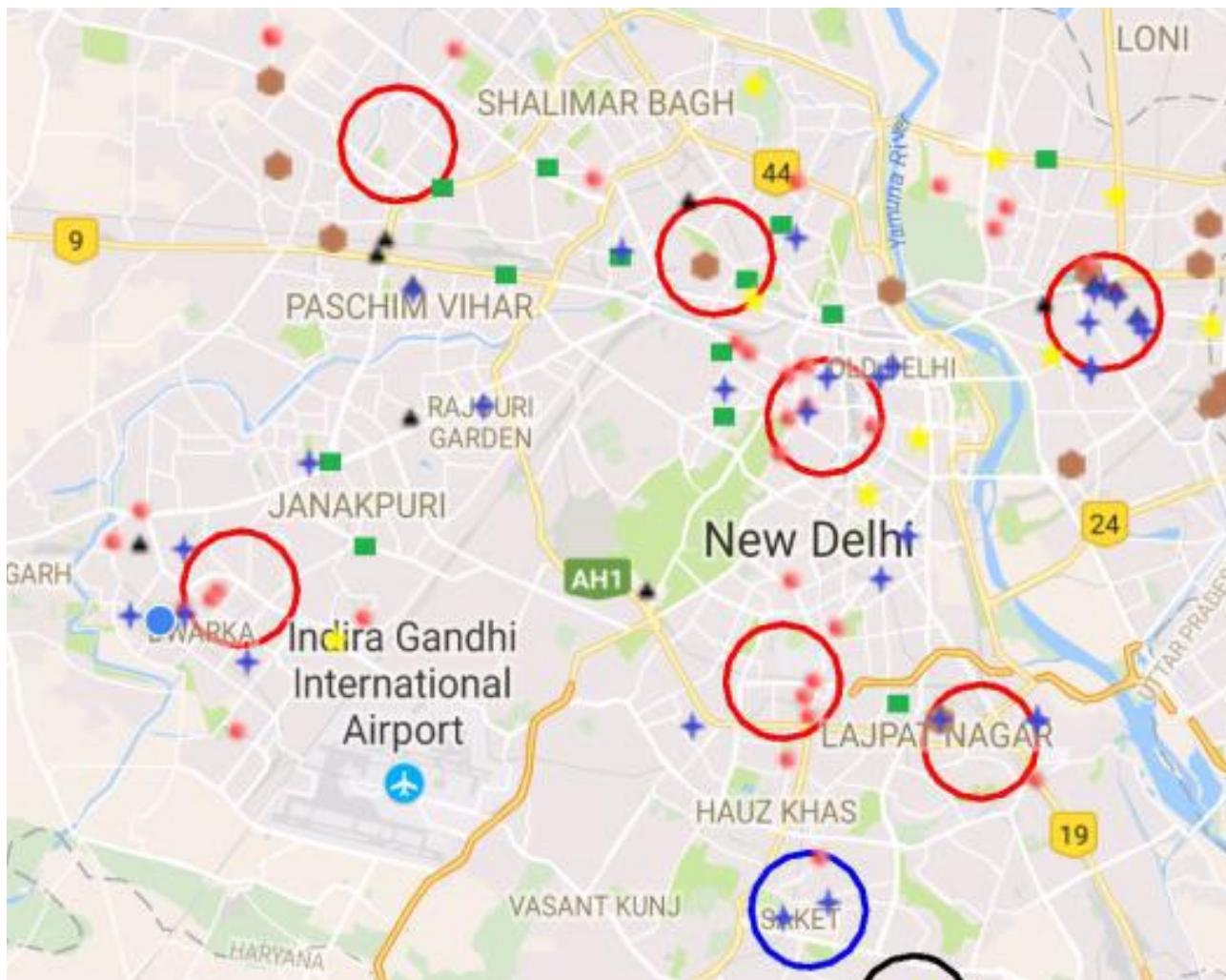
8 8 8 7 9 4 9 9 9

Task: Classification Object recognition



Task: Clustering

Crime prediction using k-means clustering

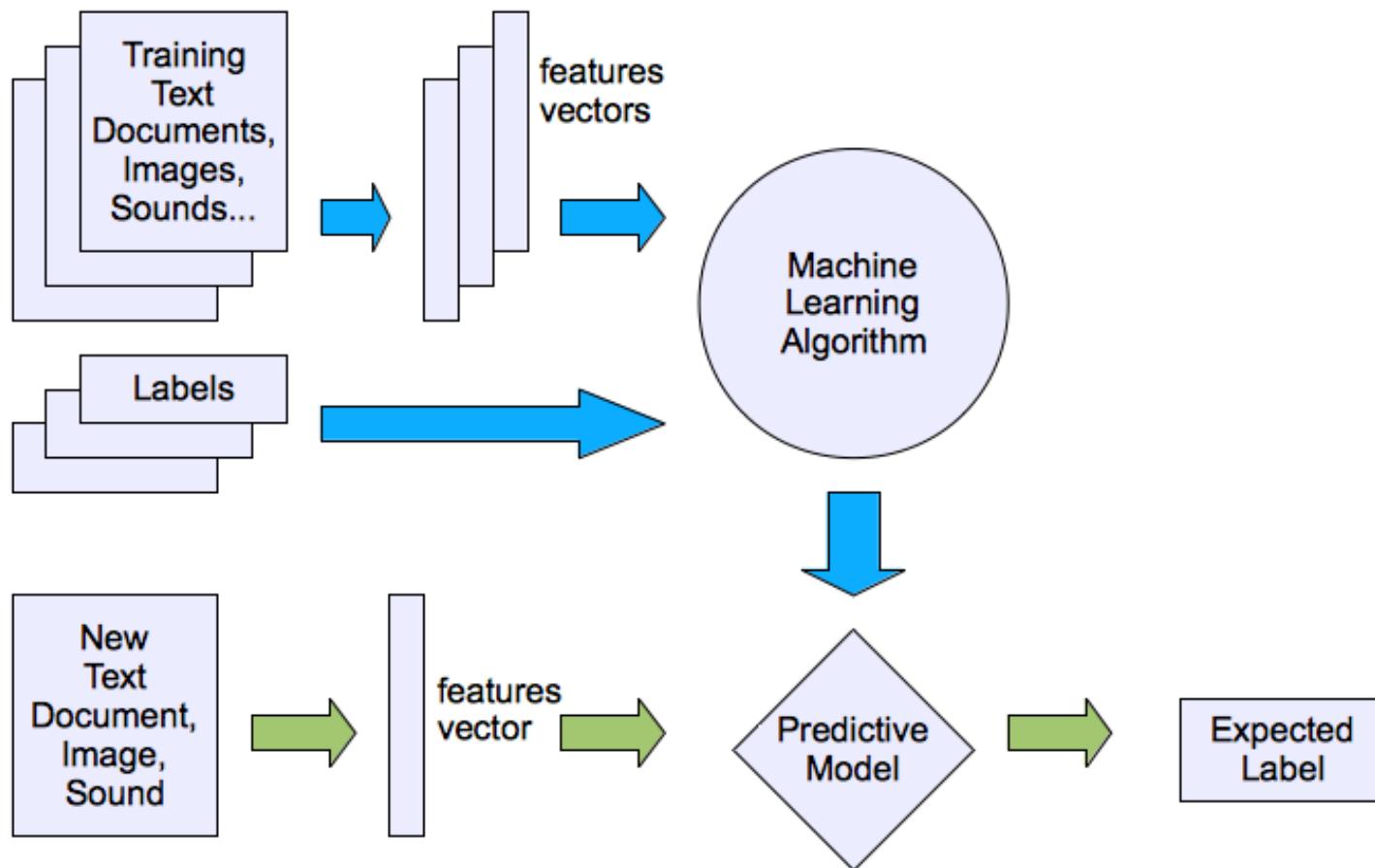


More Sample Applications

- Web search
- Healthcare
- Robotics
- Information extraction
- Finance
- E-commerce
- Computational biology
- Space exploration
- Social networks
- [Your favorite area]

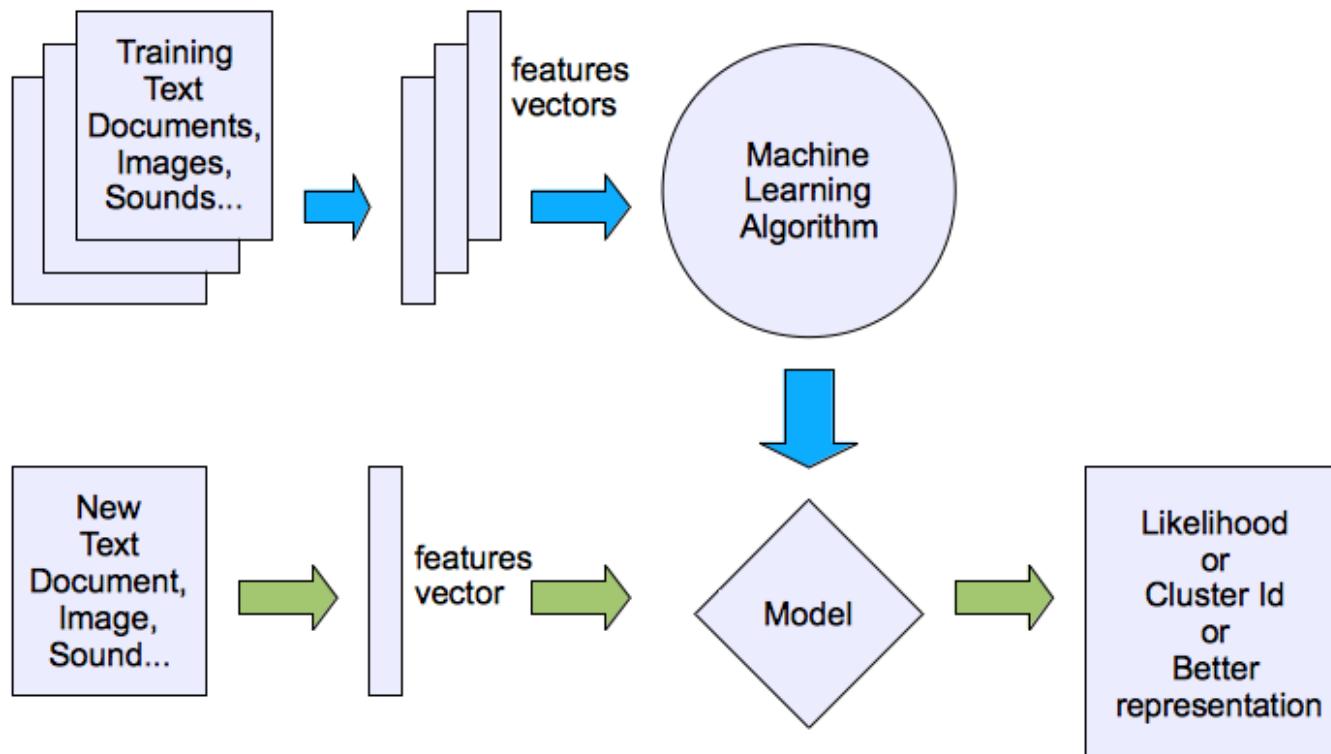
Types of Learning

- Supervised learning



Types of Learning

- Unsupervised learning



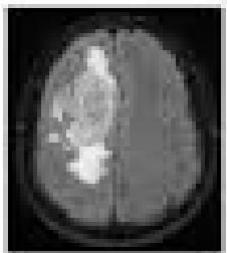
Types of Learning

- **Supervised (inductive) learning**
 - Training data includes desired outputs
- **Unsupervised learning**
 - Training data does not include desired outputs
 - This is the new frontier of machine learning because most big datasets do not come with labels
- **Semi-supervised learning**
 - Training data includes a few desired outputs

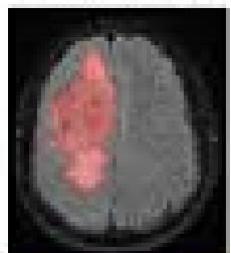
Semi-supervised Learning

Motivation: labeled data are expensive

unlabeled data are free



unlabeled



labeled

Medical field:

- The process of labeling medical images requires **specialized expertise** and is often **time-consuming and expensive**.
- This is due to the **complexity** of medical images, the need for **high-quality annotations**, and **the limited availability** of trained medical professionals.

Framework

- **Programming languages**
 - Python
 - C++
 - ...
- **Many libraries**
 - PyTorch
 - TensorFlow
 - ...

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Entropy
- etc.

Issues

- Many machine learning/AI projects fail
(Gartner claims 85%)



- Ethics



Reasons for Failure

- Asking the wrong question
- Trying to solve the wrong problem
- Not having enough data
- Not having the right data
- Hiring the wrong people
- Using the wrong tools
- Not having the right model
- Not having the right yardstick



Any Questions?





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 2 – Design a Learning System

Ying Weng
2024 Autumn

Human Learning vs Machine Learning

- Cognitive science vs computational science
 - Animal learning vs machine learning
 - Don't fly like birds
 - Many ML models are based on human types of learning
- Evolution vs machine learning
 - Adaptation vs learning

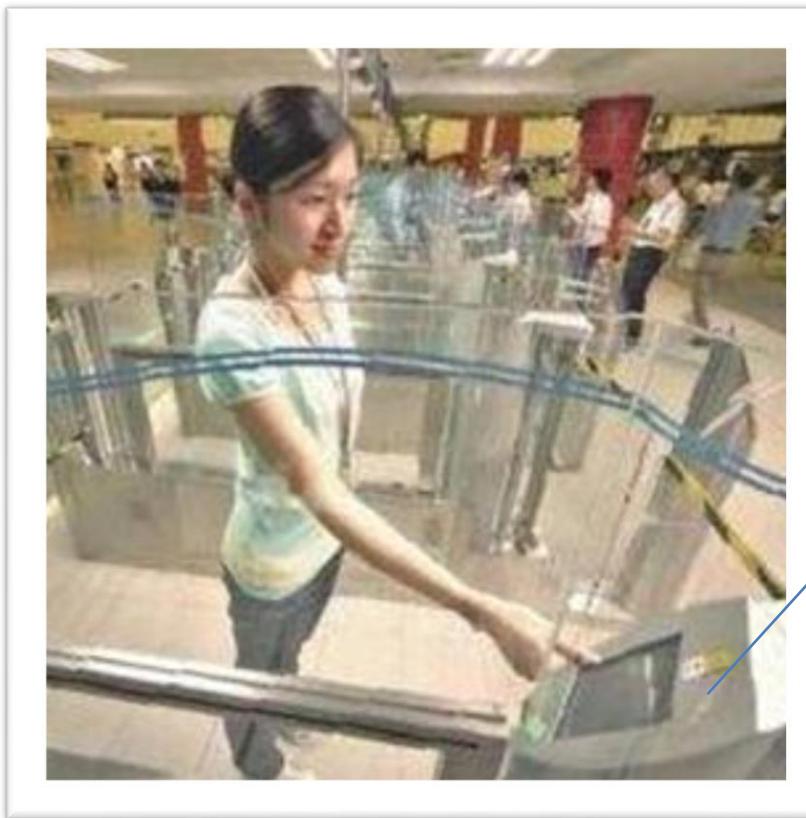
Motivating Problems

Face Recognition



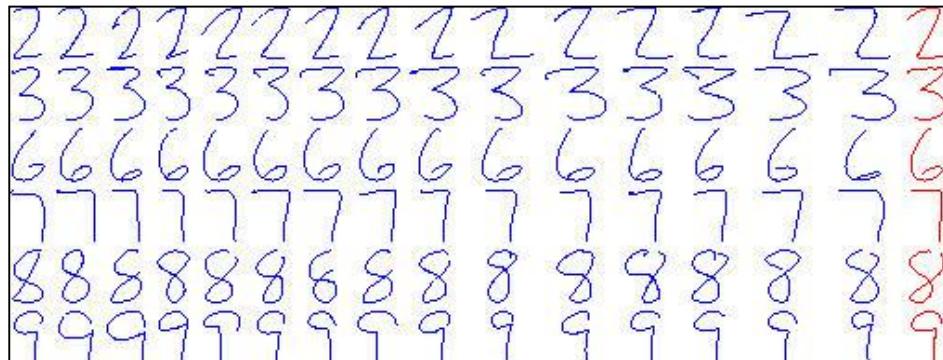
Motivating Problems

Fingerprint Recognition (e.g., border control)



Motivating Problems

Handwritten Character Recognition



0	1	2	3	5	6	8	9
C	G	I	J	L	M	N	O
P	S	U	V	W	Z	N	O
0	1	2	3	M	6	8	9
C	G	I	J	L	M	N	O
P	S	U	V	W	Z	Z	S

あいうえお
えいすせえ
かきくけえ
ないぬの
新年あめでテラ!

完全非限定脱机手写汉字的识别
完全非限定脱机手写汉字的识别
完全非限定脱机手写汉字的识别

Motivating Problems

DaVinci surgical robot by Intuitive Surgical.

St. Elizabeth Hospital is one of the local hospitals using this robot. You can see this robot in person during an open house ([website](#)).



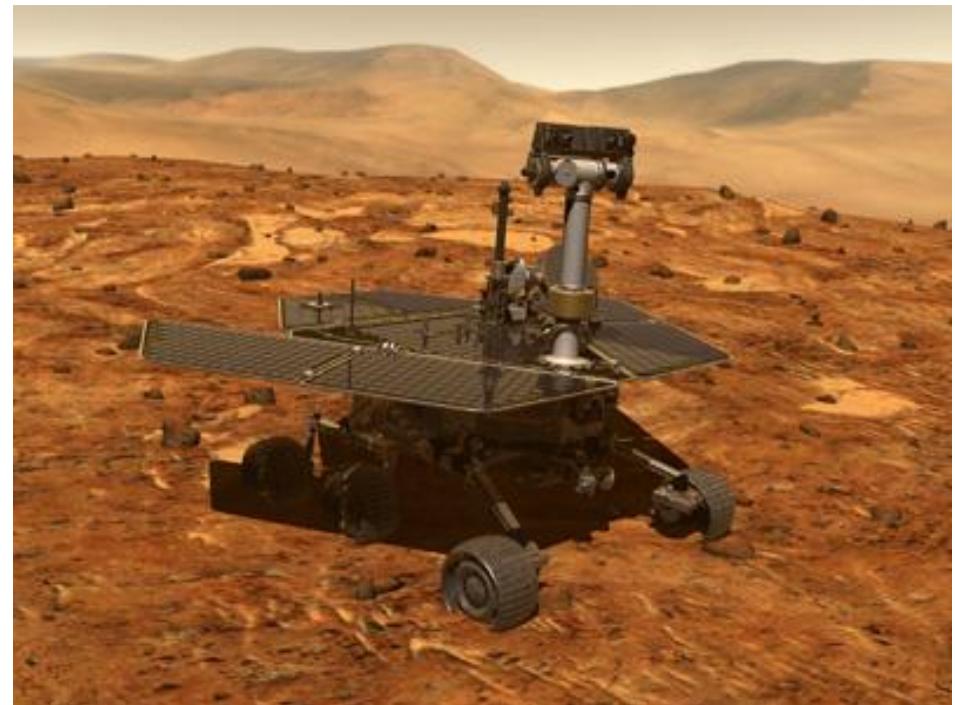
Japanese health care assistant suit
(HAL - Hybrid Assistive Limb)



Also... Mind-controlled wheelchair using NI LabVIEW

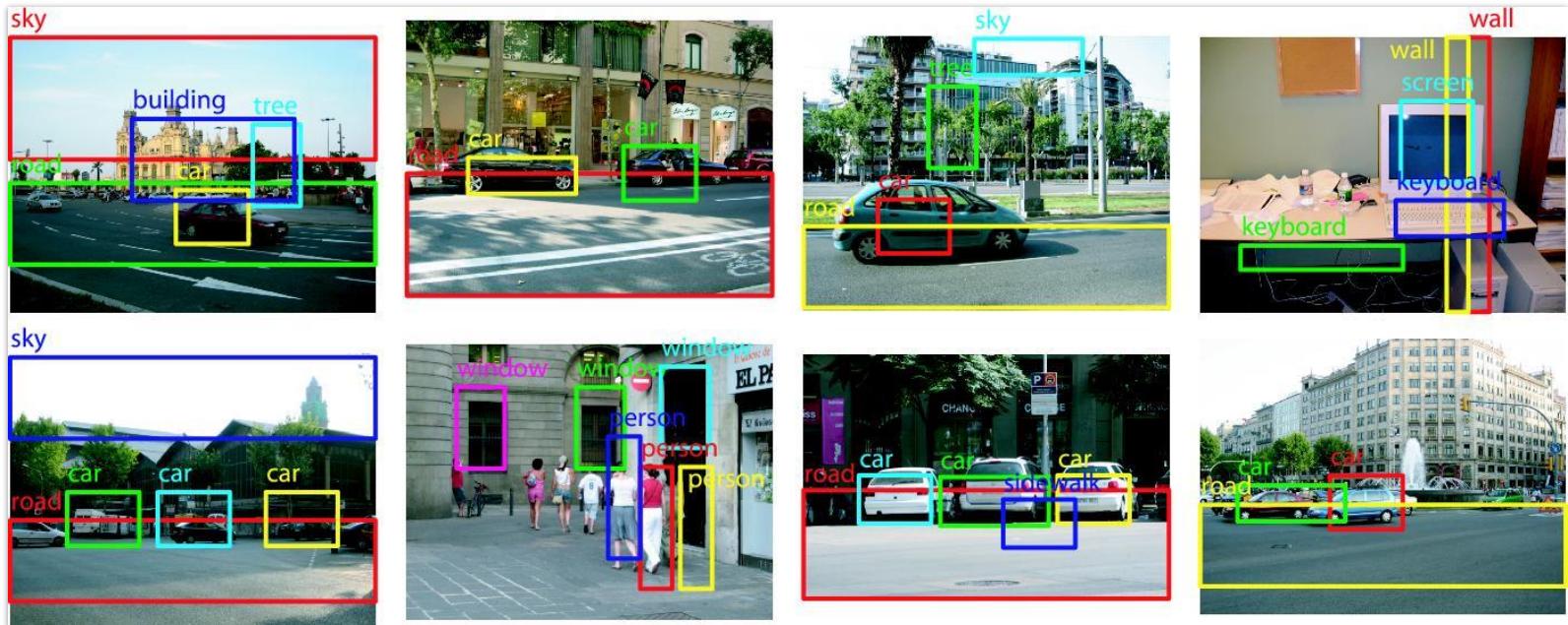
Motivating Problems

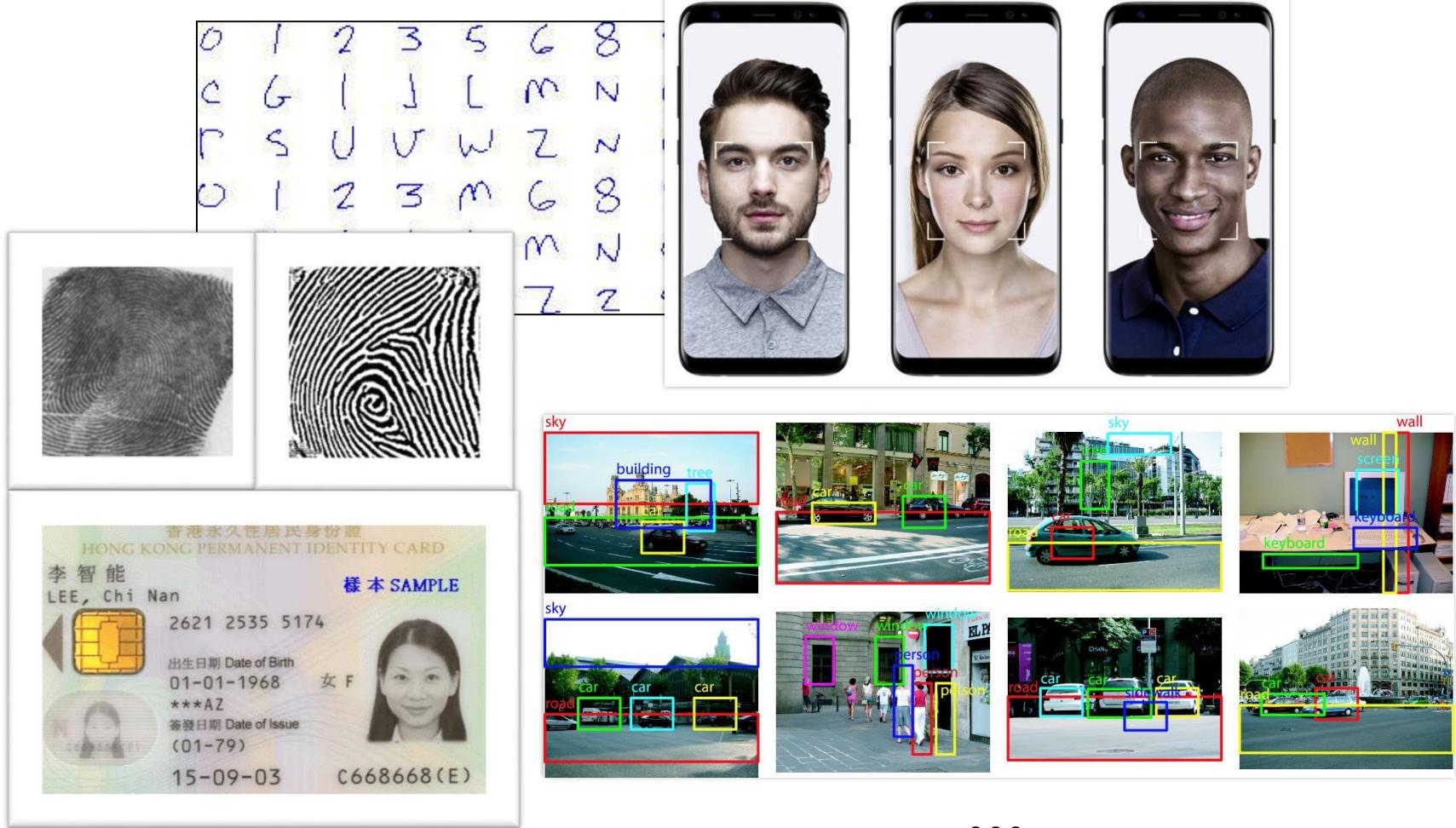
- Space Robot – Mars Autonomous navigation features with human remote control and oversight



Motivating Problems

Object Recognition





Can Machines Learn to Solve These Problems?

Definition of Learning

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

-- from Mitchell, Machine Learning, McGraw-Hill, 1997

Definition of Learning

What does this mean exactly?

For example, handwriting recognition problem

- Task T : Recognizing hand written characters
- Performance measure P : percent of characters correctly classified
- Training experience E : a database of handwritten characters with given classifications

0	1	2	3	5	6	8	9
C	G	I	J	L	M	N	O
P	S	U	V	W	Z	N	O
0	1	2	3	M	6	8	9
C	G	I	J	L	M	N	O
P	S	V	V	W	Z	Z	8

Definition of Learning

What are design issues and approaches?

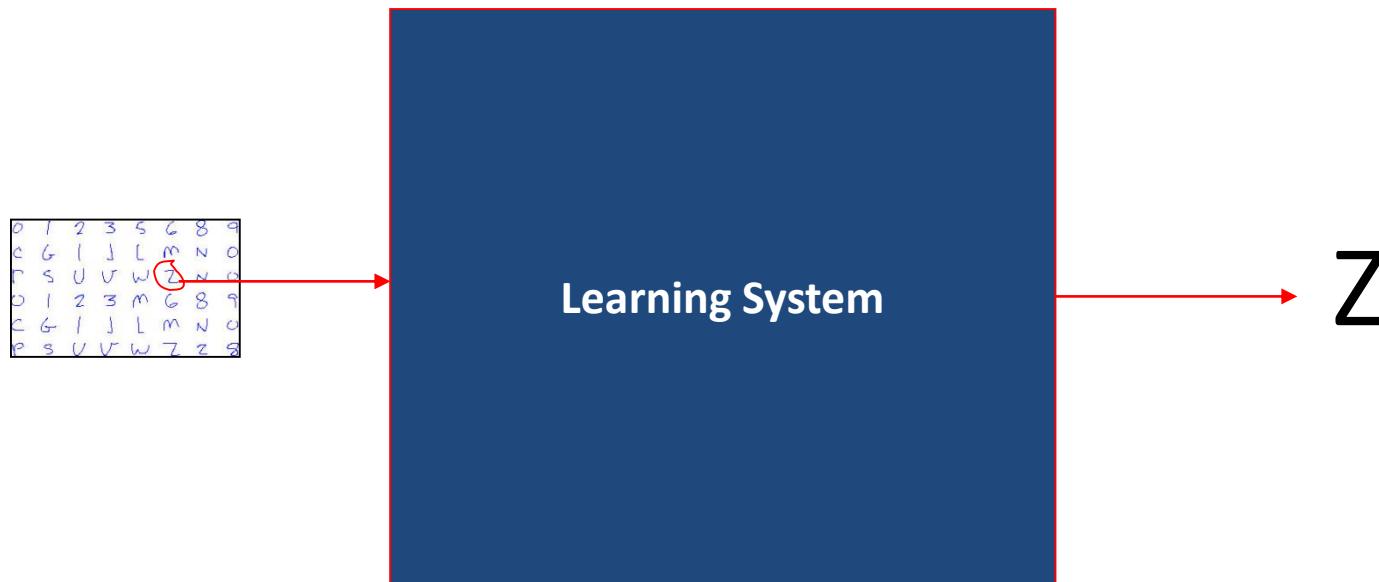
For example, handwriting recognition problem

0	1	2	3	5	6	8	9
C	G	I	J	L	M	N	O
P	S	U	V	W	Z	N	O
0	1	2	3	M	6	8	9
C	G	I	J	L	M	N	O
P	S	U	V	W	Z	Z	8

Design a Learning System

Step 0:

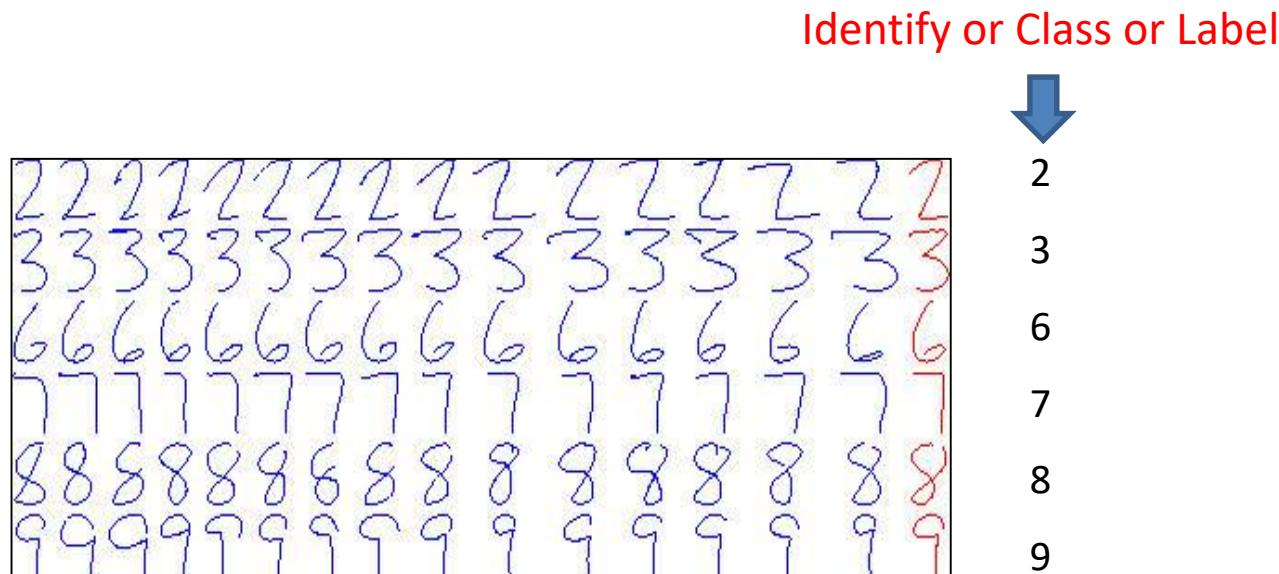
- Lets treat the learning system as a black box



Design a Learning System

Step 1: Collect Training Examples (Experience).

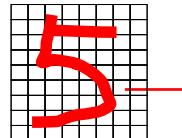
- Without examples, our system will not learn
 - so-called learning from examples



Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience / examples



$X = (1,1,0,1,1,1,1,1,1,1,0,0,0,0,1,1, \dots, 1);$ 64-d Vector

The sensor input represented by an n-d vector,
called the **feature vector**, $X = (x_1, x_2, x_3, \dots, x_n)$

Design a Learning System

Step 2: Representing Experience

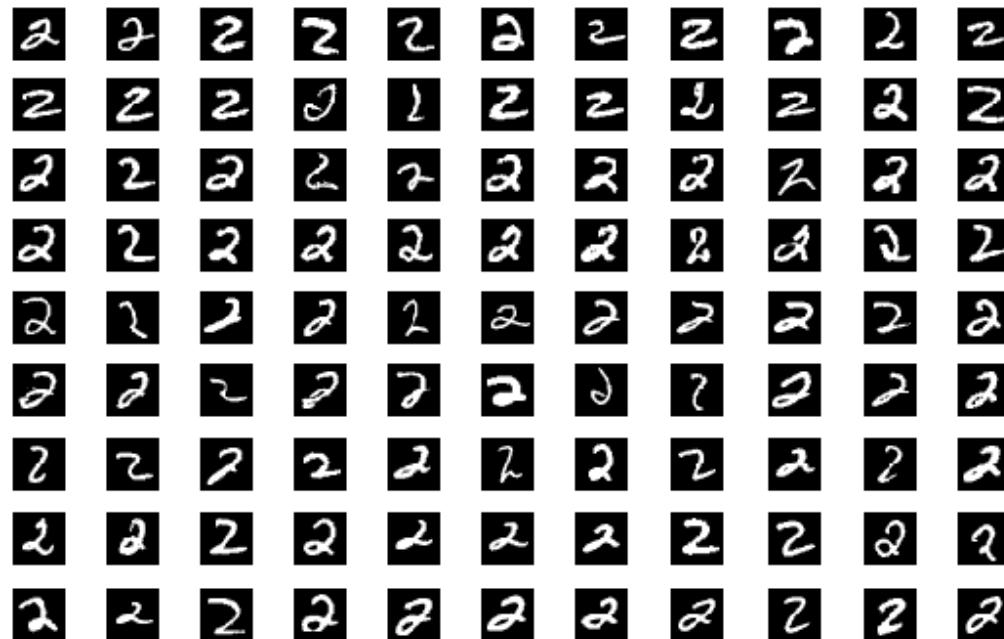
- THE MNIST DATABASE
<http://yann.lecun.com/exdb/mnist/>
- The original black and white (bi-level) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

Design a Learning System

Step 2: Representing Experience

- THE MNIST DATABASE

<http://yann.lecun.com/exdb/mnist/>



The feature vector of
input data is a 784
dimensional vector

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience/examples
 - The sensor input represented by an n-d vector, called the feature vector, $\mathbf{X} = (x_1, x_2, x_3, \dots, x_n)$
 - To represent the experience, we need to know what \mathbf{X} is.
 - So we need a corresponding vector \mathbf{D} , which will record our knowledge (experience) about \mathbf{X} .
 - The experience \mathbf{E} is a pair of vectors $\mathbf{E} = (\mathbf{X}, \mathbf{D})$.

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience/examples.
 - The experience \mathbf{E} is a pair of vectors $\mathbf{E} = (\mathbf{X}, \mathbf{D})$.
- So, what would \mathbf{D} be like? There are many possibilities.

Design a Learning System

Step 2: Representing Experience

- So, what would **D** be like? There are many possibilities.
- Assuming our system is to recognise 10 digits only, then D can be a 10-d binary vector; each correspond to one of the digits.

$$D = (d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9)$$

e.g,

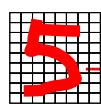
if X is digit 5, then $d_5=1$; all others =0

Design a Learning System

Step 2: Representing Experience

- So, what would **D** be like? There are many possibilities.
- Assuming our system is to recognise 10 digits only, then D can be a 10-d binary vector; each correspond to one of the digits.

$$D = (d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9)$$



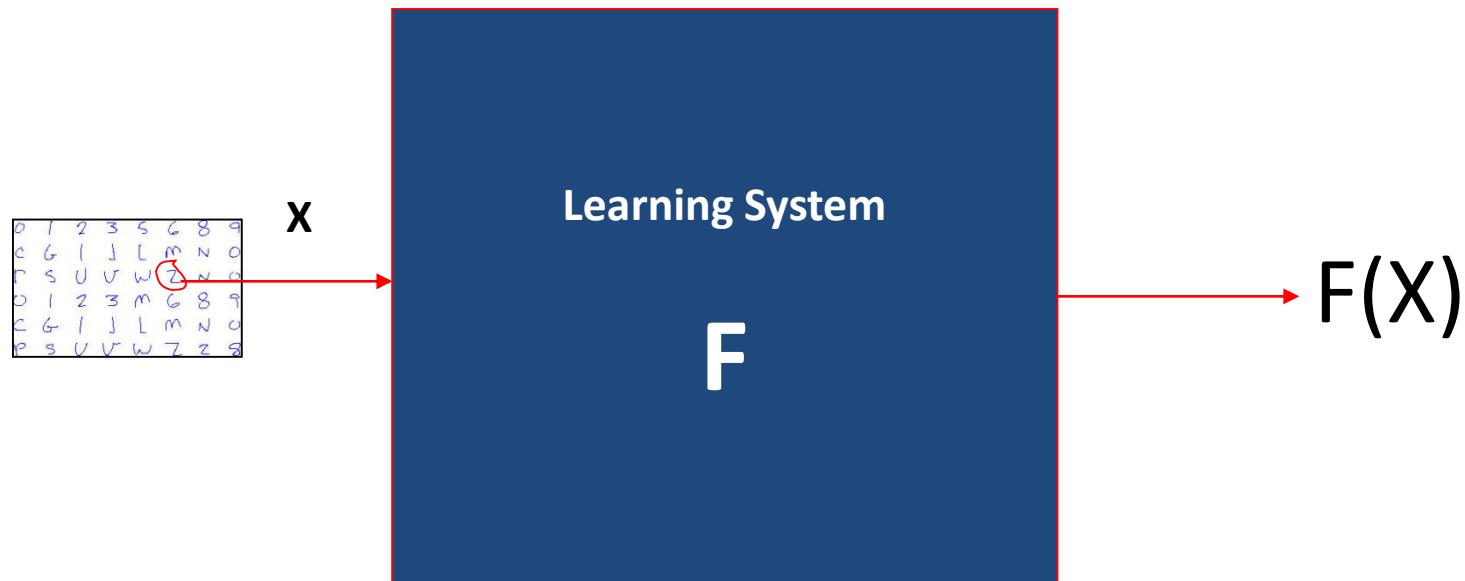
X = (1,1,0,1,1,1,1,1,1,0,0,0,0,1,1, ..., 1); 64-d Vector

D = (0,0,0,0,0,1,0,0,0,0)

Design a Learning System

Step 3: Choose a Representation for the Black Box

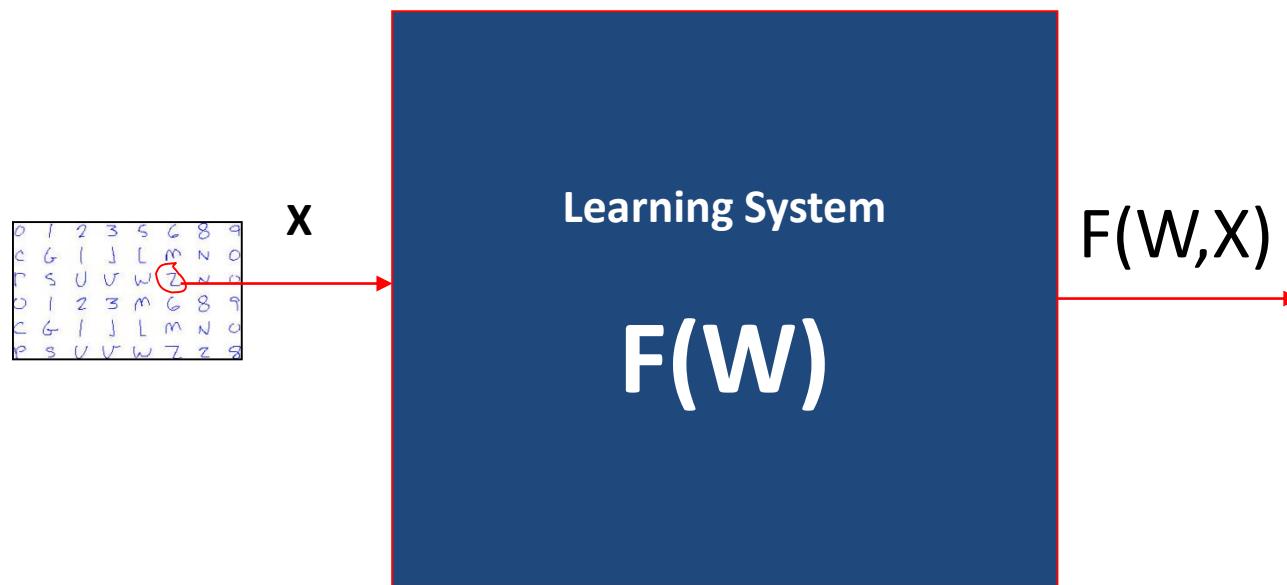
- We need to choose a function F to approximate the black box. For a given X , the value of F will give the classification of X . There are considerable flexibilities in choosing F .



Design a Learning System

Step 3: Choose a Representation for the Black Box

- F will be a function of some adjustable parameters, or weights, $W = (w_1, w_2, w_3, \dots, w_N)$, which the learning algorithm can modify or learn



Design a Learning System

Step 4: Learning/Adjusting the Weights

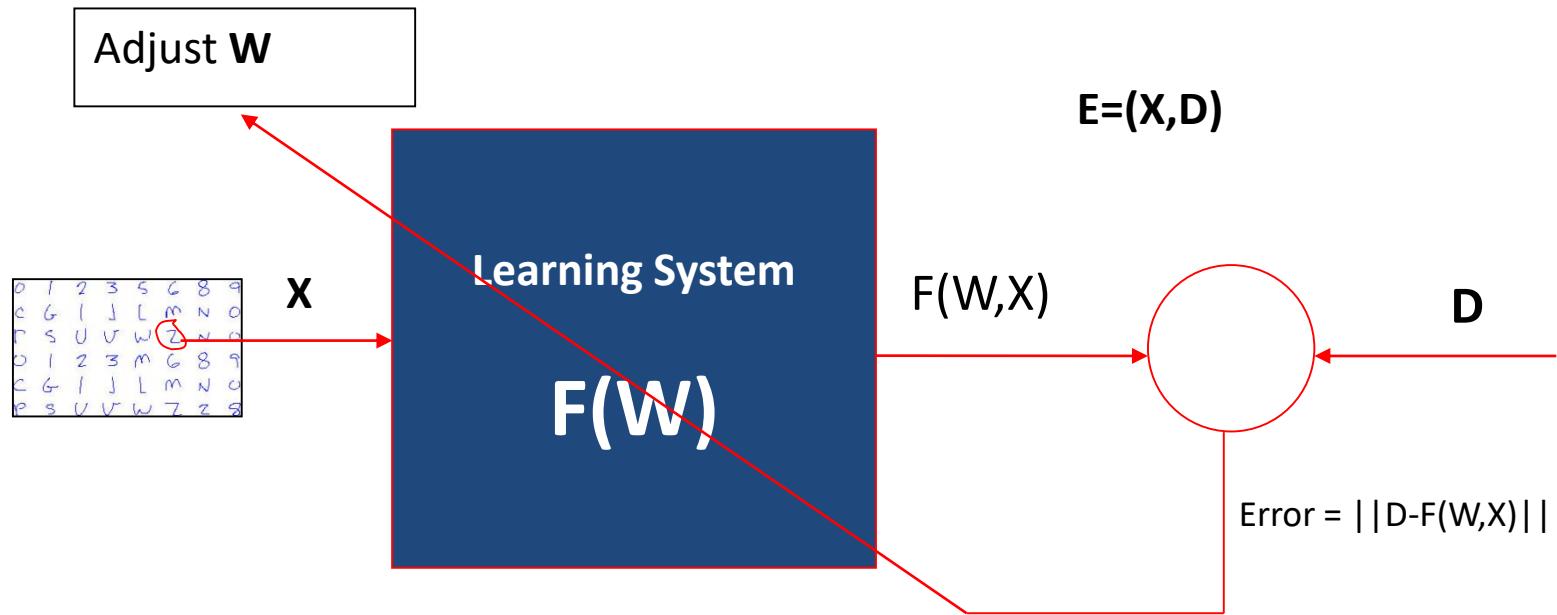
- We need a learning algorithm to adjust the weights such that the experience/prior knowledge from the training data can be learned into the system:

$$E = (X, D)$$

$$F(W, X) = D$$

Design a Learning System

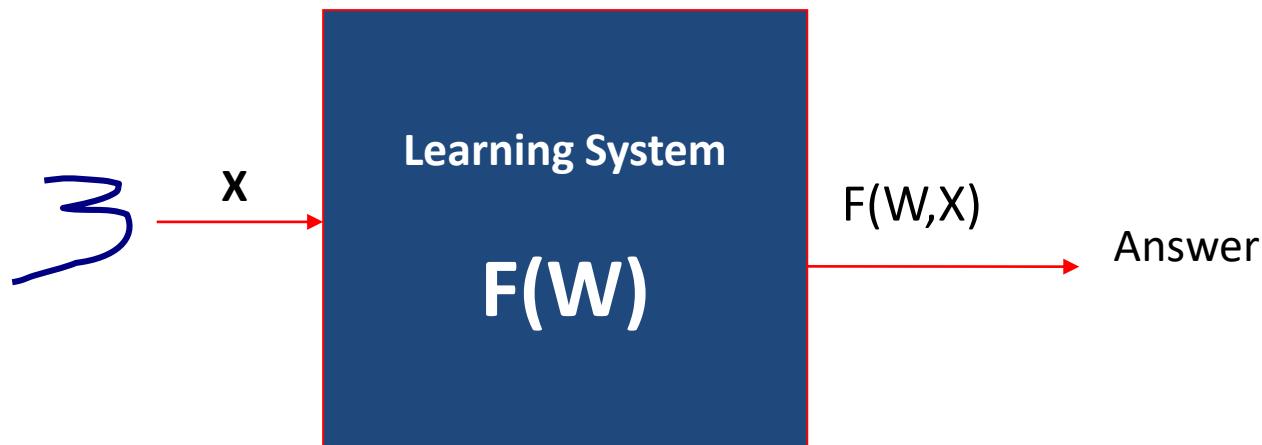
Step 4: Learning/Adjusting the Weights



Design a Learning System

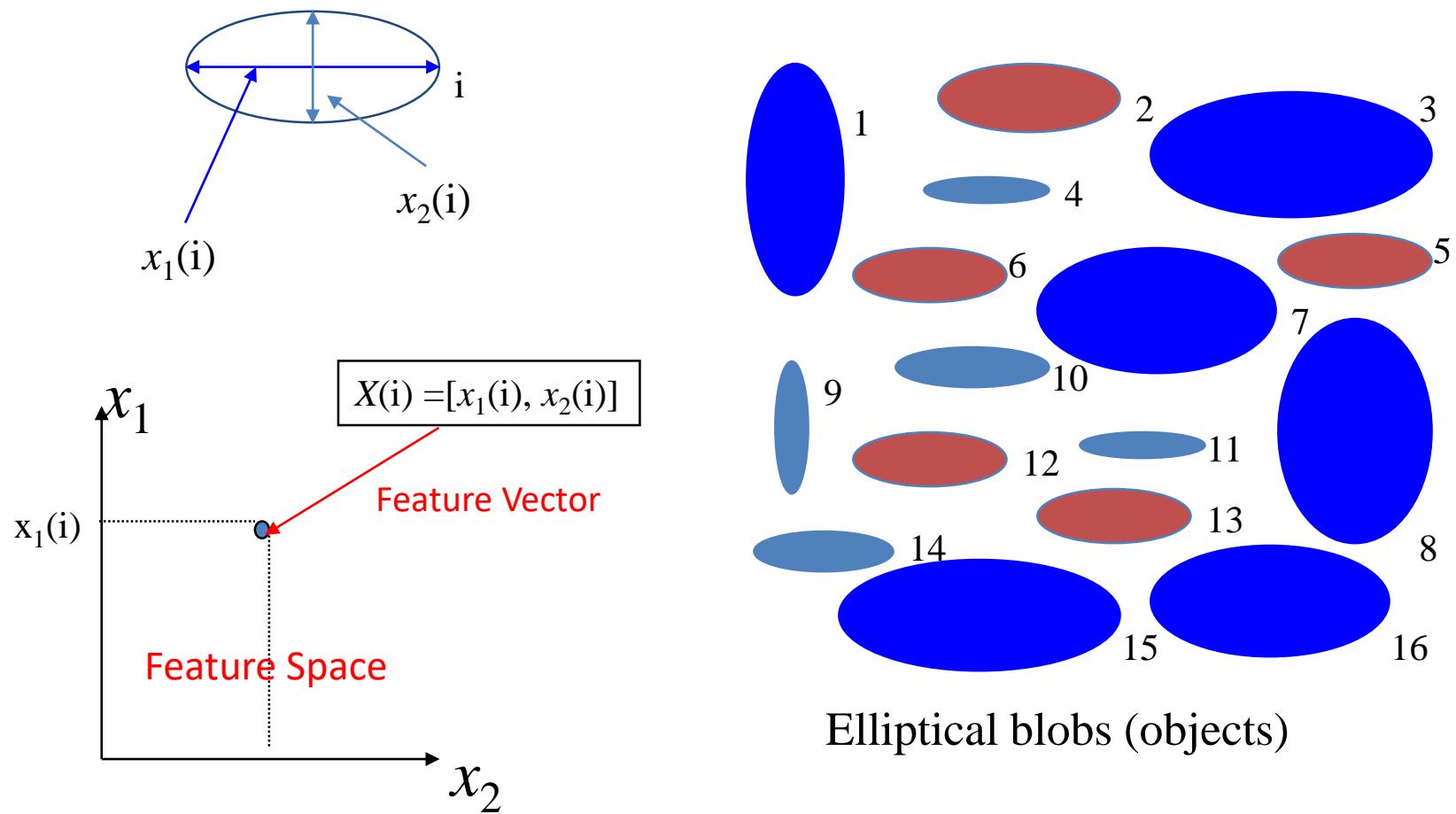
Step 5: Use/Test the System

- Once learning is completed, all parameters are fixed. An unknown input \mathbf{X} is presented to the system, the system computes its answer according to $F(\mathbf{W}, \mathbf{X})$



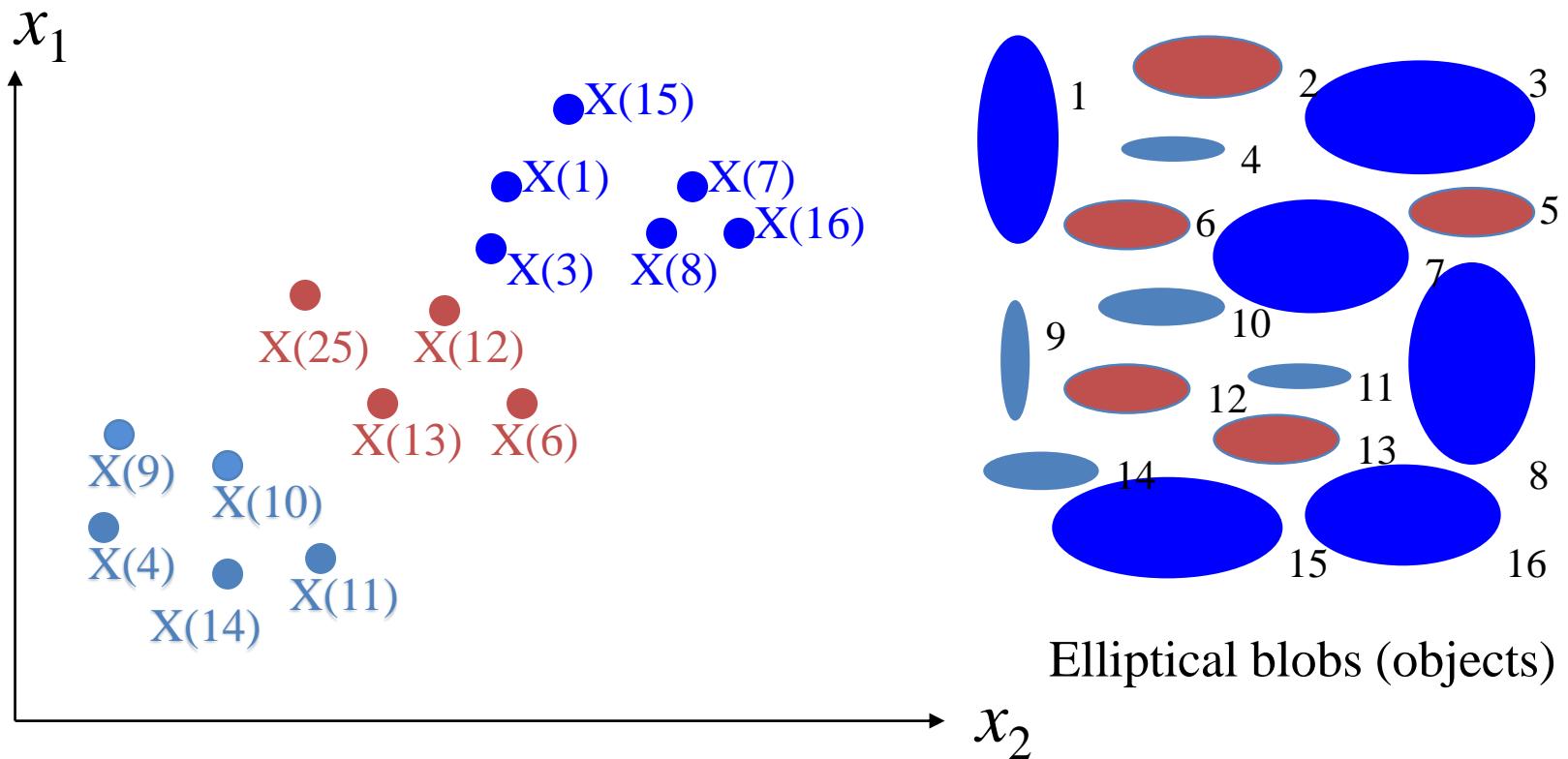
Feature Space

Representing real world objects using **feature vectors**



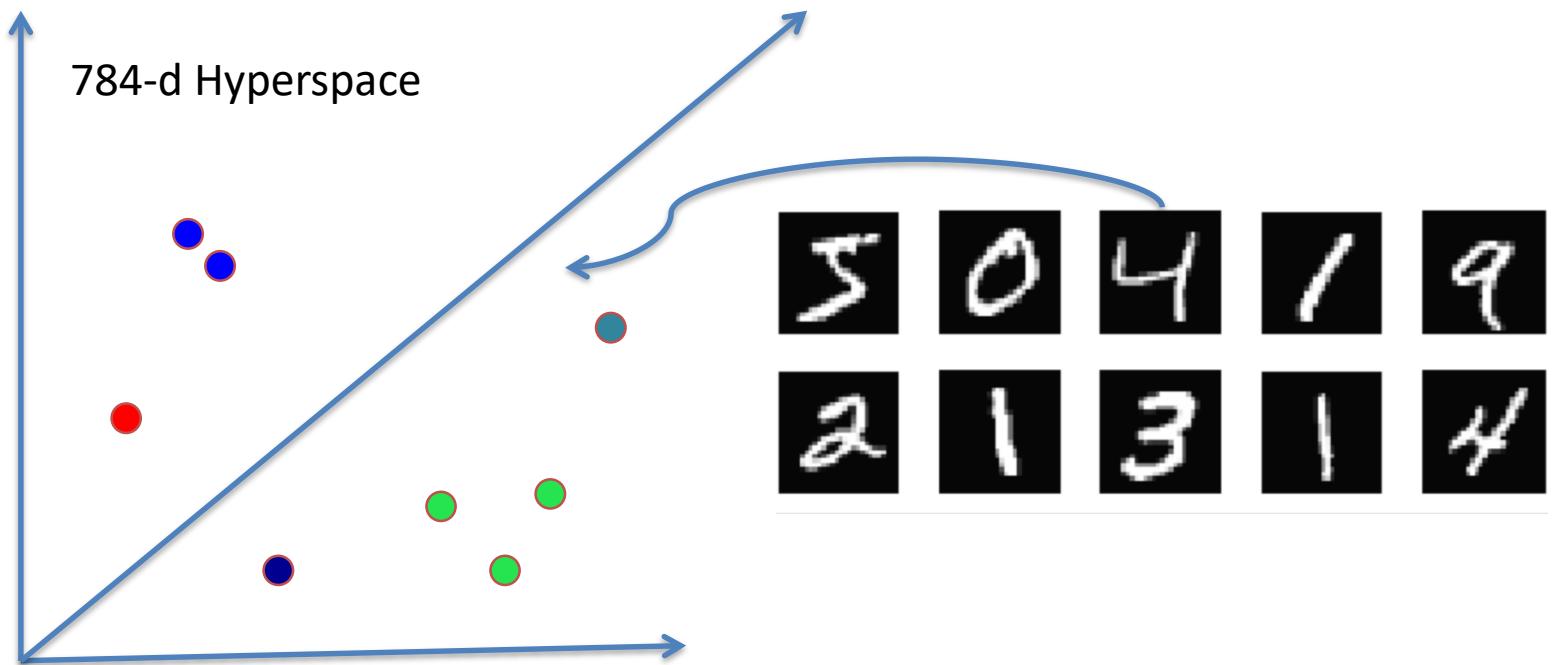
Feature Space

From Objects to Feature Vectors to Points in the Feature Spaces



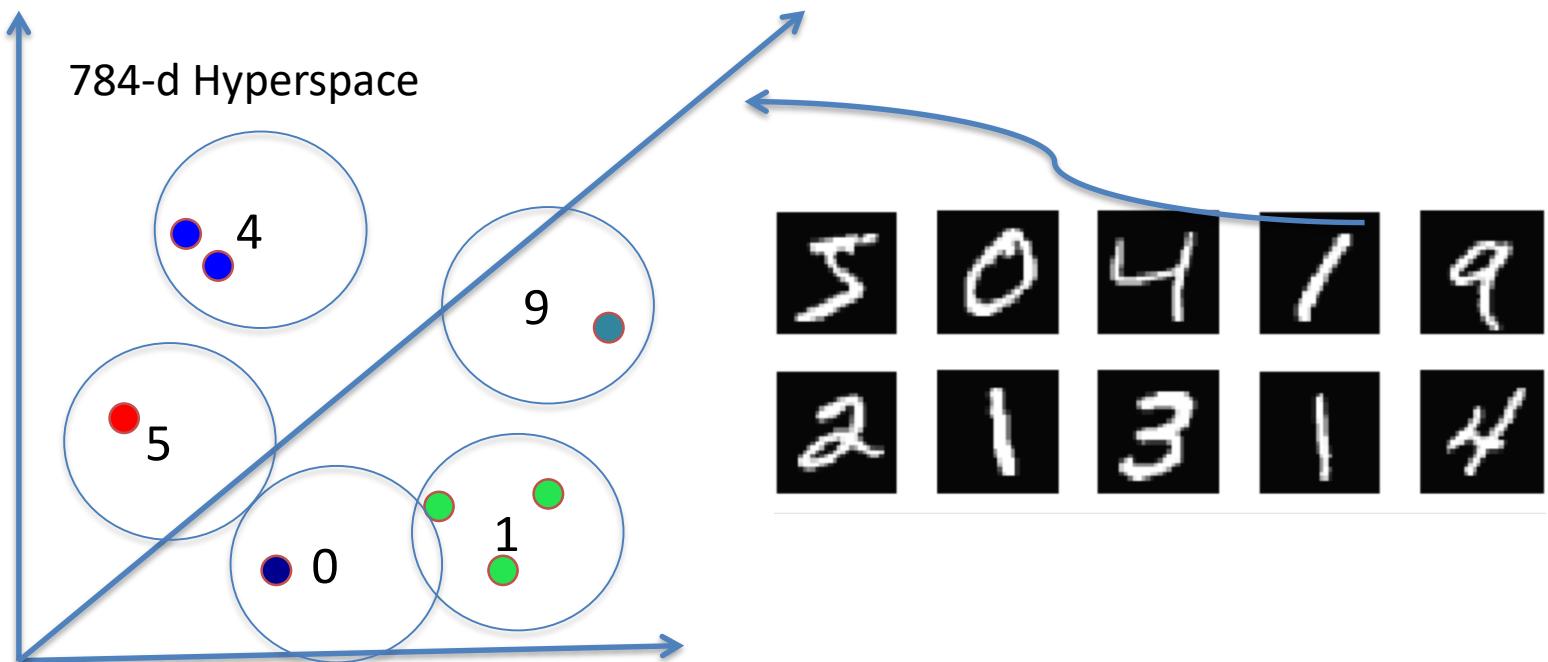
Feature Space

From Objects to Feature Vectors to Points in the Feature Spaces



Feature Space

From Objects to Feature Vectors to Points in the Feature Spaces



Representing General Objects

Feature vectors of

Faces

Cars

Fingerprints

Gestures

Emotions (a smiling face, a sad expression etc)

...

Further Reading

Chapter 1, T. M. Mitchell, Machine Learning, McGraw-Hill International Edition, 1997

Any Questions?





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 3 – Data Collection

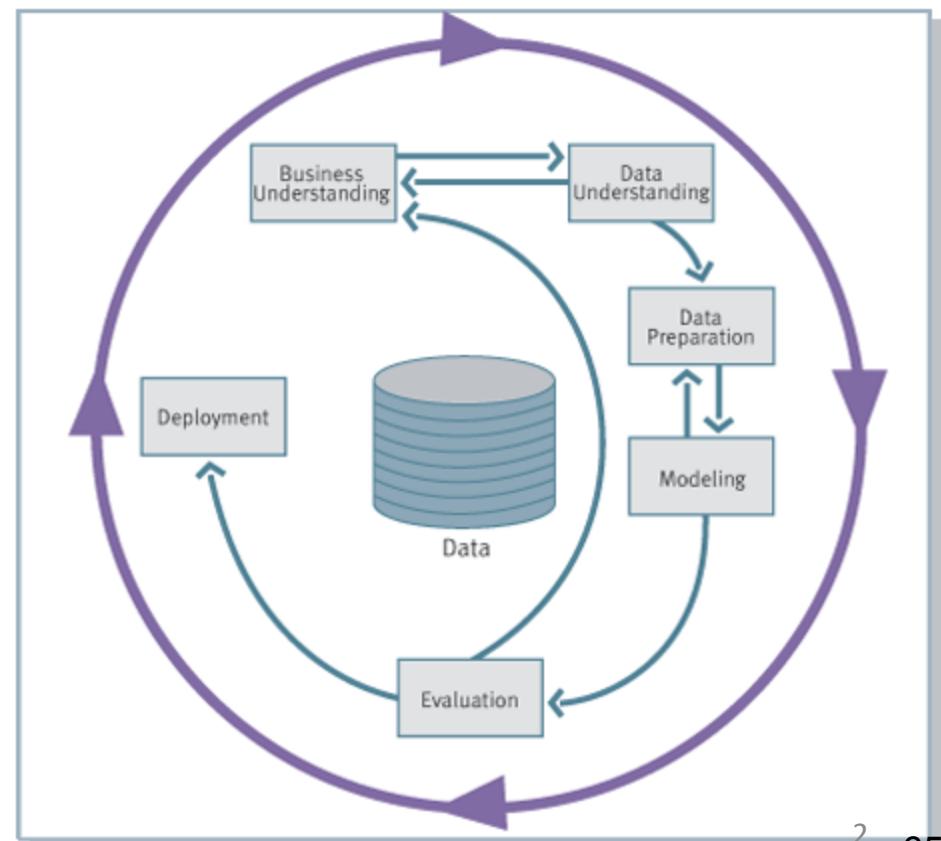
Ying Weng
2024 Autumn

Data Mining Process Model

CRoss Industry Standard Process for Data Mining (**CRISP-DM**)

Industry Standard

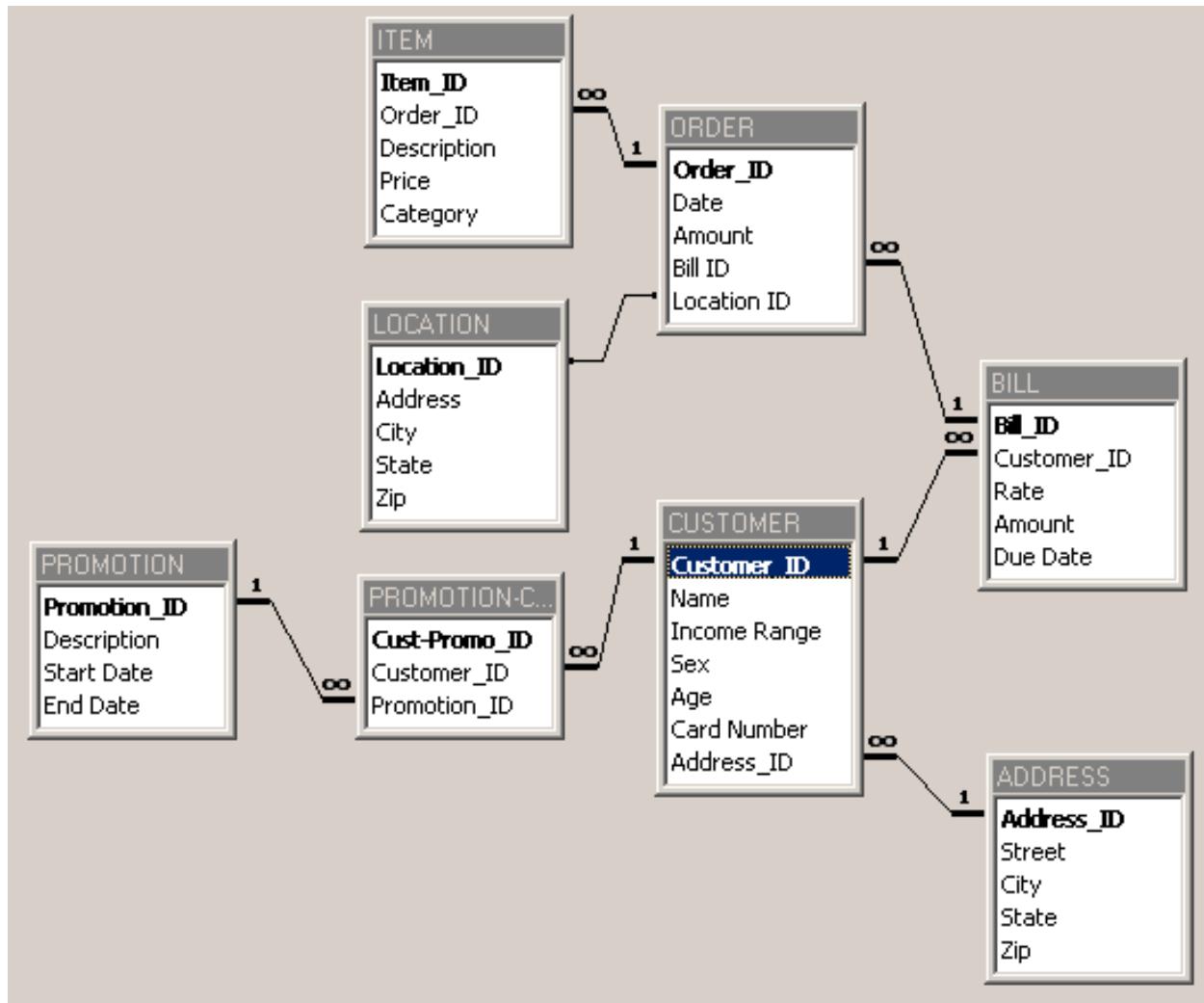
The most popular data mining methodology model according to [KDNuggets.com](https://www.kdnuggets.com)



Step 1: Business Understanding

- Define the problem
- Choose a machine learning model(s)
- Estimate project **cost**
- Estimate project completion **time**
- Address **legal** issues
- Develop a maintenance plan

Step 2: Data Understanding



Step 3: Data Preprocessing

- Noisy data
 - Locate **duplicate** records
 - Locate incorrect attribute values
 - **Smooth** data
- Missing data
 - **Discard** records with missing values
 - **Replace** missing real-valued items with the class mean
 - **Replace** missing values with values found within *highly similar* instances
- Data transformation
 - Data **normalization**
 - Data **type conversion**
 - Attribute and instance selection

Step 4: Modeling

- Choose **training** and **test** data
- Designate a set of **input** attributes
- If learning is **supervised**, choose one or more output attributes
- Select **learning parameter** values
- Train the model

Step 5: Evaluation

- Statistical analysis
- Heuristic analysis
- Experimental analysis
- Human analysis

Measures of Effectiveness of the Model

- **Accuracy**

Percentage of total predictions that were correct

- **Return on investment**

Cost-benefit ratios

- **Explanation**

Able to justify intuition

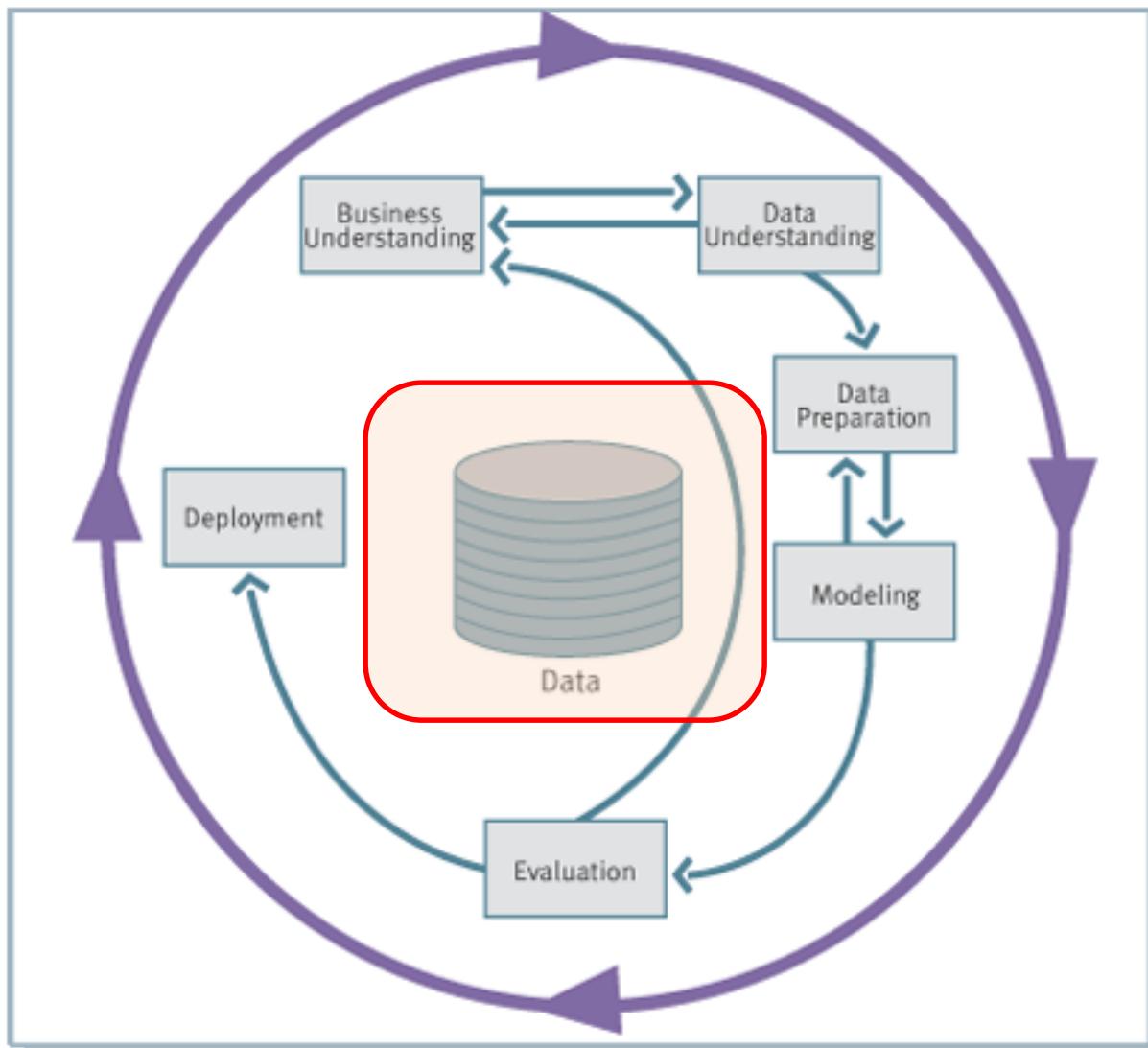
- **Validation**

Automated checking of correctness, indexes

Step 6: Deployment

- Apply the model to real world usage
- Apps, API, etc.
- Regularly update the model with new data
- ...

You Need Collect Data before Learning Starts!



What is Data?

Collection of data objects and their associated attributes

An **attribute** is a property or characteristic of an object

- Examples: eye color of a person, temperature, etc.
- Attribute is also known as variable, field, characteristic, or feature

A **collection of attributes** describe an object

- Object is also known as record, point, case, sample, entity, or instance

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Types of Attributes

There are different types of attributes

NO NEED TO EXPLAIN

- Nominal
 - Examples: ID numbers, eye color, zip codes
- Ordinal
 - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
- Interval
 - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
- Ratio
 - Examples: temperature in Kelvin, length, time, counts

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another.	zip codes, employee ID numbers, eye color, sex: { <i>male</i> , <i>female</i> }	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects.	hardness of minerals, { <i>good</i> , <i>better</i> , <i>best</i> }, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists.	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, <i>t</i> and <i>F</i> tests
Ratio	For ratio variables, both differences and ratios are meaningful.	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Discrete and Continuous Attributes

Discrete attribute

- Has only a **finite** or countable set of values.
- Examples: zip codes, number of employees, or sale counts.
- Often represented as **integer** variables.
- Note: binary attributes are a special case of discrete attributes.

Continuous attribute

- Has **real** numbers as attribute values.
- Examples: temperature, stock prices, or net income.
- Continuous attributes are typically represented as **floating-point** variables.

Types of Data

Record

- Data matrix
- Transaction data

Graph

- Social networks
- Molecular structures

Ordered

- Spatial data
- Temporal data
- Sequential data
- Genetic sequence data

We often deal with
a mixture of
different types data

Record Data

- Data that consists of a collection of records, each of which consists of a fixed set of attributes.

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Data Matrix

- If data objects have the same fixed set of **numeric** attributes, the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute.
- Such data set can be represented as an m by n matrix, where there are m rows (one for each object) and n columns (one for each attribute).

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Transaction Data

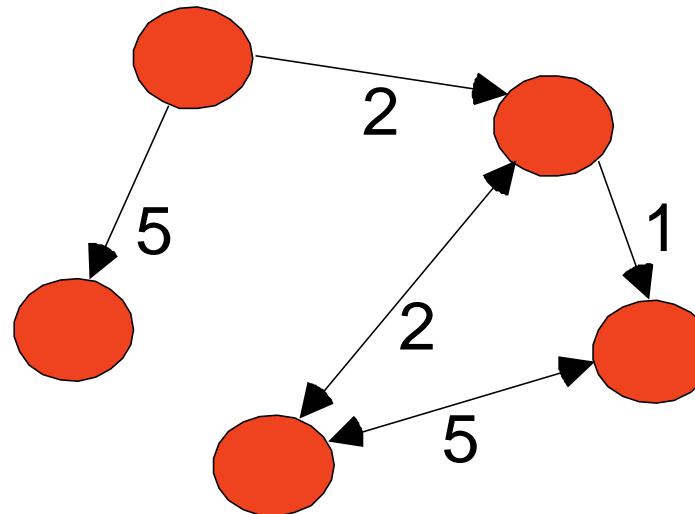
- A special type of record data, where
 - Each record (transaction) involves a set of items.
 - For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Graph Data

Examples:

- Representation of HTML Links
- Social Networks



Ordered Data

Genomic sequence data

```
GGTTCCGCCCTTCAGCCCCGCGGCC  
CGCAGGGCCCGCCCCGCGGCCGTC  
GAGAAGGGCCCGCCTGGCGGGCG  
GGGGGAGGCAGGGCCGCCGAGC  
CCAACCGAGTCCGACCAGGTGCC  
CCCTCTGCTCGGCCTAGACCTGA  
GCTCATTAGGCAGCAGCGGACAG  
GCCAAGTAGAACACGCGAAGCGC  
TGGGCTGCCTGCTGCGACCAGGG
```

Obtaining Data

- **Data sources**
 - Obtained directly from owner: text file or relational database.
 - Collect public available data: web.
- **Unstructured VS. structured**
 - Text file: can be unstructured or structured.
 - Relational database: structured.
 - Web: unstructured.
 - Need to structure the data if necessary.
- **Data need to be cleaned**
- **Tools for data collection and cleaning**
 - Python, R, Excel, SQL, etc.

Text Files

- Most companies use proprietary software to store data that can be exported into text files.
 - Usually with .txt extension.
 - Sometimes with .csv extension
- Commonly used text file format
 - Fixed-width: each attribute value starts and stops a fixed positions (columns) in the lines.
 - Delimited: there is a delimiter character, usually a tab, space, or **comma**, that separates different values in the lines.
 - **CSV**: comma separated values

The image shows two side-by-side Notepad windows. The left window displays a fixed-width text file named 'Transactions.txt' with columns for Purchase Date, Customer ID, Gender, Marital Status, and Homeow. The right window displays a CSV file with columns for Annual Income, City, State or Province, Country, Product, and various item details like Snack Foods, Produce Vegetables, etc.

Transactions.txt - Notepad (Left):

Purchase Date	Customer ID	Gender	Marital Status	Homeow
12/18/2007	7223	F	S	Y
12/20/2007	7841	M	M	Y
12/21/2007	8374	F	M	N
12/21/2007	9619	M	M	Y
12/22/2007	1900	F	S	Y
12/22/2007	6696	F	M	Y
12/23/2007	9673	M	S	Y
12/25/2007	354	F	M	Y
12/25/2007	1293	M	M	Y
12/25/2007	7938	M	S	N
12/26/2007	9357	F	M	N
12/26/2007	3097	M	M	Y
12/26/2007	2741	M	S	N

Transactions.txt - Notepad (Right):

Annual Income	City	State or Province	Country	Product
CA	USA	Food	Snack Foods	Snack Foods
CA	USA	Food	Produce Vegetables	5
WA	USA	Food	Snack Foods	Snack Foods
OR	USA	Food	Snacks Candy	\$4.44
CA	USA	Drink	Beverages	Carbonated Beverages
CA	USA	Food	Deli	Side Dishes
USA	Food	Frozen Foods	Breakfast Foods	\$4.3
USA	Food	Canned Foods	Canned Soup	\$7.3
WA	USA	Non-Consumable	Household	Cleaning Sup
CA	USA	Non-Consumable	Health and Hygiene	Pain
CA	USA	Food	Snack Foods	Snack Foods
CA	USA	Food	Baking Goods	Baking Goods
WA	USA	Food	Canned Foods	Canned Tuna

CSV Example

- Yahoo stock price historical data
 - Example: Sina stock price from Jan 1st, 2016 to Jan 1st 2017
 - <http://finance.yahoo.com/quote/SINA/history?period1=1451577600&period2=1483200000&interval=1d&filter=history&frequency=1d>

SINA Corporation (SINA)
NasdaqGS - NasdaqGS Real Time Price. Currency in USD [Add to watchlist](#)

68.50 0.00 (0.00%)
At close: January 14 4:00PM EST

Summary Conversations Statistics Profile Financials Options Holders **Historical Data** Analysts

 ★ 全面涵蓋考試+改卷+策略分析
★ 每星期免費DSE解難服務
★ 經驗導師一對一改善建議

Time Period: Jan 01, 2016 - Jan 01, 2017 Show: Historical Prices Frequency: Daily [Apply](#)

Currency in USD [Download Data](#)

Date	Open	High	Low	Close	Adj Close*	Volume
Dec 30, 2016	62.21	62.83	60.46	60.79	60.79	927,100
Dec 29, 2016	62.29	62.67	61.76	62.13	62.13	516,100
Dec 28, 2016	63.60	63.60	61.80	62.02	62.02	601,400

CSV Example

- Manual download from Yahoo
 - <http://chart.finance.yahoo.com/table.csv?s=SINA&a=0&b=1&c=2016&d=0&e=1&f=2017&g=d&ignore=.csv>
 - Save to local disk as csv file, e.g. “table.csv”
 - Use Python to read csv file
- Automatically download from Yahoo
 - Use Python for both downloading and reading

```
# Load Yahoo stock price for Sina
import numpy as np
# URL for SINA from Jan 1st 2016 to Jan 1st 2017
# url = "table.csv"
url = "http://chart.finance.yahoo.com/table.csv?s=SINA&a=0&b=1&c=2016&d=0&e=1&f=2017&g=d&ignore=.csv"
# load the CSV file as a numpy matrix
dataset = np.genfromtxt(url, dtype=None, skip_header=1, delimiter=",")
print(dataset[0])
```

Relational Database

- A **relational database** is a set of related tables, where each table is a rectangular arrangement of fields and records.
 - Rows corresponding to records.
 - Columns corresponding to fields.
 - Primary key contains unique values to index data.
 - Foreign key links tables and can contain duplicate values.
- Python allows you to import data from many database packages
 - You may need to install drive for different database, e.g. MySQLdb for MySQL database, before use.
- Basic steps
 - Make connection with host name, user name, password, database name, etc.
 - Create a cursor to the connected database
 - Execute SQL queries and fetch the data through the cursor
 - Close the database

Relational Database

```
import MySQLdb

db = MySQLdb.connect(host="localhost", # your host, usually localhost
                     user="john", # your username
                     passwd="megajonhy", # your password
                     db="jonhydb") # name of the data base

# you must create a Cursor object. It will let
# you execute all the queries you need
cur = db.cursor()

# Use all the SQL you like
cur.execute("SELECT * FROM YOUR_TABLE_NAME")
# print all the first cell of all the rows

for row in cur.fetchall():
    print row[0]

db.close()
```

Data from the Web

- Web sites containing data are structured in all sorts of ways, and the steps required to collect the data for analysis vary greatly.
- No matter the server side is a program or file, what the client side received (and the browser shows) are all files.
 - Usually in HTML (hypertext markup language) format.
 - HTML uses tags for displaying various items on the web page.



HTML File and Representation

- Most **html tags** control font, color, images, actions, etc., which are not related to the content of data.
- The layout of html is often controlled using table (old fashion) or CSS (Cascading Style Sheets, new fashion).
- Tablet data are normally put inside table tag, but many data we are interested are not in the table.

```
<html>
<body>
<p>
    Each table starts with a table tag.
    Each table row starts with a tr tag.
    Each table data starts with a td tag.
</p>

<h4>One row and three columns:</h4>
<table border="1">
<tr>
    <td>100</td>
    <td>200</td>
    <td>300</td>
</tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="1">
<tr>
    <td>100</td>
    <td>200</td>
    <td>300</td>
</tr>
<tr>
    <td>400</td>
```

Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.

One row and three columns:

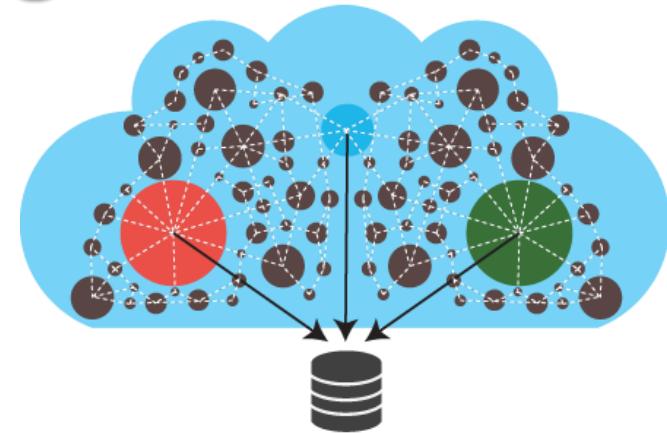
100	200	300
-----	-----	-----

Two rows and three columns:

100	200	300
400	500	600

Web Crawling

- Find out the patterns of URL, e.g.
 - <http://finance.yahoo.com/q?s=SINA>
 - actual address: <http://finance.yahoo.com/q>
 - Parameter: **s=SINA**
 - **name=value** pairs separated by &
- Write a program to generate the URLs and download.
 - By changing parameters and name=value pairs
- Sometimes need to find and download the URLs inside a web page (spidering).
 - URLs normally start with “http” in raw html files.
- Parsing the web pages to collect useful data
 - **Identify** where the useful data is in a web page.
 - **Finding patterns** to help the program auto locate the data by looking at raw html files.
 - Use Python Regular Expression to locate the data according to the pattern.



Unstructured Data Parsing

Regular Expression

- Python, Java, etc.
- Find patterns from the “unstructured” data, including text file, HTML file, or other files.
- Define such patterns using regular expression grammar.
- Read files into a string. The processing engine will extract data from the string that meets this pattern.
- More on this using Python later!

```
import urllib.request
import re

url="http://google.com"

# regular expression for locating title
these_regex=b"<title>(.+?)</title>"
pattern=re.compile(these_regex)

# load the url
with urllib.request.urlopen(url) as response:
    html = response.read()

# find the pattern in the downloaded file
titles=re.findall(pattern, html)
print(titles)
```

Any Questions?





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

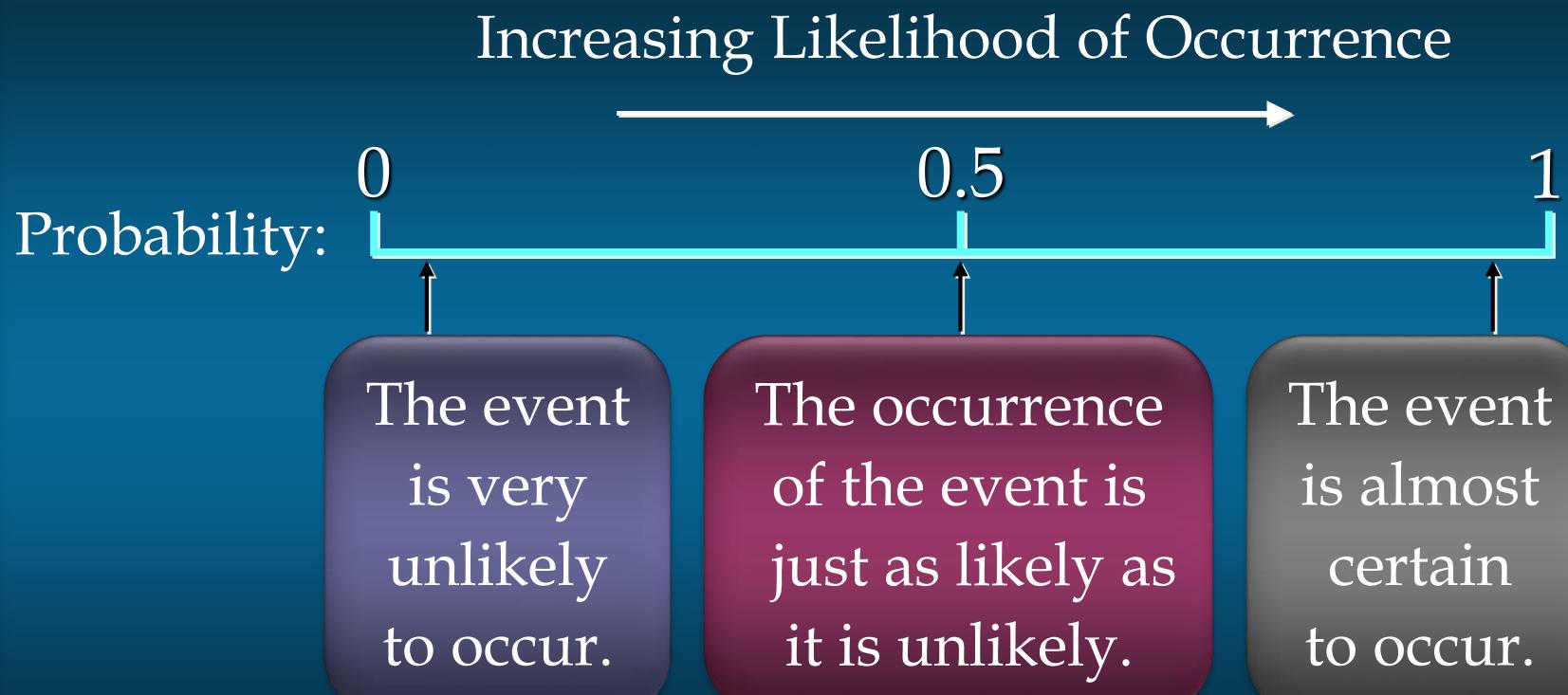
Topic 4 – Probability Theory

Ying Weng
2024 Autumn

Probability

- ▶ Probability is a numerical measure of the likelihood that an event will occur.
- ▶ Probability values are always assigned on a scale from 0 to 1.
- ▶ A probability near zero indicates an event is quite unlikely to occur.
- ▶ A probability near one indicates an event is almost certain to occur.

Probability as a Numerical Measure



Sample Space

- The set of all possible outcomes of an experiment.
- A sample space can be finite or infinite, & discrete or continuous.

An Experiment and Its Sample Space

Experiment

Flip a coin

Inspect a product

Conduct a sales call

Roll a die

Play a football game

Experiment Outcomes

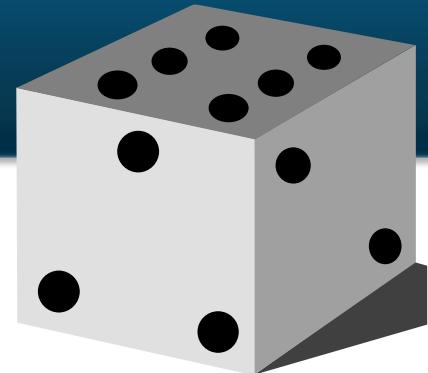
Head, tail

Defective, non-defective

Purchase, no purchase

1, 2, 3, 4, 5, 6

Win, lose, tie



Experiment

- Something capable of replication under stable conditions.
- Example: Flipping a coin



Example: discrete, finite sample space

- Experiment: Flipping a coin once.
- Sample space: {Head, Tail}.
- This sample space is **discrete**
- This sample space is also finite. There are just two elements: that's a **finite** number.



Example: continuous sample space

- Experiment: Burning a light bulb until it burns out.
Suppose there is a theoretical maximum number of hours that a bulb can burn and that is 10,000 hours.
- Sample space: The set of all real numbers between 0 & 10,000.
- Between any two numbers you can pick in the sample space, there is another number.
- For example, the bulb could burn for 99.777 hours or 99.778 hours. But it could also burn for 99.7775, which is in between.
- This sample space is **continuous**.
- It is also **infinite**, since there are an infinite number of possibilities.
- ***All continuous sample spaces are infinite.***

Two Basic Properties of Probability

- 1. $0 \leq \Pr(E) \leq 1$
for every subset of the sample space S
- 2. $\Pr(S) = 1$

Assigning Probabilities

■ Basic Requirements for Assigning Probabilities

- ▶ 1. The probability assigned to each experimental outcome must be between 0 and 1, inclusively.

$$▶ 0 \leq P(E_i) \leq 1 \text{ for all } i$$

where:

E_i is the i th experimental outcome
and $P(E_i)$ is its probability

Assigning Probabilities

- Basic Requirements for Assigning Probabilities
 - ▶ 2. The sum of the probabilities for all experimental outcomes must equal to 1.
$$P(E_1) + P(E_2) + \dots + P(E_n) = 1$$

where:

n is the number of experimental outcomes

Assigning Probabilities

▶ Classical Method

Assigning probabilities based on the assumption of equally likely outcomes

▶ Relative Frequency Method

Assigning probabilities based on experimental or historical data

▶ Subjective Method

Assigning probabilities based on judgment

Classical Method

- Example: Roll a die
 - ▶ If an experiment has n possible outcomes, the classical method would assign a probability of $1/n$ to each outcome.
 - ▶ Sample Space: $S = \{1, 2, 3, 4, 5, 6\}$
 - ▶ Probabilities: Each sample point has a $1/6$ chance of occurring

Relative Frequency Method

Example: Lucas Tool Rental

- Lucas Tool Rental would like to assign probabilities to the number of car polishers it rents each day. Office records show the following frequencies of daily rentals for the last 40 days.



<u>Number of Polishers Rented</u>	<u>Number of Days</u>
0	4
1	6
2	18
3	10
4	2

Relative Frequency Method

Example: Lucas Tool Rental

- Each probability assignment is given by dividing the frequency (number of days) by the total frequency (total number of days).

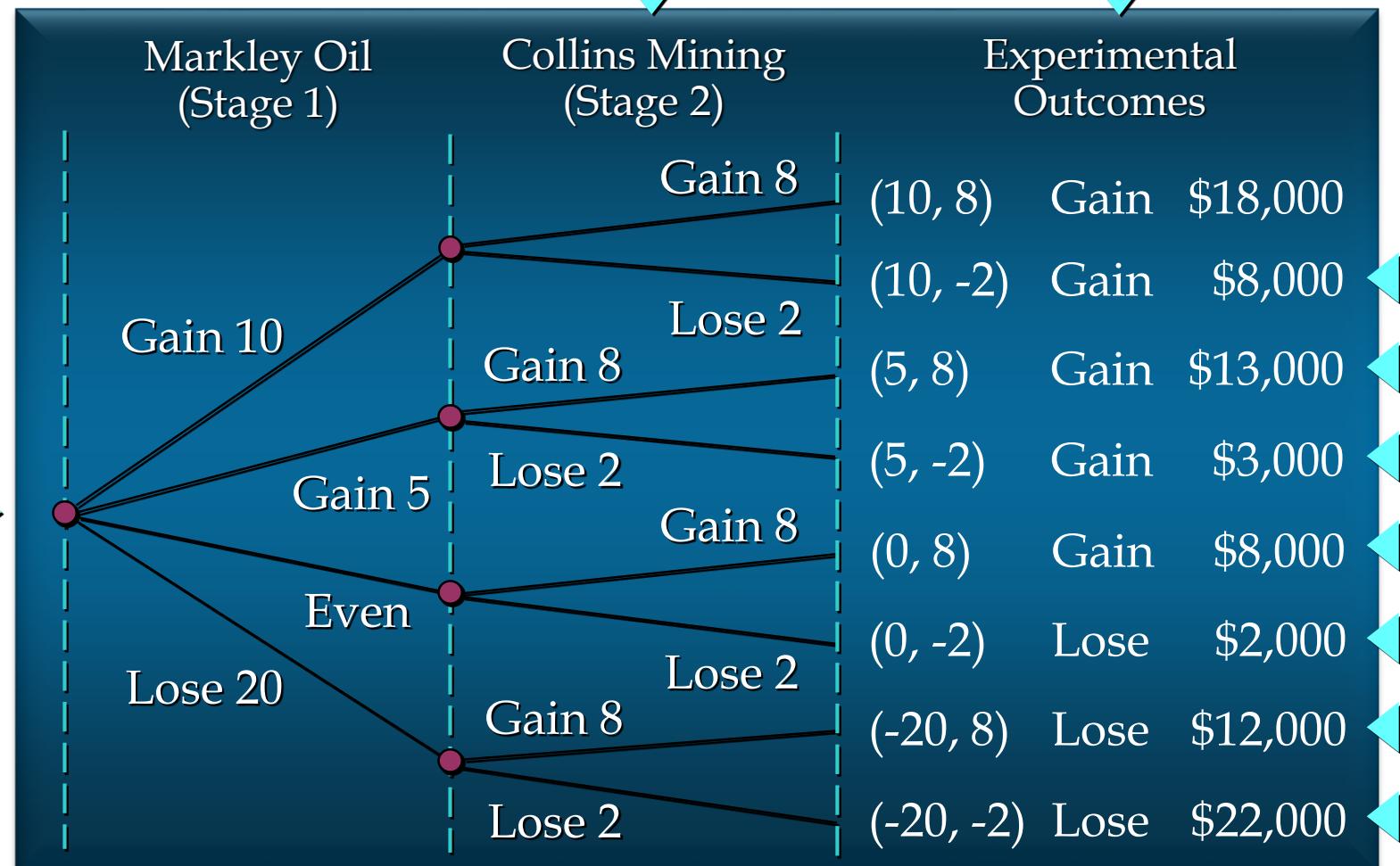
<u>Number of Polishers Rented</u>	<u>Number of Days</u>	<u>Probability</u>
0	4	.10
1	6	.15
2	18	.45
3	10	.25
4	$\frac{2}{40}$	$\frac{.05}{1.00}$

Subjective Method

- ▶ ■ When economic conditions and a company's circumstances change rapidly it might be inappropriate to assign probabilities based solely on historical data.
- ▶ ■ We can use any data available as well as our experience and intuition, but ultimately a probability value should express our degree of belief that the experimental outcome will occur.
- ▶ ■ The best probability estimates often are obtained by combining the estimates from the classical or relative frequency approach with the subjective estimate.

Tree Diagram

Example: Bradley Investments



Subjective Method

■ Example: Bradley Investments

► An analyst made the following probability estimates.

<u>Experimental Outcome</u>	<u>Net Gain or Loss</u>	<u>Probability</u>
(10, 8)	\$18,000 Gain	.20
(10, -2)	\$8,000 Gain	.08
(5, 8)	\$13,000 Gain	.16
(5, -2)	\$3,000 Gain	.26
(0, 8)	\$8,000 Gain	.10
(0, -2)	\$2,000 Loss	.12
(-20, 8)	\$12,000 Loss	.02
(-20, -2)	\$22,000 Loss	.06

Counting Rules

- We'll look at:

Basic multiplication rule

Multiplication Rule Example

In general,

- If we have an experiment with k parts (such as 3 flips)
- and each part has n possible outcomes (such as heads & tails)
- then the total number of possible outcomes for the experiment is

$$\bullet n^k$$

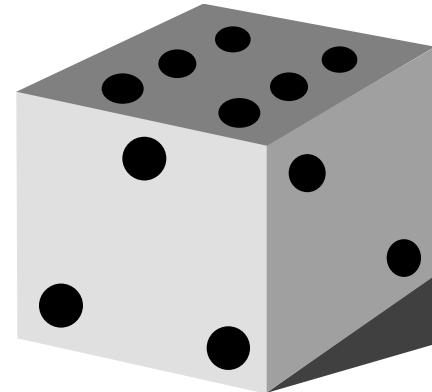
- This is the simplest multiplication rule

As a variation, suppose that we have an experiment with 2 parts, the 1st part has m possibilities, & the 2nd part has n possibilities.

- How many possibilities are there for the experiment?

In particular, we might have a coin & a die.

- So one possible outcome could be H6 (a head on the coin & a 1 on the die).
- How many possible outcomes does the experiment have?



We have $2 \times 6 = 12$ possibilities.

- Return to our more general question about the 2-part experiment with m & n possibilities for each part.
- We now see that the total number of outcomes for the experiment is

• mn

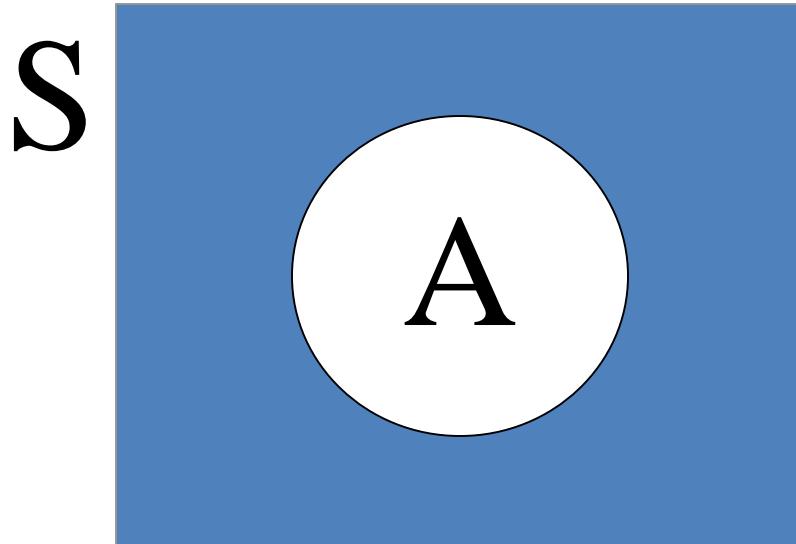
If we had a 3-part experiment,
the 1st part has **m** possibilities,
the 2nd part has **n** possibilities,
& the 3rd part has **p** possibilities,
how many possible outcomes would the
experiment have?

$$\bullet m \ n \ p$$

Complements, Unions, & Intersections

- Suppose A & B are events.

The ***complement*** of A is everything in the sample space S that is NOT in A.

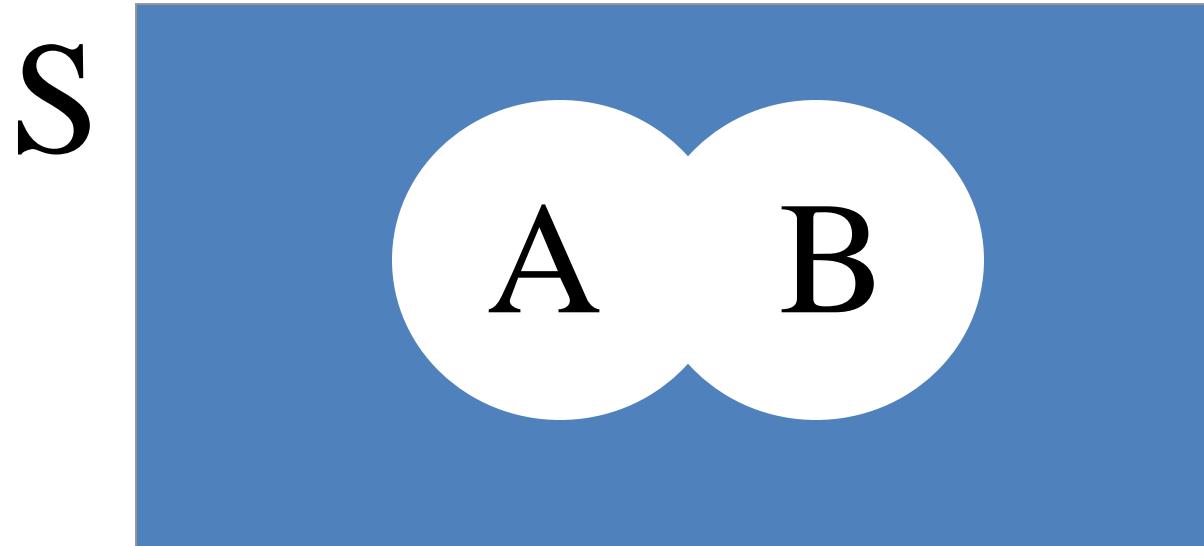


- If the rectangular box is S, and the white circle is A, then everything in the box that's outside the circle is A^c , which is the complement of A.

Theorem

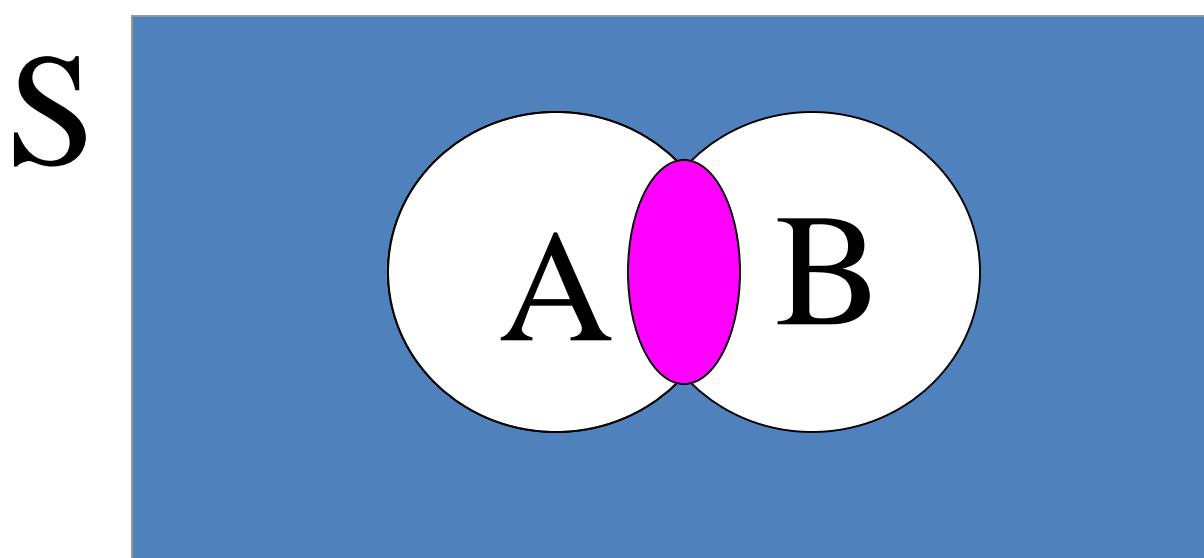
- $\Pr (A^c) = 1 - \Pr (A)$
- Example:
- If A is the event that a randomly selected student is male, and the probability of A is 0.6, what is A^c and what is its probability?
- A^c is the event that a randomly selected student is female, and its probability is 0.4.

The ***union*** of A & B (denoted $A \cup B$)
is everything in the sample space that is in either A or
B or both.



- The union of A & B is the whole white area.

The ***intersection*** of A & B (denoted $A \cap B$) is everything in the sample space that is in both A & B.



- The intersection of A & B is the pink overlapping area.

Example

- A family is planning to have 2 children.
- Suppose boys (B) & girls (G) are equally likely.
- What is the sample space S?
 - $S = \{BB, GG, BG, GB\}$

Example continued

- If E is the event that both children are the same sex, what does E look like & what is its probability?
 - $E = \{BB, GG\}$
- Since boys & girls are equally likely, each of the four outcomes in the sample space $S = \{BB, GG, BG, GB\}$ is equally likely & has a probability of $1/4$.
 - So $\Pr(E) = 2/4 = 1/2 = 0.5$

Example continued:

Recall that $E = \{BB, GG\}$ & $\Pr(E) = 0.5$

- What is the complement of E and what is its probability?

- $E^c = \{BG, GB\}$

- $\Pr(E^c) = 1 - \Pr(E) = 1 - 0.5 = 0.5$

Example continued

- If F is the event that at least one of the children is a girl, what does F look like & what is its probability?

- $F = \{BG, GB, GG\}$

- $\Pr(F) = 3/4 = 0.75$

Recall: $E = \{BB, GG\}$ & $\Pr(E) = 0.5$

$F = \{BG, GB, GG\}$ & $\Pr(F) = 0.75$

- What is $E \cap F$?
 - $\{GG\}$
- What is its probability?
 - $1/4 = 0.25$

Recall: $E = \{BB, GG\}$ & $\Pr(E) = 0.5$

$F = \{BG, GB, GG\}$ & $\Pr(F) = 0.75$

- What is the EUF?
 - $\{BB, GG, BG, GB\} = S$
- What is the probability of EUF?
 - 1
- If you add the separate probabilities of E & F together, do you get $\Pr(EUF)$? Let's try it.
- $\Pr(E) + \Pr(F) = 0.5 + 0.75 = 1.25 \neq 1 = \Pr(EUF)$
- Why doesn't it work?
- We counted GG (the intersection of E & F) twice.

A formula for $\Pr(E \cup F)$

- $\Pr(E \cup F) = \Pr(E) + \Pr(F) - \Pr(E \cap F)$
- If E & F do not overlap, then the intersection is the empty set, & the probability of the intersection is zero.
- When there is no overlap,
 $\Pr(E \cup F) = \Pr(E) + \Pr(F)$.

Conditional Probability of A given B

$$\Pr(A|B)$$

- $\Pr(A|B) = \Pr(A \cap B) / \Pr(B)$

Example

Suppose there are 10,000 students at a university.

2,000 are seniors (S). 3,500 are female (F).

800 are seniors & female.

- Determine the probability that a randomly selected student is (1) a senior, (2) a female, (3) a senior & female.
- 1. $\Pr(S) = 2,000/10,000 = 0.2$
- 2. $\Pr(F) = 3,500/10,000 = 0.35$
- 3. $\Pr(S \cap F) = 800/10,000 = 0.08$

Use the definition of conditional probability

$$\Pr(A|B) = \Pr(A \cap B) / \Pr(B)$$

& the previously calculated information

$$\Pr(S) = 0.2; \quad \Pr(F) = 0.35; \quad \Pr(S \cap F) = 0.08$$

to answer the questions below.

- 1. If a randomly selected student is female, what is the probability that she is a senior?
 - $\Pr(S|F) = \Pr(S \cap F) / \Pr(F)$
 - $= 0.08 / 0.35 = 0.228$
- 2. If a randomly selected student is a senior, what is the probability the student is female?
 - $\Pr(F|S) = \Pr(F \cap S) / \Pr(S)$
 - $= 0.08 / 0.2 = 0.4$
- Notice that $S \cap F = F \cap S$, so the numerators are the same, but the denominators are different.

Joint Probability

Distributions

&

Marginal

Distributions

- **Example:**

Suppose a firm's employees in three departments :

10% are male & in dept. 1,

30% are male & in dept. 2,

20% are male & in dept. 3,

15% are female & in dept. 1,

20% are female & in dept. 2,

5% are female & in dept. 3

Then the ***joint probability distribution*** of gender & dept. is as in the table below:

	D ₁	D ₂	D ₃
M	0.10	0.30	0.20
F	0.15	0.20	0.05

Example continued: What is the probability that a randomly selected employee is *male*?

	D ₁	D ₂	D ₃
M	0.10	0.30	0.20
F	0.15	0.20	0.05

Example continued: What is the probability that a randomly selected employee is *male*?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	

Example continued: What is the probability that a randomly selected employee is *female*?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	

Example continued: What is the probability that a randomly selected employee is *female*?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40

Example continued: What is the probability that a randomly selected employee is in *dept.* 1?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40

Example continued: What is the probability that a randomly selected employee is in *dept.* 1?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25			

Example continued: What is the probability that a randomly selected employee is in *dept. 2*?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25			

Example continued: What is the probability that a randomly selected employee is in *dept. 2*?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25	0.50		

Example continued: What is the probability that a randomly selected employee is in *dept.* 3?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25	0.50		

Example continued: What is the probability that a randomly selected employee is in *dept.* 3?

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25	0.50	0.25	

Example continued: The *marginal distribution* of gender is in first & last columns (or left & right *margins* of the table) & gives the probability of each possibility for gender.

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25	0.50	0.25	

Example continued: The *marginal distribution* of department is in first & last rows (or top & bottom *margins* of the table) & gives the probability of each possibility for dept.

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25	0.50	0.25	

Notice that when you add the numbers in the last column or the last row, you must get one, because you're adding all the probabilities for all the possibilities.

	D ₁	D ₂	D ₃	
M	0.10	0.30	0.20	0.60
F	0.15	0.20	0.05	0.40
	0.25	0.50	0.25	1.00

Independence

- Two events are independent, if knowing that one event happened doesn't give you any information on whether the other happened.

- **Example:**

- A: It rained a lot in Beijing, China last year.

- B: You did well in your courses last year.

- These two events are independent

One of these events occurring tells you nothing about whether the other occurred.

**So in terms of probability, two events
A & B are independent if and only if**

● * $\Pr(A|B) = \Pr(A)$

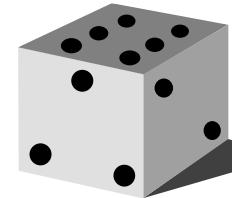
- Using the definition of conditional probability,
this statement is equivalent to
 - $\Pr(A \cap B) / \Pr(B) = \Pr(A).$
 - Multiplying both sides by $\Pr(B)$, we have
 - * $\Pr(A \cap B) = \Pr(A) \Pr(B).$
 - Dividing both sides by $\Pr(A)$, we have
 - $\Pr(A \cap B) / \Pr(A) = \Pr(B),$
 - which is equivalent to
 - * $\Pr(B|A) = \Pr(B).$
- This makes sense. If knowing about B tells us nothing about A, then knowing about A tells us nothing about B.

We now have 3 equivalent statements for 2 independent events A & B

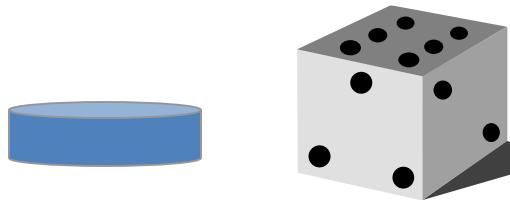
- $\Pr(A|B) = \Pr(A)$
- $\Pr(B|A) = \Pr(B)$
- $\Pr(A \cap B) = \Pr(A) \Pr(B).$
- The last equation says that you can calculate the probability that both of two independent events occurred by multiplying the separate probabilities.

Example: flip a fair coin & roll a fair die

- A: You get a H on the coin.
- B: You get a 6 on the die.
- Recall that we counted 12 possible outcomes for this experiment.
- Since the coin & the die are fair, each outcome is equally likely, & the probability of getting a H & a 6 is $1/12$.



Example continued



- The probability of a H on the coin is $1/2$
 - The probability of a 6 on the die is $1/6$.
 - $\Pr(H) \Pr(6) = (1/2)(1/6)$
 - $= 1/12$
 - $= \Pr(H \cap 6),$
- & we can see that these 2 events are independent of each other.

Mutually Exclusive

- Two events are mutually exclusive if you know that one occurred, then you know that the other could not have occurred.
- Example: You selected a student at random.
- A: The student is a male.
- B: The student is a female.
- These 2 events are mutually exclusive, because you know that if A occurred, B did not.

Mutually exclusive events are NOT independent!

- Remember that for independent events, knowing that one event occurred tells you nothing about whether the other occurred.
- For mutually exclusive events, knowing that one event occurred tells you that the other definitely did not occur!

The Birthday Problem

The Birthday Problem

- What is the probability that in a group of k people at least two people have the same birthday?
- (We are going to ignore leap day, which complicates the analysis, but doesn't have much effect on the answer.)

For our group of k people, let
 $p = \Pr(\text{at least 2 people have the same birthday}).$

- At least 2 people having the same birthday is the complement (opposite) of no 2 people having the same birthday, or everyone having different birthdays.
- It's easier to calculate the probability of different birthdays.
- So we can do that & then subtract the answer from one to get the probability we want.

$p = 1 - \Pr(\text{all different birthdays})$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365}{\dots}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364}{\dots}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364 \cdot 363}{}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364 \cdot 363 \cdot 362 \cdot 361 \cdot \dots (\text{until you have } k \text{ #'s})}{}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364 \cdot 363 \cdot 362 \cdot 361 \cdot \dots (\text{until you have } k \text{ #'s})}{365}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364 \cdot 363 \cdot 362 \cdot 361 \cdot \dots (\text{until you have } k \text{ #'s})}{365 \cdot 365}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364 \cdot 363 \cdot 362 \cdot 361 \cdot \dots (\text{until you have } k \text{ #'s})}{365 \cdot 365 \cdot 365}$$

$$p = 1 - \Pr(\text{all different birthdays})$$

$$= 1 - \frac{(\# \text{ of ways } k \text{ people can pick different bdays})}{(\# \text{ of ways } k \text{ people can pick any bdays})}$$

$$= 1 - \frac{365 \cdot 364 \cdot 363 \cdot 362 \cdot 361 \cdot \dots (\text{until you have } k \text{ #'s})}{365 \cdot 365 \cdot 365 \cdot 365 \cdot 365 \cdot \dots (\text{until you have } k \text{ #'s})}$$

- This is very messy, but you can calculate the answer for any number k .
- I have the answers computed for some sample values.

Birthday Problem Probabilities

	<u>k</u>	<u>p</u>
•	5	0.027
•	10	0.117
•	15	0.253
•	20	0.411
•	22	0.476
•	23	0.507
•	25	0.569
•	30	0.706
•	40	0.891
•	50	0.970
•	100	0.9999997

Any Questions?



COMP3055

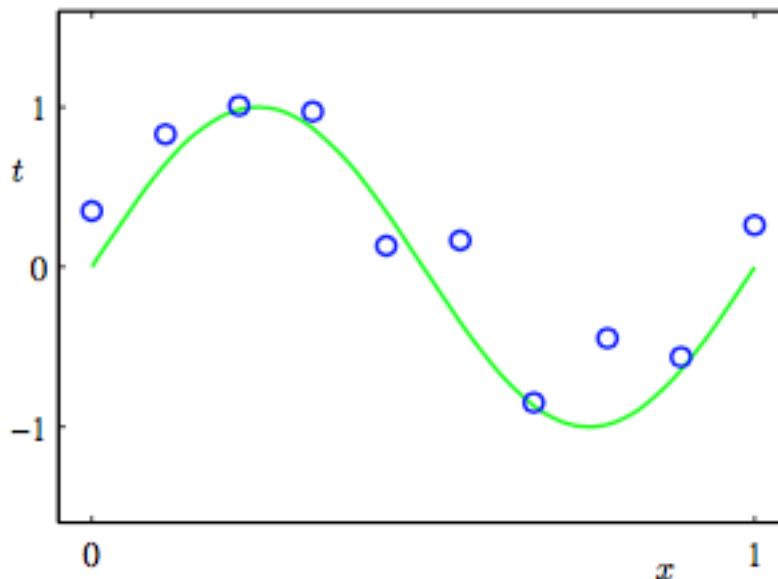
Machine Learning

Topic 5 – Machine Learning Theory and Practice

Ying Weng
2024 Autumn

Motivating Example

Polynomial Curve Fitting



Plot of a training data set of $N = 10$ points, each comprising an observation of the input variable x along with the corresponding target variable t . The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x , **without** knowledge of the green curve.

Classification VS Regression

Classification
predict a label (discrete value)

Regression
predict a response (continuous value)

Regression Problem

- $X \equiv (x_1, \dots, x_N)^T, T \equiv (t_1, \dots, t_N)^T$
- **Training set** is generated with $t_n = \sin(2\pi x_n) + b_n, n = 1, 2, \dots, N$, where b_n is **random noise** having a Gaussian Distribution.
- **Goal**: predict the target value of t for some new input value $x \rightarrow$ implicitly trying to discover the underlying function $\sin(2\pi x_n)$.
- Generalize from a finite data set ($N=10$)
- **Uncertainty**: the observed data are corrupted with noise.

Polynomial Function

- We fit the training data using a **polynomial function** (linear models) of the form:

$$y(x, \mathbf{w}) = w_0 + w_1 x^1 + \cdots + w_M x^M = \sum_{j=0}^M w_j x^j$$

- Where M is the *order of the polynomial* (degree of freedom), and the *polynomial coefficients* w_0, w_1, \dots, w_M are collectively denoted by the vector \mathbf{w}

Solve a Polynomial Function

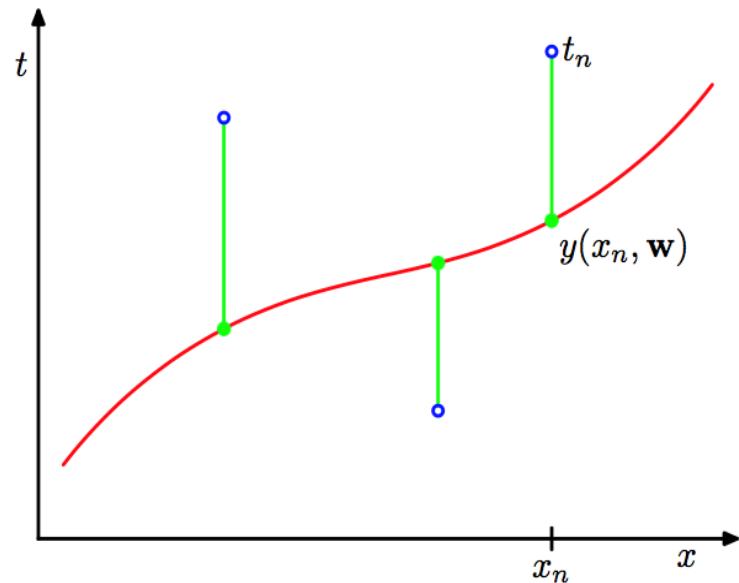
Target: find the values of
polynomial coefficients

- Step 1: Fit the polynomial to the training data
- Step 2: minimize the *error function*

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

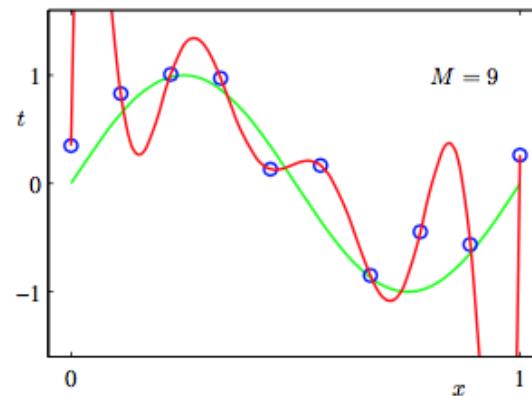
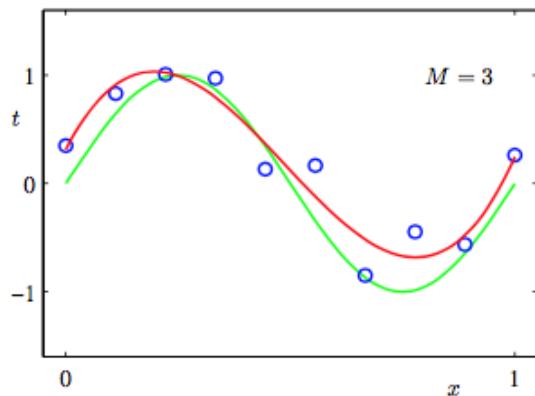
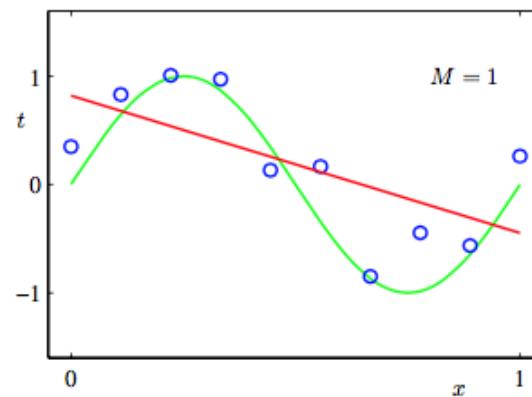
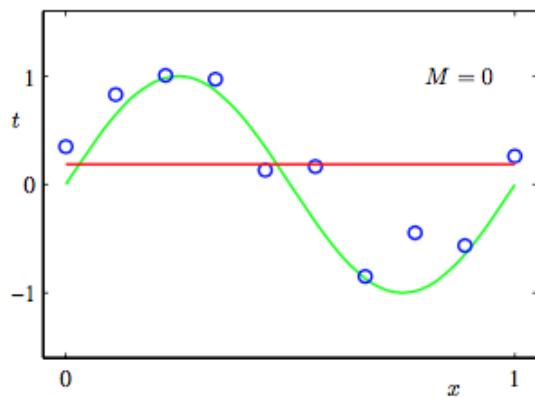
Where $E(\mathbf{w})$ measures the misfit between the function $y(x, \mathbf{w})$ and the training set data points

$\frac{1}{2}$ is included for later convenience. $E(\mathbf{w})=0$ if and only if the function $y(x, \mathbf{w})$ were to pass exactly through each training data point. The error function is a quadratic function of \mathbf{w} , there is only one solution.



Model Selection

Choosing order M of the polynomial



Model Selection

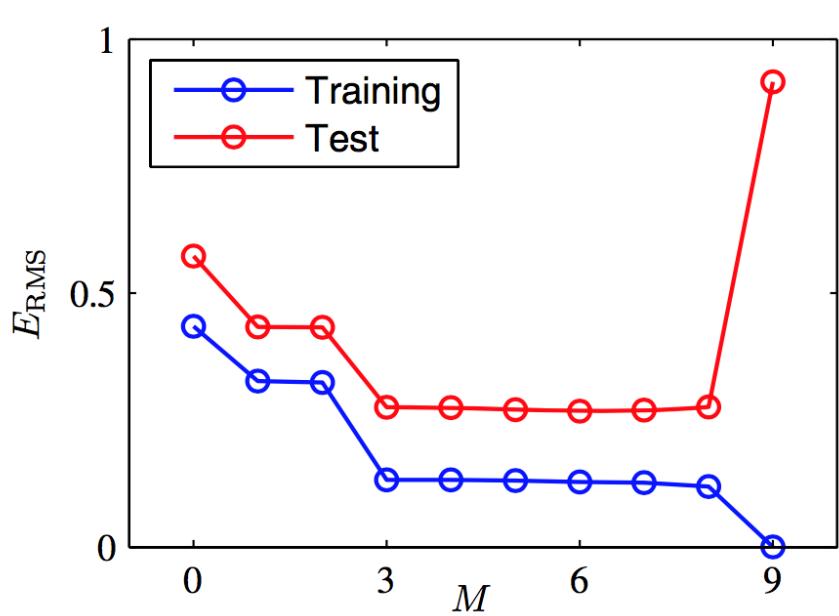
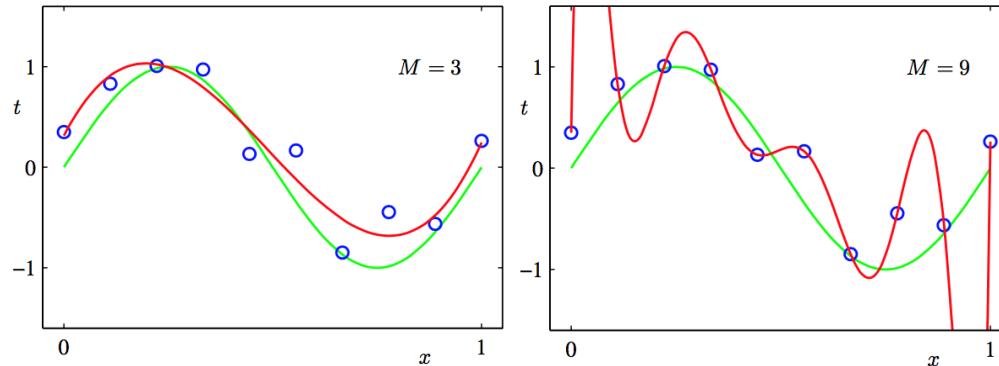
- **Goal:** achieve good *generalization* by making accurate predictions for new data.

- We use Root-mean-square (RMS) error on data

$$E_{RMS} = \sqrt{2E(w^*)/N}$$

- Where N allows us to compare different sizes of data set, and w^* is the *solution* of minimizing $E(w)$ (*hypothesis*).
- It measures how well the model w^* doing in predicting the values of t for new data observations of x .

Overfitting



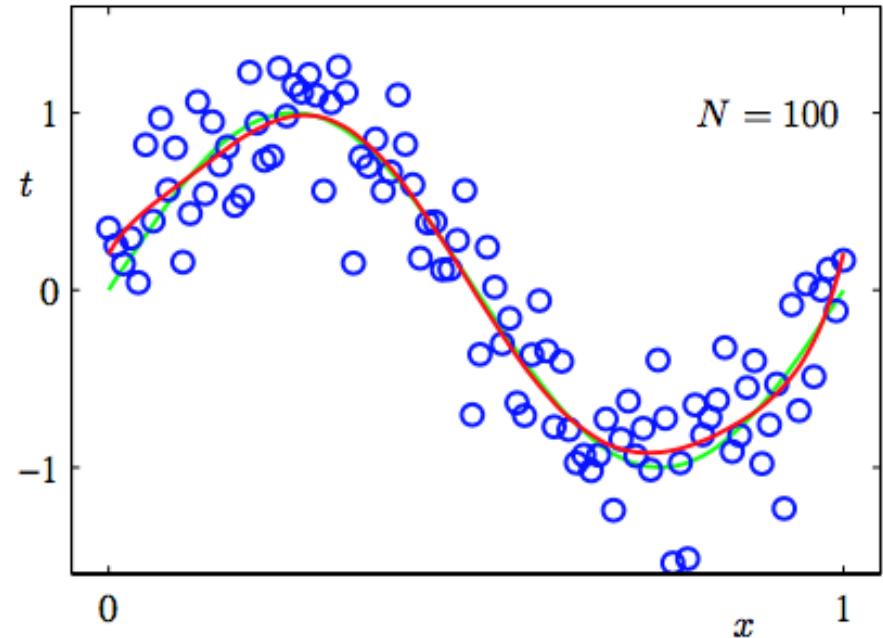
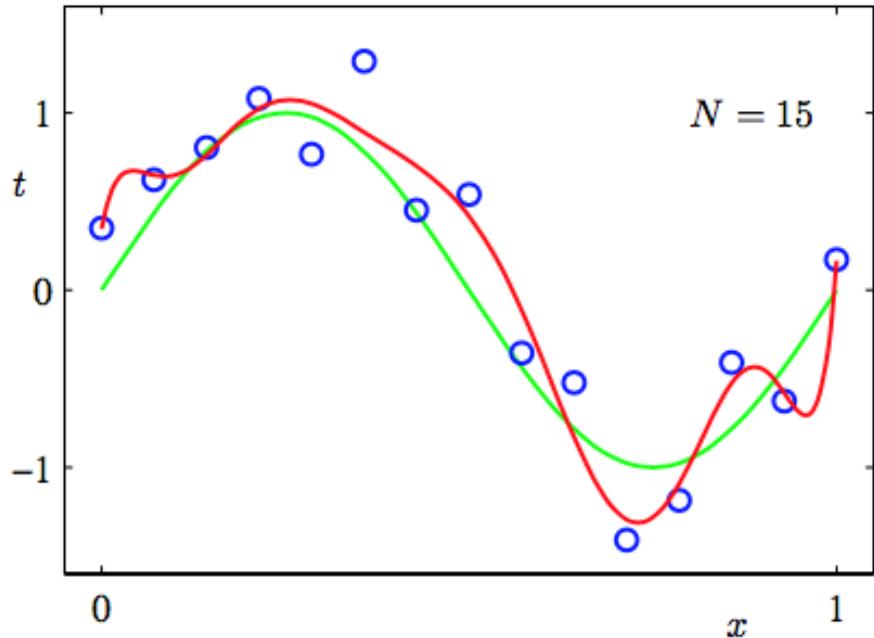
	$M = 0$	$M = 1$	$M = 6$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Overfitting

Overfitting can occur when:

- Learning is performed for **too long** (e.g. in Neural Networks).
- The examples in the training set are **not representative** of all possible situations (is usually the case!).
- Model parameters are adjusted to **uninformative features** in the training set that have no causal relation to the true underlying target function!

Overfitting



Increasing the size of the data set reduces the overfitting problem.

Regularization

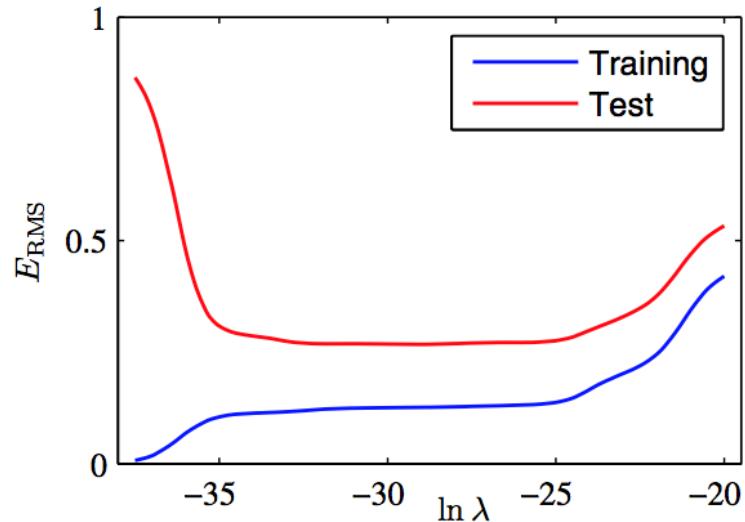
- *Regularization* can control the overfitting phenomenon, by adding **penalty term** to the error function to discourage the coefficients from reaching large values.

$$E(w) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2} \|w\|^2$$

- Where $\|w\|^2 \equiv w^T w = w_0^2 + w_1^2 + \dots + w_M^2$, and the coefficient λ governs the relative importance of the regularization term compared with the sum-of-squares error term.

Regularization

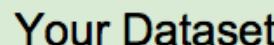
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01



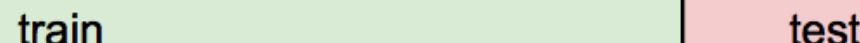
- Table of the coefficients w^* for $M = 9$ polynomials with various values for the regularization parameter λ .
- λ controls the *effective complexity* of the model and hence determines the *degree of overfitting*

Cross Validation

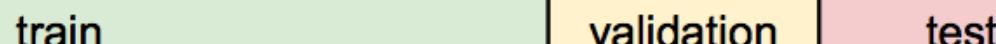
- **Idea #1:** Choose hyperparameters that work best on the data



- **Idea #2:** Split data into **train** and **test**, choose hyperparameters that work best on test data



- **Idea #3:** Split data into **train**, **val**, and **test**; choose hyperparameters on **val** and evaluate on **test**



Cross Validation

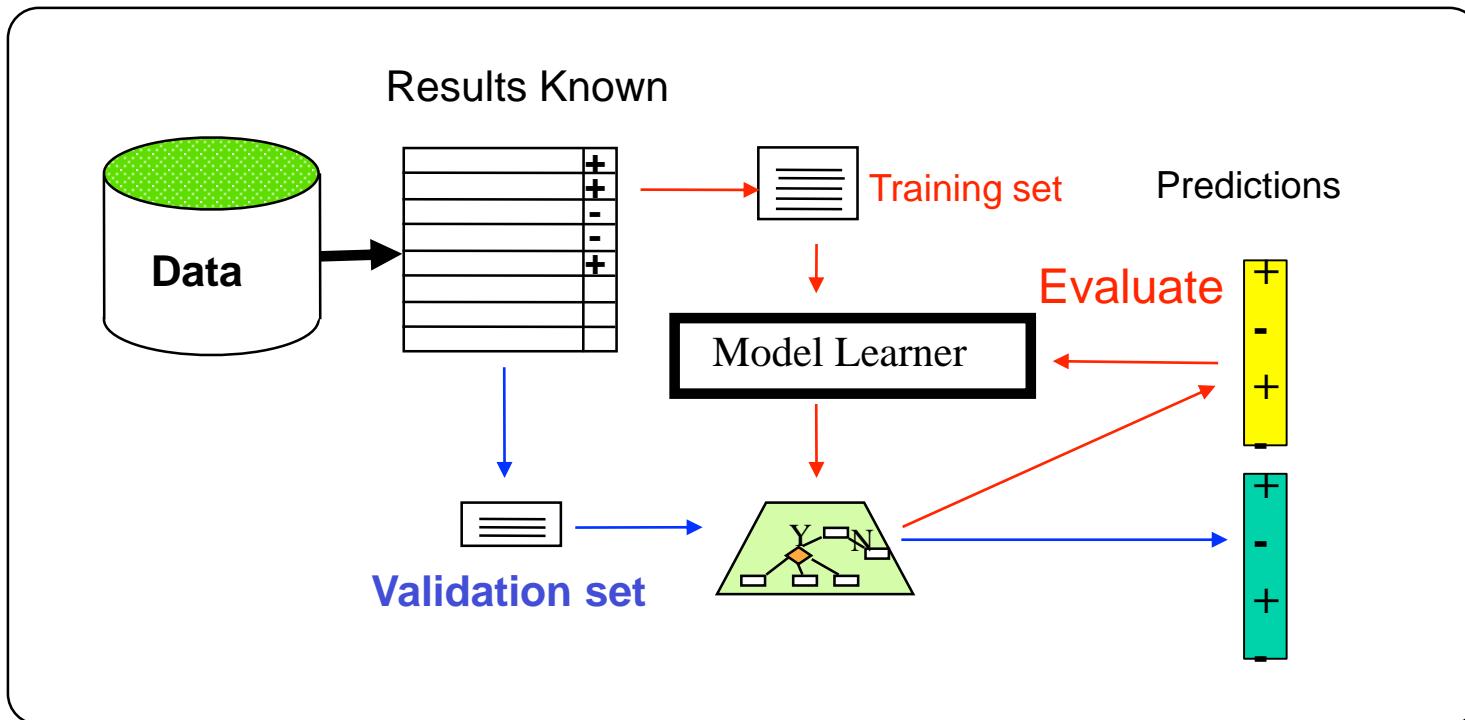
- **Idea #4: Cross-Validation:** Split data into **folds**, try each fold as validation and **average the results**

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Cross Validation

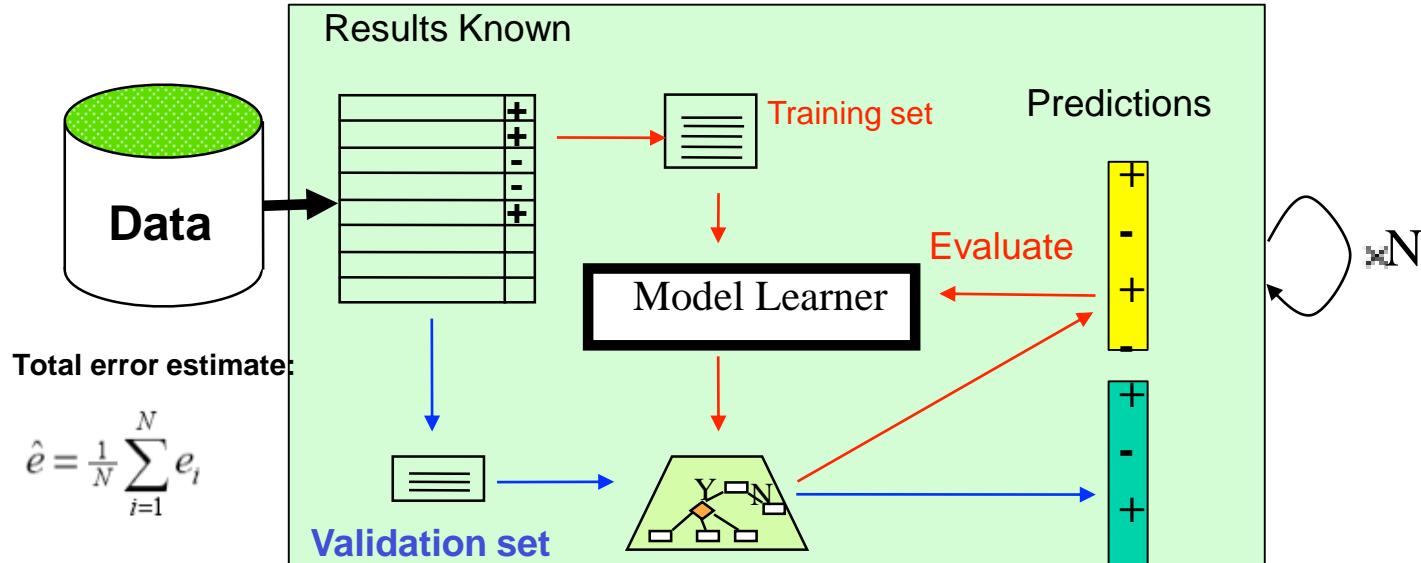
- **Cross Validation** is often used to counter overfitting.
- Partition the dataset into S groups, with $(S-2)$ training sets, a validation set and a testing set.
 - The training set is used to determine the coefficients w
 - The validation set is used to optimize the model complexity (hyperparameters, either M or λ in the previous example)
 - The testing set is used to evaluate the final selected mode
- The procedure is then repeated for all S possible choices, the performance scores from the S runs are then averaged.

Evaluation procedure



- For large datasets, a single split is usually sufficient.
- For smaller datasets, rely on cross validation

Cross validation procedure



- Split the data into training, validation and test sets in a repeated fashion.
- Estimate the total error as the average of each fold error.

Classification Measures - Error Rate

- Common performance measure for classification problems
 - Success: instance's class is predicted correctly (True Positives (**TP**) / Negatives (**TN**))
 - Error: instance's class is predicted incorrectly (False Positives (**FP**) / Negatives (**FN**))
 - False positives - **Type I error**. False Negative - **Type II error**
- Classification **error rate**: proportion of instances misclassified over the whole set of instances
$$\text{Error Rate} = (\text{FP} + \text{FN}) / (\text{P} + \text{N})$$
a.k.a, **accuracy**
$$\text{Accuracy} = 1 - \text{Error Rate}$$
- Classification Error Rate on the Training Set can be too optimistic!

Unbalanced data

- Balanced set: (roughly) equal number of positive / negative examples:

Classifier	TP	TN	FP	FN	Recall Rate
A	25	25	25	25	50%
B	37	37	13	13	74%

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

Unbalanced data

- Unbalanced set: unequal number of positive / negative examples

Classifier	TP	TN	FP	FN	Recall Rate
A	25	75	75	25	50%
B	0	150	0	50	0%

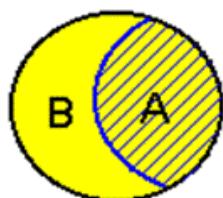
Classifier B cannot predict any positive examples!

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

Classification Measures

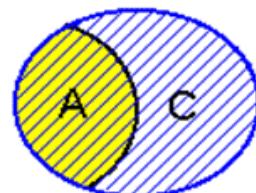
- Error Rate (Accuracy)
- Precision/Recall
- F-measure
- ROC curve
- Confusion matrix
- ...

Recall/Precision



B: Number of positive examples not retrieved
A: Number of positive examples retrieved

$$\text{RECALL}: \frac{A}{A+B} \times 100\%$$



C: Number of negative examples retrieved
A: Number of positive examples retrieved

$$\text{PRECISION}: \frac{A}{A+C} \times 100\%$$

More insight over a classifier's behavior

For the positive class:

Classifier A: Recall = 50%, Precision = 25%

Classifier B: Recall = 0%, Precision = 0%

Classifier B is useless!

F-measure

- Comparing different approaches is difficult when using multiple evaluation measures (e.g. Recall and Precision)
- F-measure combines recall and precision into a single measure:

$$f_\beta = \frac{(1 + \beta^2) \frac{P}{R}}{(\beta^2 P) + R}$$

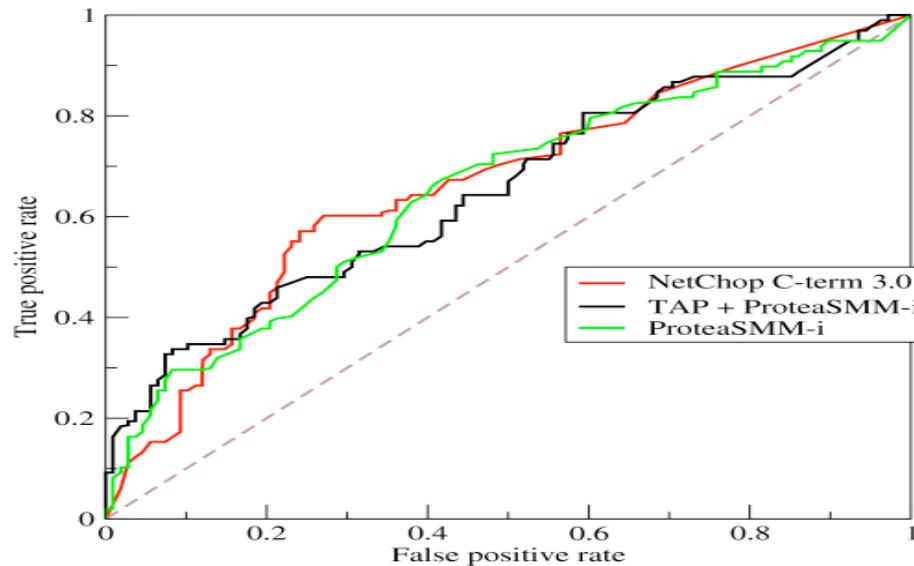
The diagram illustrates the components of the F-beta score. It features two green-outlined circles, one labeled 'Precision' and the other 'Recall'. A green line connects the center of the 'Precision' circle to the center of the 'Recall' circle. Overlaid on these circles is a larger green-outlined circle containing the letters 'P' and 'R' respectively. This visual metaphor represents how precision and recall are combined into a single F-beta score.

β is a non-negative real values

- We often use f1 measure.

ROC curves

- Receiver Operator Characteristic (ROC) curves plot TP vs FP rates



- Can be achieved by e.g. varying decision threshold of a classifier
- Area under the curve is often used as measure of goodness

Confusion matrix

- A visualization tool used to present the results attained by a learner.
- Easy to see if the system is commonly mislabeling one class as another.

Predicted True	A	B	C
A	5	3	0
B	2	3	1
C	0	2	11

Curse of Dimensionality

- As dimension D increases, the number of independent coefficients grows proportionally

$$y(x, \mathbf{w}) = w_0 + w_1 x^1 + \cdots + w_M x^M = \sum_{j=0}^M w_j x^j$$


$$y(x, \mathbf{w}) = w_0 + \sum_{i=1}^D w_i x_i + \sum_{i=1}^D \sum_{j=1}^D w_{ij} x_i x_j + \sum_{i=1}^D \sum_{j=1}^D \sum_{k=1}^D w_{ijk} x_i x_j x_k \dots$$

- The amount of data needed to support the result often grows exponentially with the dimensionality.

Curse of Dimensionality

- Consider a sphere of radius $r = 1$ in a space of D dimensions, what is the fraction of the volume of the sphere that lies between radius $r = 1 - \epsilon$ and $r = 1$?
- The volume of a sphere of radius r in D dimensions must scale as r^D , we have

$$V_3(r) = \frac{4}{3}\pi r^3$$

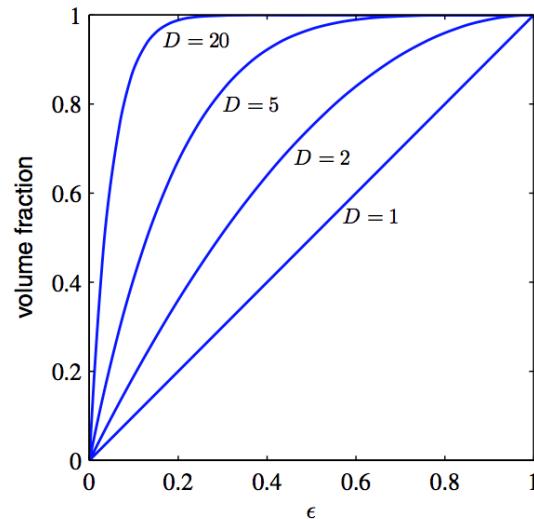
$$V_D(r) = K_D r^D$$

Curse of Dimensionality

- Where the constant K_D depends only on D, thus the required volume fraction is given by

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = 1 - (1 - \epsilon)^D$$

- For large D , the fraction *tend to 1* even for small values of ϵ .



Curse of Dimensionality

- For large D , the fraction *tend to 1* even for small values of ϵ .
- In space of high dimensionality, most of the volume of a sphere is concentrated in a *thin shell near the surface*.
- Distance functions losing their *usefulness* (for the nearest-neighbor criterion in feature-comparison algorithms, for example) in high dimensions.



University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 6 – Instance Based Learning

Ying Weng
2024 Autumn

Instance Based Learning

- Directly compare new problem instances with instances seen in training
- No explicit modeling of the training data
- Complexity grows with the training data
- Classical instance based learning technique
 - K Nearest Neighbour

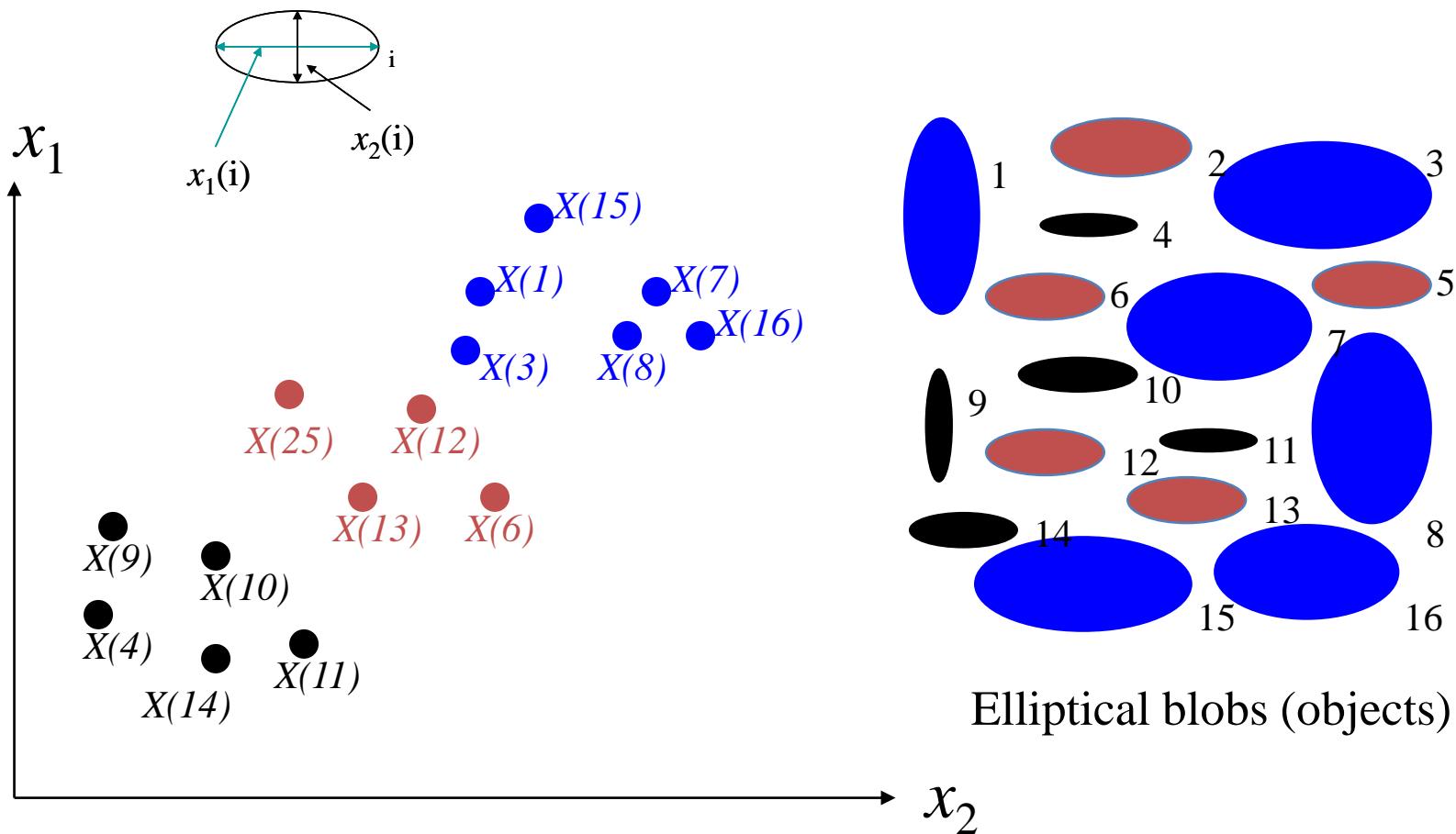
K-Nearest Neighbour Model

Example

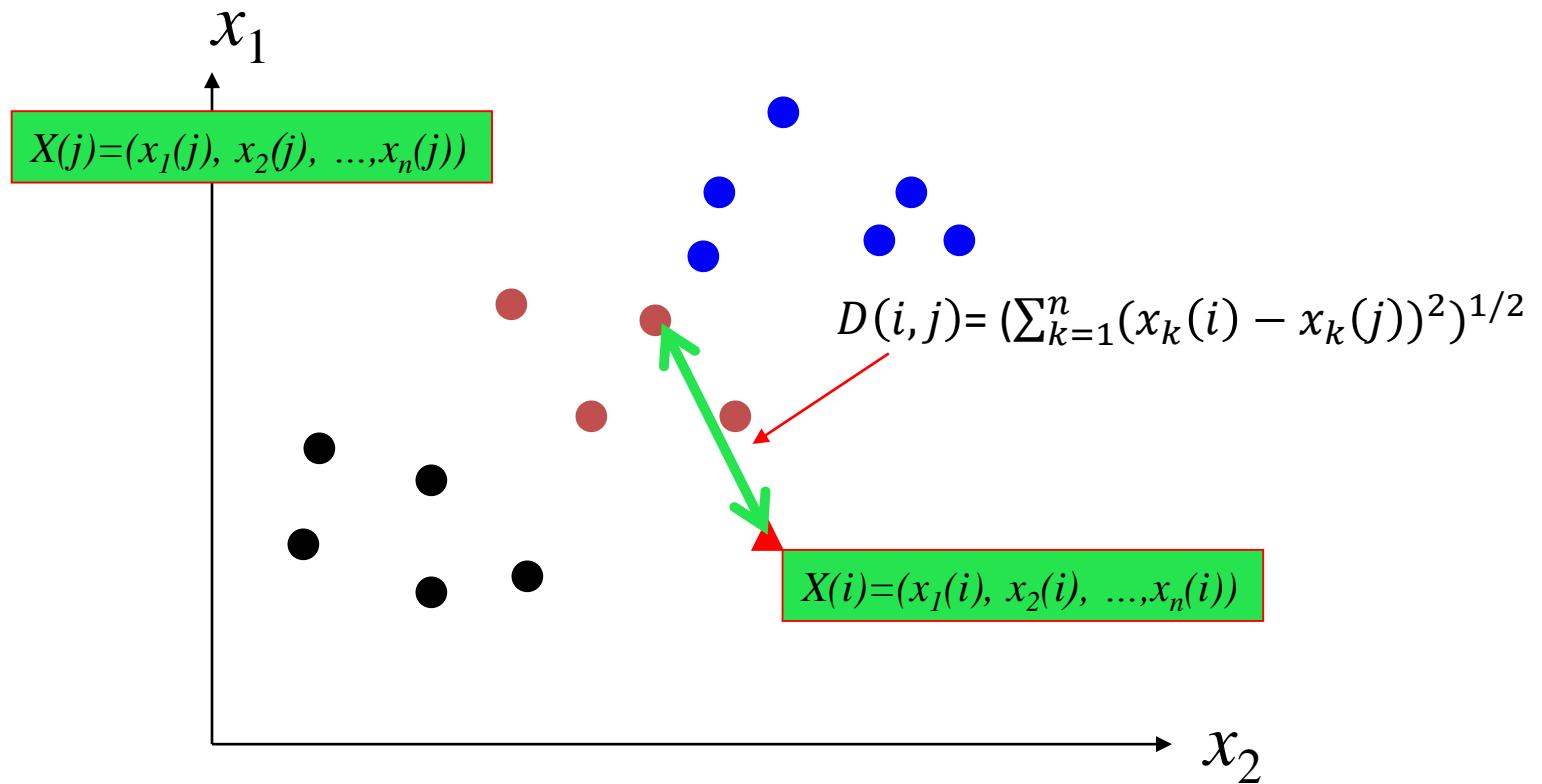
Classify whether a customer will respond to a survey question using a 3-Nearest Neighbour classifier.

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

Objects, Feature Vectors, Points



Nearest Neighbours



Nearest Neighbour Algorithm

Given training data $(X(1), D(1)), (X(2), D(2)), \dots, (X(N), D(N))$,

Define a distance metric between points in inputs space.

Common measures are:

Euclidean Distance

$$D(i, j) = (\sum_{k=1}^n (x_k(i) - x_k(j))^2)^{1/2}$$

K-Nearest Neighbour Model

Given test point X

- Find the K nearest training inputs to X
- Denote these points as

$$(X(1), D(1)), (X(2), D(2)), \dots, (X(k), D(k))$$



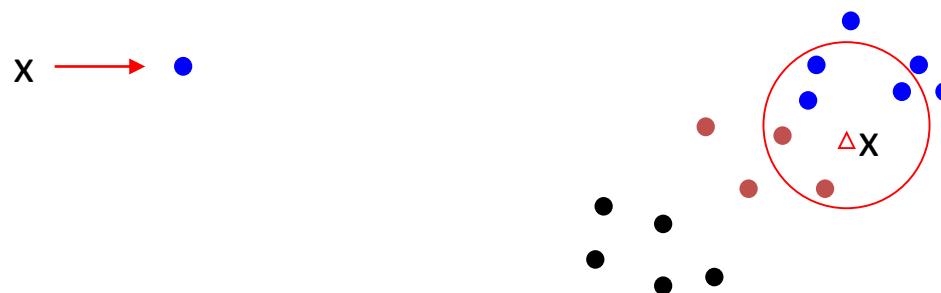
K-Nearest Neighbour Model

Instance based learning

The class identification of X

$Y = \text{most common class in set } \{D(1), D(2), \dots, D(k)\}$

Majority rule



K-Nearest Neighbour Model

Example

Classify whether a customer will respond to a survey question using a 3-Nearest Neighbour classifier.

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

K-Nearest Neighbour Model

Example

3-Nearest Neighbours

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

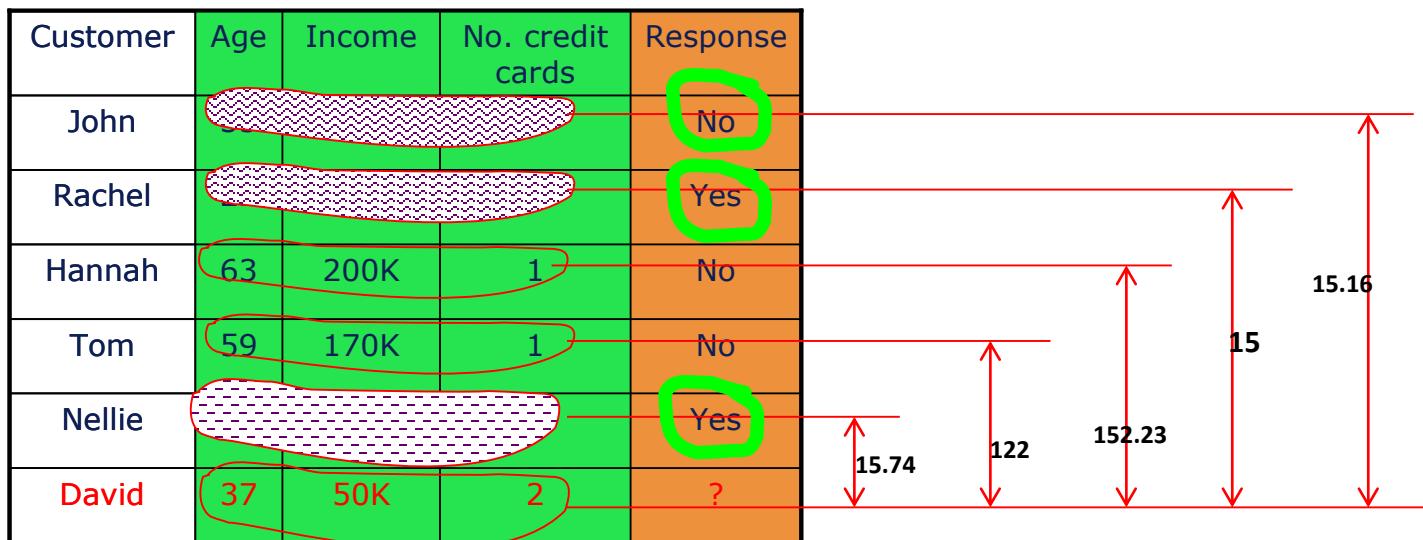
The diagram illustrates the calculation of Euclidean distances between David (37, 50K, 2, ?) and his three nearest neighbors. Red arrows point from David's row to each neighbor's row, and red double-headed vertical arrows indicate the distance between David and each neighbor.

- Distance between David and John: 15.74
- Distance between David and Rachel: 122
- Distance between David and Hannah: 152.23
- Distance between David and Tom: 15
- Total distance between David and all neighbors: 15.16

K-Nearest Neighbour Model

Example

3-Nearest Neighbours



Three nearest ones to David are: No, Yes, Yes

K-Nearest Neighbour Model

Example

3-Nearest Neighbors

Customer	Age	Income	No. credit cards	Response
John	63, 200K, 1			No
Rachel	59, 170K, 1			Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	37, 50K, 2			Yes
David	37	50K	2	Yes

The diagram illustrates the Euclidean distances between David's features and those of his three nearest neighbors. Red arrows point from David's row to each neighbor's row, with their respective distances labeled:

- Distance to Hannah: 15.74
- Distance to Tom: 122
- Distance to Nellie: 152.23
- Total distance to all three neighbors: 15
- Total distance to all four neighbors (including himself): 15.16

Three nearest ones to David are: No, Yes, Yes

K-Nearest Neighbour Model

Picking K

- Use *N fold cross validation* – Pick K to minimize the cross validation error
- For each of N training example
 - Find its K nearest neighbours
 - Make a classification based on these K neighbours
 - Calculate classification error
 - Output average error over all examples
- Use the K that gives lowest average error over the N training examples

K-Nearest Neighbour Model

Q&A

For the example we saw earlier, pick the best K from the set {1, 2, 3} to build a K-NN classifier.

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

K-Nearest Neighbour Model

Q&A

For the example we saw earlier, pick the best K from the set {1, 2, 3} to build a K-NN classifier.

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

The diagram illustrates the Euclidean distances between the features of David and the other five customers. Red arrows connect each row to a vertical axis on the right, with the corresponding distance values labeled:

- Distance between David and John: 15.74
- Distance between David and Rachel: 122
- Distance between David and Hannah: 152.23
- Distance between David and Tom: 15
- Distance between David and Nellie: 15.16

Further Readings

Chapter 8, T. M. Mitchell, Machine Learning,
McGraw-Hill International Edition, 1997

Any Questions?





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 6 – Bayesian Learning

Ying Weng
2024 Autumn

Probability

- The world is a very uncertain place.
- 30 years of Artificial Intelligence research danced around this fact.
- And then a few AI researchers decided to use some ideas from the eighteenth century.

Random Variables

- A variable whose possible values are outcomes of a random phenomenon.
- Denotes a quantity (event) that is uncertain.
- Maybe result of experiment (flipping a coin) or a real world observation (measuring attendance rate).
- If observe several instances of a random variable, we get different values.
- Some values occur more than others and this information is captured by a probability distribution.

Random Variables

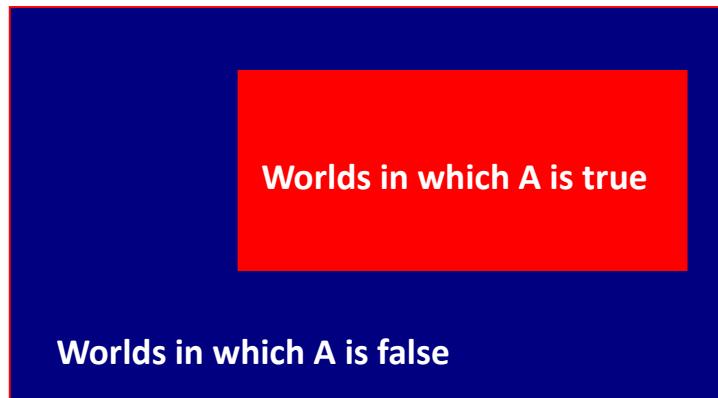
- A is a Boolean-valued random variable
 - if A denotes such event that there is some degree of uncertainty as to whether A occurs.
- Examples
 - A = The US president in 2024 will be Trump
 - A = You wake up tomorrow with a headache
 - A = You have won a prize

Probabilities

We write $P(A)$ as

“the fraction of possible worlds in which A is true”

Event space of all
possible worlds



$P(A) = \text{Area of}$
 red rectangle

The whole area is 1

All probabilities between 0 and 1

$$0 \leq P(A) \leq 1$$

True proposition has probability 1, false has probability 0.

$$P(\text{not } A) = P(\sim A) = 1 - P(A)$$

Multivalued Random Variables

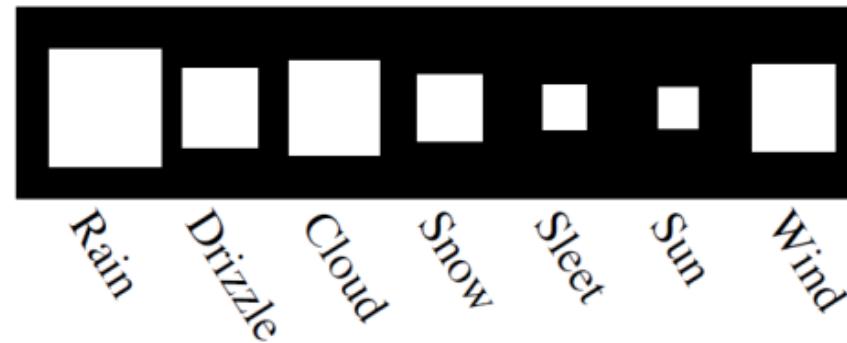
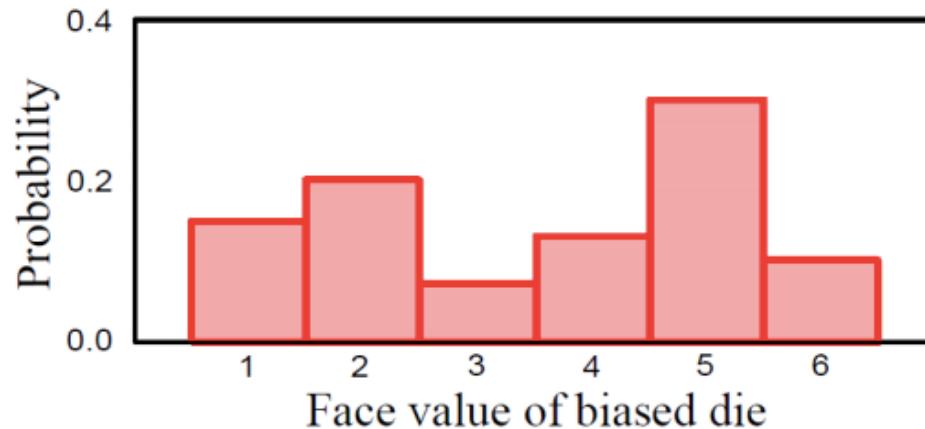
Suppose a random variable A can take on exactly one value out of $\{v_1, v_2, \dots, v_k\}$

$$P(A = v_i \cap A = v_j) = 0 \quad \text{if } i \neq j$$

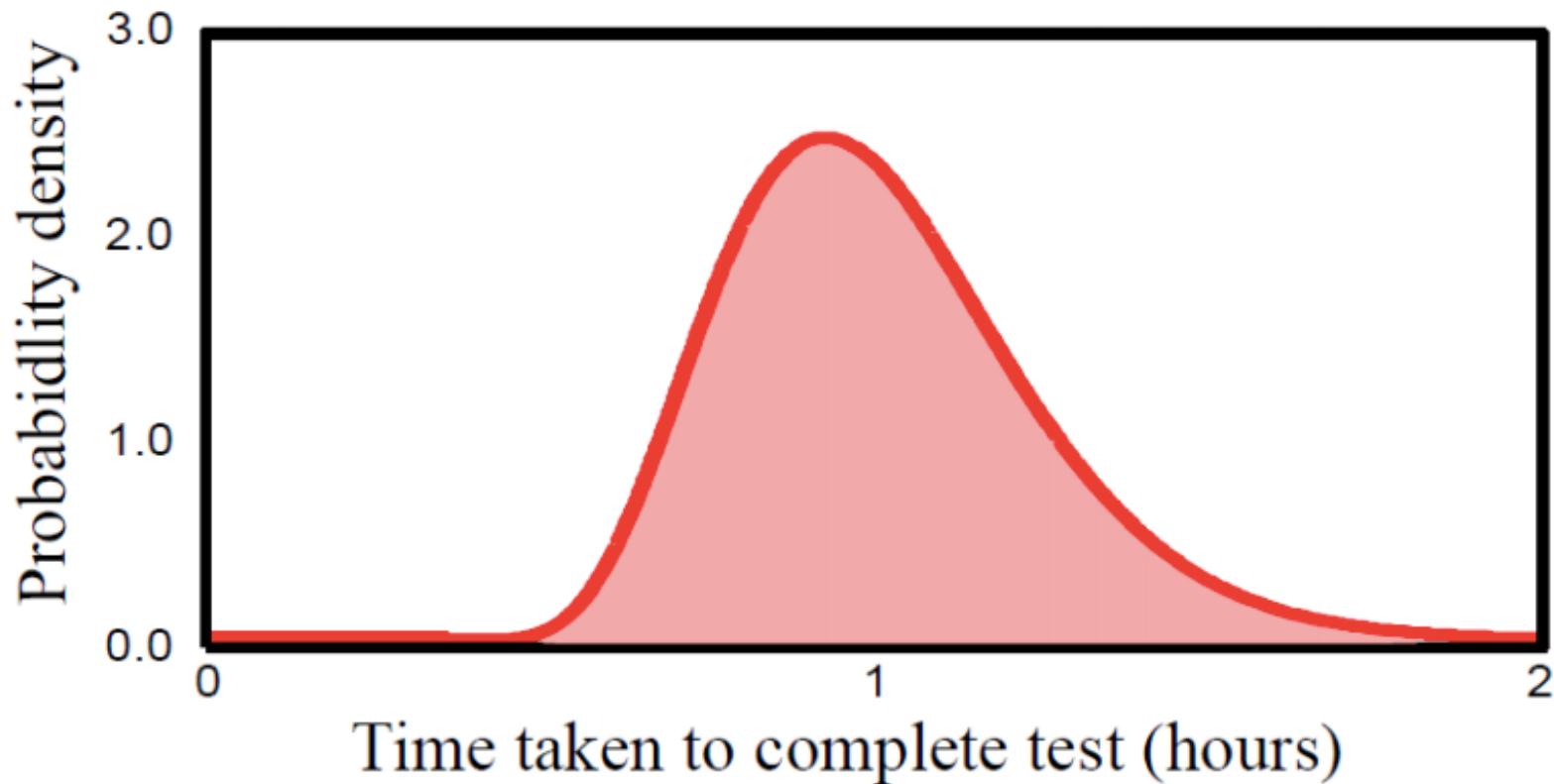
$$P(A = v_1 \cup A = v_2 \cup \dots \cup A = v_k) = 1$$

Discrete vs Continuous Random Variable

Depends on weather outcome values are discrete or continuous



Continuous Random Variables

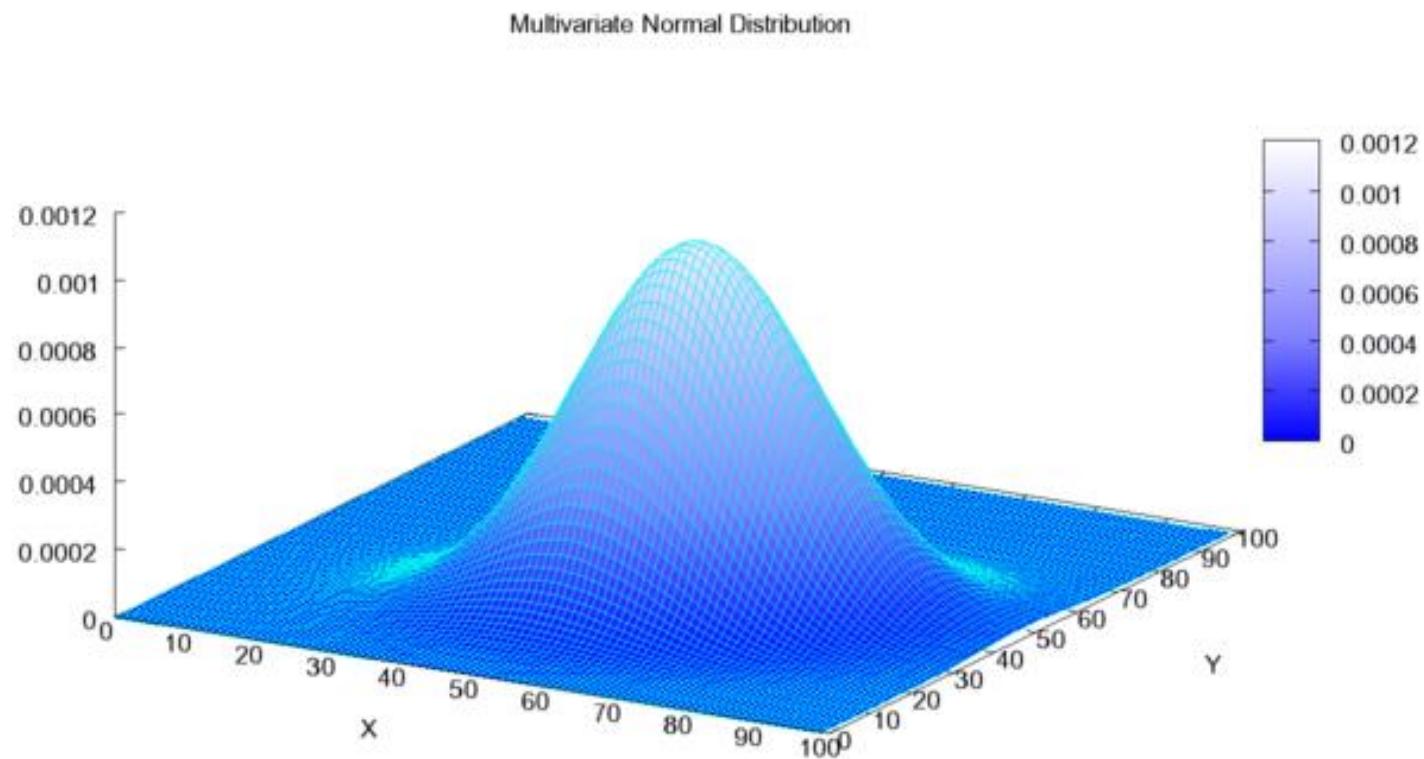


Note: Instead of summing over all possible outcome in discrete random variable, integrating the probability over all outcome should be 1 in continuous random variable.

Joint Probability

- Consider two random variables A and B.
- If we observe multiple paired instances, some combinations of outcomes are more likely than others.
- This is captured by joint probability $P(A \cap B)$

Joint Probability



Marginalization

$$P(A = v_1 \cup A = v_2 \cup \dots \cup A = v_i) = \sum_{j=1}^i P(A = v_j)$$

$$P(B \cap [A = v_1 \cup A = v_2 \cup \dots \cup A = v_i]) = \sum_{j=1}^i P(B \cap A = v_j)$$

$$P(B) = \sum_{j=1}^k P(B \cap A = v_j)$$



Marginalization: We can recover probability distribution of any variable in a joint distribution by summing over the other variables

Conditional Probability

$P(A/B)$ = Fraction of worlds in which B is true that also have A true.

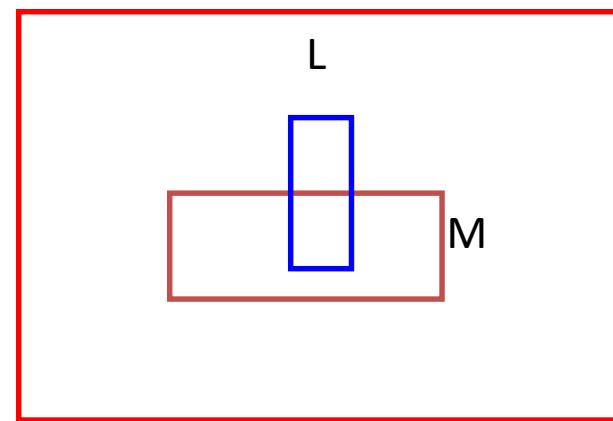
M = “Miss 9am lecture”

L = “Go to bed late”

$$P(M) = 1/4$$

$$P(L) = 1/10$$

$$P(M|L) = 1/2$$



“Missing 9am class is rare and going to bed late is rarer. But if you go to bed late, there is a 50-50 chance you will miss 9am lecture.”

Conditional Probability

$P(A|B)$ = Fraction of worlds in which B is true that also have A true.

$P(M|L)$ = Fraction of late sleeper worlds in which also miss 9am lecture

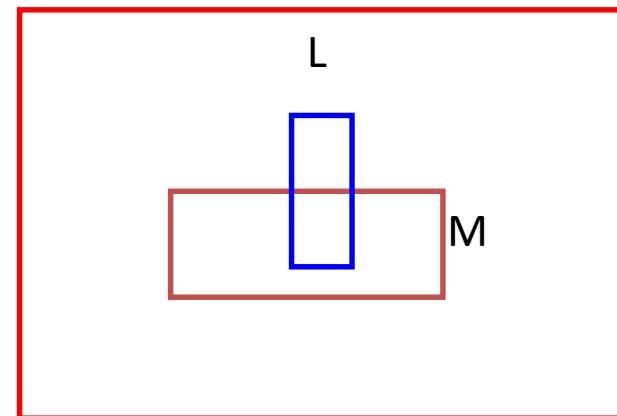
$$= \frac{\text{#worlds of late sleeper and miss 9am class}}{\text{#worlds late sleeper}}$$

$$= \frac{\text{Area of "M and L" region}}{\text{Area of "L" region}}$$

$$= \frac{P(M \cap L)}{P(L)}$$

M = “Miss 9am lecture”
L = “Go to bed late”

$$\begin{aligned} P(M) &= 1/4 \\ P(L) &= 1/10 \\ P(M|L) &= 1/2 \end{aligned}$$



Definition of Conditional Probability

Conditional probability definition

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Hence, the Chain Rule

$$P(A \cap B) = P(A|B)P(B)$$

Probabilistic Inference

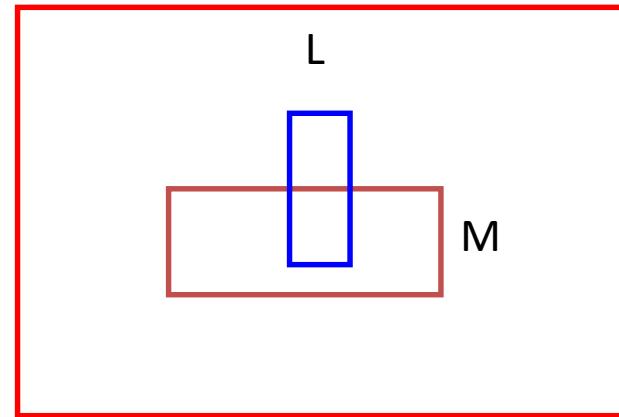
M = “Miss 9am lecture”

L = “Go to bed late”

$$P(M) = 1/4$$

$$P(L) = 1/10$$

$$P(M | L) = 1/2$$



One day you miss my 9am class.

I think, “50% of late sleepers miss my 9am lecture,
so this student (you) must have a 50-50 chance of
going to bed late last night.”

Is this thought reasonable?

Probabilistic Inference

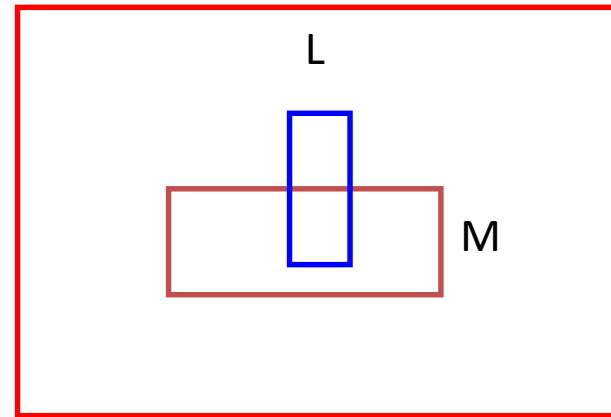
M = “Miss 9am lecture”

L = “Go to bed late”

$$P(M) = 1/4$$

$$P(L) = 1/10$$

$$P(M|L) = 1/2$$



One day you miss my 9am class.

I think, “50% of late sleepers miss my 9am lecture, so this student (you) must have a 50-50 chance of going to bed late last night.” $\leftarrow P(L|M)$

Is this thought reasonable?

$$\begin{aligned} P(M \cap L) &= P(M|L)P(L) \\ &= \frac{1}{2} * \frac{1}{10} = \frac{1}{20} \end{aligned}$$

$$\begin{aligned} P(L|M) &= \frac{P(L \cap M)}{P(M)} \\ &= \frac{1/20}{1/4} = \frac{1}{5} \end{aligned}$$

Probabilistic Inference

M = “Miss 9am lecture”

L = “Go to bed late”

$$P(M) = 1/4$$

$$P(L) = 1/10$$

$$P(M | L) = 1/2$$

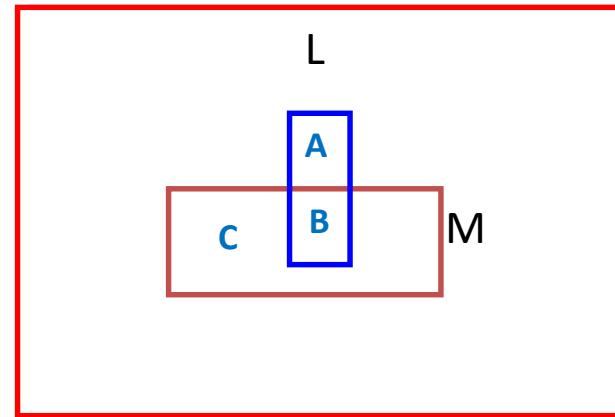
Area wise we have

$$P(L) = A + B$$

$$P(M) = B + C$$

$$P(M | L) = B / (A + B)$$

$$P(L | M) = B / (B + C) = P(M | L) P(L) / P(M)$$



Bayes Rule

- **Bayes, Thomas (1763)** An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London, **53:370-418**
- **Bayes Rule**

$$p(A | B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B | A)P(A)}{P(B)}$$



Bayesian Learning

$$p(h | x) = \frac{P(x | h)P(h)}{P(x)}$$

Understanding Bayes' rule

x = data

h = hypothesis (model)

- rearranging

$$p(h | x)P(x) = P(x | h)P(h)$$

$$P(x, h) = P(x, h)$$

the same joint probability
on both sides

$P(h)$: prior belief (probability of hypothesis h before seeing any data)

$P(x | h)$: likelihood (probability of the data if the hypothesis h is true)

$P(x) = \sum_h P(x | h)P(h)$: data evidence (marginal probability of the data)

$P(h | x)$: posterior (probability of hypothesis h after having seen the data d)

Choosing Hypotheses

- Generally, we want the most probable hypothesis(class label) given the observed data
 - Maximum a posteriori (**MAP**) hypothesis
 - Maximum likelihood (**ML**) hypothesis

Maximum A Posteriori (MAP)

- Maximum a posteriori (**MAP**) hypothesis

$$p(h | x) = \frac{P(x | h)P(h)}{P(x)}$$

$$h_{MAP} = \arg \max_{h \in H} p(h | x) = \arg \max_{h \in H} \frac{P(x | h)P(h)}{P(x)} = \arg \max_{h \in H} P(x | h)P(h)$$

Note $P(x)$ is independent of h , hence can be ignored.

Maximum Likelihood (ML)

$$h_{MAP} = \arg \max_{h \in H} P(x | h)P(h)$$

- Assuming that each hypothesis in H is equally probable, i.e., $P(h_i) = P(h_j)$, for all i and j, then we can drop $P(h)$ in MAP. $P(x|h)$ is often called the likelihood of data x given h . Any hypothesis that maximizes $P(x|h)$ is called the maximum likelihood hypothesis

$$h_{ML} = \arg \max_{h \in H} P(x | h)$$

An Illustrating Example

Classifying days according to whether someone will play tennis.

Each day is described by the attributes, Outlook, Temperature, Humidity and Wind.

Based on the training data in the table, classify the following instance

Outlook = sunny

Temperature = cool

Humidity = high

Wind = strong

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

An Illustrating Example

Training sample pairs (X, D)

$X = (x_1, x_2, \dots, x_n)$ is the feature vector representing the instance.

$D = (d_1, d_2, \dots, d_m)$ is the desired (target) output of the classifier

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

An Illustrating Example

Training sample pairs (X, D)

$X = (x_1, x_2, \dots, x_n)$ is the feature vector representing the instance.

$n = 4$

$x_1 = \text{outlook} = \{\text{sunny}, \text{overcast}, \text{rain}\}$

$x_2 = \text{temperature} = \{\text{hot}, \text{mild}, \text{cool}\}$

$x_3 = \text{humidity} = \{\text{high}, \text{normal}\}$

$x_4 = \text{wind} = \{\text{weak}, \text{strong}\}$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

An Illustrating Example

Training sample pairs (X, D)

$D = (d_1, d_2, \dots, d_m)$ is the desired
(target) output of the classifier

$m = 1$

$d = \text{Play Tennis} = \{\text{yes}, \text{no}\}$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Bayesian Classifier

- The Bayesian approach to classifying a new instance X is to assign it to the most probable target value Y (MAP classifier)

$$\begin{aligned}Y &= \arg \max_{d_i \in d} p(d_i | X) \\&= \arg \max_{d_i \in d} p(d_i | x_1, x_2, x_3, x_4) \\&= \arg \max_{d_i \in d} \frac{p(x_1, x_2, x_3, x_4 | d_i) P(d_i)}{p(x_1, x_2, x_3, x_4)} \\&= \arg \max_{d_i \in d} p(x_1, x_2, x_3, x_4 | d_i) P(d_i)\end{aligned}$$

Bayesian Classifier

$$Y = \arg \max_{d_i \in d} p(x_1, x_2, x_3, x_4 | d_i) P(d_i)$$

$P(d_i)$ is easy to calculate: simply counting how many times each target value d_i occurs in the training set.

$$P(d = yes) = 9/14$$

$$P(d = no) = 5/14$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Bayesian Classifier

$$Y = \arg \max_{d_i \in d} p(x_1, x_2, x_3, x_4 | d_i) P(d_i)$$

$P(x_1, x_2, x_3, x_4 | d_i)$ is much more difficult to estimate.

In this simple example, there are $3 \times 3 \times 2 \times 2 \times 2 = 72$ possible terms.

To obtain a reliable estimate, we need to see each terms many times.

Hence, we need a very, very large training set! (which in most cases is impossible to get).

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Naïve Bayes Classifier

Naïve Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.

This means, we have

$$P(x_1, x_2, \dots, x_n | d_i) = \prod_i^n P(x_i | d_i)$$

Naïve Bayes Classifier

$$Y = \arg \max_{d_i \in d} P(d_i) \prod_{k=1}^4 P(x_k | d_i)$$

Back to the Example

Naïve Bayes Classifier

$$Y = \arg \max_{d_i \in D} \prod_{k=1}^4 P(x_k | d_i) P(d_i)$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

$$Y = \arg \max_{d_i \in \{yes, no\}} P(d_i) P(x_1 = suny | d_i) P(x_2 = cool | d_i) P(x_3 = high | d_i) P(x_4 = strong | d_i)$$

Back to the Example

$$Y = \arg \max_{d_i \in \{yes, no\}} P(d_i)P(x_1 = \text{sunny} | d_i)P(x_2 = \text{cool} | d_i)P(x_3 = \text{high} | d_i)P(x_4 = \text{strong} | d_i)$$

$$P(d=yes) = 9/14 = 0.64$$

$$P(d=no) = 5/14 = 0.36$$

$$P(x_1 = \text{sunny}/yes) = ?$$

$$P(x_1 = \text{sunny}/no) = ?$$

$$P(x_2 = \text{cool}/yes) = ?$$

$$P(x_2 = \text{cool}/no) = ?$$

$$P(x_3 = \text{high}/yes) = ?$$

$$P(x_3 = \text{high}/no) = ?$$

$$P(x_4 = \text{strong}/yes) = ?$$

$$P(x_4 = \text{strong}/no) = ?$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

$$Y = \arg \max_{d_i \in \{yes, no\}} P(d_i)P(x_1 = \text{sunny} | d_i)P(x_2 = \text{cool} | d_i)P(x_3 = \text{high} | d_i)P(x_4 = \text{strong} | d_i)$$

$$P(d=yes) = 9/14 = 0.64$$

$$P(d=no) = 5/14 = 0.36$$

$$P(x_1 = \text{sunny}/yes) = 2/9$$

$$P(x_1 = \text{sunny}/no) = 3/5$$

$$P(x_2 = \text{cool}/yes) = 3/9$$

$$P(x_2 = \text{cool}/no) = 1/5$$

$$P(x_3 = \text{high}/yes) = 3/9$$

$$P(x_3 = \text{high}/no) = 4/5$$

$$P(x_4 = \text{strong}/yes) = 3/9$$

$$P(x_4 = \text{strong}/no) = 3/5$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

$$Y = \arg \max_{d_i \in \{yes, no\}} P(d_i)P(x_1 = \text{sunny} | d_i)P(x_2 = \text{cool} | d_i)P(x_3 = \text{high} | d_i)P(x_4 = \text{strong} | d_i)$$

$$P(\text{yes})P(x_1 = \text{sunny} | \text{yes})P(x_2 = \text{cool} | \text{yes})P(x_3 = \text{high} | \text{yes})P(x_4 = \text{strong} | \text{yes}) = ?$$

$$P(\text{no})P(x_1 = \text{sunny} | \text{no})P(x_2 = \text{cool} | \text{no})P(x_3 = \text{high} | \text{no})P(x_4 = \text{strong} | \text{no}) = ?$$

$Y = \text{Play Tennis} = \text{yes or no?}$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

$$Y = \arg \max_{d_i \in \{yes, no\}} P(d_i)P(x_1 = suny | d_i)P(x_2 = cool | d_i)P(x_3 = high | d_i)P(x_4 = strong | d_i)$$

$$P(yes)P(x_1 = suny | yes)P(x_2 = cool | yes)P(x_3 = high | yes)P(x_4 = strong | yes) = 0.0053$$

$$P(no)P(x_1 = suny | no)P(x_2 = cool | no)P(x_3 = high | no)P(x_4 = strong | no) = 0.0206$$

$Y = Play Tennis = no$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Estimating Probabilities

- So far, we estimate the probabilities by the fraction of times the event is observed to occur over the entire opportunities
- In the above example, we estimated

$$P(\text{wind=strong} | \text{play tennis=no}) = N_c/N,$$

where $N = 5$ is the total number of training samples for which $\text{play tennis} = \text{no}$, and N_c is the number of these for which wind=strong

- What happens if $N_c = 0$?

Estimating Probabilities

- When N_c is small, however, such approach provides poor estimation. To avoid this difficulty, we can adopt the **m-estimate** of probability

$$\frac{N_c + mP}{N + m}$$

where P is the prior estimate of the probability we wish to estimate, m is a constant called the equivalent sample size, which determines how heavily to weight P relative to the observed data.

A typical method for choosing P in the absence of other information is to assume uniform priors: If an attribute has k possible values we set $P=1/K$.

Back to the Example

Estimating Probabilities

- For example: $P(\text{wind} = \text{strong} / \text{play tennis} = \text{no})$, we note *wind* has two possible values: *weak, strong*; so the uniform priors of *wind* $P = 1/2$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

Estimating Probabilities

- **Question:** $P(\text{outlook} = \text{sunny} / \text{play tennis} = \text{yes})$, what are the uniform priors of *outlook*?

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

Estimating Probabilities

- Solution:** $P(\text{outlook} = \text{sunny} / \text{play tennis} = \text{yes})$, we note *outlook* has three possible values: *sunny*, *overcast*, *rain*; so the uniform priors of *wind* $P = 1/3$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Naïve Bayesian Classifier

- What about using Naïve Bayesian Classifier for our handwritten digit recognition problem?
 - Each pixel is an x_i . There will be 784 x 's.
 - Digit label is d_k . Note there will be 10 possible d 's.
 - $P(d_k)$ can be calculated by counting number of training images for the digit, divided to total number of training images.
 - $P(x_i/d_k)$ can be calculated by counting number of images for a given digit, given pixel position, and given an intensity value, divided by number of training images with that digit.

Naïve Bayesian Classifier

$P(x_i|d_k)$ can be calculated by counting number of images for a given digit, given pixel position, and given an intensity value, divided by number of training images with that digit.

For example,

$$P(x_1 = 255|d = 0)$$

$$= \frac{\text{Number of images whose first pixel value is 255 and contain digit 0}}{\text{Total number of images contain digit 0}}$$

Naïve Bayesian Classifier

- For a given input image X and given digit label d_k , calculate $P(d_k)$ and all $P(x_i/d_k)$
- For each digit label d_k , calculate
$$P(d_k|X) = P(d_k)P(x_1 = 0|d_k)P(x_2 = 255|d_k) \dots P(x_{784} = 0|d_k)$$
- Choose the digit label k that give the max value according to above calculation.

Input image X

0	255	0	0	0	0	0	0	0	0	0
0	0	0	0	255	255	0	0	0	0	0
0	0	0	255	0	0	255	0	0	0	0
0	0	0	0	0	0	255	0	0	0	0
0	0	0	0	0	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0
0	0	0	255	255	255	255	255	255	255	0

Further Readings

Chapter 6, T. M. Mitchell, Machine Learning,
McGraw-Hill International Edition, 1997

Any Questions?





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 8 – Data Clustering

Ying Weng
2024 Autumn

Supervised VS Unsupervised Learning

- Supervised learning
 - Learns a function that maps an input to an output based on example input-output pairs.
 - Training data is labeled.
- Unsupervised learning
 - Learns from test data that has not been labeled.
 - Learns relationships between elements in a data set and classify the raw data without "help."
 - Typical application includes data clustering.

Motivating Problems

- ★ A true colour image – 24bits/pixel, R – 8 bits, G – 8 bits, B – 8 bits



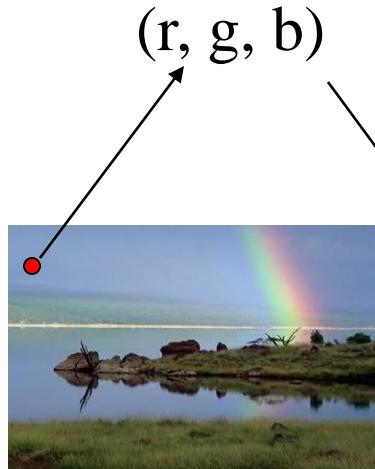
16777216 possible colours

- ★ A gif image - 8bits/pixel

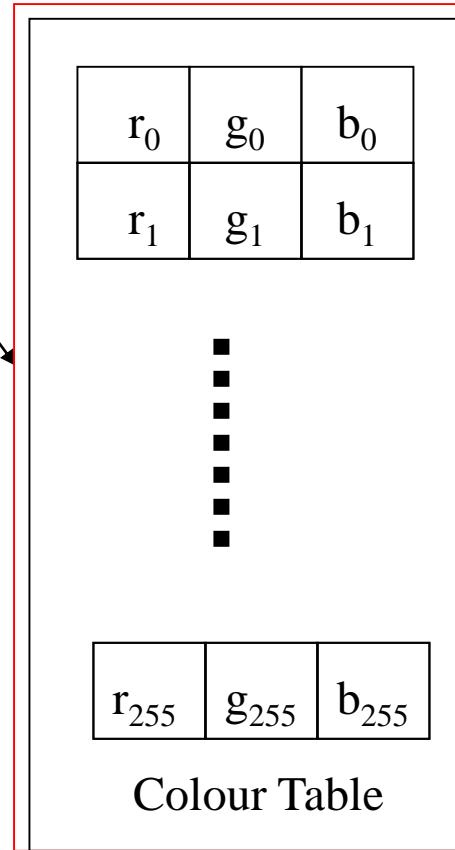


256 possible colours

Motivating Problems



(r, g, b)



For each pixel in the original image

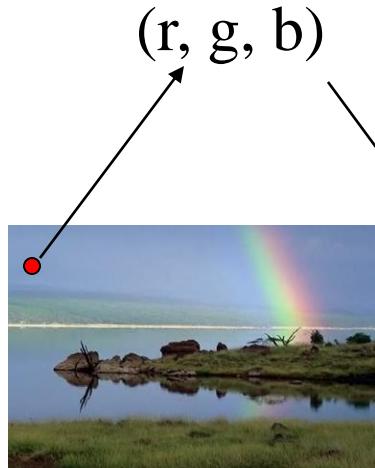
Find the closest colour in the Colour Table



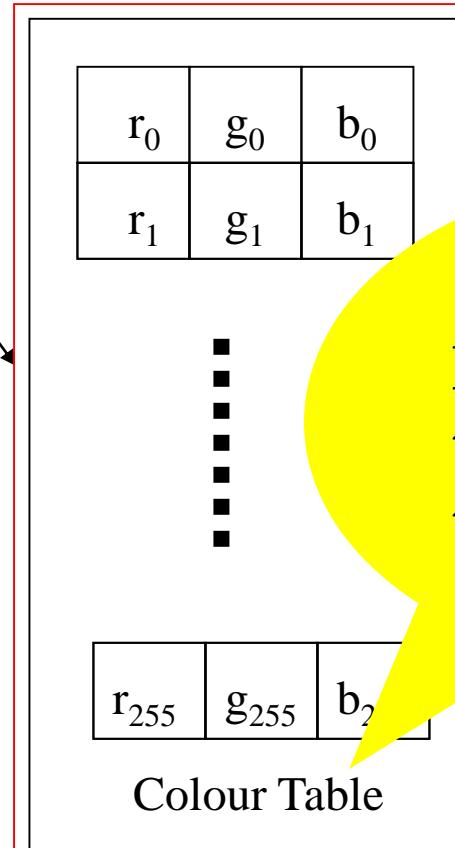
Record the index of that colour (for storage or transmission)

To reconstruct the image, place the indexed colour from the Colour Table at the corresponding spatial location

Motivating Problems



(r, g, b)



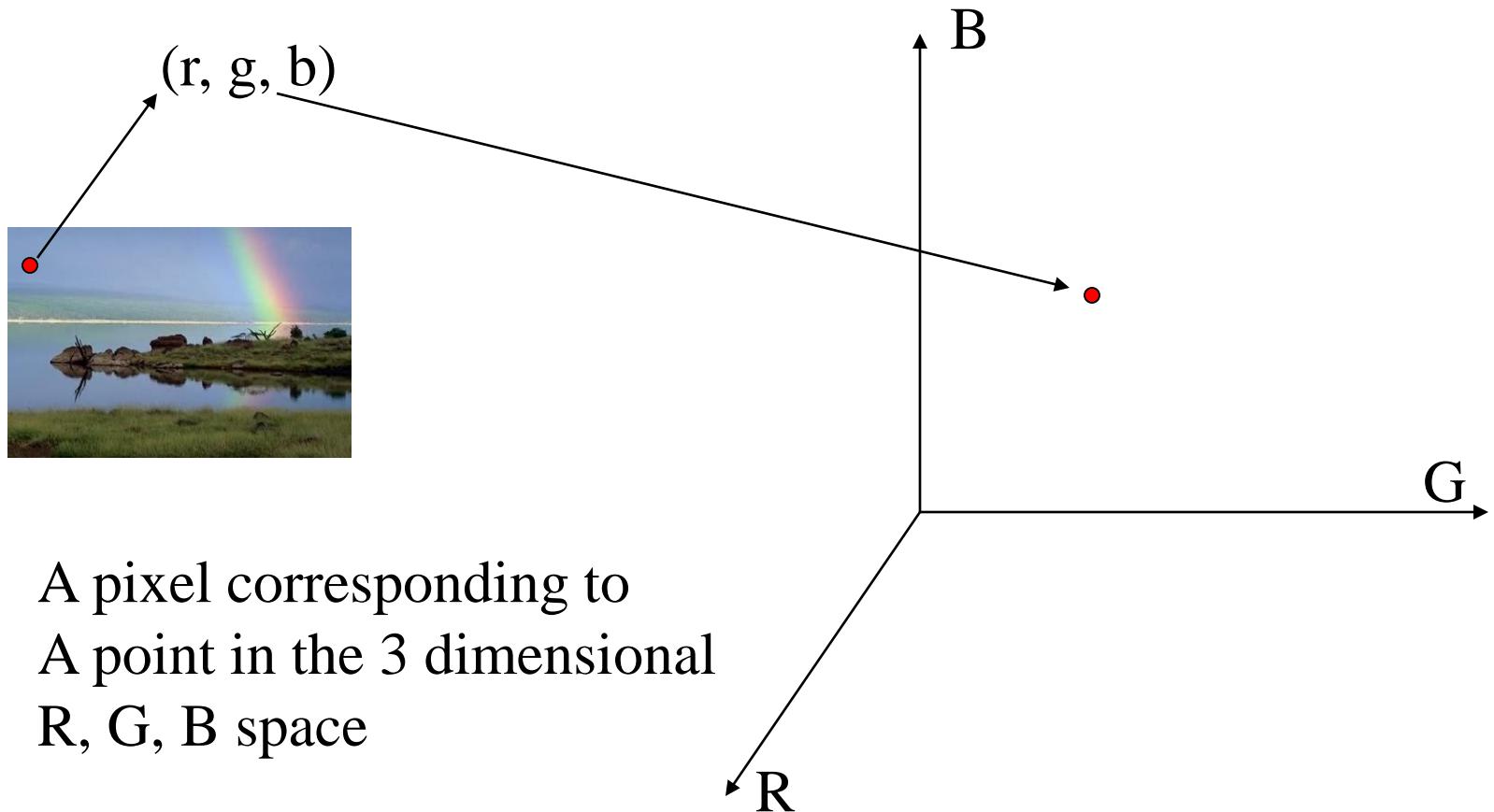
How to choose
the colours in
the table?

For each pixel in the original image

Find the closest colour in the Colour Table

Replace the indexed colour from the Colour Table at the corresponding spatial location

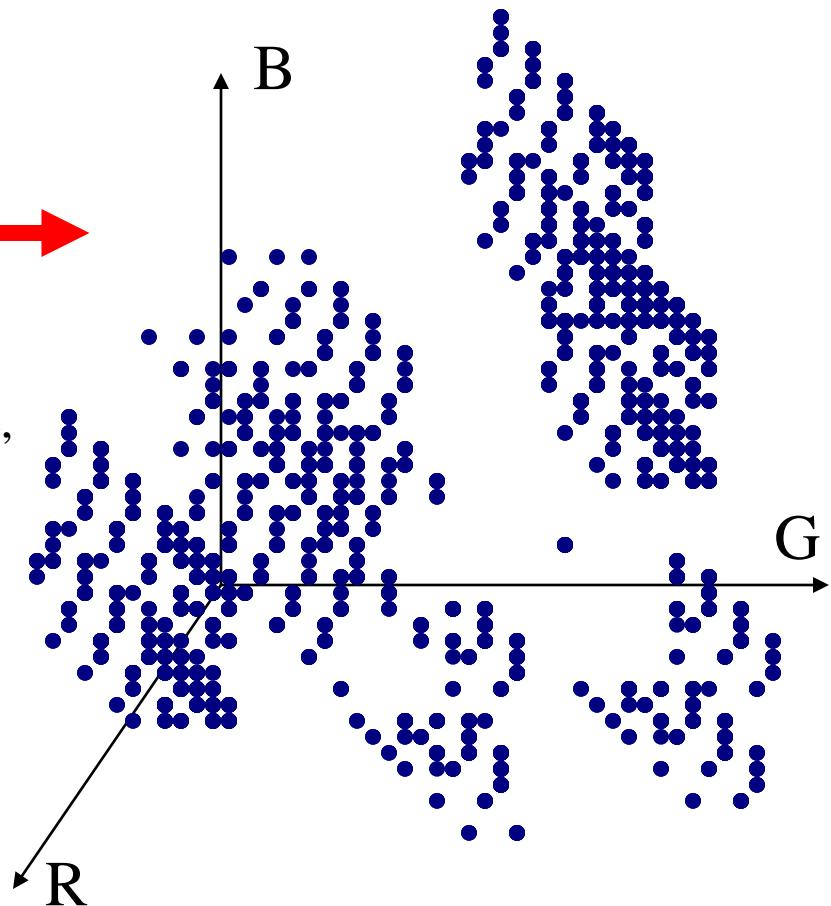
Motivating Problems



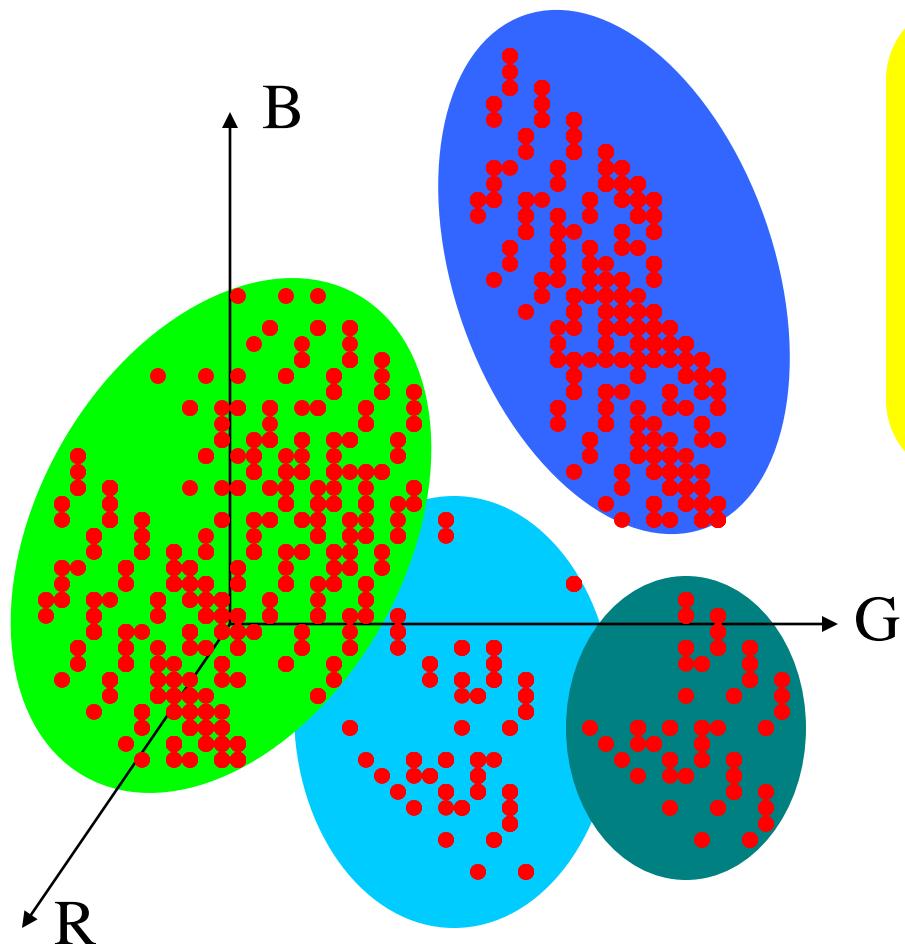
Motivating Problems



Map all pixels into the R, G, B space,
“clouds” of pixels are formed

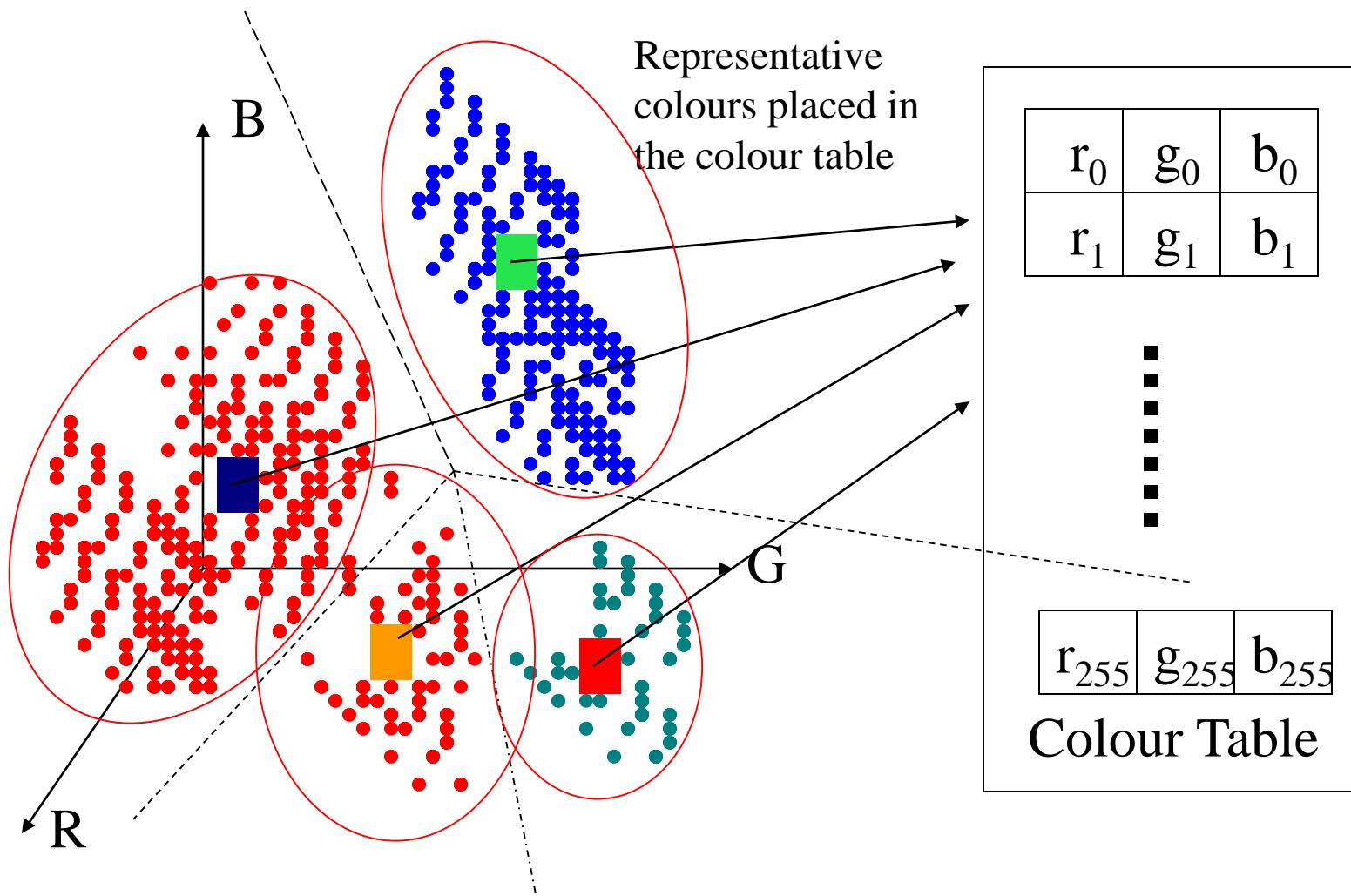


Motivating Problems



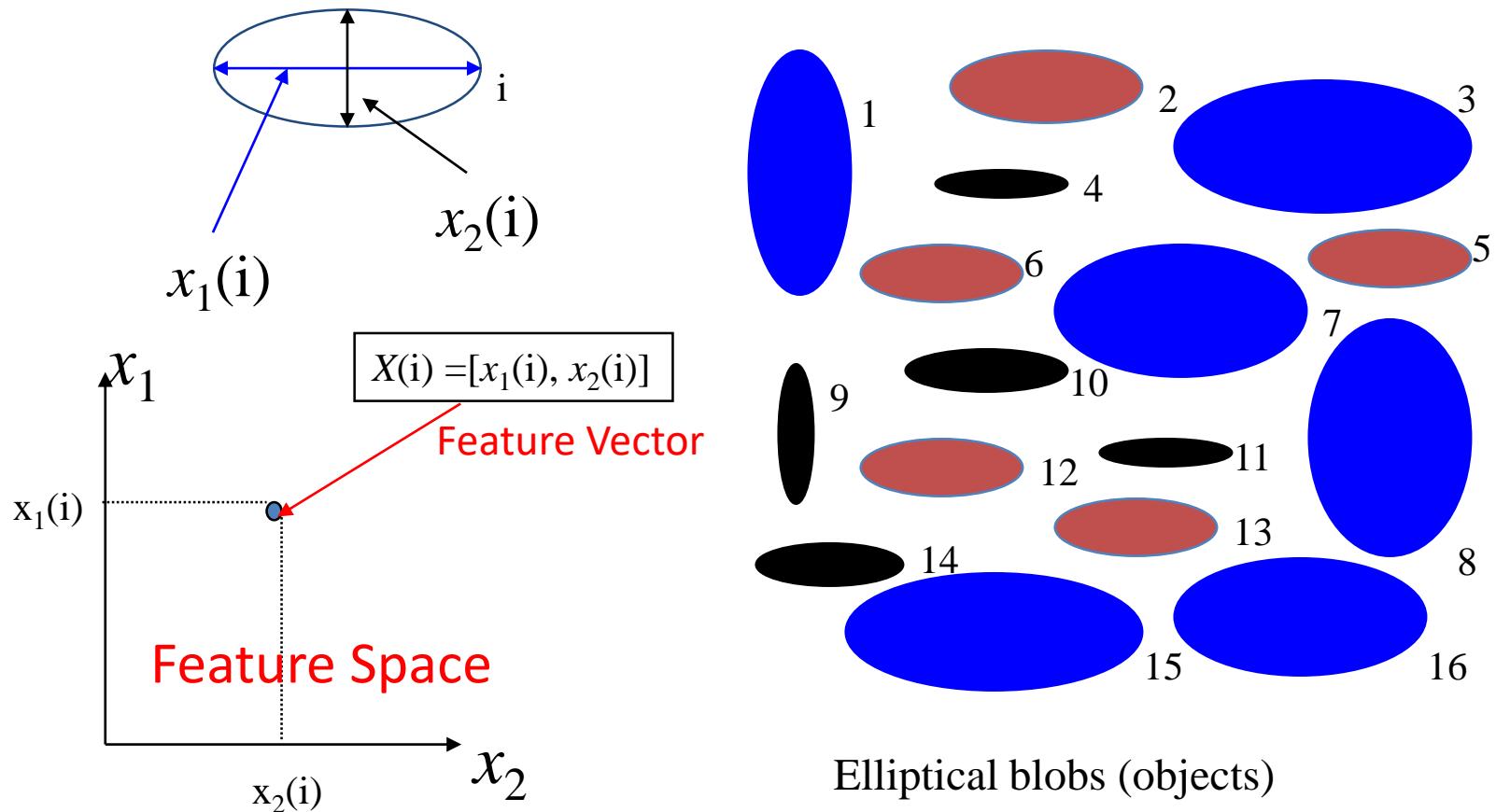
Group pixels that are close to each other, and replace them by one single colour

Motivating Problems



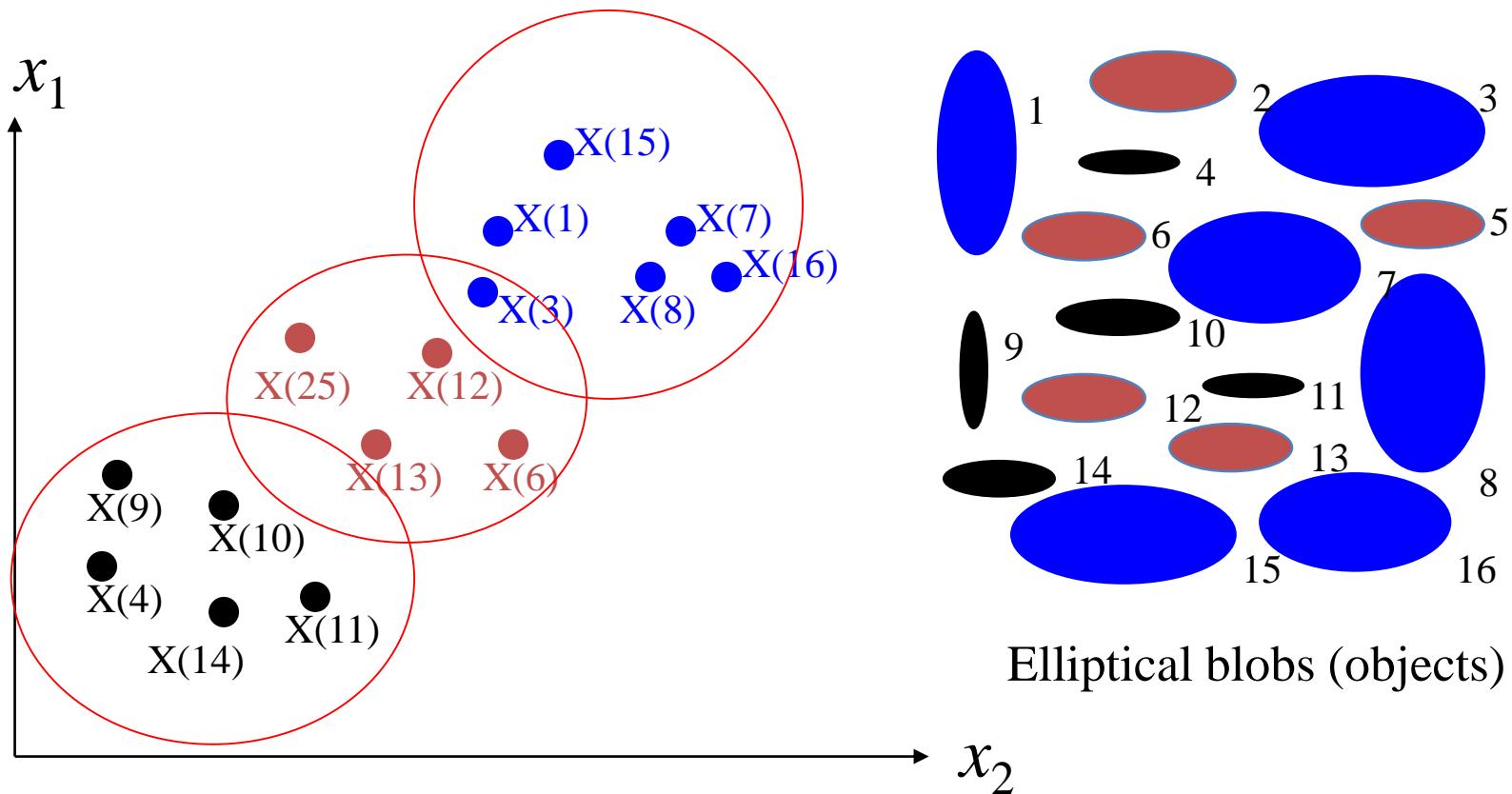
Motivating Example

- Classify objects (Oranges, Potatoes) into large, middle, small sizes



Motivating Example

- ★ From Objects to Feature Vectors to Points in the Feature Space



Motivation of Clustering

- Patterns within a valid cluster are *more similar to each other* than they are to a pattern belonging to a different cluster.
- In clustering, the problem is to group a given collection of *unlabeled patterns* into meaningful clusters. Clustering is data *driven method*, the clusters are obtained solely from the data.
- Clustering could be used in the field of pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation

K-Means

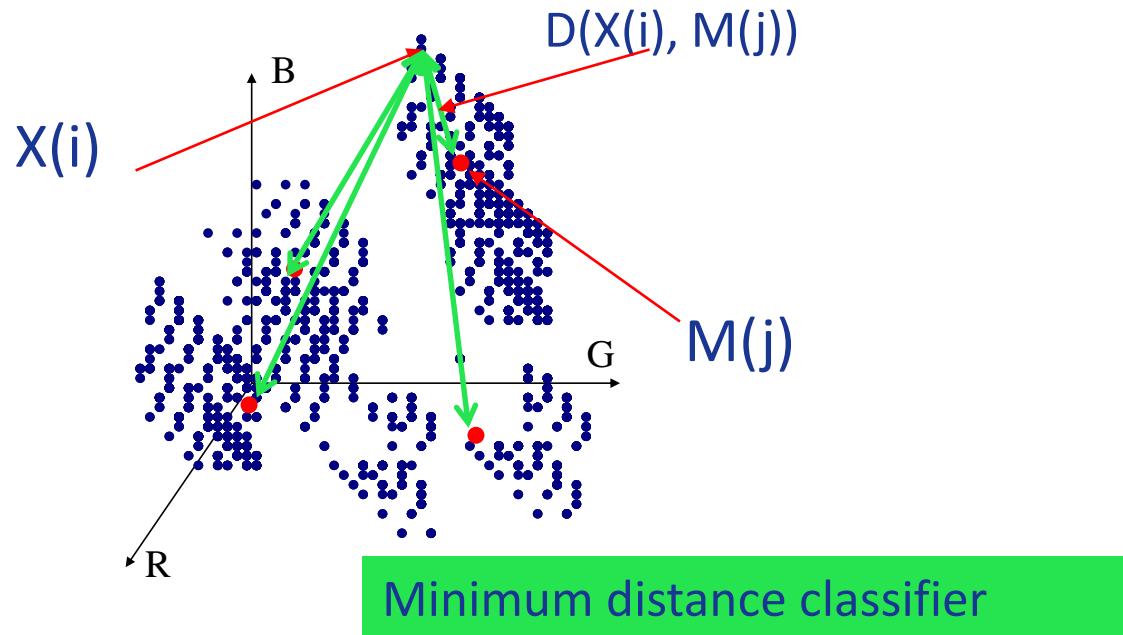
- ❖ An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing N_j data points
 - ❖ Define, $X(i) = [x_1(i), x_2(i), \dots, x_n(i)]$, $i = 1, 2, \dots, N$, as N data points
 - ❖ We want to cluster these N points into K subsets, or K clusters, where K is pre-set
 - ❖ For each cluster, we define $M(j) = [m_1(j), m_2(j), \dots, m_n(j)]$, $j=1, 2, \dots, K$, as its prototype or cluster centroids
 - ❖ Define the distance between data point $X(i)$ and cluster prototype $M(j)$ as

$$D(X(i), M(j)) = \|X(i) - M(j)\| = \sqrt{\sum_{l=1}^n (x_l(i) - m_l(j))^2}$$

K-Means

- ★ A data point $X(i)$ is assigned to the j th cluster, $C(j)$, $X(i) \in C(j)$, if following condition holds

$$D(X(i), M(j)) \leq D(X(i), M(l)) \quad \text{for all } l = 1, 2, \dots, k$$



K-Means Algorithm

Step 1

- ★ Arbitrarily choose from the given sample set k initial cluster centres,

$$M^{(0)}(j) = [m^{(0)}_1(j), m^{(0)}_2(j), \dots, m^{(0)}_n(j)] \quad j = 1, 2, \dots, K,$$

e.g., the first K samples of the sample set
or can also be generated randomly

Set $t = 0$ (t is the iteration index)

K-Means Algorithm

Step 2

- ★ Assign each of the samples $X(i) = [x_1(i), x_2(i), \dots, x_n(i)]$, $i = 1, 2, \dots, N$, to one of the clusters according to the distance between the sample and the centre of the cluster:

$$X(i) \in C^{(t)}(j)$$

if $D(X(i), M^{(t)}(j)) \leq D(X(i), M^{(t)}(l))$

for all $l = 1, 2, \dots, k$

K-Means Algorithm

Step 3

Update the cluster centres to get

$$M^{(t+1)}(j) = [m^{(t+1)}_1(j), m^{(t+1)}_2(j), \dots, m^{(t+1)}_n(j)] ; j = 1, 2, \dots, K$$

according to

$$M^{(t+1)}(j) = \frac{1}{N_j^{(t)}} \sum_{X(i) \in C^{(t)}(j)} X(i)$$

$N_j^{(t)}$ is the number of samples in $C_j^{(t)}$

K-Means Algorithm

Step 4

- Calculate the error of approximation

$$E(t) = \sum_{j=1}^K \sum_{X(i) \in C^{(t)}(j)} \|X(i) - M^{(t)}(j)\|$$

K-Means Algorithm

Step 5

- If the terminating criterion is met, then stop, otherwise

Set $t = t+1$

Go to Step 2.

K-Means Algorithm

Stopping criterions

- The K-means algorithm can be stopped based on following criterions
 1. The errors do not change significantly in two consecutive epochs
$$|E(t)-E(t-1)| < \varepsilon, \text{ where } \varepsilon \text{ is some preset small value}$$
 2. No further change in the assignment of the data points to clusters in two consecutive epochs.
 3. It can also stop after a fixed number of epochs regardless of the error

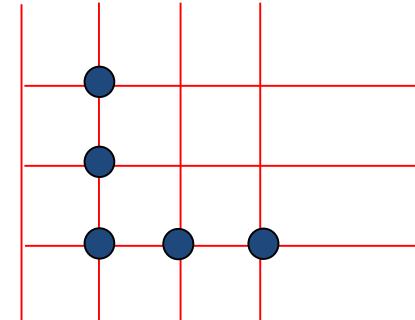
K-Means Algorithm

A worked example to see how it works exactly

Five 2-dimensional data points:

(1, 1), (2, 1), (3, 1), (1, 2), (1, 3)

Cluster them into two clusters and find the cluster centres



K-Means Algorithm

A worked example to see how it works exactly

(1) Euclidean distance to $m_1^0(1,2)$

$$\sqrt{(1-1)^2 + (1-2)^2} = 1$$

$$\sqrt{(2-1)^2 + (1-2)^2} = \sqrt{2} = 1.41$$

$$\sqrt{(3-1)^2 + (1-2)^2} = \sqrt{5} = 2.24$$

$$\sqrt{(1-1)^2 + (2-2)^2} = 0$$

$$\sqrt{(1-1)^2 + (3-2)^2} = 1$$

Euclidean distance to $m_2^0(3,1)$

$$\sqrt{(1-3)^2 + (1-1)^2} = 2$$

$$\sqrt{(2-3)^2 + (1-1)^2} = 1$$

$$\sqrt{(3-3)^2 + (1-1)^2} = 0$$

$$\sqrt{(1-3)^2 + (2-1)^2} = \sqrt{5} = 2.24$$

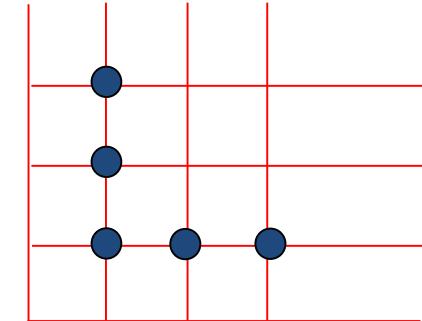
$$\sqrt{(1-3)^2 + (3-1)^2} = \sqrt{8} = 2.83$$

$\therefore C_1$

$\therefore C_2$

$\therefore C_2$

$\therefore C_1$



C_1 class: $(1,1), (1,2), (1,3) \Rightarrow m_1^{(0+1)}$:

$$\frac{1}{3} \sum_{i=1}^3 x_i = \frac{1+1+1}{3} = 1$$

$$\frac{1}{3} \sum_{i=1}^3 y_i = \frac{1+2+3}{3} = 2$$

C_2 class: $(2,1), (3,1) \Rightarrow m_2^{(0+1)}$:

$$\frac{1}{2} \sum_{i=1}^2 x_i = \frac{2+3}{2} = 2.5$$

$$\frac{1}{2} \sum_{i=1}^2 y_i = \frac{1+1}{2} = 1$$

(2) Euclidean distance to $m_1^1(1,2)$

1

$$\sqrt{2} = 1.41$$

$$\sqrt{5} = 2.24$$

0

1

$\therefore C_1$ class: $(1,1), (1,2), (1,3)$

C_2 class: $(2,1), (3,1)$

Euclidean distance to $m_2^1(2.5, 1)$

$$\sqrt{(1-2.5)^2 + (1-1)^2} = 1.5$$

$\therefore C_1$

$$\sqrt{(2-2.5)^2 + (1-1)^2} = 0.5$$

$\therefore C_2$

$$\sqrt{(3-2.5)^2 + (1-1)^2} = 0.5$$

$\therefore C_2$

$$\sqrt{(1-2.5)^2 + (2-1)^2} = \sqrt{3.25} = 1.80$$

$\therefore C_1$

$$\sqrt{(1-2.5)^2 + (3-1)^2} = \sqrt{6.25} = 2.5$$

$\therefore C_1$

K-Means Algorithm

- What is the algorithm doing exactly?
 - It tries to find the centre vectors $M(j)$'s that optimize the following cost function

$$E = \sum_{j=1}^K \sum_{X(i) \in C(j)} \|X(i) - M(j)\|$$

K-Means Algorithm

- Some remarks
 - Is a gradient descent algorithm, trying to minimize a cost function E
 - In general, the algorithm does not achieve a global minimum of E over the assignments
 - Sensitive to initial choice of cluster centers. Different starting cluster centroids may lead to different solution
 - Is a popular method, many more advanced methods derived from this simple algorithm

Any Questions?





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

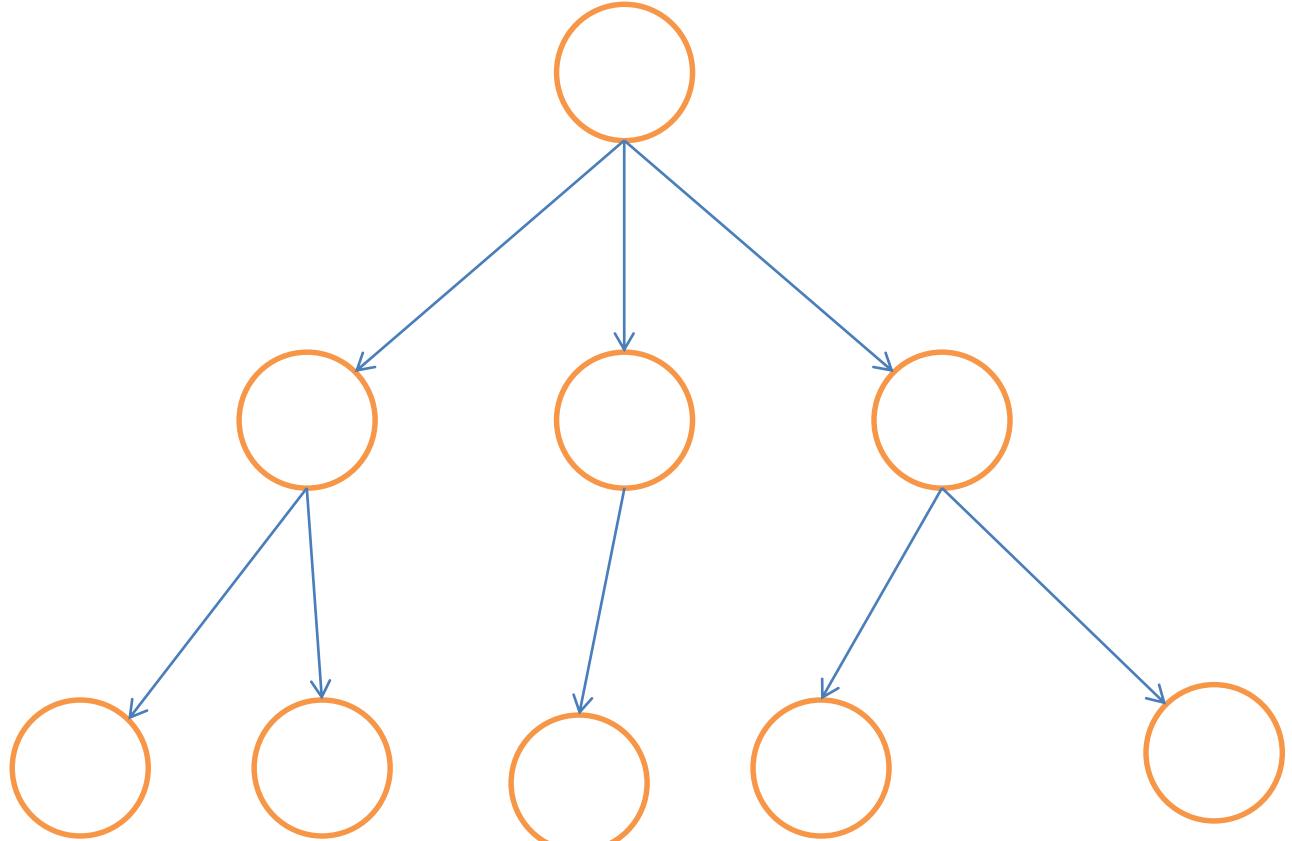
Machine Learning

Topic 9 – Decision Tree

Ying Weng
2024 Autumn

Trees

- Node
- Root
- Leaf
- Branch
- Path
- Depth



Decision Trees

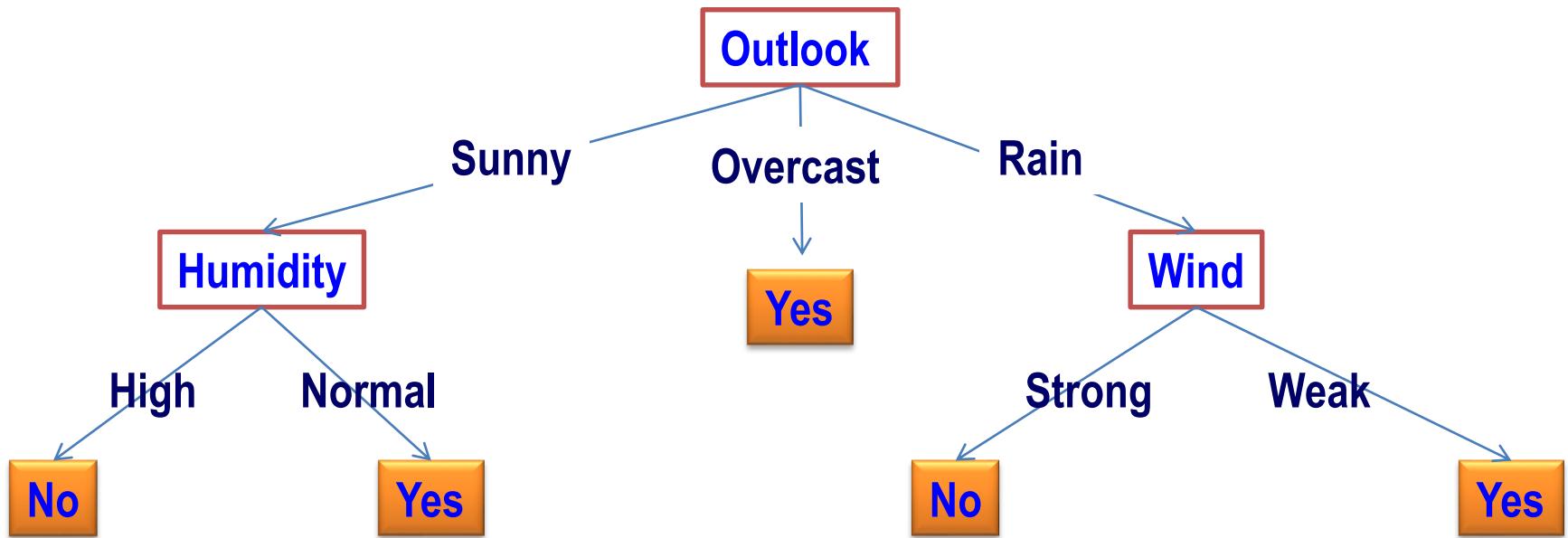
- A hierarchical data structure that represents data by implementing a divide and conquer strategy
- Can be used as a non-parametric classification method
- Given a collection of examples, learn a decision tree that represents it
- Use this representation to classify new examples

Decision Trees

- Each node is associated with a feature (one of the elements of a feature vector that represent an object)
- Each node test the value of its associated feature
- There is one branch for each value of the feature
- Leaves specify the categories (classes)
- Can categorize instances into multiple disjoint categories – multi-class

Decision Trees

- ❑ Play Tennis Example
- ❑ Feature Vector = (Outlook, Temperature, Humidity, Wind)

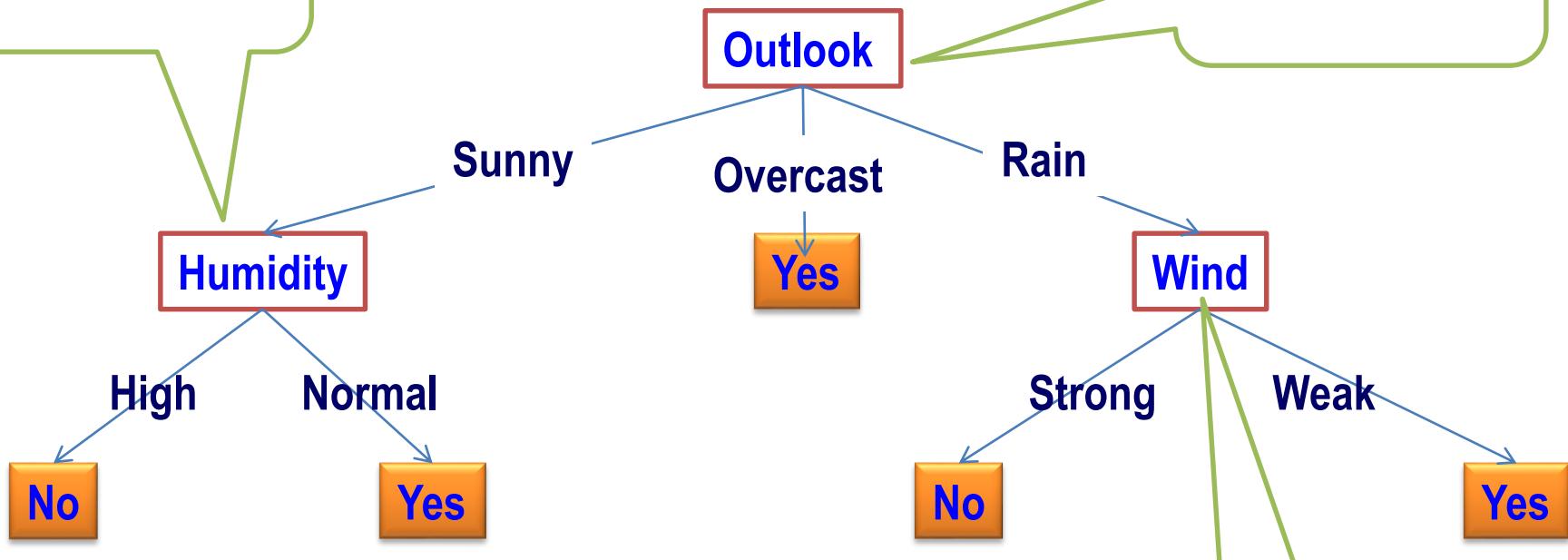


Decision Trees

Node associated
with a feature

Node associated
with a feature

Node associated
with a feature

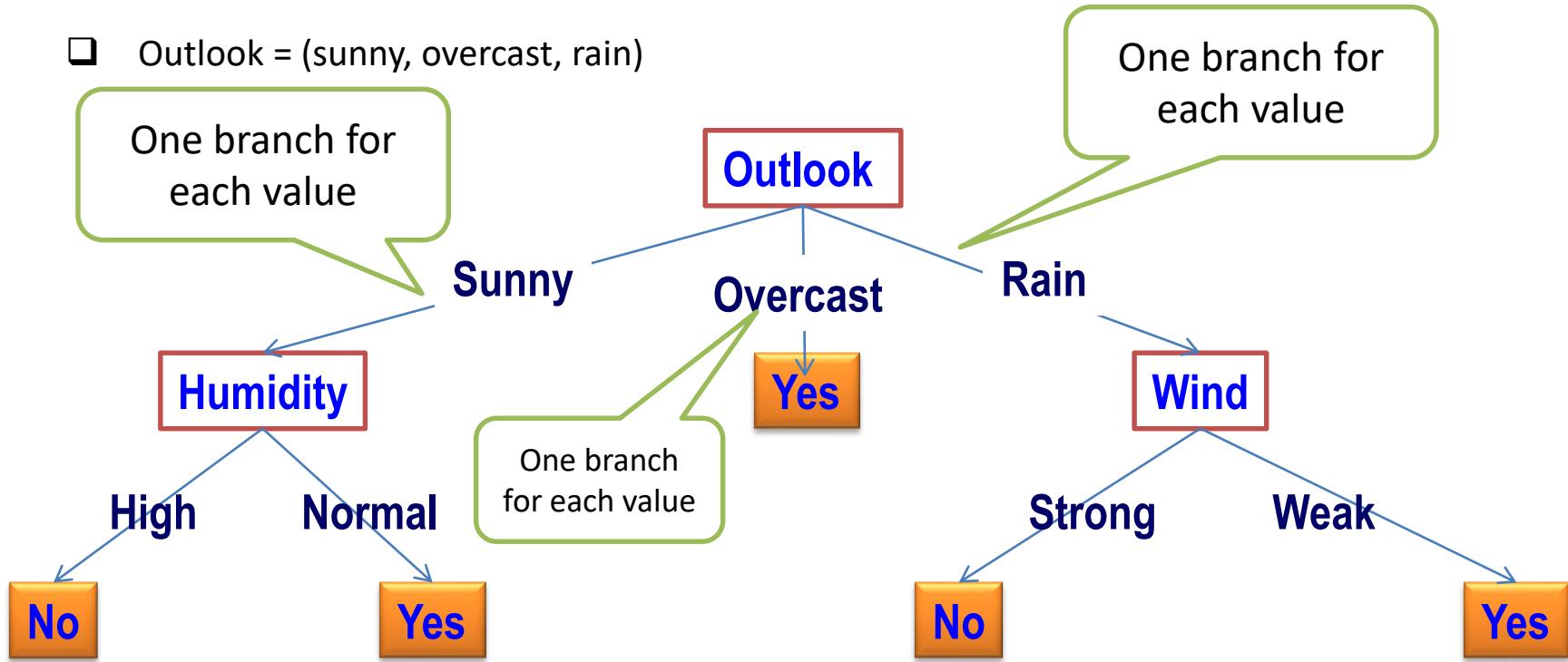


Decision Trees

- ❑ Play Tennis Example
- ❑ Feature values:
 - ❑ Outlook = (sunny, overcast, rain)
 - ❑ Temperature =(hot, mild, cool)
 - ❑ Humidity = (high, normal)
 - ❑ Wind =(strong, weak)

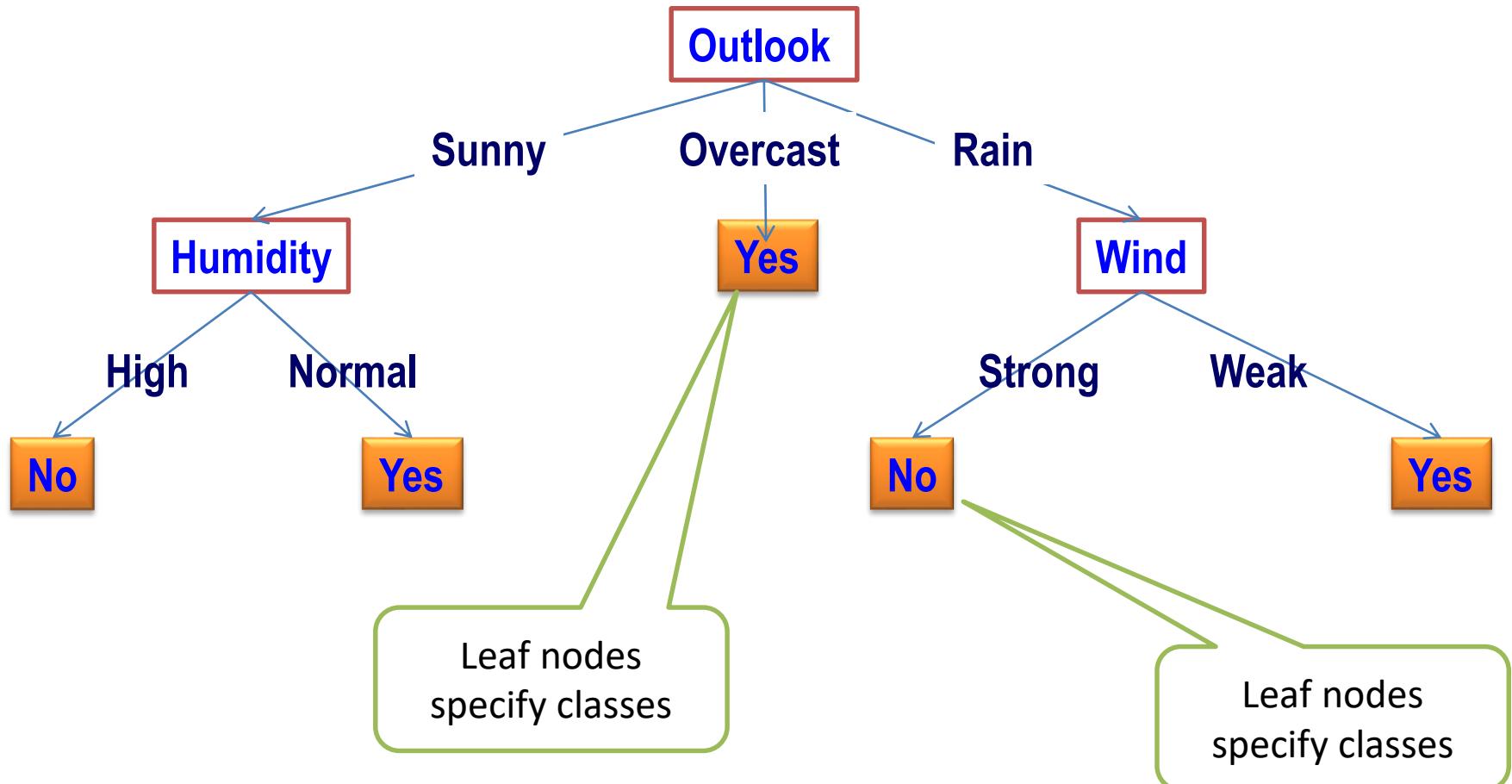
Decision Trees

- Outlook = (sunny, overcast, rain)



Decision Trees

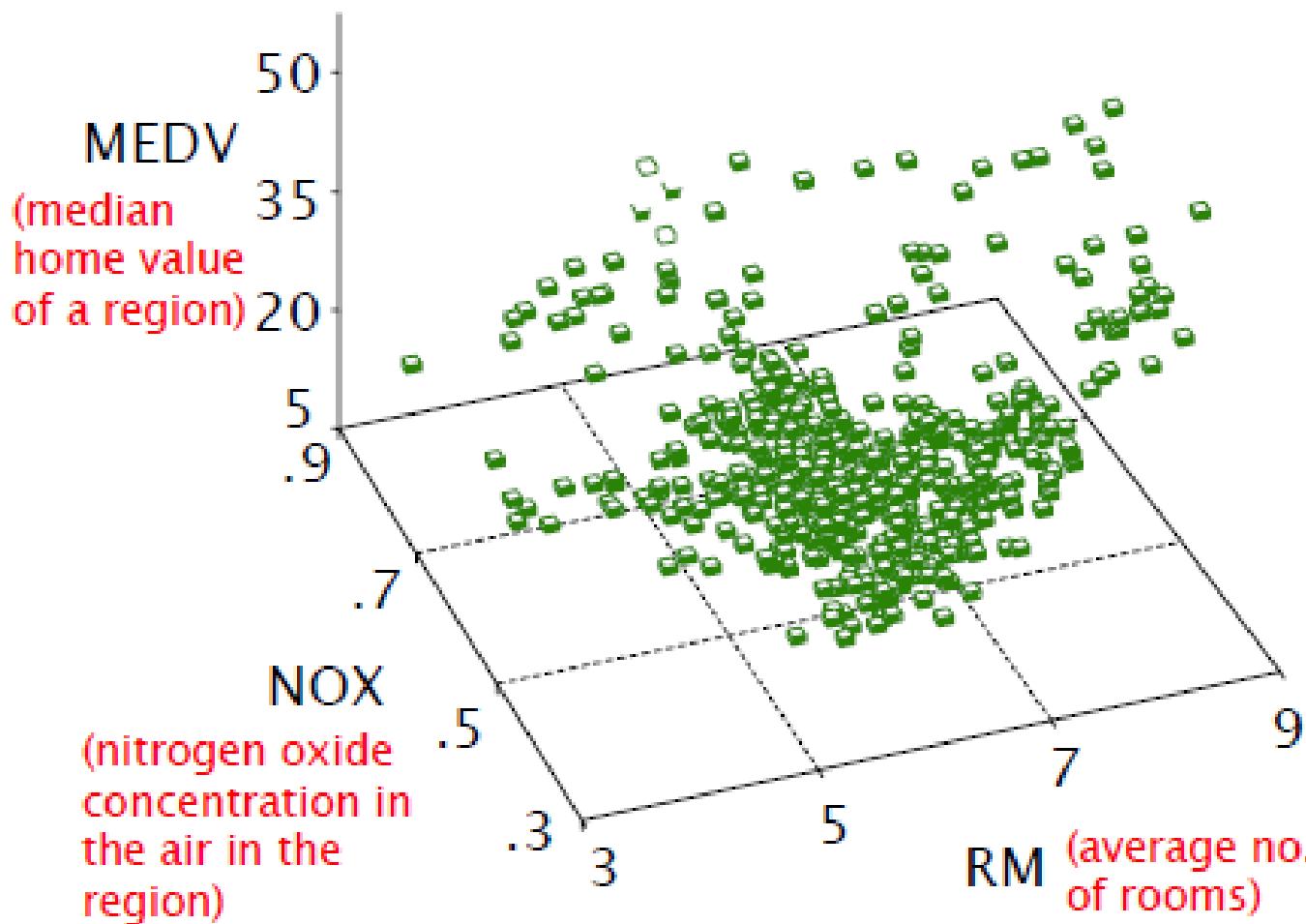
- Class = (Yes, No)



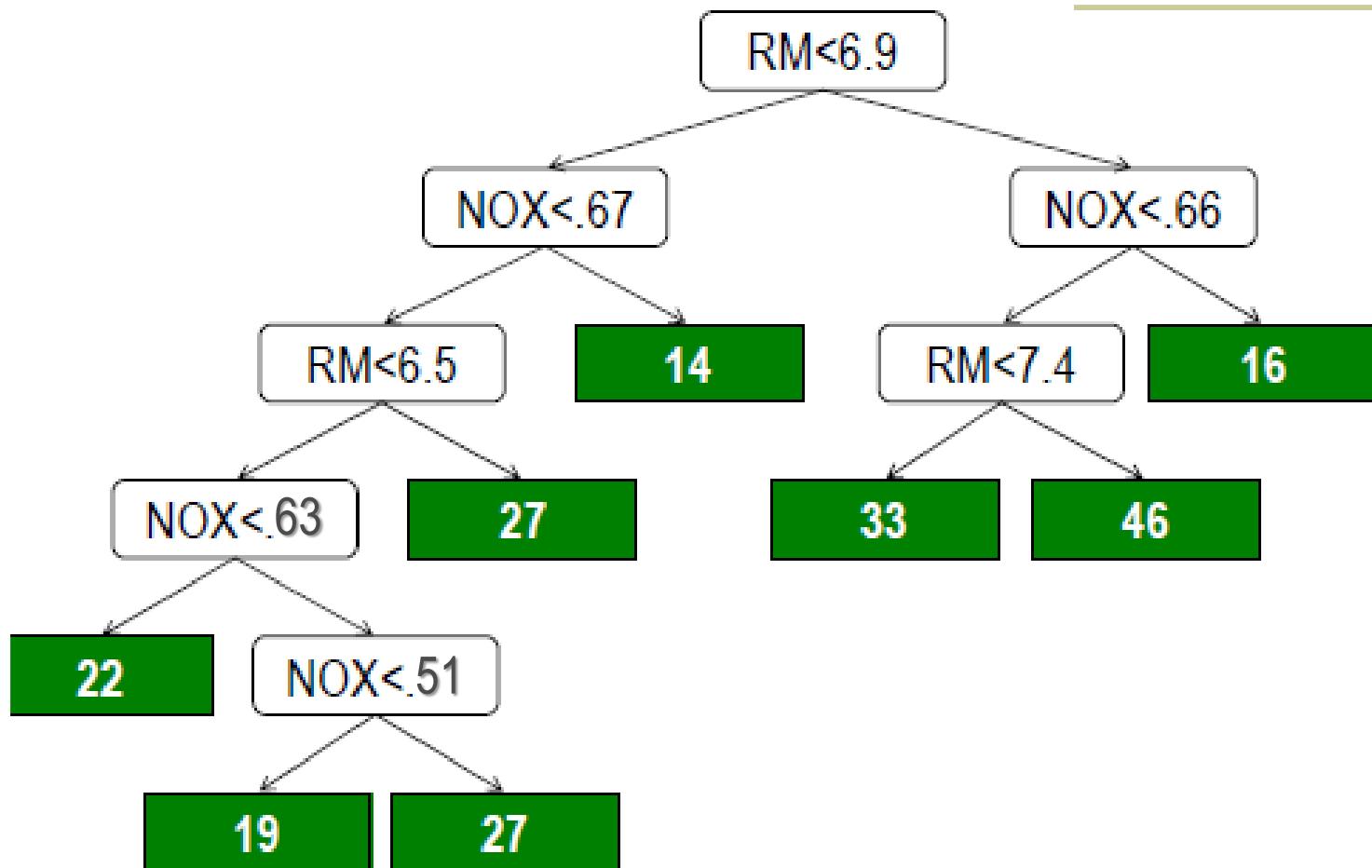
Variation of Decision Trees

- Classification tree
 - The output is discrete (label).
 - The leaves give the predicted class as well as the probability of class membership.
- Regression tree
 - The output is continuous.
 - The leaves give the predicted value of the output.
- Tree with binary splits
- Tree with multiway splits

Boston Housing Data



Regression Tree



Decision Trees

- ❑ Design Decision Tree Classifier
 - ❑ Picking the root node
 - ❑ Recursively branching

Decision Trees

❑ Picking the root node

- ❑ Consider data with two Boolean attributes (A,B) and two classes + and –

{ (A=0,B=0), - }: 50 examples

{ (A=0,B=1), - }: 50 examples

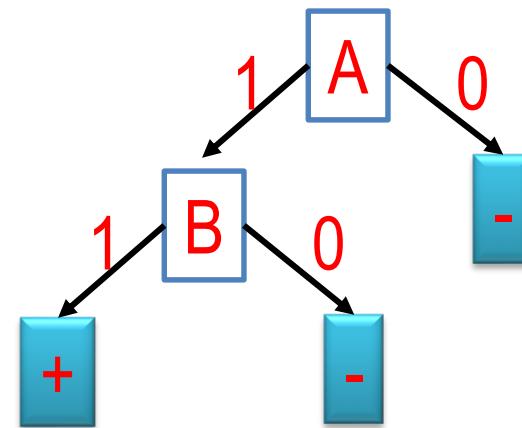
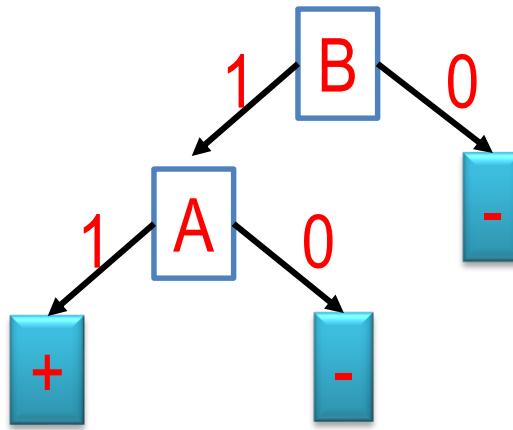
{ (A=1,B=0), - }: 3 examples

{ (A=1,B=1), + }: 100 examples

Decision Trees

❑ Picking the root node

- ❑ Trees looks structurally similar; which attribute should we choose?



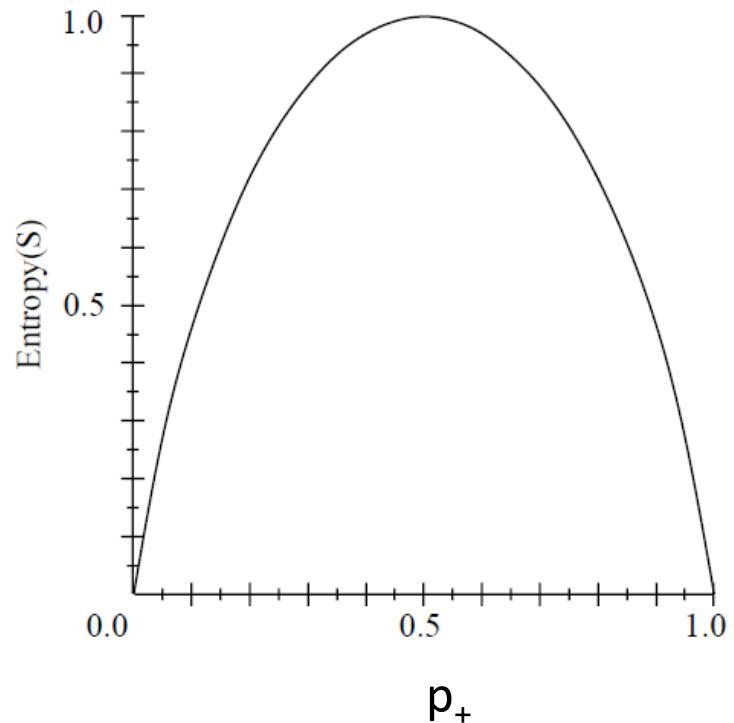
Decision Trees

❑ Picking the root node

- ❑ The goal is to have the resulting decision tree as **small** as possible (Occam's Razor)
- ❑ The main decision in the algorithm is the selection of **the next attribute** to condition on (start from the root node).
- ❑ We want attributes that split the examples to sets that are relatively **pure** in one label; this way we are closer to a leaf node.
- ❑ The most popular heuristics is based on **information gain**, originated with the ID3 system of Quinlan.

Entropy

- S is a sample of training examples
- p_+ is the **proportion** of positive examples in S
- p_- is the **proportion** of negative examples in S
- Entropy measures the impurity of S



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

Entropy(IIP)

- The idea: associate information with probability
- A random event E with probability $P(E)$ contains:

$$I(E) = \log\left(\frac{1}{P(E)}\right) = -\log(P(E))$$

units of information

- Suppose that grey level values are generated by a random variable, then r_k contains:

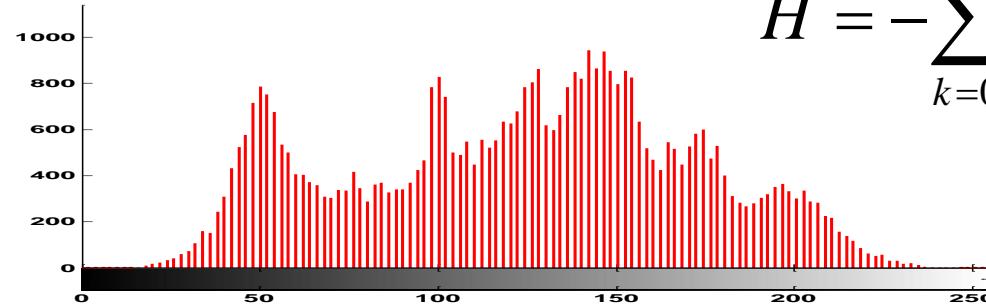
$$I(r_k) = -\log(P(r_k))$$

Note: $I(E)=0$ when $P(E)=1$

units of information

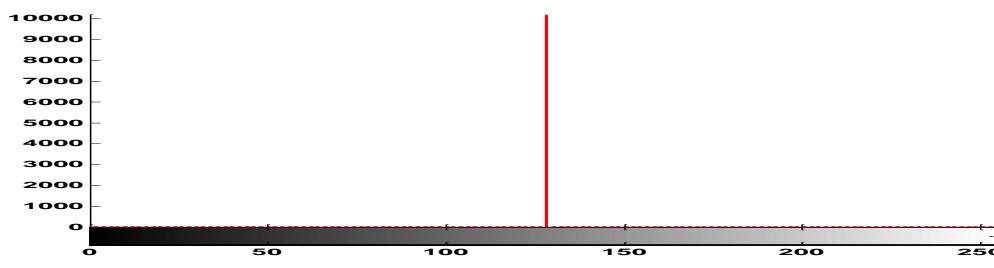
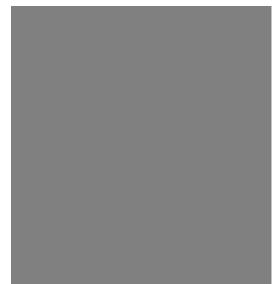
Entropy(IIP)

- Entropy is the average information content of an image, a measure of histogram dispersion



$$H = -\sum_{k=0}^{L-1} p(r_k) \log_2 p(r_k)$$

entropy=7.4635



entropy=0

- Can't compress to less than H bits/pixel without losing information

Highly Disorganized

High Entropy

Much Information Required

+ - + + + + - + - + + +
- - + + + - + - + - + -
+ - + - + + - + + + - +
+ - - + + - + + + - + +
+ - - + - + + - + - + +

- - + + + - + - +
+ - + - + + + -
+ - + - - + - +

- - + - + - +
- - + - - -
+ - - + - -

+ + + +
+ + + +

+ + + + +
+ + + + +
+ + + +

- - + - + - +
- + + +

+ + +
+ + +

Highly Organized
Low Entropy

Little Information Required

Information Gain

- $\text{Gain}(S, A)$ = expected reduction in entropy due to sorting on A

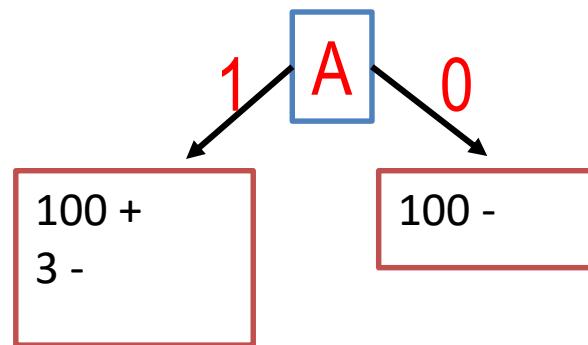
$$\text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- $\text{Values}(A)$ is the set of all possible values for attribute A, S_v is the subset of S which attribute A has value v, $|S|$ and $|S_v|$ represent the number of samples in set S and set S_v respectively
- $\text{Gain}(S, A)$ is the expected reduction in entropy caused by knowing the value of attribute A.

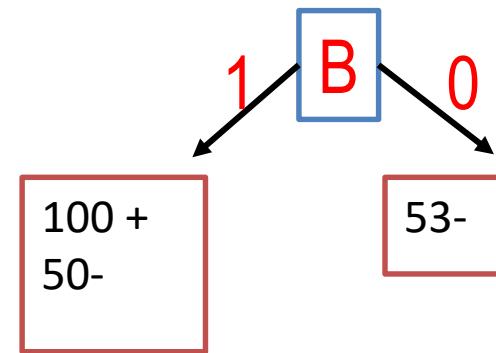
Information Gain

Example: Choose A or B ?

Split on A



Split on B



Example

Play Tennis Example

Entropy(S) =

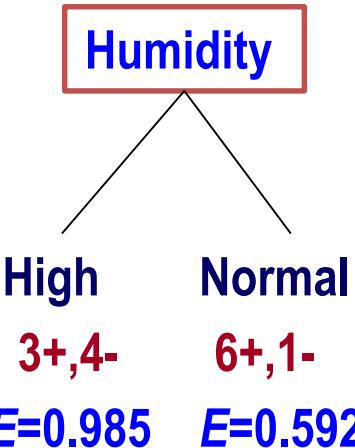
$$-\frac{9}{14} \log\left(\frac{9}{14}\right)$$
$$-\frac{5}{14} \log\left(\frac{5}{14}\right)$$

$$= 0.94$$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

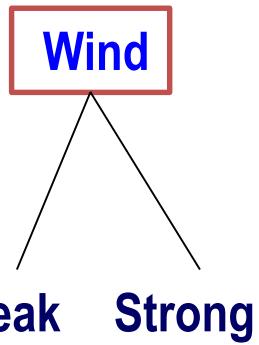


$$Gain(S, \text{Humidity}) = .94 - 7/14 * 0.985 - 7/14 * 0.592 = 0.151$$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



Weak Strong

6+,2- 3+,3-

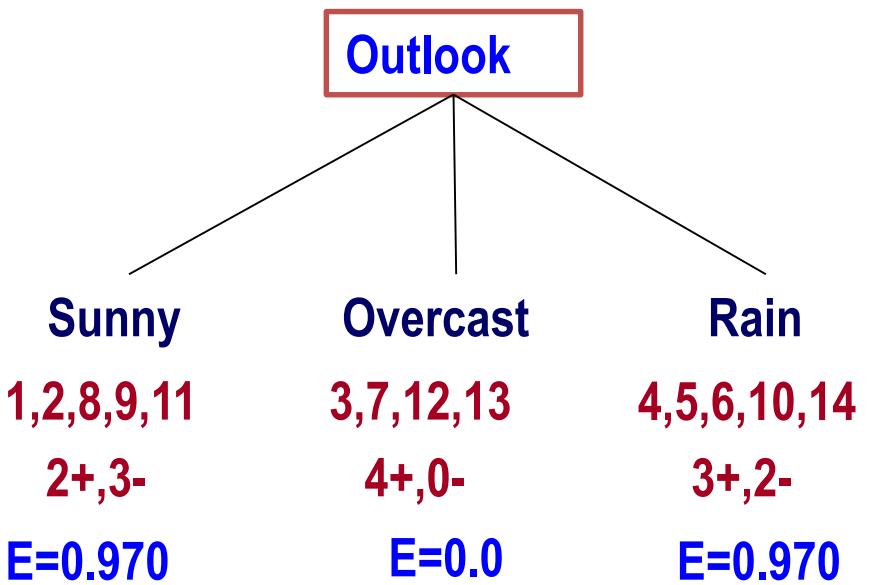
E=0.811 E=1.0

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

$$Gain(S, \text{Wind}) = .94 - 8/14 * 0.811 - 6/14 * 1.0 = 0.048$$

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

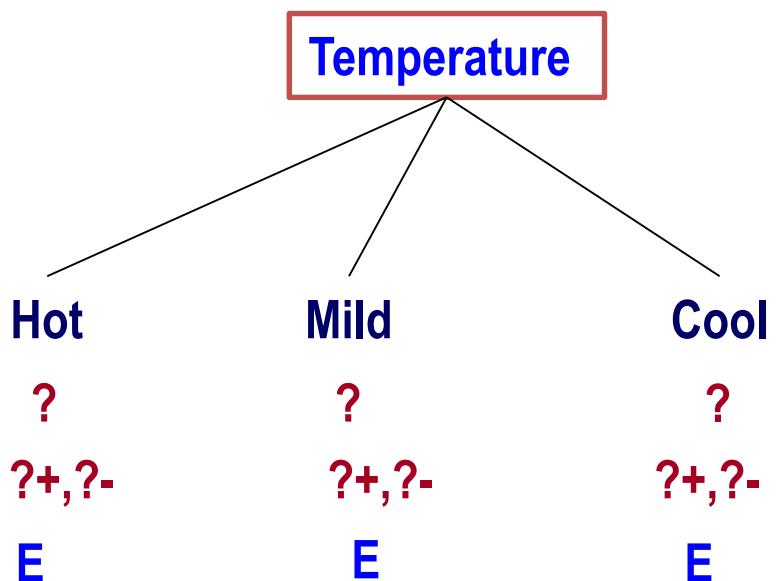


$$Gain(S, \text{Outlook}) = 0.246$$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

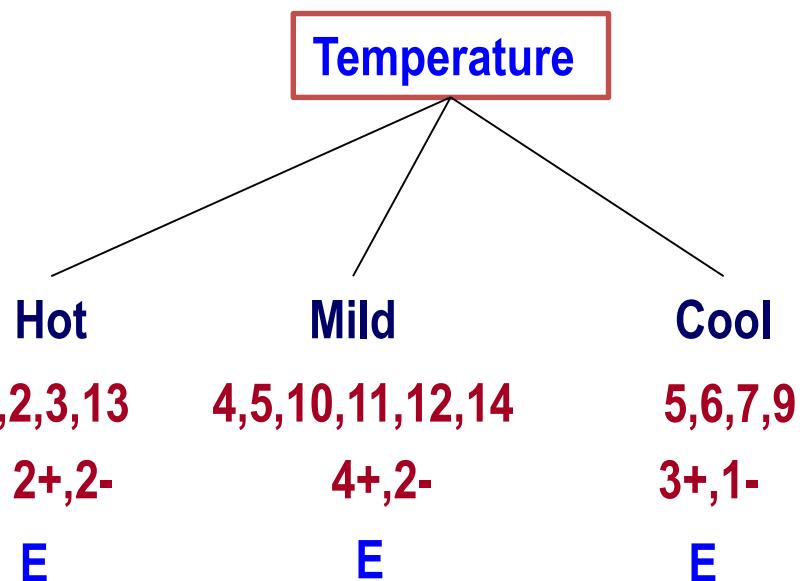


$$Gain(S, \text{Temperature}) = ?$$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

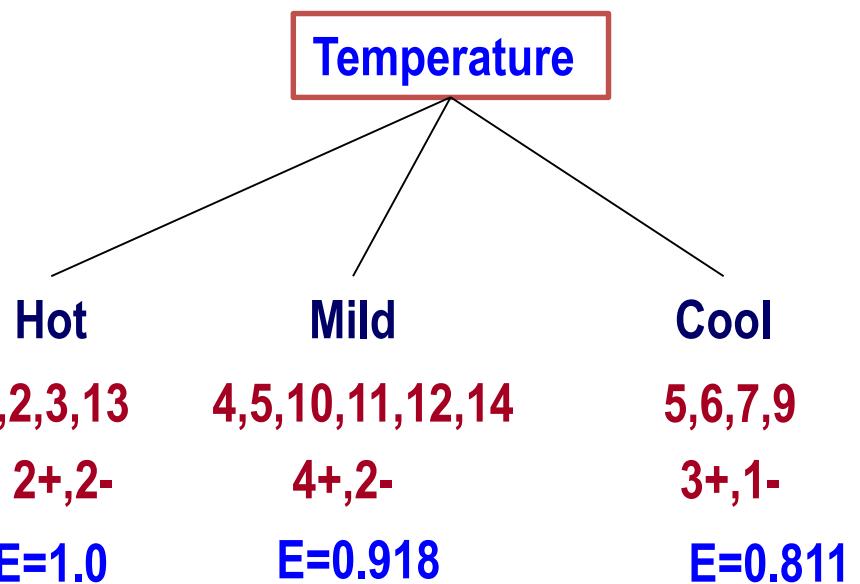


$$Gain(S, \text{Temperature}) = ?$$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Example

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

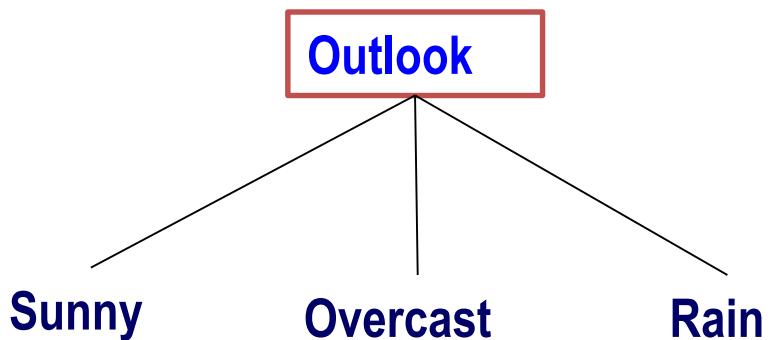


$$Gain(S, \text{Temperature}) = 0.029$$

| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Example

Pick Outlook as the root



| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

$$\text{Gain}(S, \text{Outlook}) = 0.246$$

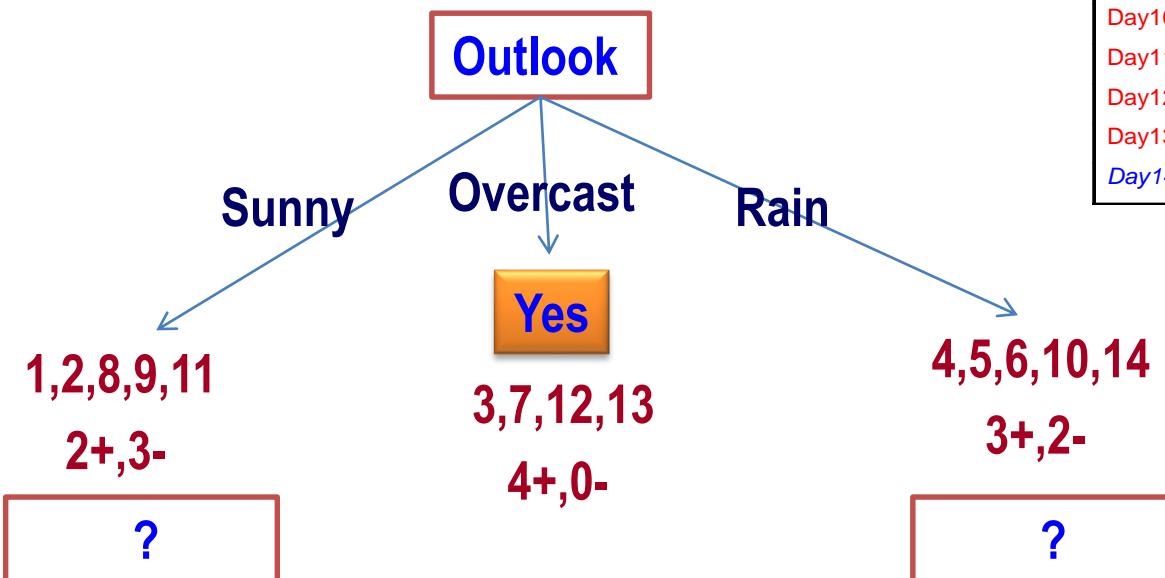
$$\text{Gain}(S, \text{Temperature}) = 0.029$$

$$\text{Gain}(S, \text{Humidity}) = 0.151$$

$$\text{Gain}(S, \text{Wind}) = 0.048$$

Example

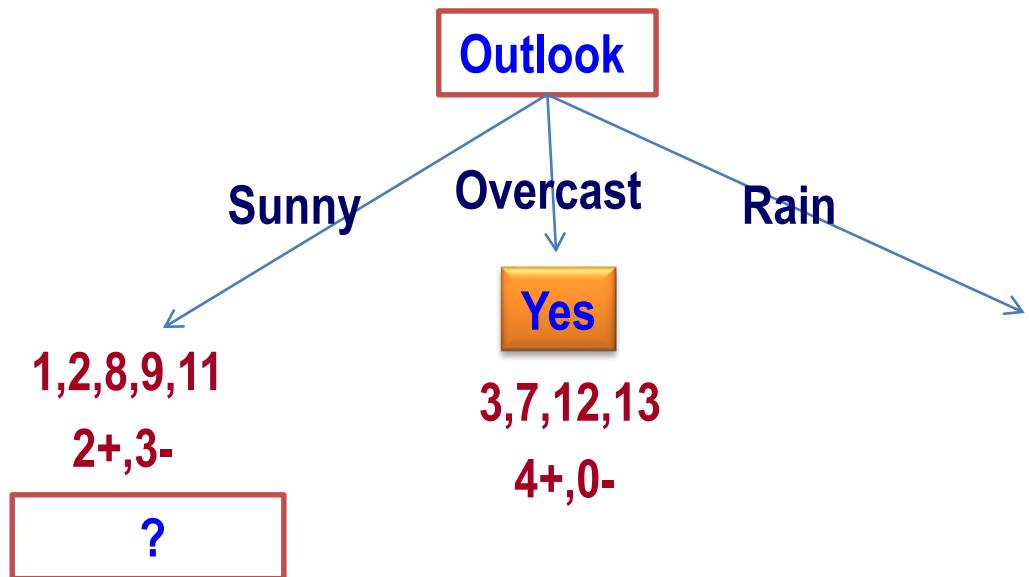
Pick Outlook as the root



| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Continue until: Every attribute is included in path, or, all examples in the leaf have same label

Example



| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

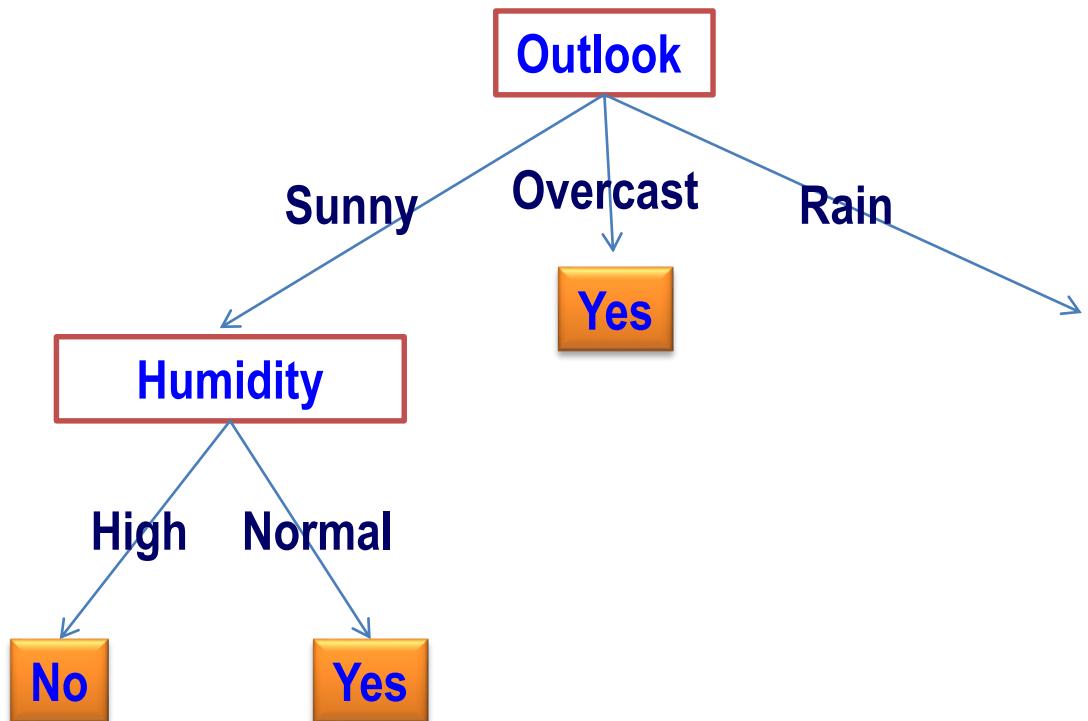
| | | | | | |
|----|-------|------|--------|--------|-----|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 8 | Sunny | Mild | High | Weak | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |

$$\text{Gain } (S_{\text{sunny}}, \text{Humidity}) = .97 - (3/5) * 0 - (2/5) * 0 = .97$$

$$\text{Gain } (S_{\text{sunny}}, \text{Temp}) = .97 - 0 - (2/5) * 1 = .57$$

$$\text{Gain } (S_{\text{sunny}}, \text{Wind}) = .97 - (2/5) * 1 - (3/5) * .92 = .02$$

Example



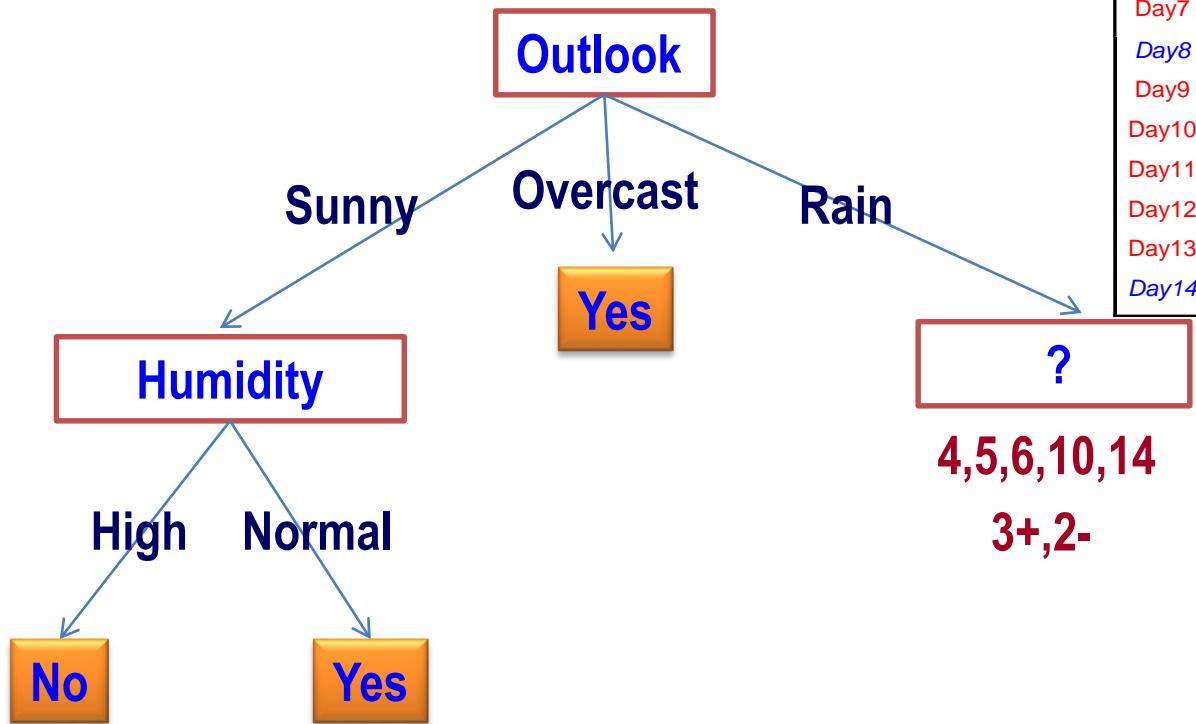
| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

$$\text{Gain } (S_{\text{sunny}}, \text{Humidity}) = .97 - (3/5) * 0 - (2/5) * 0 = .97$$

$$\text{Gain } (S_{\text{sunny}}, \text{Temp}) = .97 - 0 - (2/5) * 1 = .57$$

$$\text{Gain } (S_{\text{sunny}}, \text{Wind}) = .97 - (2/5) * 1 - (3/5) * .92 = .02$$

Example



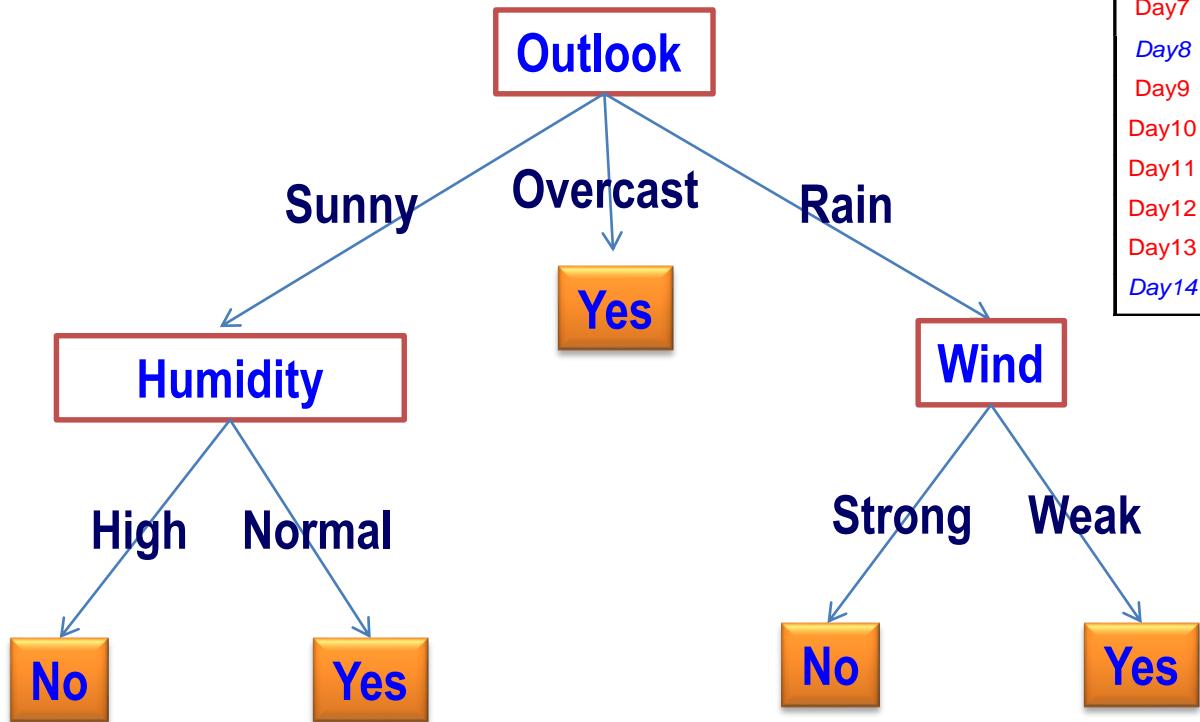
| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

4,5,6,10,14

3+,2-

Gain (S_{rain} , Humidity) =
 Gain (S_{rain} , Temp) =
 Gain (S_{rain} , Wind) =

Example



| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|-------|----------|-------------|----------|--------|-------------|
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

Summary on Build a Decision Tree

- Recursive partitioning
 - a top-down, greedy algorithm to fit the decision tree for the data.
- Top-down
 - Starting at the root node, split the data into subgroups that are as homogeneous as possible with respect to the output.
- Greedy method
 - Always make a locally optimal choice in the hope that this will lead to a globally optimal solution.

Three Steps in Building Decision Tree

- Selection of the best split
 - Which feature could give the “best” split – information gain?
- Stop-splitting rule
 - When should the splitting stop?
- Assignment of each leaf node to a class
 - Predict the value of the output at each leaf node.

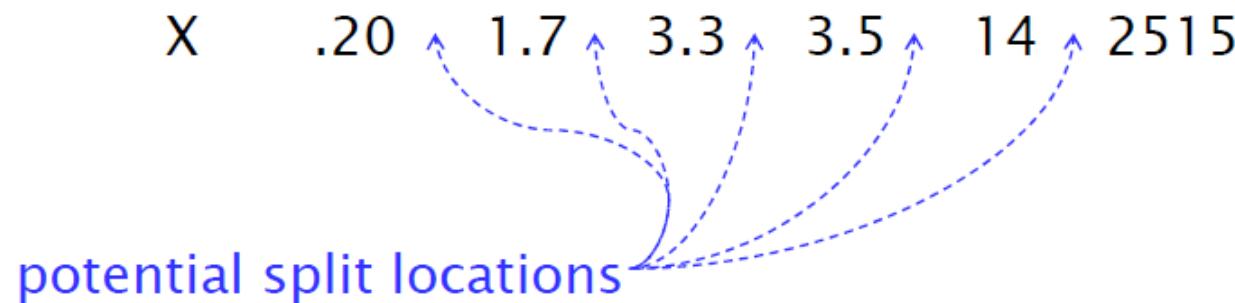
Possible Splits

Split on a discrete feature with B branches.

| Branch (B) | | |
|----------------|----------|------------|
| 2 | 3 | 4 |
| A, BCD | A, B, CD | A, B, C, D |
| B, ACD | A, C, BD | |
| C, ABD | A, D, BC | |
| D, ABC | B, C, AD | |
| AB, CD | B, D, AC | |
| AC, BD | C, D, AB | |
| AD, BC | | |

Possible Splits

Split on a continuous feature.



Selection of Best Splits

- Exhaustively examining all possible splits is time consuming.
- By default, algorithms will use exhaustive search if no. of possible splits is less than a given number (e.g. 5000).
- Otherwise, a clustering of levels of a feature is used to limit the possible splits to consider.
- An alternative way is to consider binary splits only

Remarks

- The **process** of selecting the best split on a node:
 1. Select the best split on each feature (i.e. choose number of branches and cut-off points).
 2. Select the best of these.

Stopping Criteria for Tree Induction

- Stop expanding a node when all the data belong to the same class
- Stop expanding a node when all the data have similar feature values
- Early termination

Stop Splitting Rule

- A simple method:
 - Continue splitting until every node is pure or contains only one instance.
 - Fit training data perfectly but may predict poorly on new data.
- Two approaches:
 - Top-down stopping rules (pre-pruning).
 - Bottom-up assessment criteria (post-pruning).

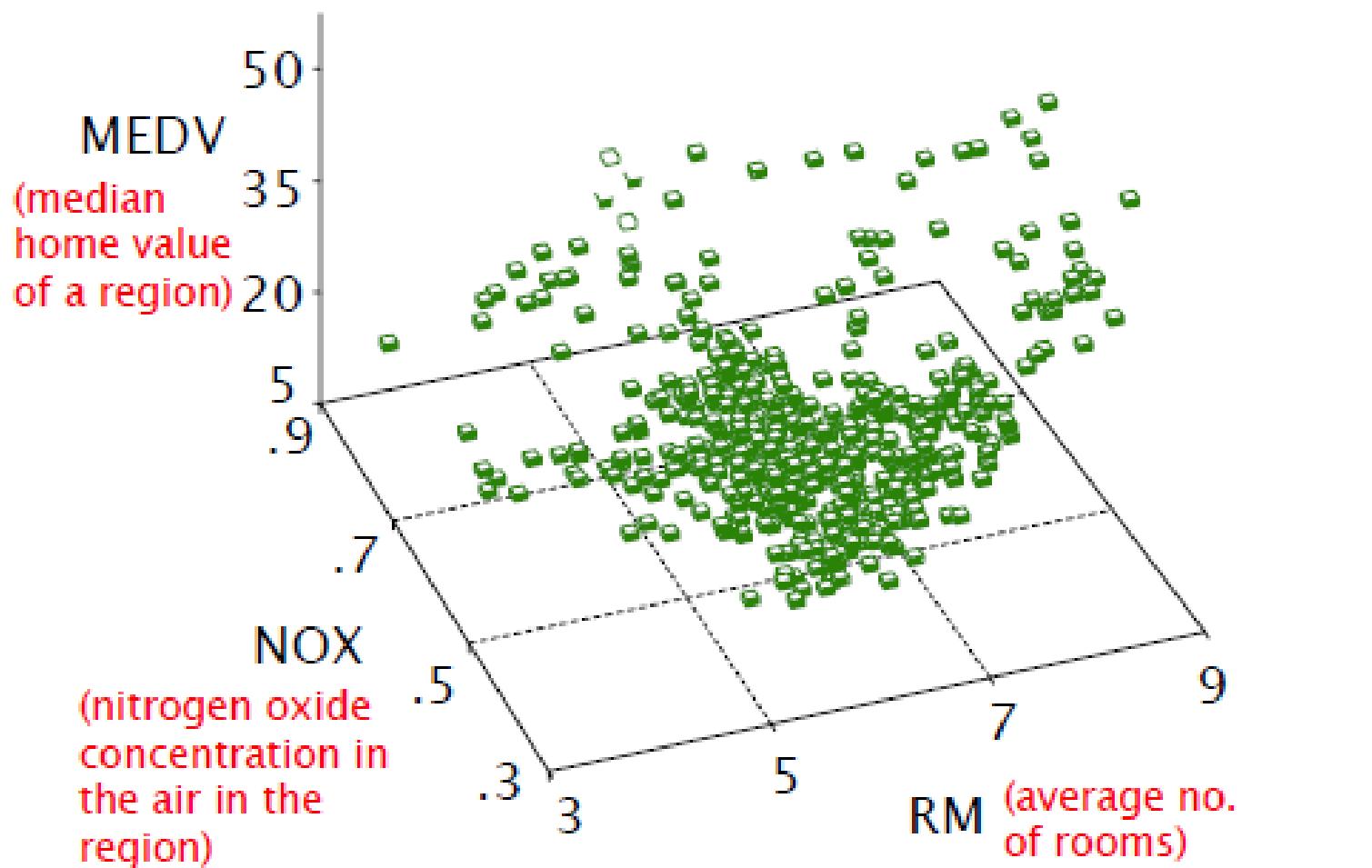
Assignment of Each Leaf Node to A Class

- **Classification tree**
 - Classify an input sample with the most common label in the leaf node (the percentage of the label is the probability).
- **Regression tree**
 - Predict an input in a node by the sample mean of the output values in the node.

Advantages of Trees

- **Easy to interpret**
 - Tree structured presentation.
- **Allow mixed input data types**
 - Discrete or continuous.
- **Allow discrete or continuous output**
- **Robust to outliers in feature vectors**
- **No problem with missing values**
- **Automatically**
 - Accommodates nonlinearity.
 - Selects input variables.

Boston Housing Data



Disadvantages of Trees

- **Most algorithms use univariate splits (split on only one variable)**
 - Solution: Linear combination split ($a_1x_1 + a_2x_2 < c?$).
- **Unstable fitted tree**
 - Often a small change in the data result in a very different series of splits.
- **Lack of smoothness (step function) in regression tree**
- **Splitting turns continuous input features into discrete features**
- **Spitting using a “greedy” algorithm**
 - While each split is optimal, the overall tree is not.

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary.
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen data.
- Need new ways for estimating errors.

How to Address Overfitting

Pre-Pruning (Early Stopping Rule)

- Stop the algorithm before it becomes a fully-grown tree.
- Typical stopping conditions for a node
 - Stop if all samples belong to the same class.
 - Stop if all the features values are the same.
- More restrictive conditions
 - Stop if number of samples is less than some user-specified threshold.
 - Stop if expanding the current node does not improve impurity measures, e.g., information gain.

How to Address Overfitting

Post-pruning

- Grow decision tree to its entirety.
- Trim the nodes of the decision tree in a bottom-up fashion.
- If generalization error improves after trimming, replace sub-tree by a leaf node.
- Class label of leaf node is determined from majority class of samples in the sub-tree.

Any Questions?



COMP3055

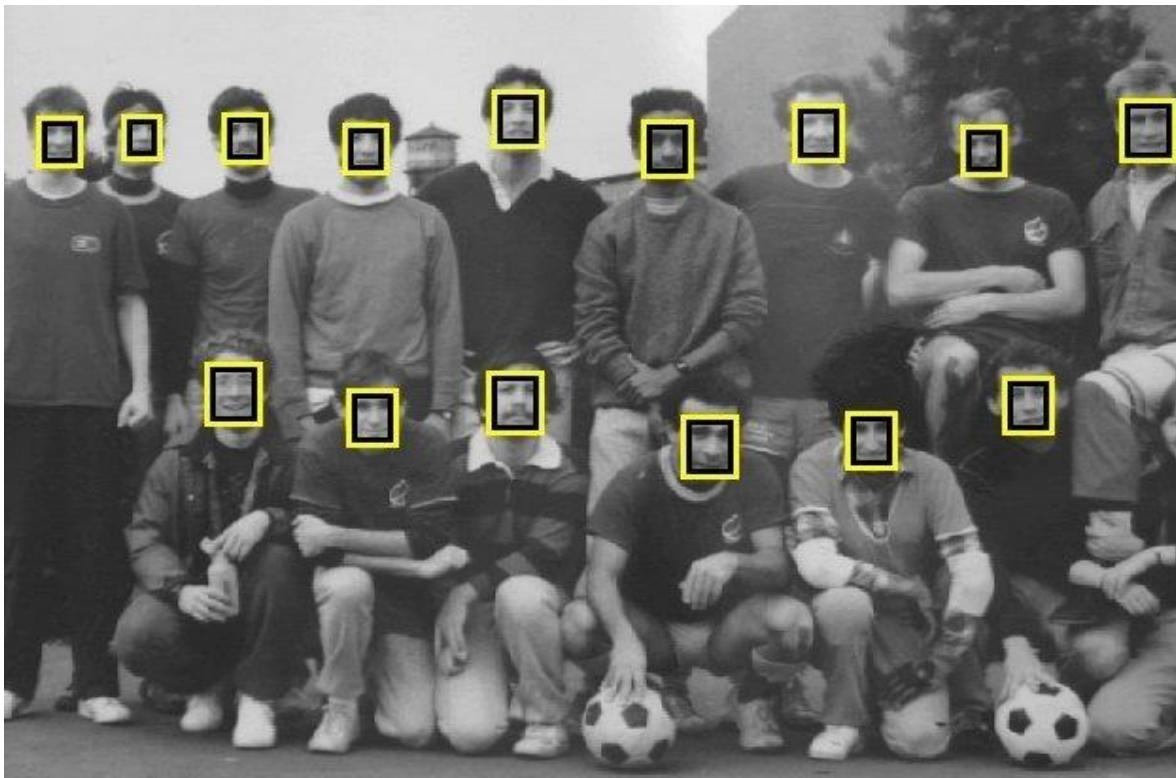
Machine Learning

Topic 10 – Data Processing and Representation

Ying Weng
2024 Autumn

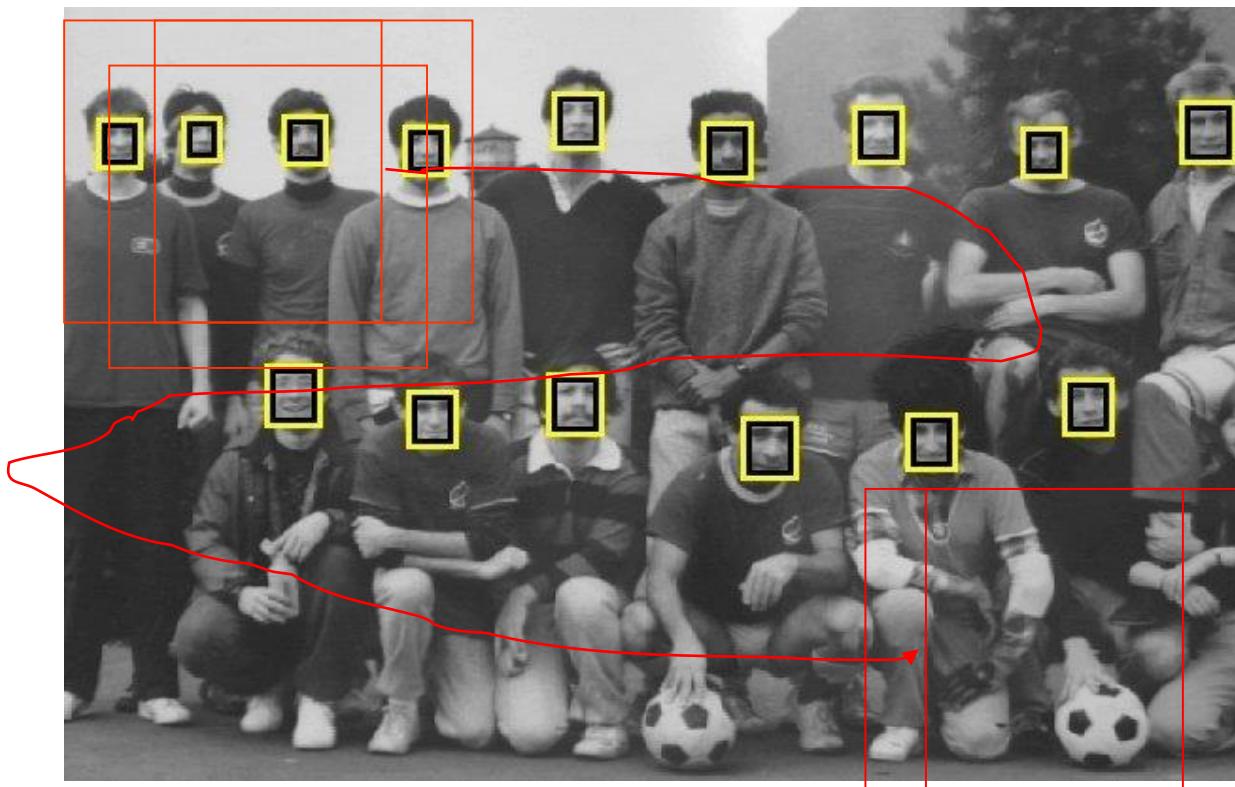
Problems

- Object Detection



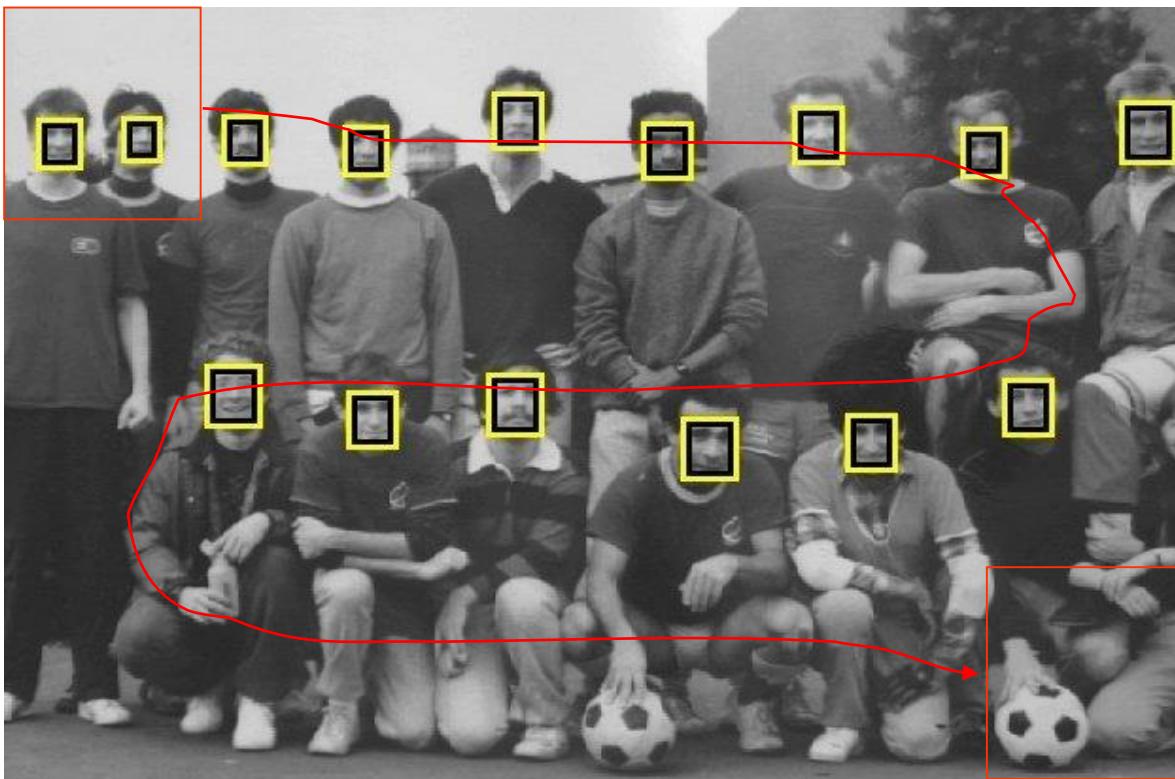
Problems

- Object Detection: Many detection windows



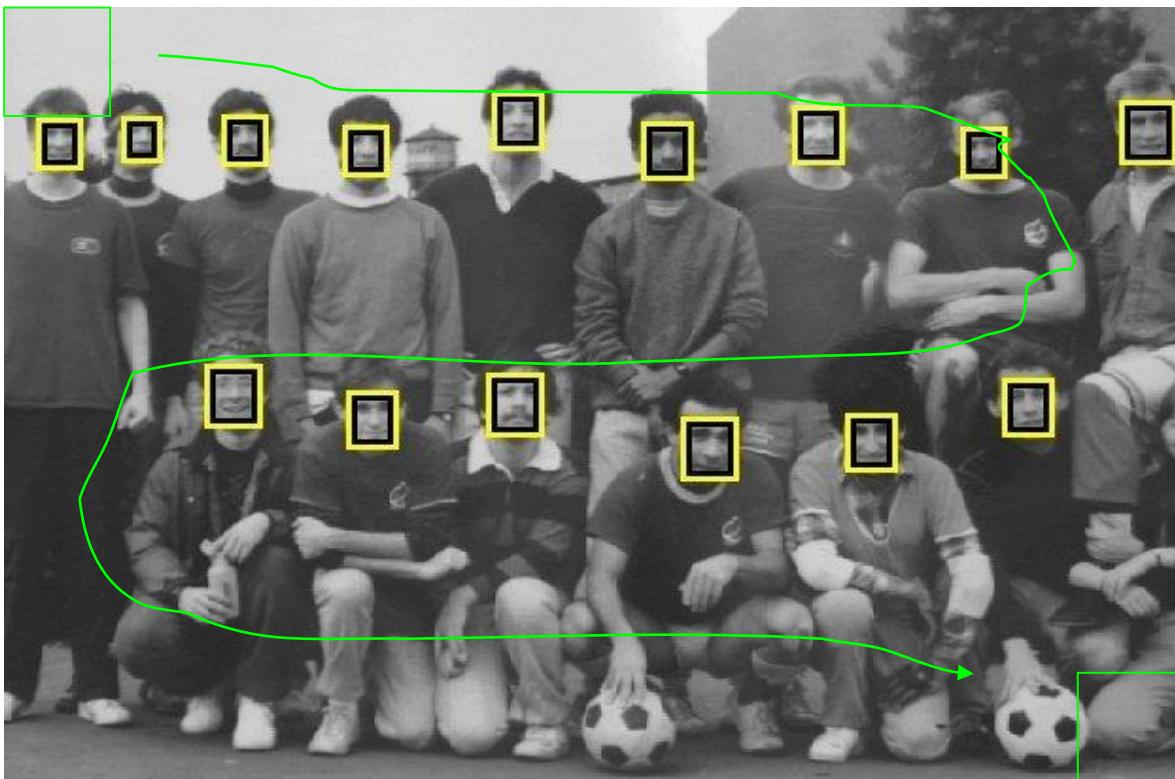
Problems

- Object Detection: Many detection windows



Problems

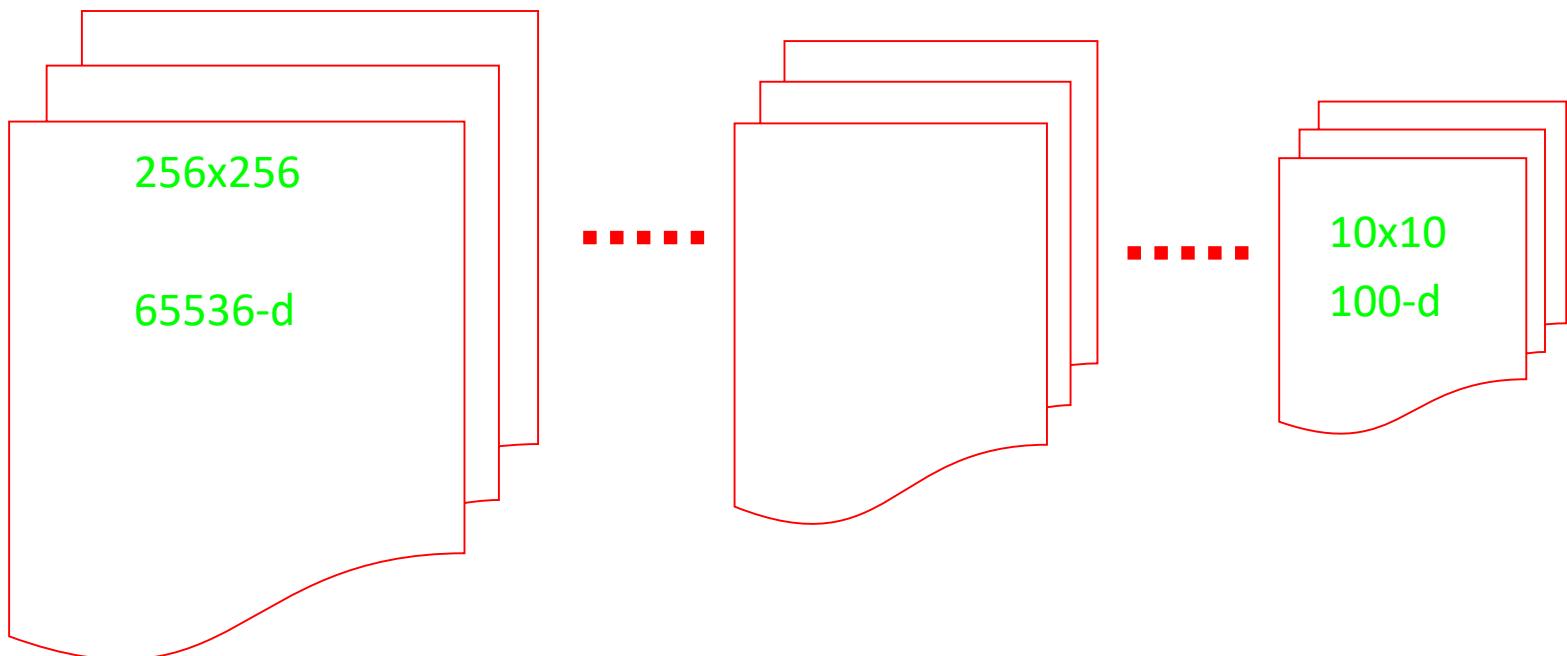
- Object Detection: Many detection windows



Problems

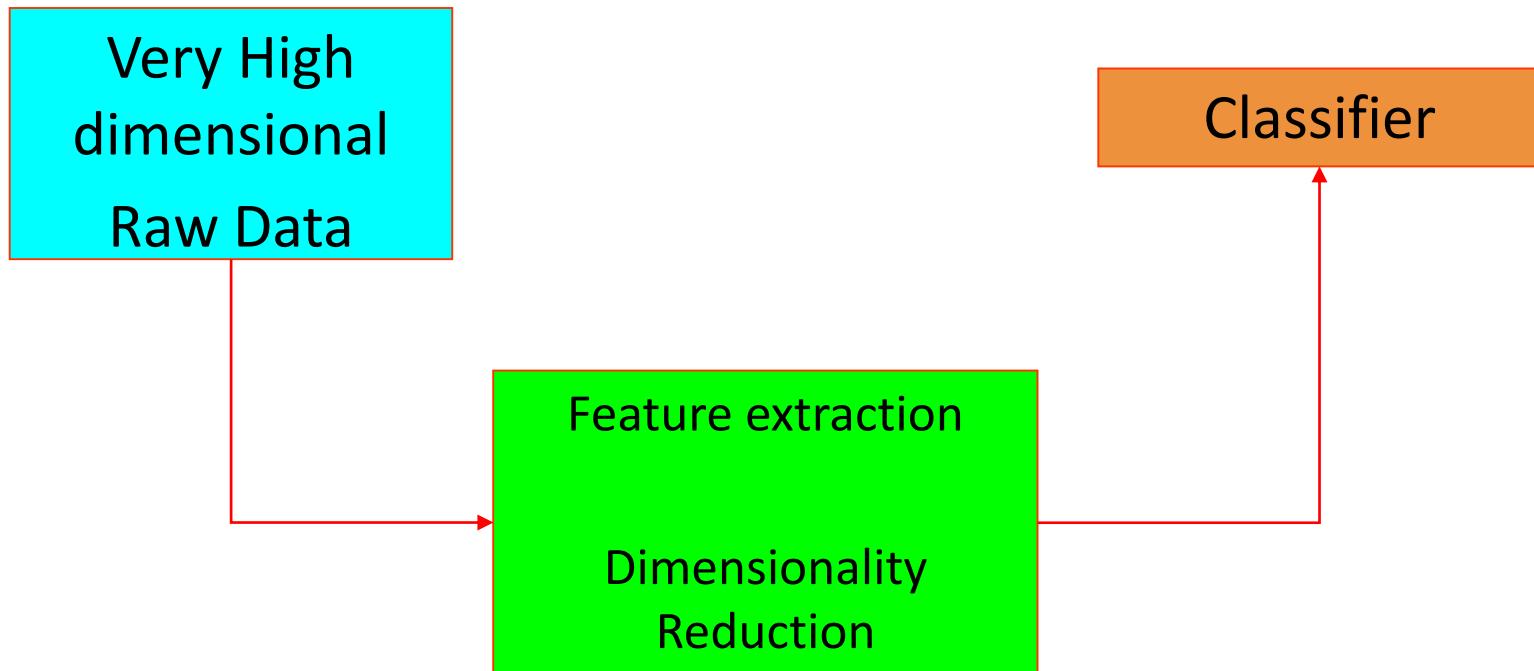
- Object Detection:

Each window is very high dimension data



Processing Methods

- General framework



Feature extraction/Dimensionality reduction

- It is impossible to process raw image data (pixels) directly
 - Too many of them (or data dimensionality too high)
 - Curse of dimensionality problem
- Process the raw pixel to produce a smaller set of numbers which will capture most information contained in the original data – this is often called a feature vector (feature extraction)

Feature extraction/Dimensionality reduction

- Basic Principle
 - From a raw data (vector) X of N-dimension to a new vector Y of n-dimension ($n \ll N$) via a transformation matrix A such that Y will **capture most information** in X

$$Y = AX = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & & & a_{1N} \\ & \Lambda & & & \\ \Lambda & & \Lambda & & \Lambda \\ & & & \Lambda & \Lambda \\ a_{n1} & & \Lambda & & a_{nN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

PCA

- Principal Component Analysis (PCA) is one of the most often used dimensionality reduction technique.

PCA Goals

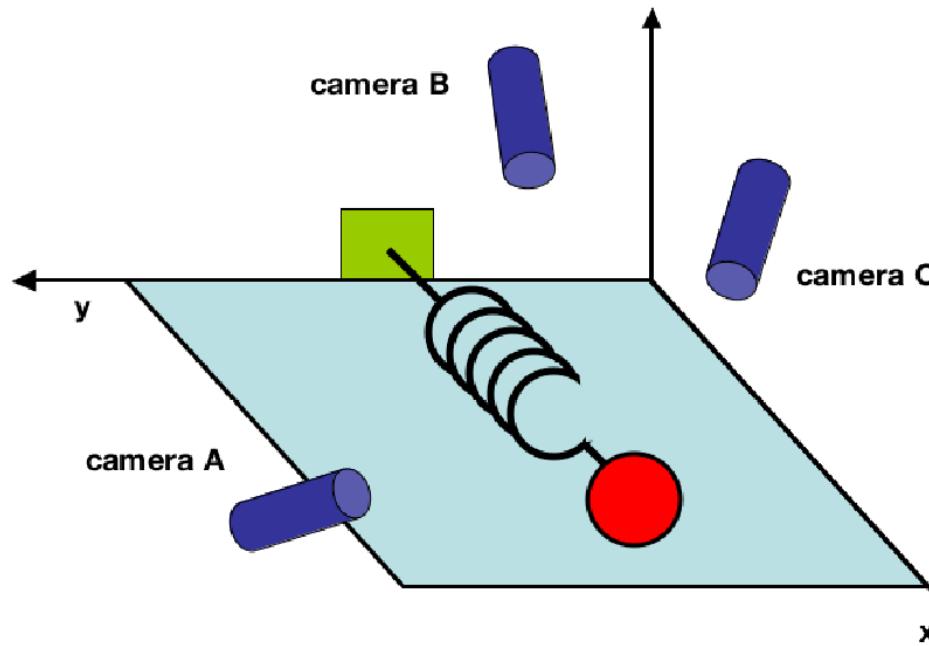
- We wish to explain/summarize the underlying variance-covariance structure of a large set of variables through a few linear combinations of these variables.

Applications

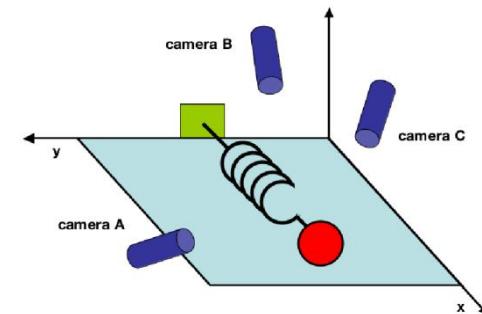
- Data Visualization
- Data Reduction
- Data Classification
- Trend Analysis
- Factor Analysis
- Noise Reduction

An Example

- A toy example: The movement of an ideal spring, the underlying dynamics can be expressed as a function of a single variable x .



An Example



- But, pretend that we are ignorant of that and
- Using 3 cameras, each records 2d projection of the ball's position. We record the data for 1 minutes at 200Hz
- We have 12,000, 6-d data
- How can we work out the dynamic is only along the x-axis
- Thus determining that only the dynamics along x are important and the rest are redundant.

An Example

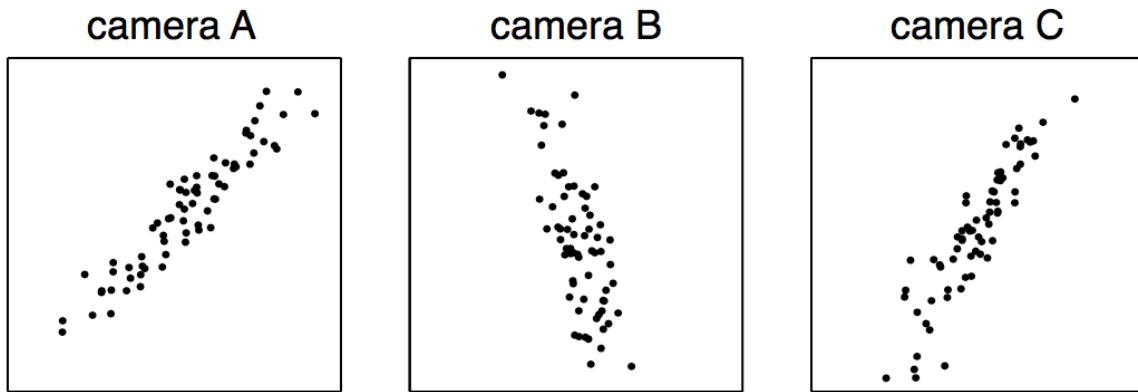


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

$$\vec{X} = \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

An Example

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & a_{16} \\ & \Lambda & a_{61} \\ & & a_{66} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

An Example

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & a_{16} \\ \vdots & \ddots & \vdots \\ a_{61} & \Lambda & a_{66} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

1st Eigenvector of
the Covariance
matrix

2nd Eigenvector
of the Covariance
matrix

6th Eigenvector
of the Covariance
matrix

An Example

1st Principal Component
2nd Principal Component

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & a_{16} \\ \vdots & \vdots & \vdots \\ a_{61} & \Lambda & a_{66} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

1st Eigenvector of
the Covariance
matrix

2nd Eigenvector
of the Covariance
matrix

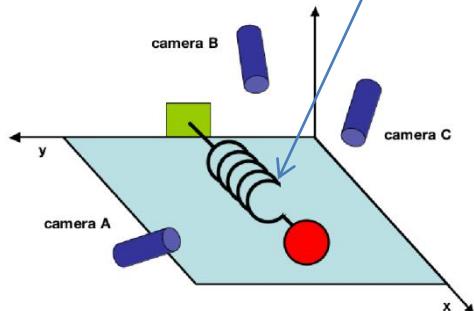
6th Eigenvector
of the Covariance
matrix

PCA

1st Eigenvector of
the Covariance
matrix

2nd Eigenvector
of the Covariance
matrix

Dynamic of the spring



$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & a_{16} \\ & \Lambda & \\ & & a_{61} & \Lambda & a_{66} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

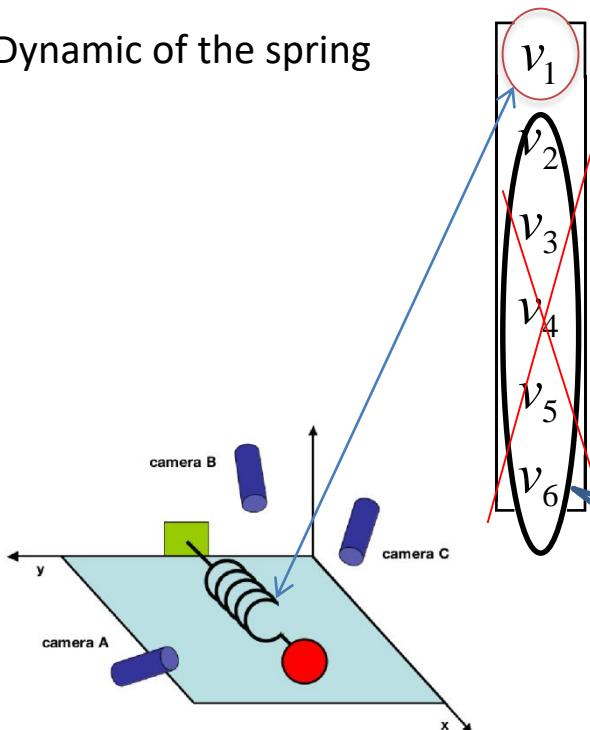
6th Eigenvector
of the Covariance
matrix

PCA

1st Eigenvector of
the Covariance
matrix

2nd Eigenvector
of the Covariance
matrix

Dynamic of the spring



$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & a_{16} \\ & \Lambda & \\ & & a_{66} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

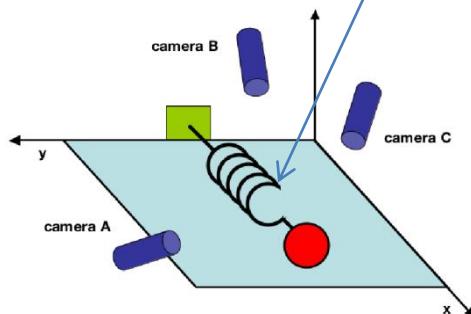
They contain no
useful
information and
can be discarded!

6th Eigenvector
of the Covariance
matrix

PCA

We only need
ONE number

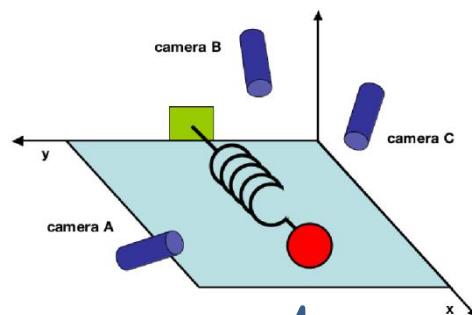
Dynamic of the spring



$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & a_{16} \\ & \Lambda & \\ & & \Lambda \\ a_{61} & \Lambda & a_{66} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix}$$

Instead of
SIX
Numbers!

PCA



Capture the
data patterns
of SIX
Numbers!

Linear
combination
(scaling) of ONE
variable

$$\begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \\ x_C \\ y_C \end{bmatrix} = M \begin{bmatrix} a_{11} \\ v_1 \\ a_{16} \end{bmatrix}$$

Variance and Covariance

- Consider two sets of measurements with zero means

$$A = \{a_1, a_2, \dots, a_n\}, \quad B = \{b_1, b_2, \dots, b_n\}$$

- The variance of A and B are individually defined as

$$\sigma_A^2 = \frac{1}{n} \sum_i a_i^2, \quad \sigma_B^2 = \frac{1}{n} \sum_i b_i^2$$

- The covariance (redundancy)

$$\text{covariance of } A \text{ and } B \equiv \sigma_{AB}^2 = \frac{1}{n} \sum_i a_i b_i$$

Noise(Signal-to-noise ratio)

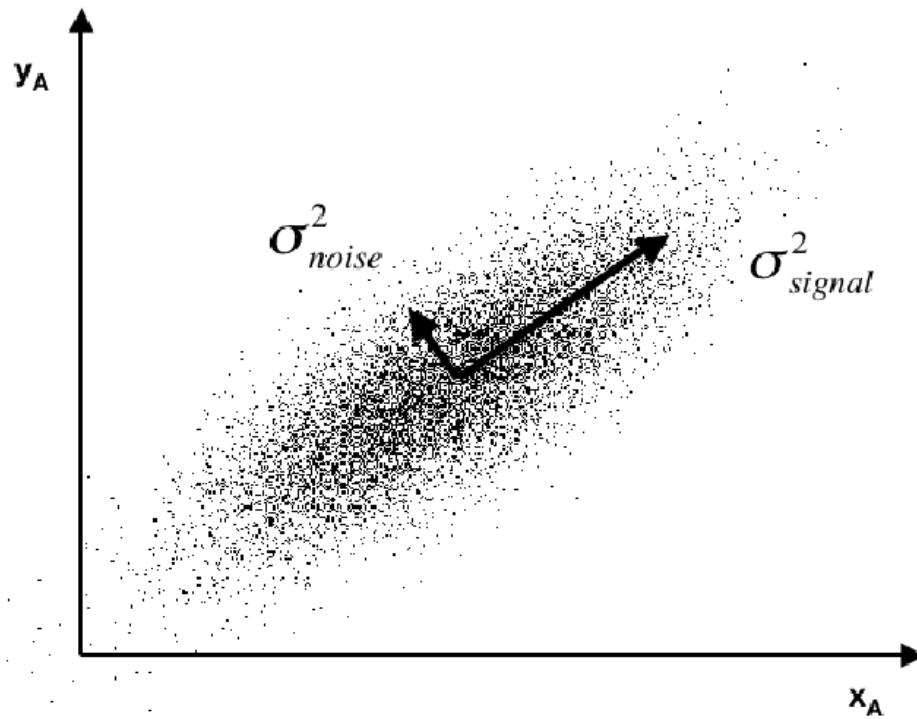


Figure 2: A simulated plot of (x_A, y_A) for camera A . The signal and noise variances σ_{signal}^2 and σ_{noise}^2 are graphically represented.

Redundancy

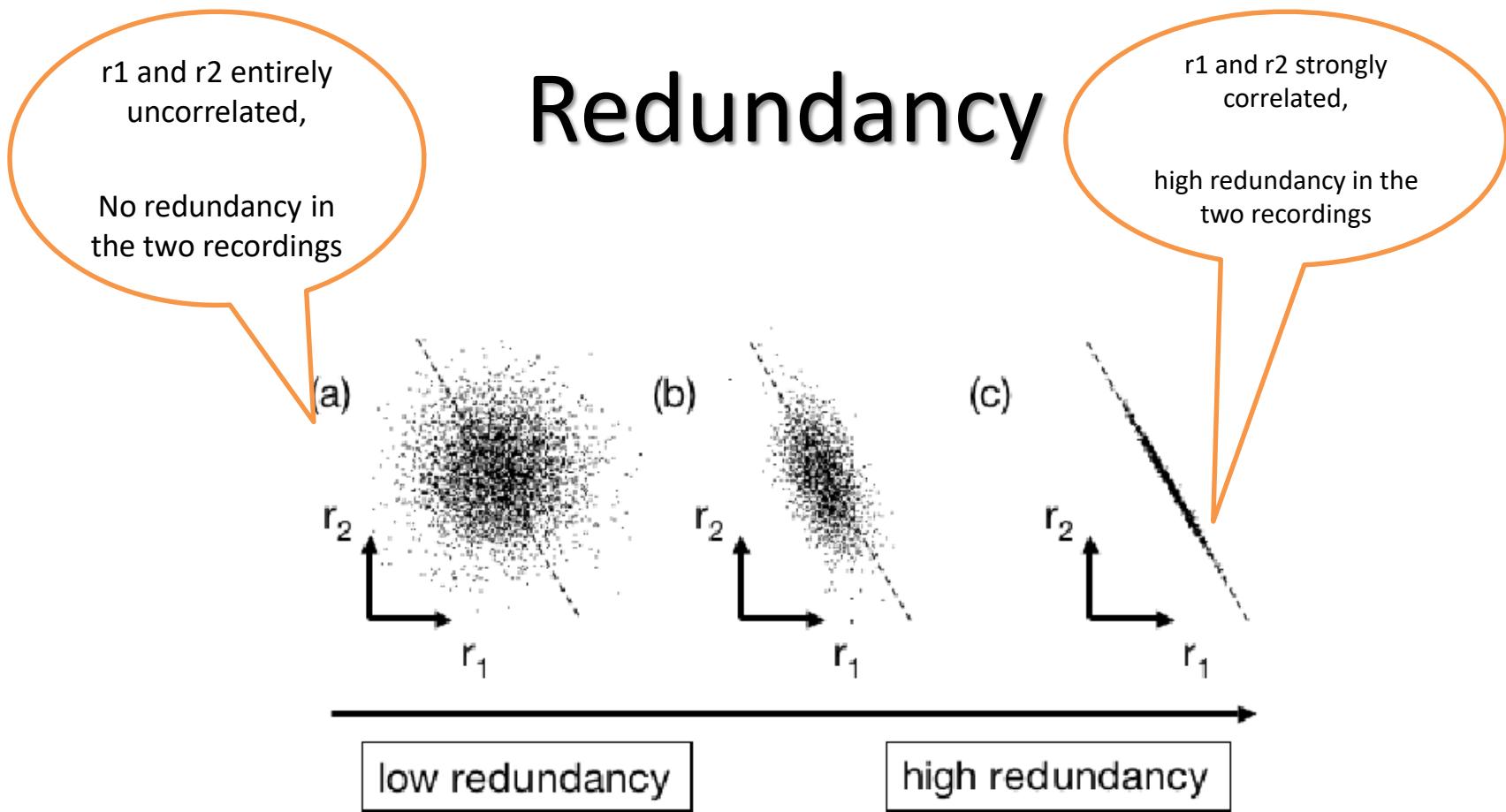


Figure 3: A spectrum of possible redundancies in data from the two separate recordings r_1 and r_2 (e.g. x_A, y_B). The best-fit line $r_2 = kr_1$ is indicated by the dashed line.

Covariance matrix

$$X = \begin{bmatrix} x_{11} & x_{12} & \Lambda & x_{1n} \\ x_{21} & x_{22} & \Lambda & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \Lambda & x_{mn} \end{bmatrix}$$

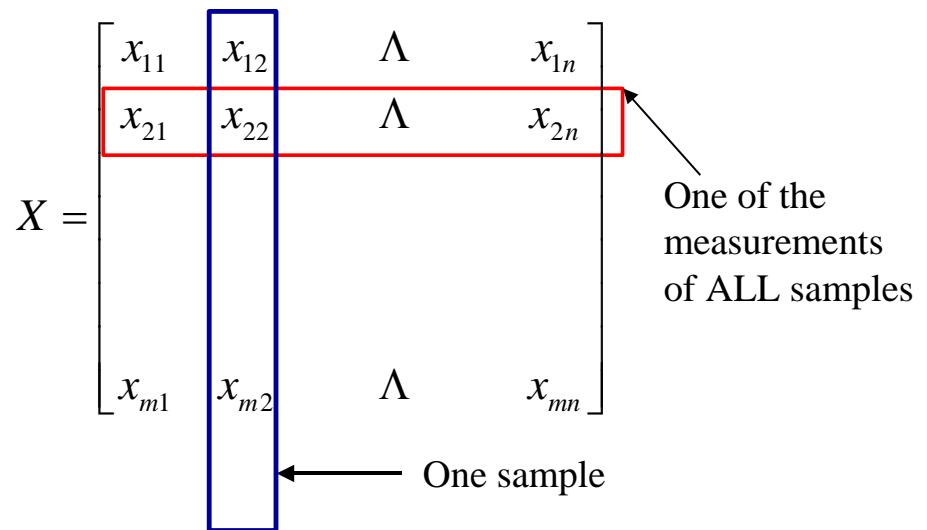
One sample (m-d) ←

One of the measurements of ALL samples (n samples)

Covariance matrix

$$S_X = \frac{1}{n-1} XX^T$$

is the covariance matrix of
the data



Covariance matrix

- Covariance measures the **degree of the linear relationship** between variables. A large positive value indicates positively correlated data. The absolute magnitude of the covariance measures the degree of redundancy.
- The ij^{th} element of S_x is the **dot product** between the vector of the i^{th} measurement type with the vector of the j^{th} measurement type.

Covariance matrix

$$S_x = \frac{1}{n-1} XX^T$$

- S_x is an $m \times m$ square matrix, m is the dimensionality of the measures (feature vectors)
- The **diagonal terms** of S_x are the variance of particular measurement type
- The **off-diagonal terms** of S_x are the covariance between measurement types

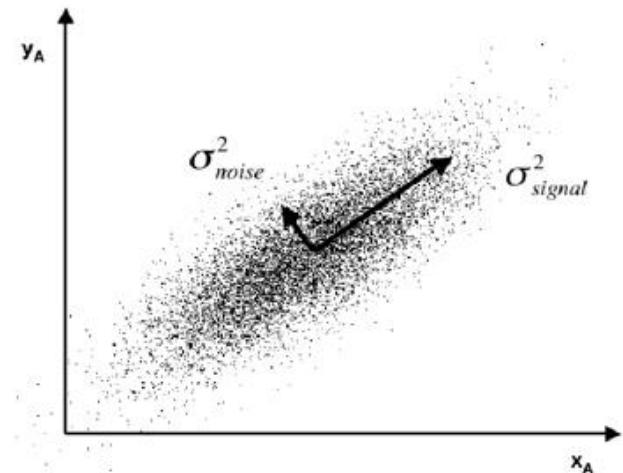
Covariance matrix

$$S_X = \frac{1}{n-1} XX^T$$

- S_x is special.
- It describes all **relationships** between pairs of measurements in our data set.
- A larger covariance indicates large correlation (**more redundancy**), zero covariance indicates entirely uncorrelated data.

Objective of PCA

- Minimize **redundancy**, measured by the magnitude of the covariance
- Maximize the **signal**, measured by the variance (diagonal element of the covariance matrix)



Covariance matrix

- If our **goal** is to **reduce redundancy**, then we want each variable co-vary a little as possible
- Precisely, we want the covariance between separate measurements to be **zero**
- **Diagonalise** the covariance matrix

Feature extraction/Dimensionality reduction

- Remove redundancy

$$Y = AX = \begin{bmatrix} y_1 \\ y_2 \\ M \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & & a_{1m} \\ & \Lambda & & \\ \Lambda & \Lambda & \Lambda & \Lambda \\ a_{m1} & \Lambda & & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ M \\ x_m \end{bmatrix}$$

- Optimal covariance matrix S_Y - off-diagonal terms set zero
- **Therefore removing redundancy, diagonalises S_Y**

»

How to find the transformation matrix

- Remove redundancy,

$$Y = AX = \begin{bmatrix} y_1 \\ y_2 \\ M \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & & & a_{1m} \\ & \Lambda & & \\ \Lambda & & \Lambda & \\ & & & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ M \\ x_m \end{bmatrix}$$

- Optimal covariance matrix S_Y - off-diagonal terms set zero
- **Therefore removing redundancy, diagonalises S_Y**

Solving PCA: Diagonalising the Covariance Matrix

- There are many ways to **diagonalizing** S_Y , PCA choose the simplest method. (**eigenvector decomposition**)
- PCA assumes all **basis vectors** are orthonormal. P is an **orthonormal matrix**

$$p_i = [p_{i1} \quad p_{i2} \quad \Lambda \quad p_{im}]$$
$$p_i p_j = \delta_{ij} \quad \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

- PCA assumes the directions with the **largest variances** are the most important or most **principal**.

Solving PCA: Diagonalising the Covariance Matrix

- PCA works as follows
 - PCA first selects a normalised direction in m -dimensional space along which the variance of X is maximised – it saves the direction as p_1
 - It then finds another direction, along which variance is maximised subject to the orthonormal condition – it restricts its search to all directions perpendicular to all previous selected directions.
 - The process could continue until m directions are found. The resulting ORDERED set of p 's are the ***principal components***
 - The variances associated with each direction p_i quantify how principal (important) each direction is – thus rank-ordering each basis according to the corresponding variance.

Karhunen-Loeve Transform (KLT) (Eigenvector Transform)

- Forward KLT

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & p_{1m} \\ & \Lambda & & \\ \Lambda & \Lambda & \Lambda & \Lambda \\ p_{m1} & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

- Inverse KLT

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & p_{m1} \\ & \Lambda & & \\ \Lambda & \Lambda & \Lambda & \Lambda \\ p_{1m} & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Karhunen-Loeve Transform (KLT) (Eigenvector Transform)

- Forward KLT

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & p_{1m} \\ & \Lambda & & \\ \Lambda & \Lambda & \Lambda & \Lambda \\ p_{m1} & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

1st Eigenvector of the Covariance matrix

- Inverse KLT

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & p_{m1} \\ & \Lambda & & \\ \Lambda & \Lambda & \Lambda & \Lambda \\ p_{1m} & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

m-th Eigenvector of the Covariance matrix

Karhunen-Loeve Transform (KLT) (Eigenvector Transform)

- Forward KLT

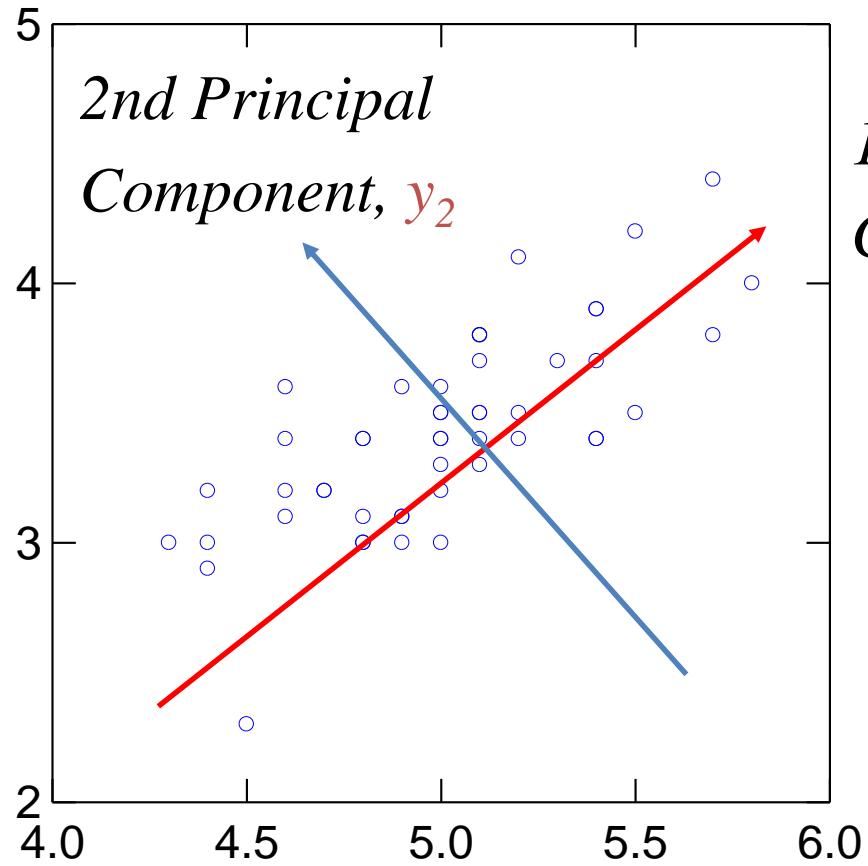
$$\begin{bmatrix} y_1 \\ y_2 \\ M \\ y_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & p_{1m} \\ & \Lambda & & \\ \Lambda & & \Lambda & \\ p_{m1} & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ M \\ x_m \end{bmatrix}$$

1st Eigenvector of the Covariance matrix

- Inverse KLT

$$\begin{bmatrix} x_1 \\ x_2 \\ M \\ x_m \end{bmatrix} = \begin{bmatrix} p_{11} \\ \Lambda \\ p_{1m} \\ \Lambda \\ \Lambda \\ \Lambda \\ \Lambda \\ \Lambda \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ M \\ y_m \end{bmatrix}$$

m-th Eigenvector of the Covariance matrix



1st Principal Component, y_1

2nd Principal Component, y_2

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & & p_{1m} \\ \Lambda & \ddots & \Lambda & & x_1 \\ & & \ddots & \ddots & x_2 \\ & & & \Lambda & M \\ p_{m1} & & & & x_m \end{bmatrix}$$

Solving PCA Eigenvectors of Covariance

$$Y = PX \quad S_Y = \frac{1}{n-1} YY^T$$

- Find some orthonormal matrix P such that S_Y is diagonalized.
- The row of P are the principal components of X

Solving PCA Eigenvectors of Covariance

$$S_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} (PX)(PX)^T$$

$$S_Y = \frac{1}{n-1} PXX^T P^T$$

$$S_Y = \frac{1}{n-1} P(XX^T)P^T$$

$$S_Y = \frac{1}{n-1} PAP^T$$

where $A = XX^T$

- A is a symmetric matrix, which can be diagonalised by an orthonormal matrix of its eigenvectors.

Solving PCA Eigenvectors of Covariance

$$A = EDE^T$$

- D is a diagonal matrix, E is a matrix of eigenvectors of A arranged as **columns**
- The matrix A has $r \leq m$ orthonormal eigenvectors, where r is the rank of A .
- r is less than m when A is degenerate or all data occupy a subspace of dimension $r < m$

Solving PCA Eigenvectors of Covariance

$$A = EDE^T \quad P \equiv E^T \quad A = P^T DP$$

- Select the matrix P to be a matrix where each row p_i is an eigenvector of XX^T .

$$S_Y = \frac{1}{n-1} PAP^T$$

$$S_Y = \frac{1}{n-1} P(P^T DP)P^T$$

$$S_Y = \frac{1}{n-1} PP^T DPP^T = \frac{1}{n-1} (PP^T) D (PP^T)$$

$$S_Y = \frac{1}{n-1} D$$

Solving PCA Eigenvectors of Covariance

$$S_Y = \frac{1}{n-1} D$$

- The **principal component** of X are the eigenvectors of the covariance matrix of X (S_x , XX^T); or the rows of P
- The *i*th diagonal value of S_Y is the **variance** of X along p_i

PCA Procedures

- Get data (example)
- Step 1
 - Subtract the mean (example)
- Step 2
 - Calculate the covariance matrix
- Step 3
 - Calculate the eigenvectors and eigenvalues of the covariance matrix

A 2D Numerical Example

PCA Example – Data

- Original data

| x | y |
|-----|-----|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

STEP 1

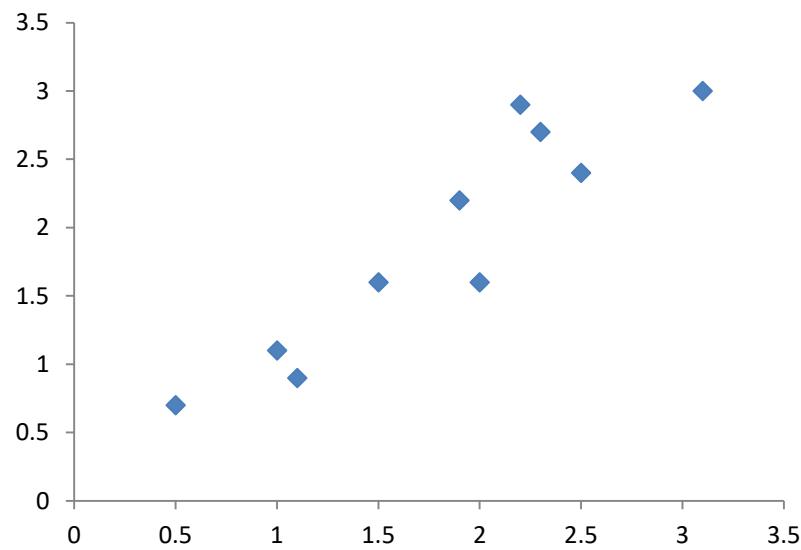
- Subtract the mean
- from each of the data dimensions. All the x values have average (x) subtracted and y values have average (y) subtracted from them. This produces a data set whose mean is zero.
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.

STEP 1

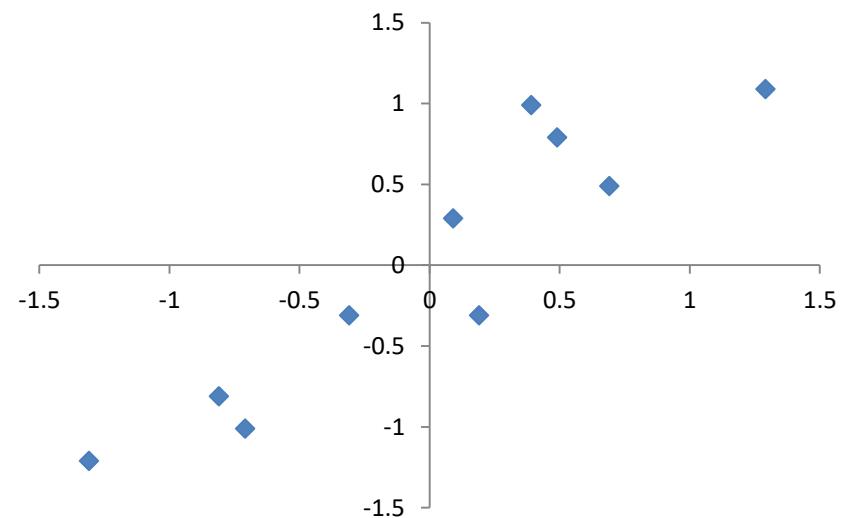
- Zero-mean data

| | |
|-------|-------|
| 0.69 | 0.49 |
| -1.31 | -1.21 |
| 0.39 | 0.99 |
| 0.09 | 0.29 |
| 1.29 | 1.09 |
| 0.49 | 0.79 |
| 0.19 | -0.31 |
| -0.81 | -0.81 |
| -0.31 | -0.31 |
| -0.71 | -1.01 |

STEP 1



Original



Zero-mean

STEP 2

- Calculate the covariance matrix

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- since the non-diagonal elements in this covariance matrix are **positive**, we should expect that both the x and y variable **increase together**.

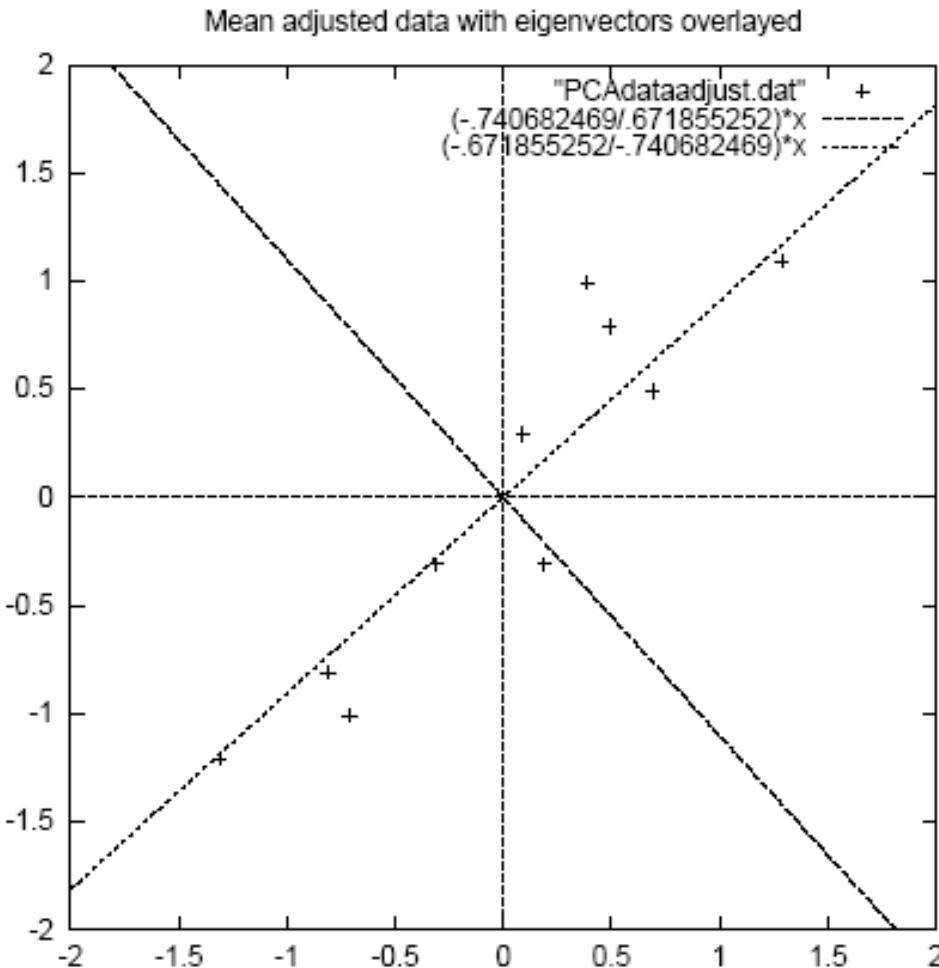
STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

STEP 3



- eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.
- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

Feature Extraction

- Reduce dimensionality and form *feature vector*
 - the eigenvector with the *highest* eigenvalue is the *principal component* of the data set.
 - In our example, the eigenvector with the largest eigenvalue was the one that pointed down the middle of the data.
 - Once eigenvectors are found from the covariance matrix, the next step is to **order them by eigenvalue**, highest to lowest. This gives you the components in order of significance.

Feature Extraction

- Eigen Feature Vector

$$\text{FeatureVector} = (\text{eig}_1 \text{ eig}_2 \text{ eig}_3 \dots \text{ eig}_n)$$

We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} - .677873399 \\ - .735178656 \end{pmatrix}$$

Eigen-analysis/ Karhunen Loeve Transform

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & & p_{1m} \\ & \Lambda & & & \\ \Lambda & & \Lambda & & \Lambda \\ p_{m1} & & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Eigen
Matrix

Eigen-analysis/ Karhunen Loeve Transform

Back to our example: Transform data to eigen-space (x', y')

$$x' = -0.68x - 0.74y$$

$$y' = -0.74x + 0.68y$$

- .827970186

- .175115307

x
0.69

y
0.49

-1.31 -1.21

1.77758033

.142857227

0.39 0.99

- .992197494

.384374989

0.09 0.29

- .274210416

.130417207

1.29 1.09

-1.67580142

- .209498461

0.49 0.79

- .912949103

.175282444

0.19 -0.31

.0991094375

- .349824698

-0.81 -0.81

1.14457216

.0464172582

-0.31 -0.31

.438046137

.0177646297

-0.71 -1.01

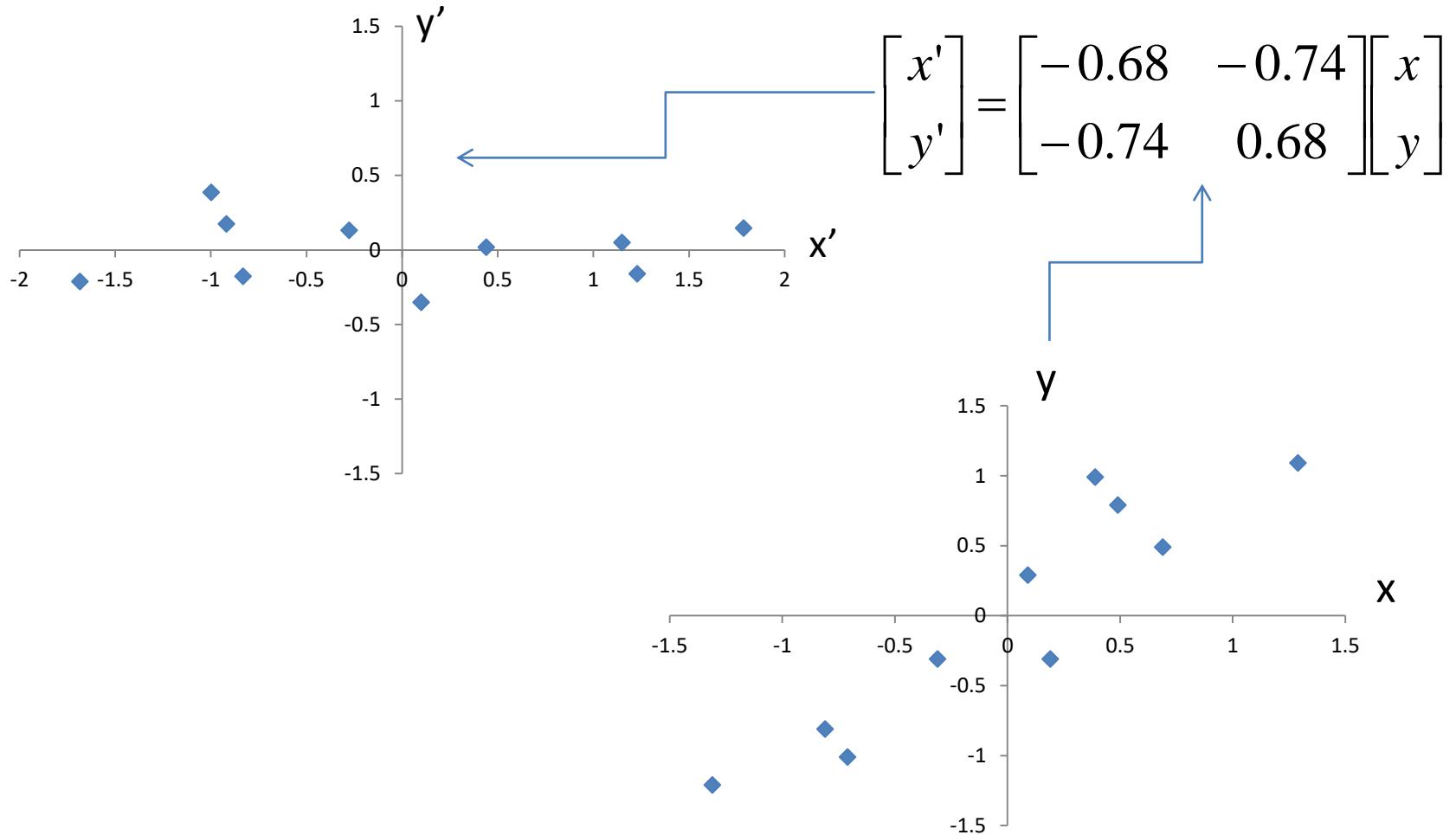
1.22382056

- .162675287

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Eigen-analysis/ Karhunen Loeve Transform



Reconstruction of original Data/Inverse Transformation

- Forward Transform

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Inverse Transform

$$\begin{bmatrix} x_{construction} \\ y_{construction} \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Reconstruction of original Data/Inverse Transformation

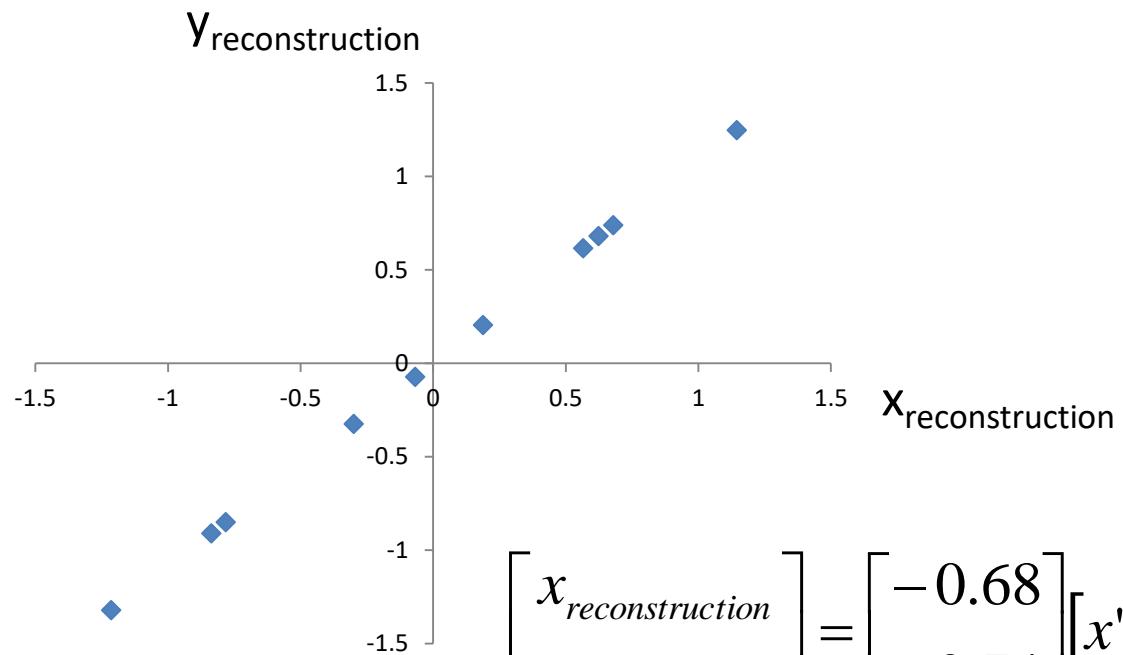
- If we reduced the dimensionality, obviously, when reconstructing the data we would lose those dimensions we chose to discard.
- Throw away the less important one, throw away y' and only keep x'

$$\begin{bmatrix} x_{reconstruction} \\ y_{reconstruction} \end{bmatrix} = \begin{bmatrix} -0.68 \\ -0.74 \end{bmatrix} [x']$$

Reconstruction of original Data/Inverse Transformation

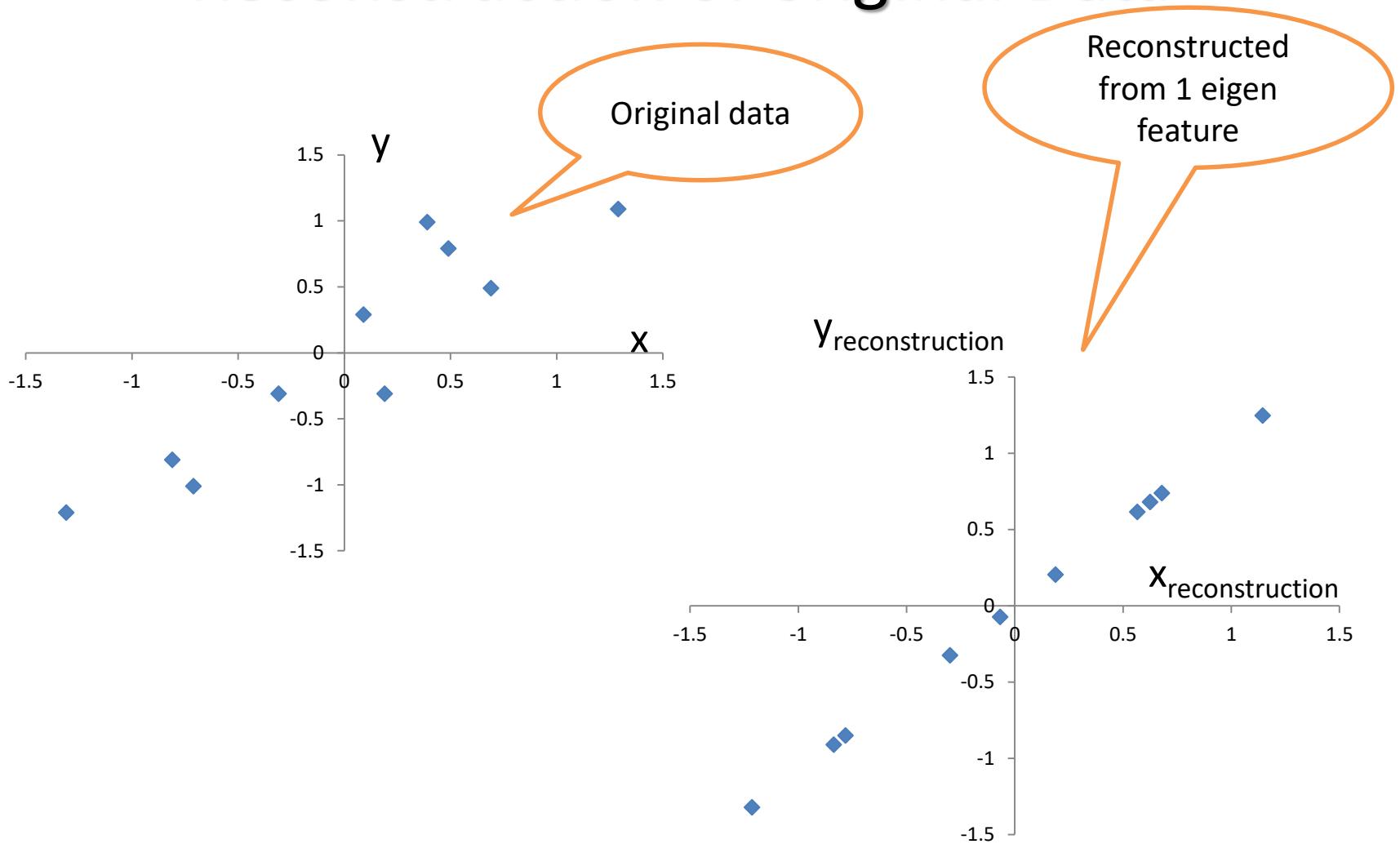
x'

-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056



$$\begin{bmatrix} x_{\text{reconstruction}} \\ y_{\text{reconstruction}} \end{bmatrix} = \begin{bmatrix} -0.68 \\ -0.74 \end{bmatrix} [x']$$

Reconstruction of original Data



Feature Extraction/Eigen-features

Eigen
Feature
vector

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} p_{11} & \Lambda & & p_{1m} \\ & \Lambda & & \\ \Lambda & & \Lambda & \Lambda \\ p_{m1} & \Lambda & & p_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

PCA Applications –General

1st eigenvector

$$Y = PX \quad X = P^T Y$$

$$\begin{bmatrix} x_1 \\ x_2 \\ M \\ M \\ M \\ x_m \end{bmatrix} = \begin{bmatrix} p_{11} & p_{21} & \Lambda & p_{m1} \\ p_{12} & p_{22} & \Lambda & p_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1m} & p_{2m} & \Lambda & p_{mm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ M \\ M \\ M \\ y_m \end{bmatrix}$$

mth eigenvector

- Data compression/dimensionality reduction

PCA Applications -General

- Data compression/dimensionality reduction

$$p_i = [p_{i1} \quad p_{i2} \quad \Lambda \quad p_{im}]$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ M \\ x_i \\ M \\ x_m \end{bmatrix} = [y_1 p_1^T + y_2 p_2^T + \Lambda + y_m p_m^T]$$

PCA Applications -General

- Data compression/dimensionality reduction
- Reduce the number of features needed for effective data representation by discarding those features having small variances
- The most interesting dynamics occur only in the first l dimensions ($l \ll m$).

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ M \\ M \\ \hat{x}_m \end{bmatrix} = \begin{bmatrix} p_{11} & p_{21} & \Lambda & p_{l1} \\ p_{12} & p_{22} & \Lambda & p_{l2} \\ M & M & \Lambda & M \\ M & M & \Lambda & M \\ p_{1m} & p_{2m} & \Lambda & p_{lm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ M \\ M \\ y_l \end{bmatrix} = [y_1 p_1^T + y_2 p_2^T + \Lambda + y_l p_l^T]$$
$$p_i = [p_{i1} \quad p_{i2} \quad \Lambda \quad p_{im}] \qquad \qquad X = [y_1 p_1^T + y_2 p_2^T + \Lambda + y_m p_m^T]$$

PCA Applications -General

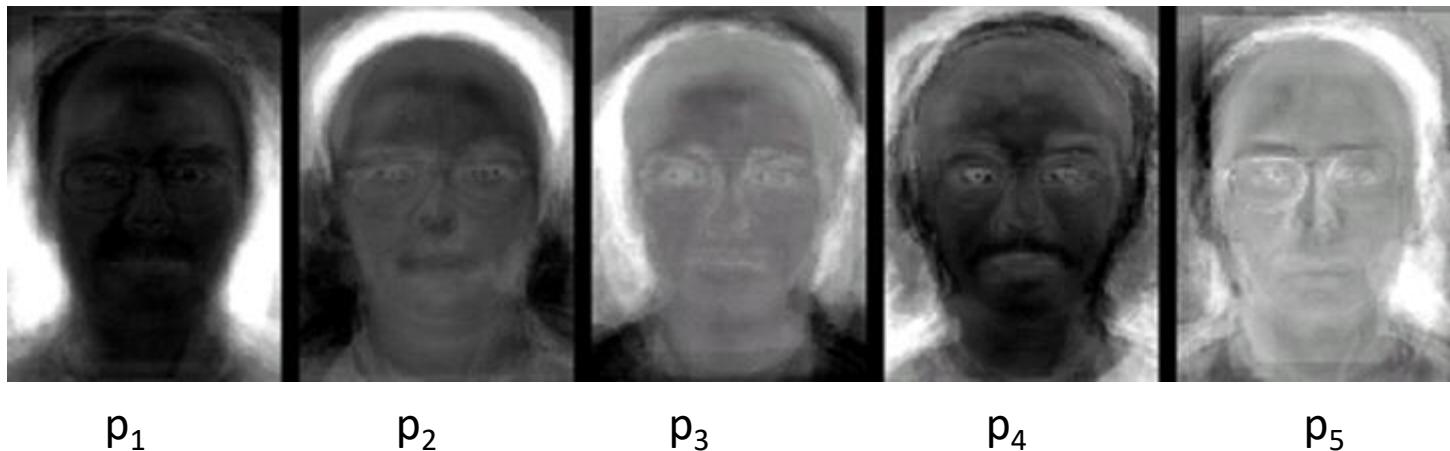
- Data compression/dimensionality reduction
- Reduce the number of features needed by discarding those features having small variance.
- The most interesting dynamics occur only in the first l dimensions ($l \ll m$).

We know what can be thrown away; or do we?

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_m \end{bmatrix} = \begin{bmatrix} p_{11} & p_{21} \\ p_{12} & p_{22} \\ \vdots & \vdots \\ p_{1m} & p_{2m} \end{bmatrix} \begin{bmatrix} p_{l1} \\ p_{l2} \\ \vdots \\ p_{lm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_l \end{bmatrix} = [y_1 p_1^T + y_2 p_2^T + \Lambda + y_l p_l^T]$$
$$p_i = [p_{i1} \quad p_{i2} \quad \Lambda \quad p_{im}]$$
$$X = [y_1 p_1^T + y_2 p_2^T + \Lambda + y_m p_m^T]$$

Eigenface Example

- A 256x256 face image, 65536 dimensional vector, X , representing the face images with much lower dimensional vectors for analysis and recognition
 - Compute the covariance matrix, find its eigenvector and eigenvalue
 - Throw away eigenvectors corresponding to small eigenvalues, and keep the first l ($l \ll m$) principal components (eigenvectors)



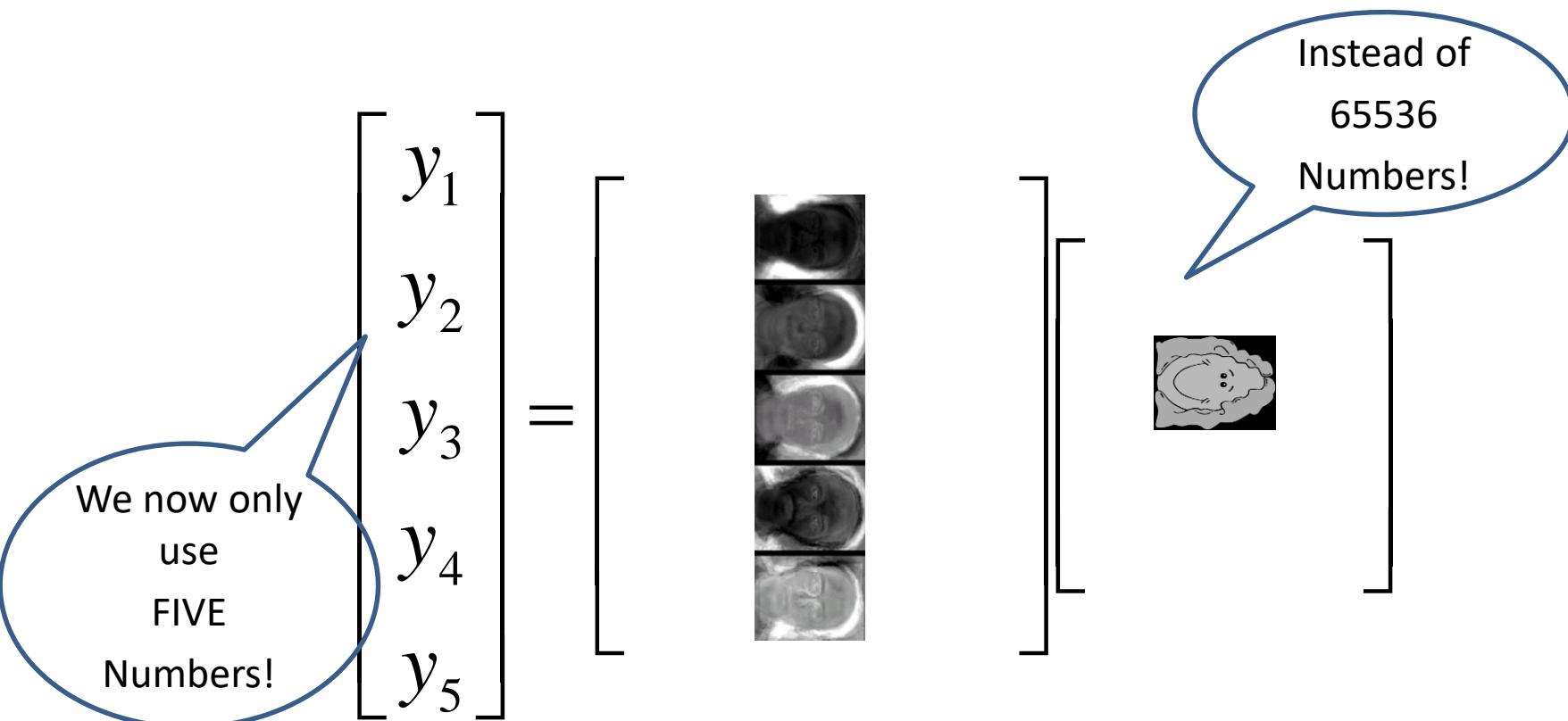
Eigenface Example

- A 256x256 face image, 65536 dimensional vector, X , representing the face images with much lower dimensional vectors for analysis and recognition

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} \text{Face Image} \\ \text{Face Image} \\ \text{Face Image} \\ \text{Face Image} \\ \text{Face Image} \end{bmatrix}$$

We now only use FIVE Numbers!

Instead of 65536 Numbers!



Eigen Analysis - General

- The same principle can be applied to the analysis of many other data types

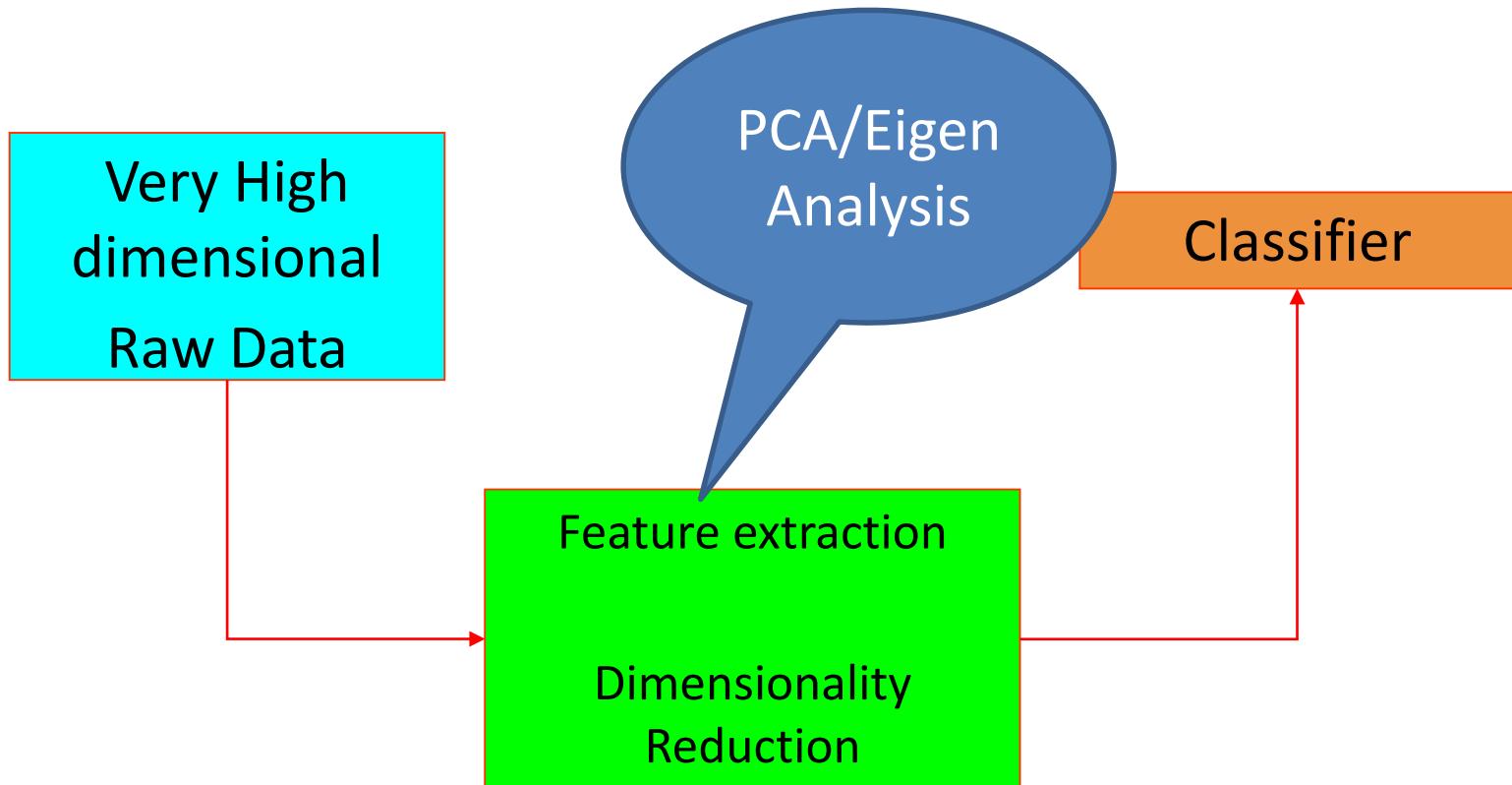
Reduce the dimensionality of biomarkers for analysis and classification

Raw data representation

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & \Lambda & & a_{1N} \\ \Lambda & \Lambda & \Lambda & a_{nN} \\ a_{n1} & \Lambda & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Processing Methods

- General framework



PCA

- Some remarks about PCA
 - PCA computes projection directions in which **variances of the data** can be ranked
 - The first few principal components capture the most “**energy**” or largest variance of the data
 - In classification/recognition tasks, which **principal component** is more **discriminative** is unknown

PCA

- Some remarks about PCA
 - Traditional popular practice is to use the **first few principal components** to represent the original data.
 - However, the subspace spanned by the first few principal components is not necessarily the most discriminative.
 - Therefore, throwing away the principal components with small variances may not be a good idea!

COMP3055

Machine Learning

Topic 11 – Perceptron, ADLINE, and Delta Rule

Zheng Lu
2024 Autumn

Recall: Machine Learning Concept

- Artificial Intelligence: AI is the science of making machines do things that require intelligence if done by men (Minsky 1986).
- Machine Learning is an area of AI concerned with development of techniques which allow machines to learn.
- Why Machine Learning?
 - To build machines exhibiting intelligent behaviour (i.e., able to reason, predict, and adapt) while helping humans work, study, and entertain themselves

Machine Learning Algorithms

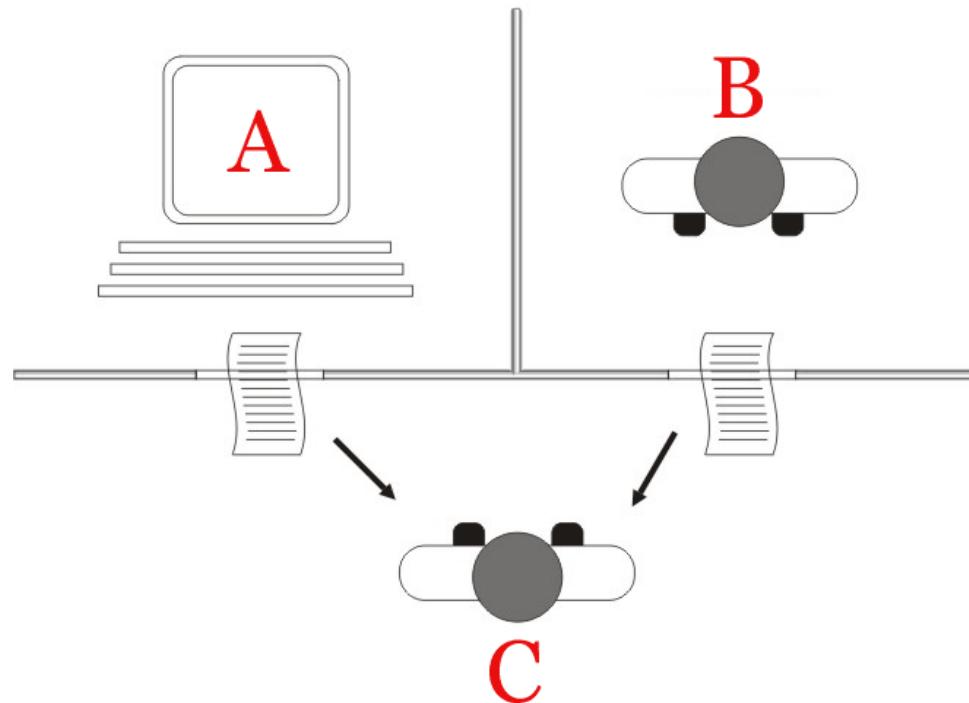
- The success of machine learning system depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.

Machine Learning Algorithms

- **Supervised learning** (generates a function that maps inputs to desired outputs)
 - Prediction
 - Classification (discrete labels), Regression (real values)
- **Unsupervised learning** (models a set of inputs, labelled examples are not available)
 - Probability distribution estimation
 - Finding association (in features)
 - Dimension reduction
 - Clustering
- **Reinforcement learning** (learning by education / experience)
 - Decision making (robot, chess machine)

Artificial General Intelligence

- **Artificial General Intelligence** (AGI) is the hypothetical intelligence of a machine that has the capacity to understand or learn any intellectual task that a human being can. AGI can also be referred to as **strong AI**
- Example: Turing Test



Purposeful Problem Solver

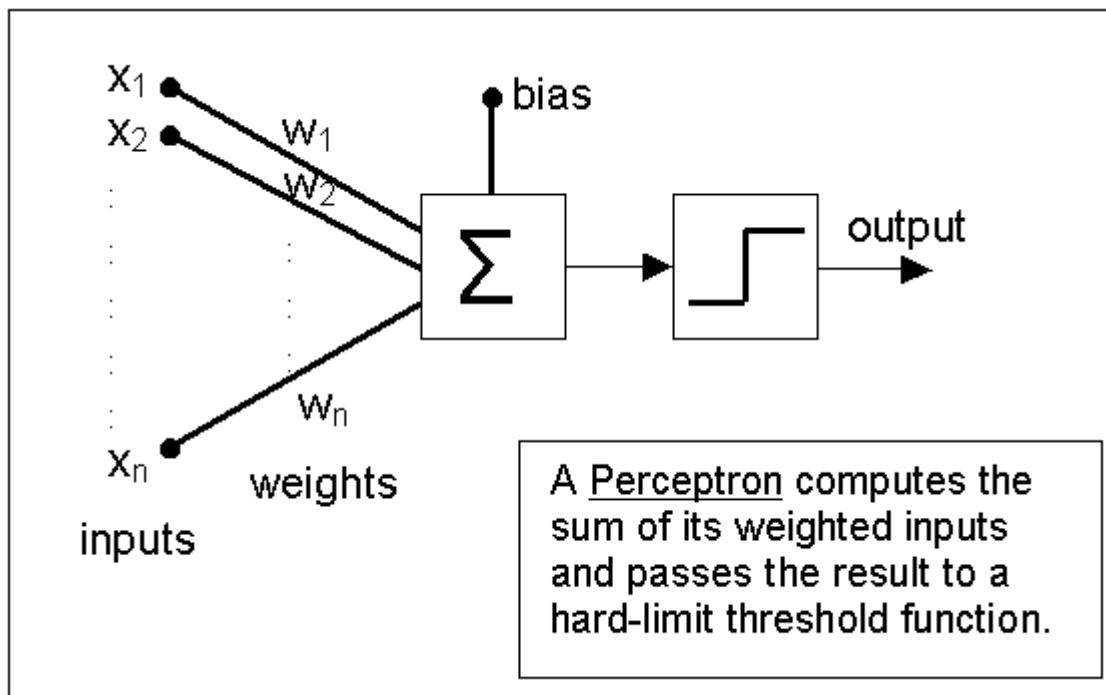
- In contrast to strong AI, **weak AI** is not intended to perform human cognitive abilities, rather, weak AI is limited to the use of software to study or accomplish specific problem solving or reasoning tasks.
- Examples: face recognition, speech recognition, games, etc.
- Application domains: security, medicine, education, finances, genetics, etc.

Feature Engineering

- Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithms.
- Process:
 - Brainstorming or testing features;
 - Deciding what features to create;
 - Creating features;
 - Checking how the features work with your model;
 - Improving your features if needed;
 - Go back to brainstorming/creating more features until the work is done.

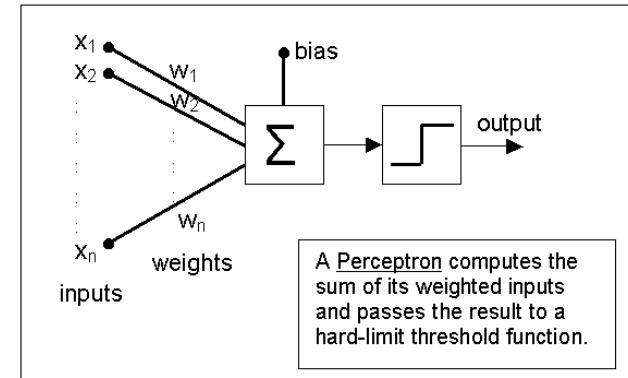
Perceptron - Basic

Perceptron is a type of Artificial Neural Network (ANN)



Perceptron - Operation

Perceptron takes a vector of real-valued inputs, calculates a *linear combination* of these inputs. Then it outputs **1** if the result is greater than some threshold and **-1** otherwise.



$$R = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = w_0 + \sum_{i=1}^n w_i x_i$$

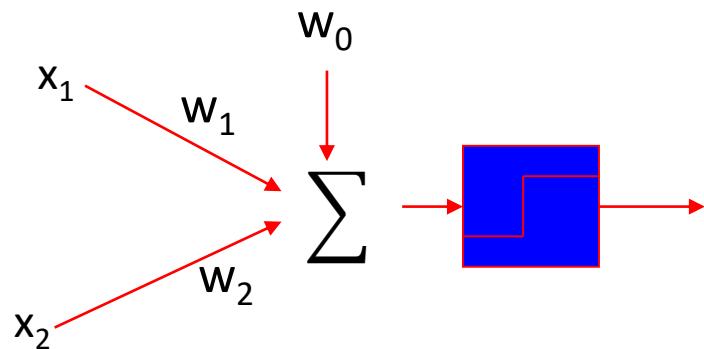
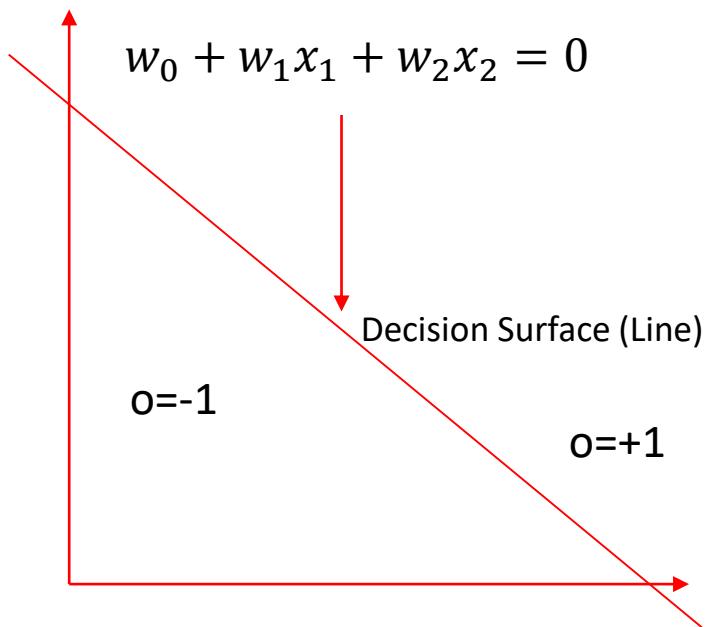
$$\text{output } o = \text{sign}(R) = \begin{cases} +1, & \text{if } R > 0 \\ -1, & \text{otherwise} \end{cases}$$

Perceptron – Decision Surface

- Perceptron can be regarded as representing a hyperplane decision surface in the n-dimensional **feature space** of instances.
- The perceptron outputs a 1 for instances lying on one side of the hyperplane and a -1 for instances lying on the other side.
- This hyperplane is called the **Decision Surface**.

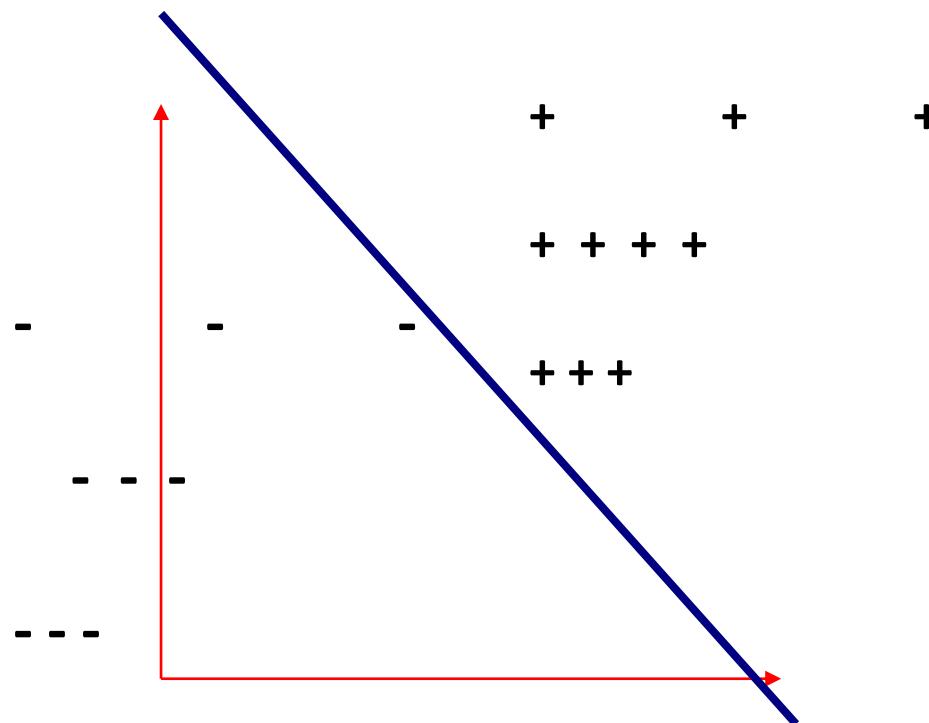
Perceptron – Decision Surface

In 2-dimensional space:



Perceptron – Representation Power

- The Decision Surface is linear.
- Perceptron can only solve **Linearly Separable Problems.**

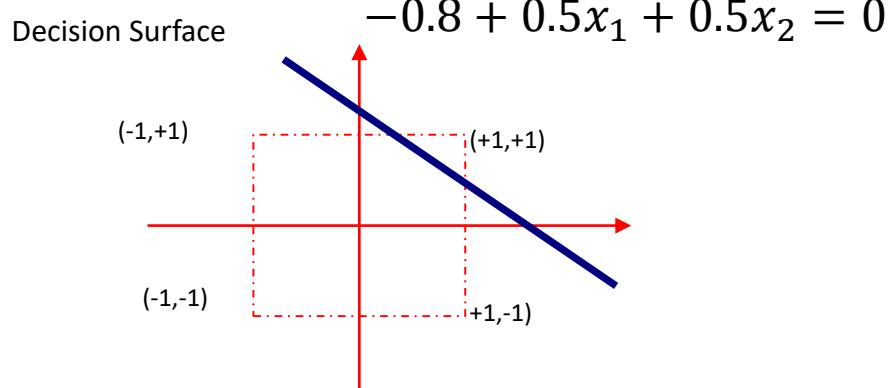
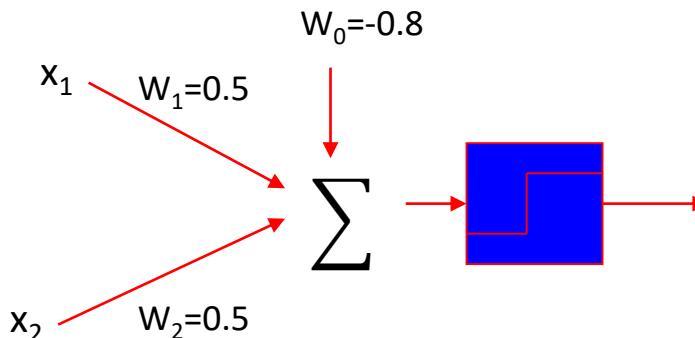


Perceptron – Representation Power

Can represent many Boolean functions, assuming Boolean values of 1 (true) and -1 (false).

AND

| x ₁ | x ₂ | D |
|----------------|----------------|----|
| -1 | -1 | -1 |
| -1 | +1 | -1 |
| +1 | -1 | -1 |
| +1 | +1 | +1 |

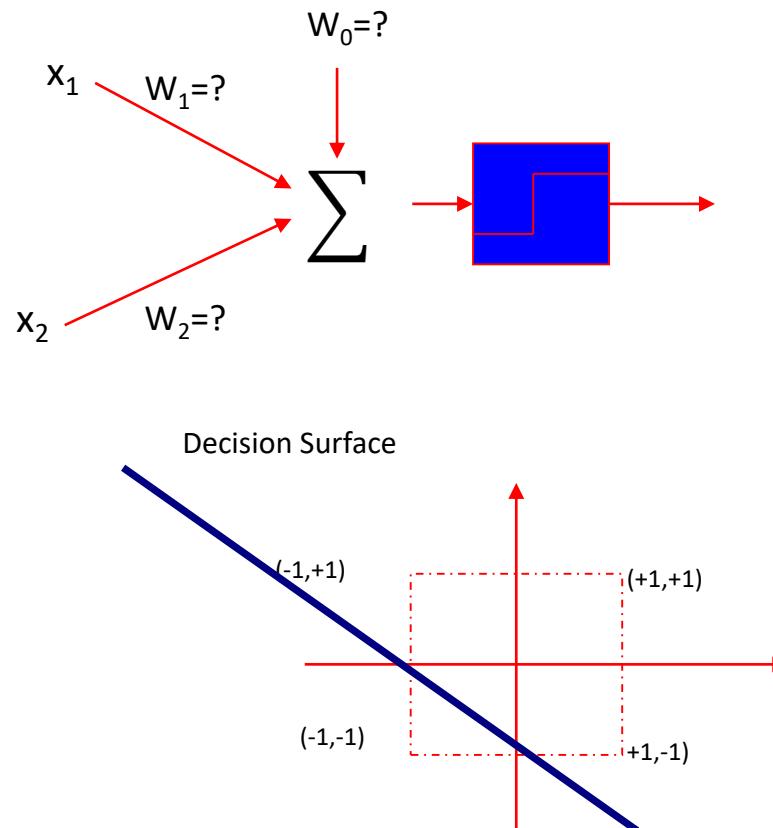


Perceptron – Representation Power

Can represent many Boolean functions, assuming Boolean values of 1 (true) and -1 (false).

OR

| x1 | x2 | D |
|----|----|----|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | +1 |

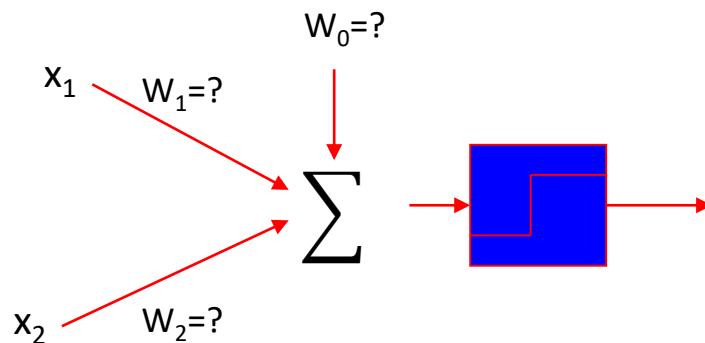


Perceptron – Representation Power

Some problems are **linearly non-separable**.

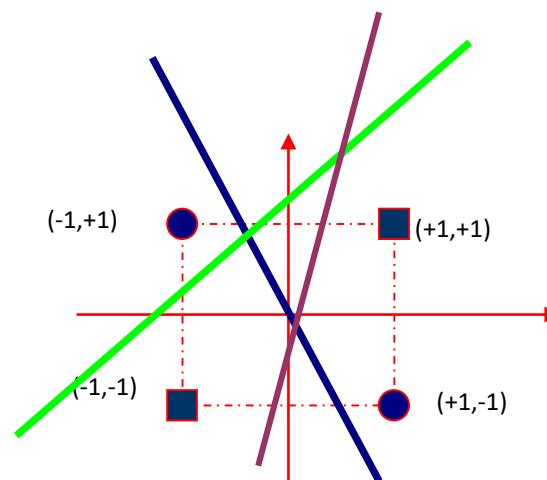
XOR

| x1 | x2 | D |
|----|----|----|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | -1 |



Decision Surface:

It doesn't matter where you place the line (decision surface). It is impossible to separate the space such that on one side we have $o = 1$ and on the other we have $o = -1$.



Perceptron Cannot Solve such Problem!

Perceptron – Training Algorithm

Training sample pairs (X, d) , where X is the input vector, d is the input vector's classification (+1 or -1) is iteratively presented to the network for training, *one at a time*, until the process converges.

Perceptron – Training Algorithm

The Procedure is as follows:

1. Set the weights to small random values, e.g., in the range (-1, 1)
2. Present X, and calculate

$$R = \sum_{i=0}^n w_i x_i , \quad o = sign(R) = \begin{cases} +1, & \text{if } R > 0 \\ -1, & \text{otherwise} \end{cases}$$

let $x_0=1$

1. Update the weights

$$w_i \leftarrow w_i + \eta(d - o)x_i, i = 0, 1, \dots, n$$

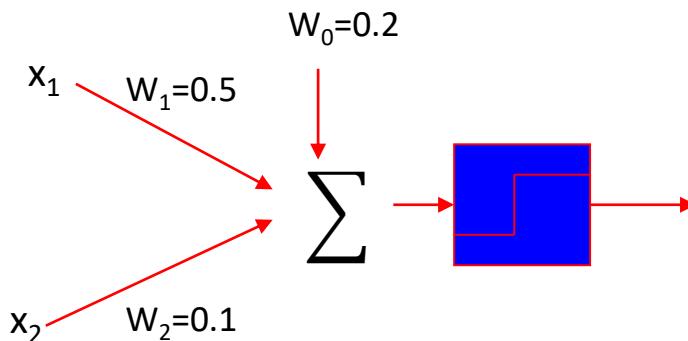
$0 < \eta < 1$, is the training rate

2. Repeat by going to step 2

Perceptron – Training Algorithm

Example

| x1 | x2 | D |
|----|----|----|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | +1 |



$$w_i \leftarrow w_i + \eta(d - o)x_i, i = 1, 2, \dots, n$$

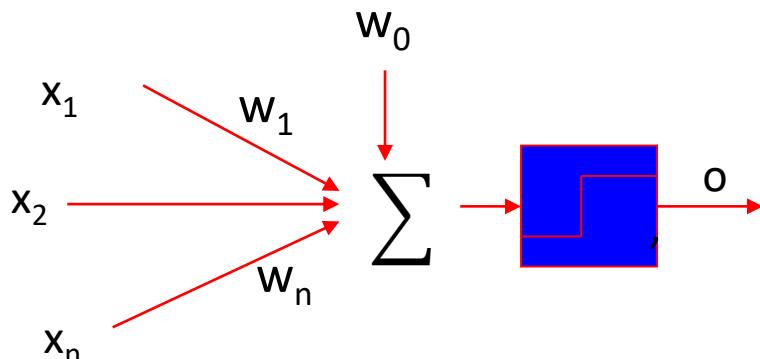
Perceptron – Training Algorithm

Convergence Theorem

The perceptron training rule will converge (finding a weight vector correctly classifies all training samples) within a finite number of iterations, **provided the training examples are linearly separable** and provided a sufficiently small η is used.

Adaptive Linear Element (ADLINE)

Perceptron

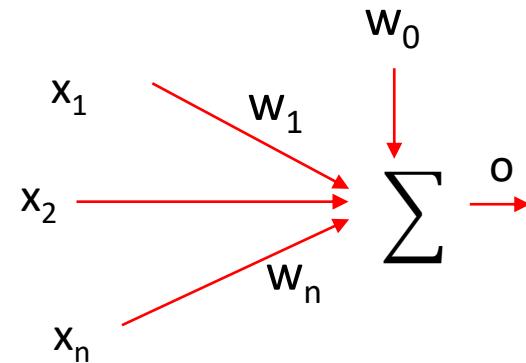


$$R = w_0 + \sum_{i=1}^n w_i x_i$$

$$o = sign(R) = \begin{cases} +1, & \text{if } R > 0 \\ -1, & \text{otherwise} \end{cases}$$

VS

ADLINE



$$o = w_0 + \sum_{i=1}^n w_i x_i$$

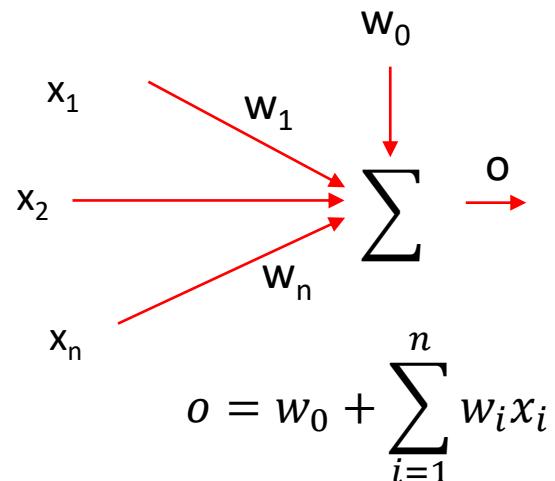
ADLINE VS Perceptron

- When the problem is not linearly separable, perceptron will fail to converge.
- ADLINE can overcome this difficulty by finding a best fit approximation to the target.

The ADLINE Error Function

- We have training pairs $(X(k), d(k), k = 1, 2, \dots, K)$, where K is the number of training samples, the **training error** specifies the difference between the output of the ADLINE and the desired target.
- The error is defined as

$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$



$o(k) = W^T X(k)$ is the output of presenting the training input $X(k)$

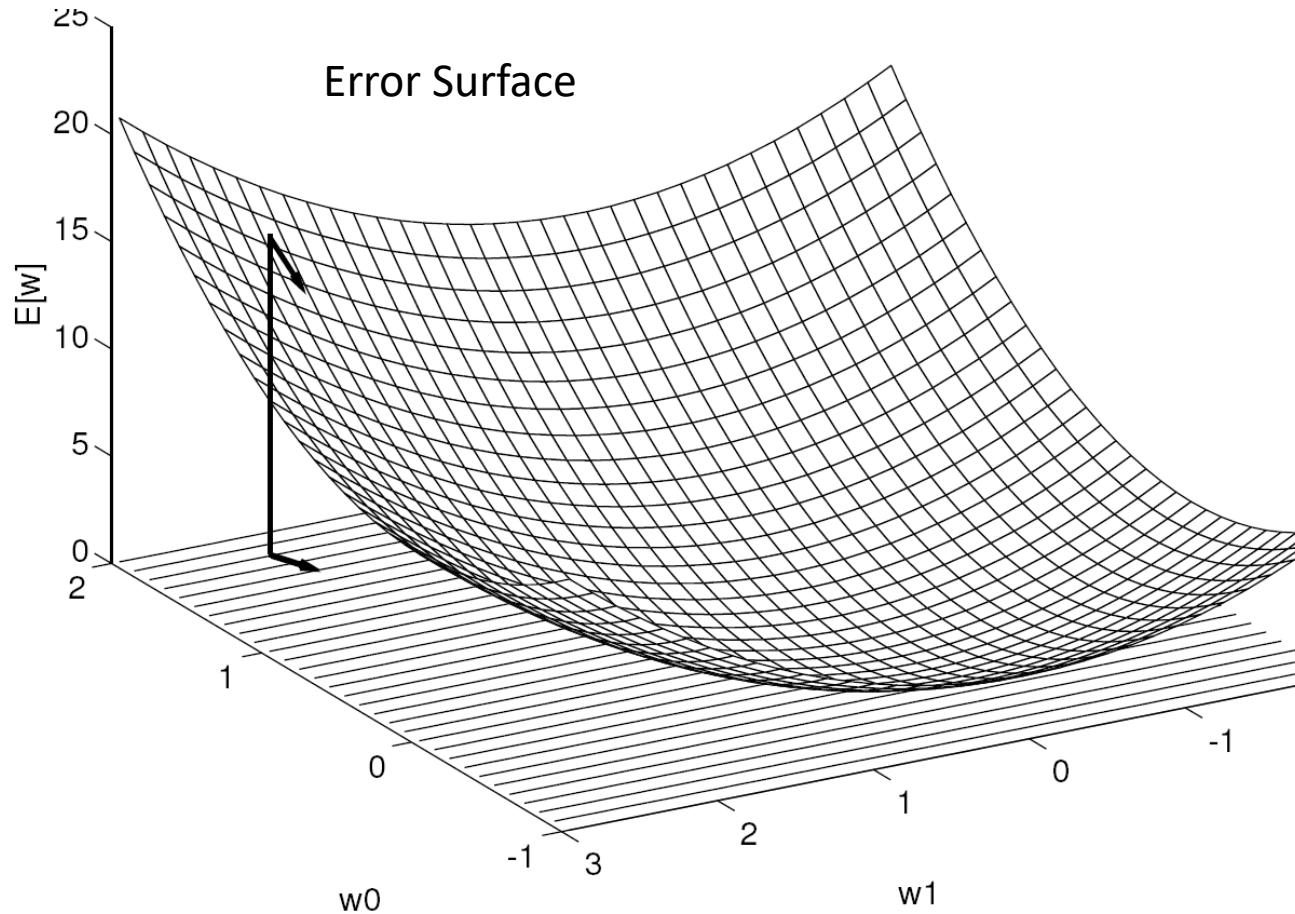
The ADLINE Error Function

- The error is defined as

$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$

- The smaller $E(W)$ is, the closer is the approximation.
- We need to find W , based on the given training set, that minimizes the error $E(W)$.

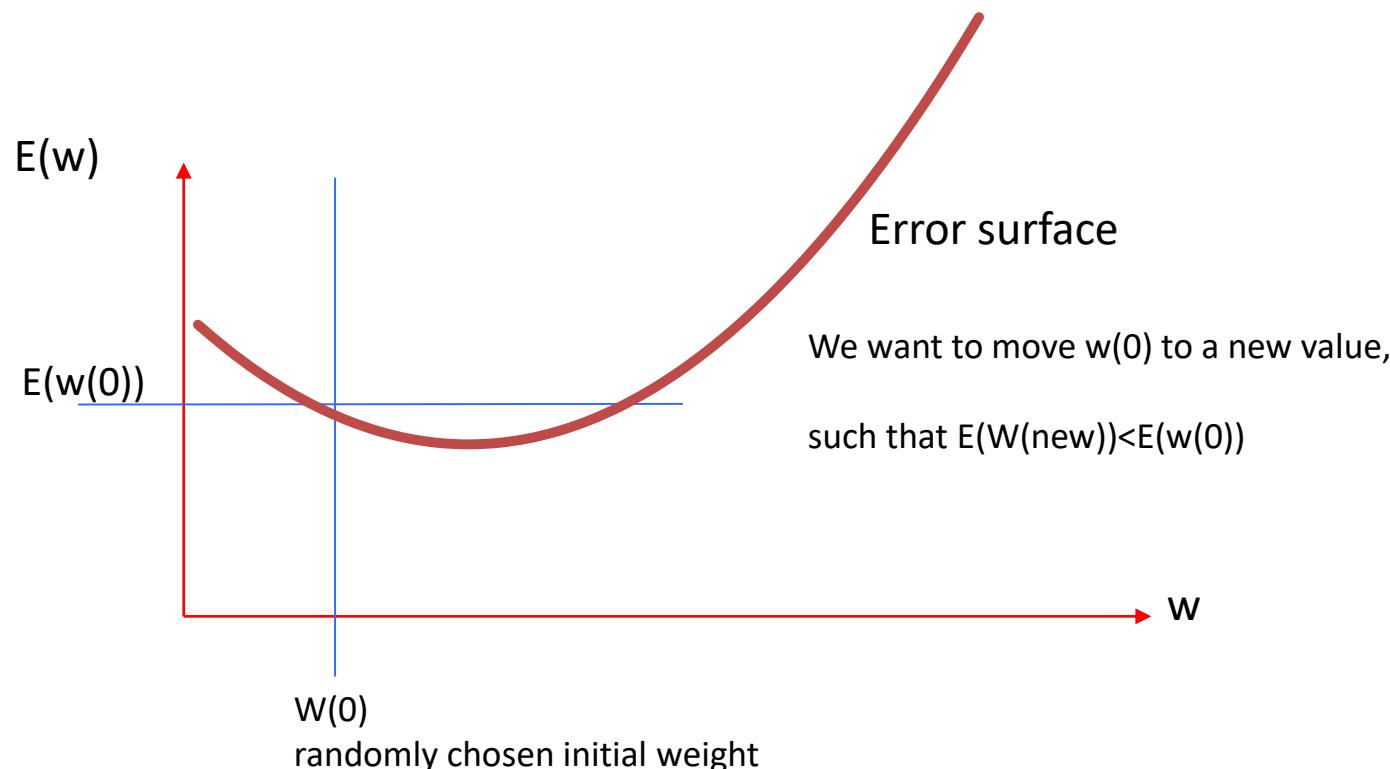
The ADLINE Error Function



The Gradient Descent Rule

An intuition

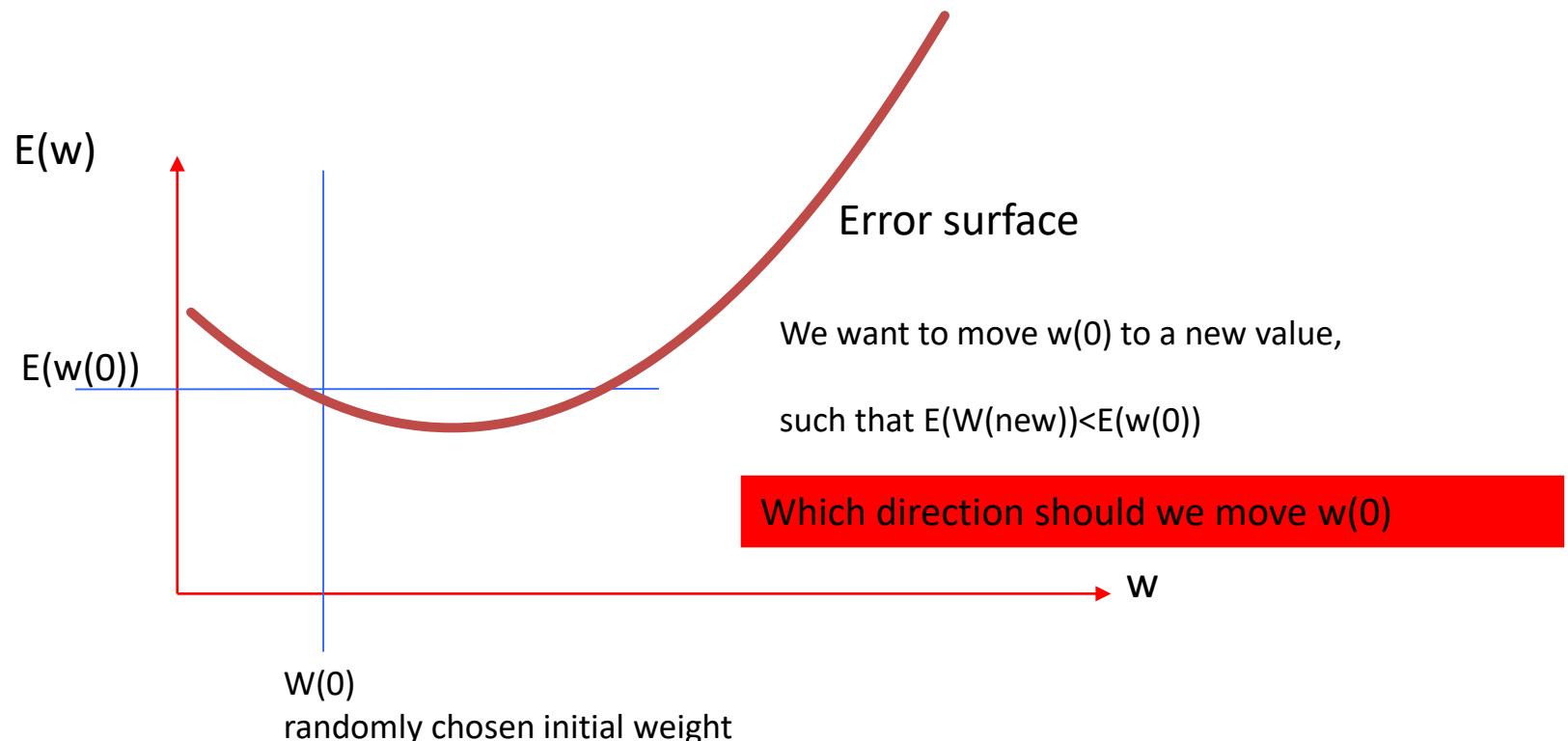
Before we formally derive the gradient decent rule, here is an intuition of what we should be doing.



The Gradient Descent Rule

An intuition

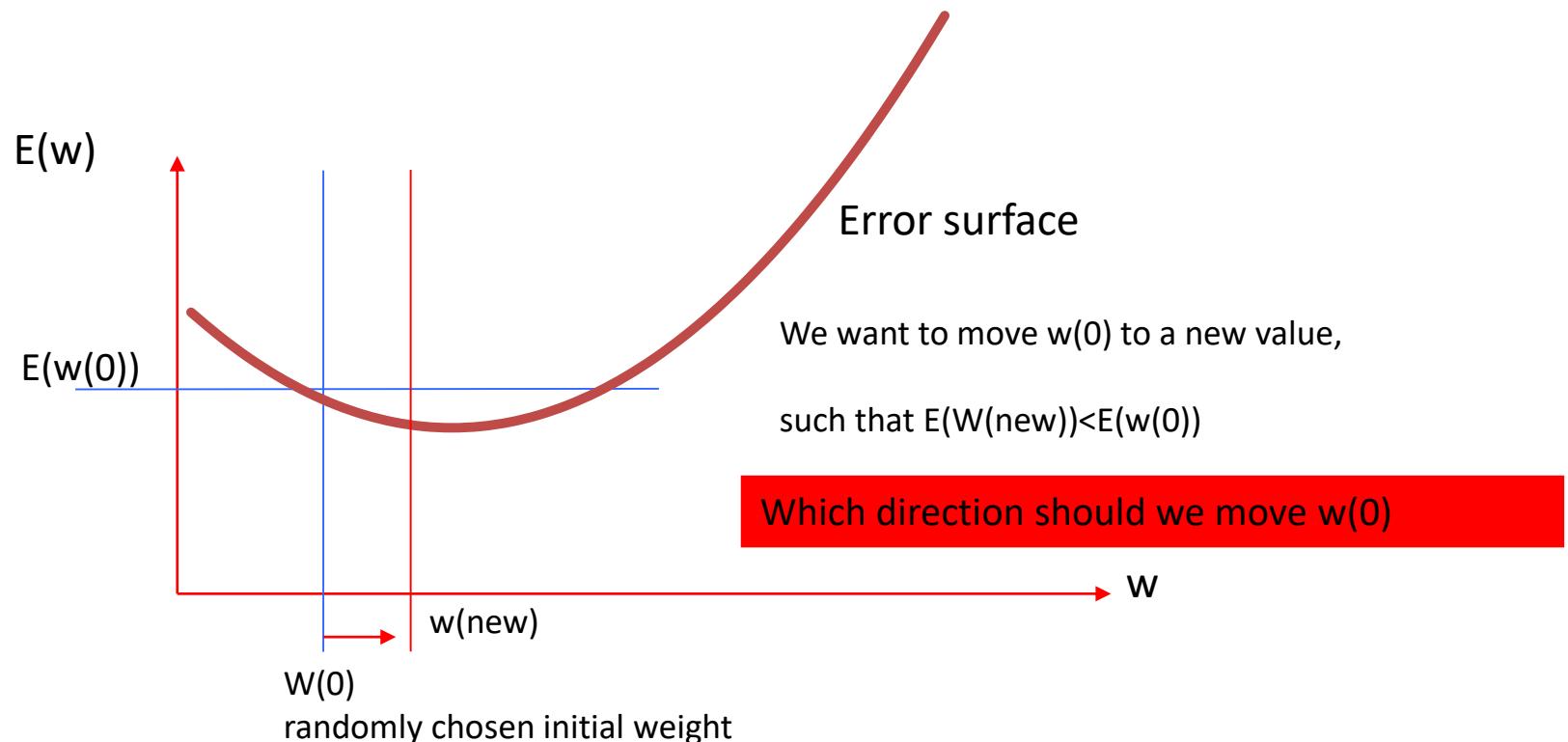
Before we formally derive the gradient decent rule, here is an intuition of what we should be doing.



The Gradient Descent Rule

An intuition

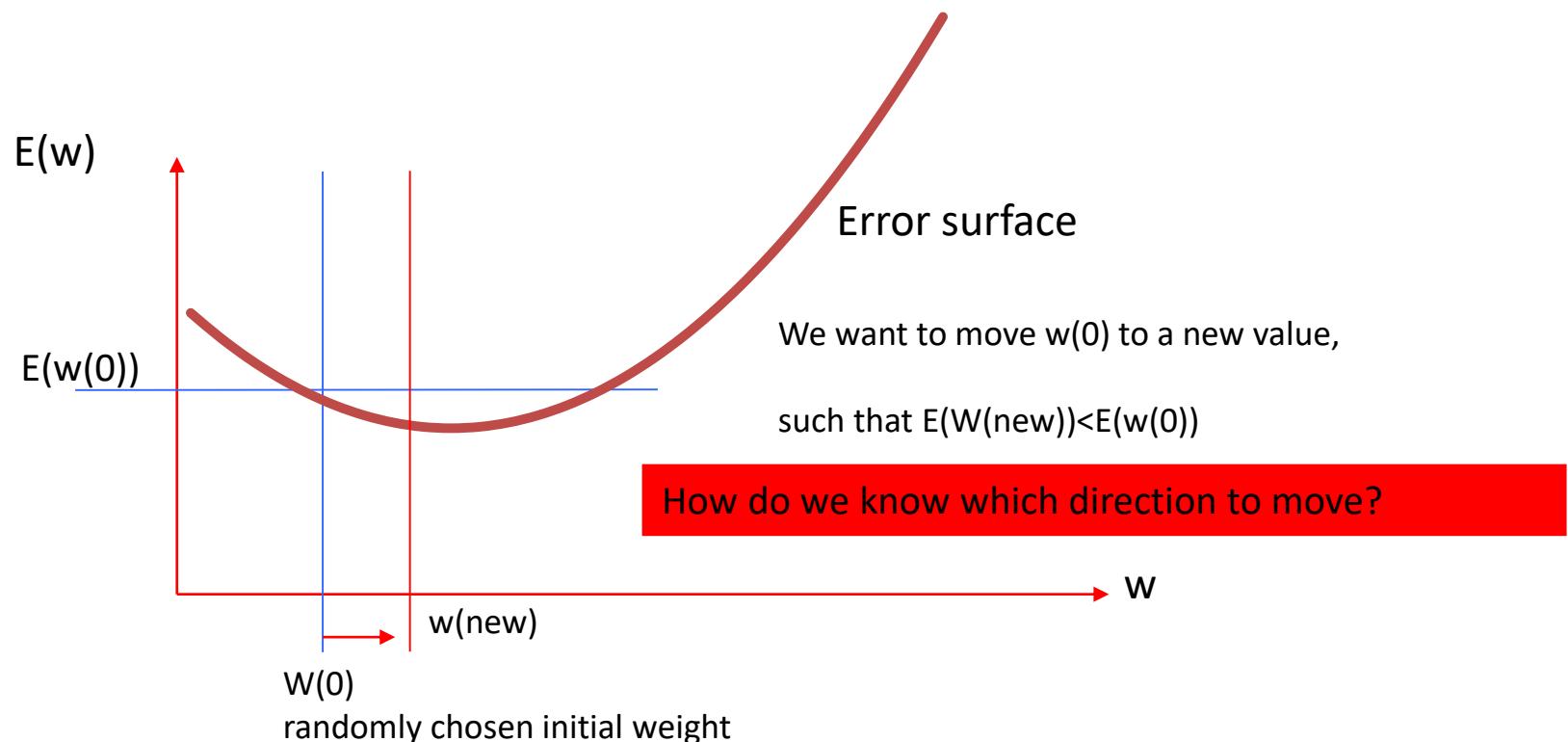
Before we formally derive the gradient decent rule, here is an intuition of what we should be doing.



The Gradient Descent Rule

An intuition

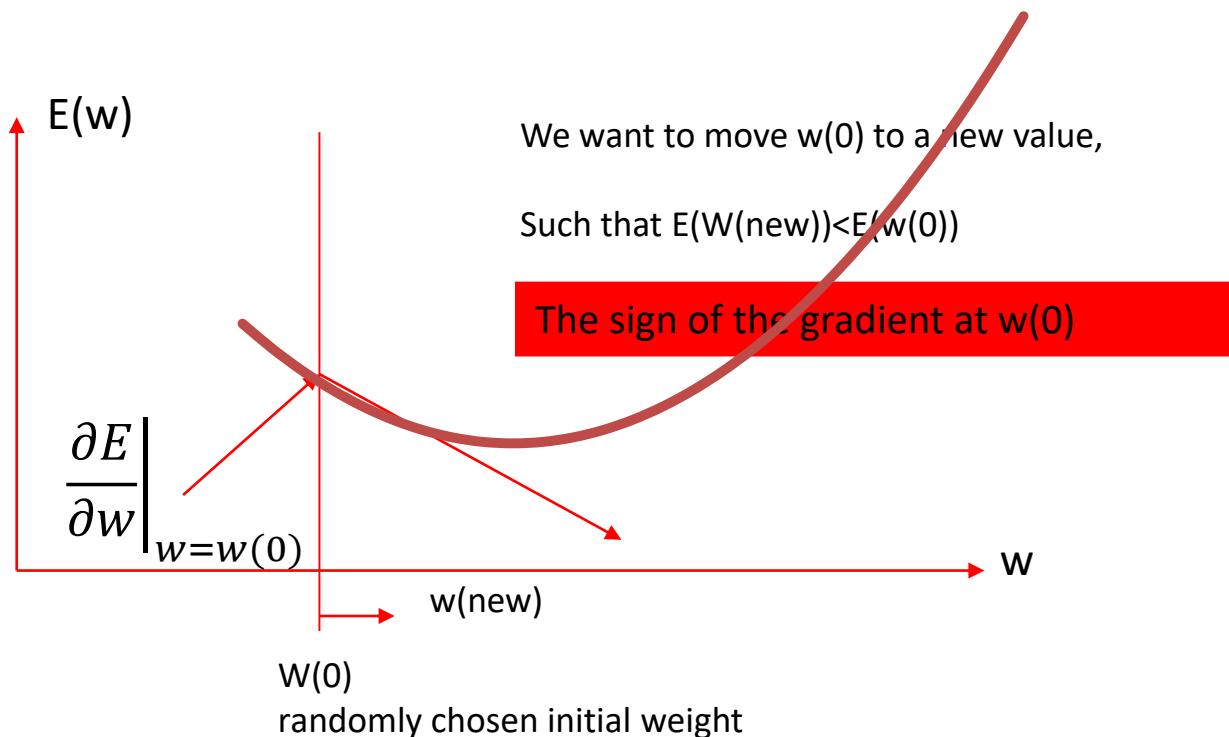
Before we formally derive the gradient decent rule, here is an intuition of what we should be doing.



The Gradient Descent Rule

An intuition

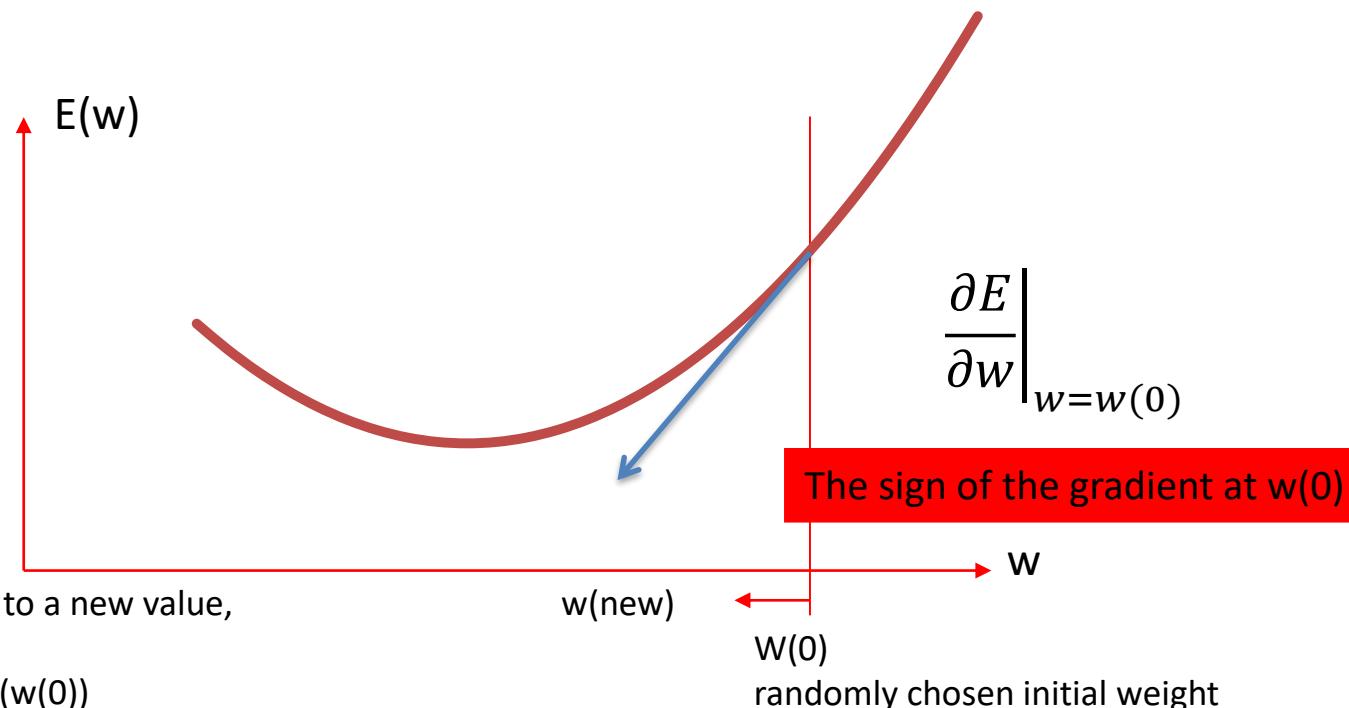
Before we formally derive the gradient decent rule, here is an intuition of what we should be doing.



The Gradient Descent Rule

An intuition

Before we formally derive the gradient decent rule, here is an intuition of what we should be doing.

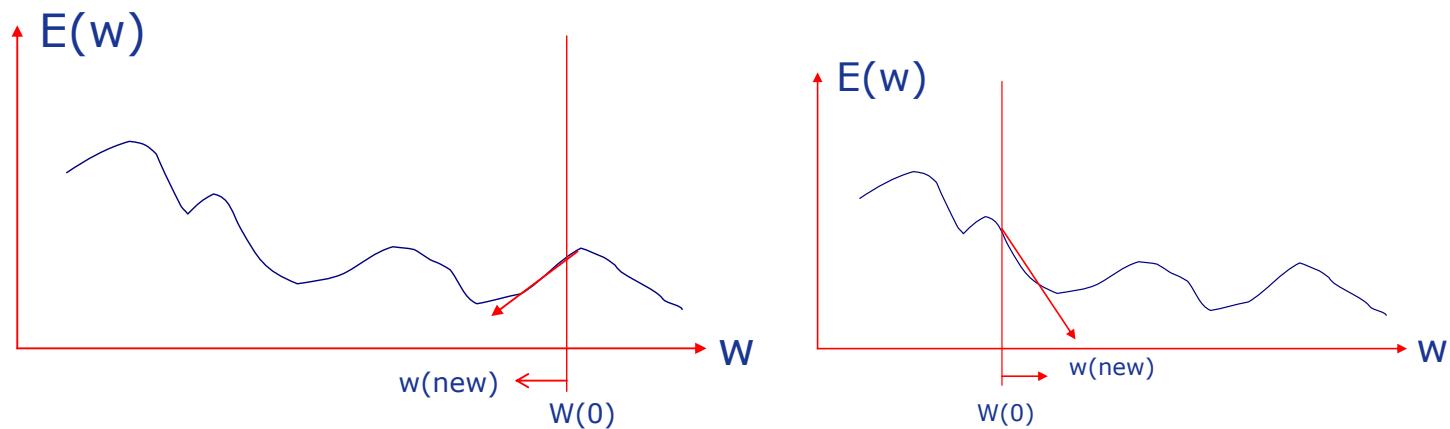


The Gradient Descent Rule

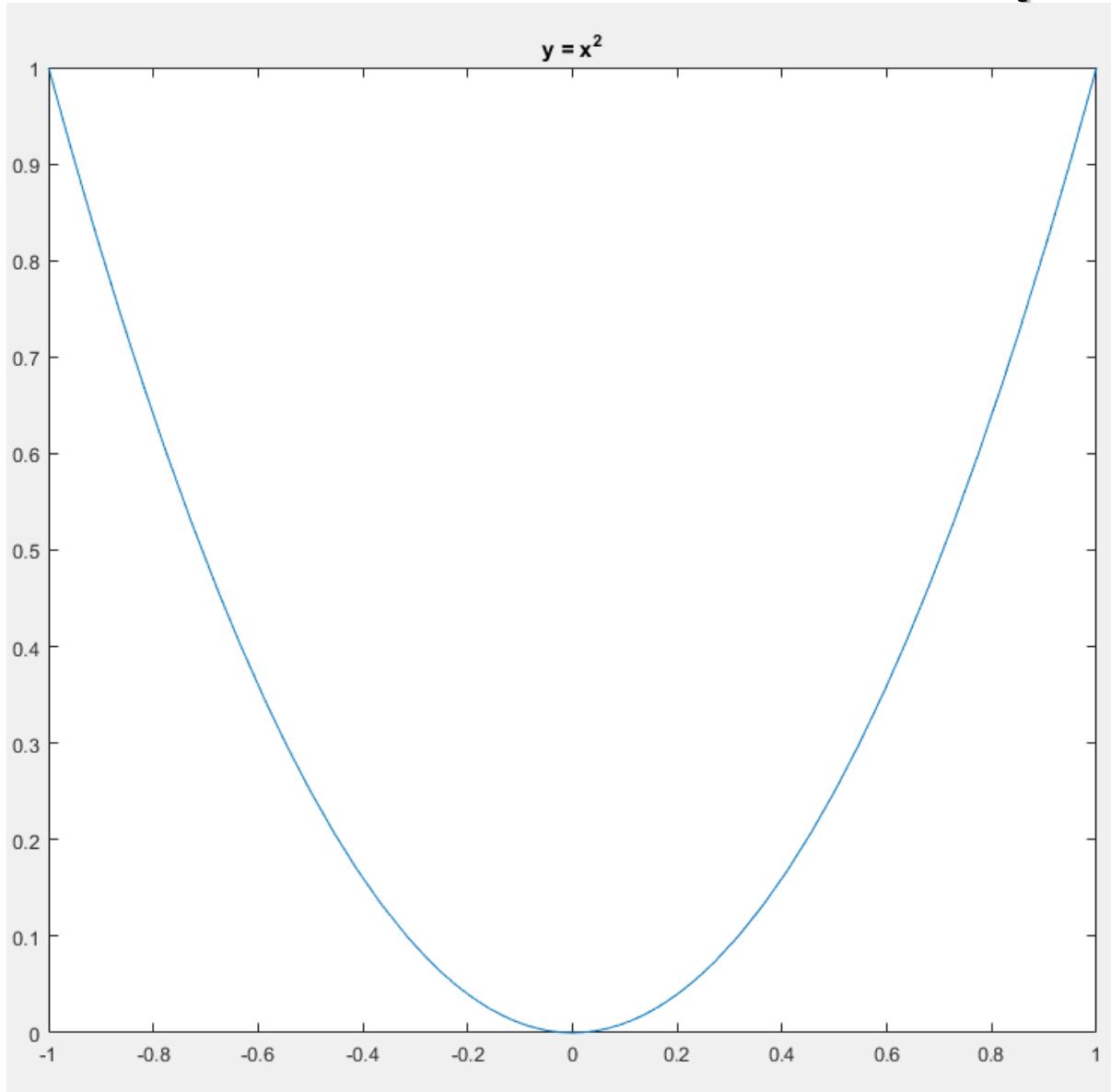
An intuition

The intuition leads to the following rule:

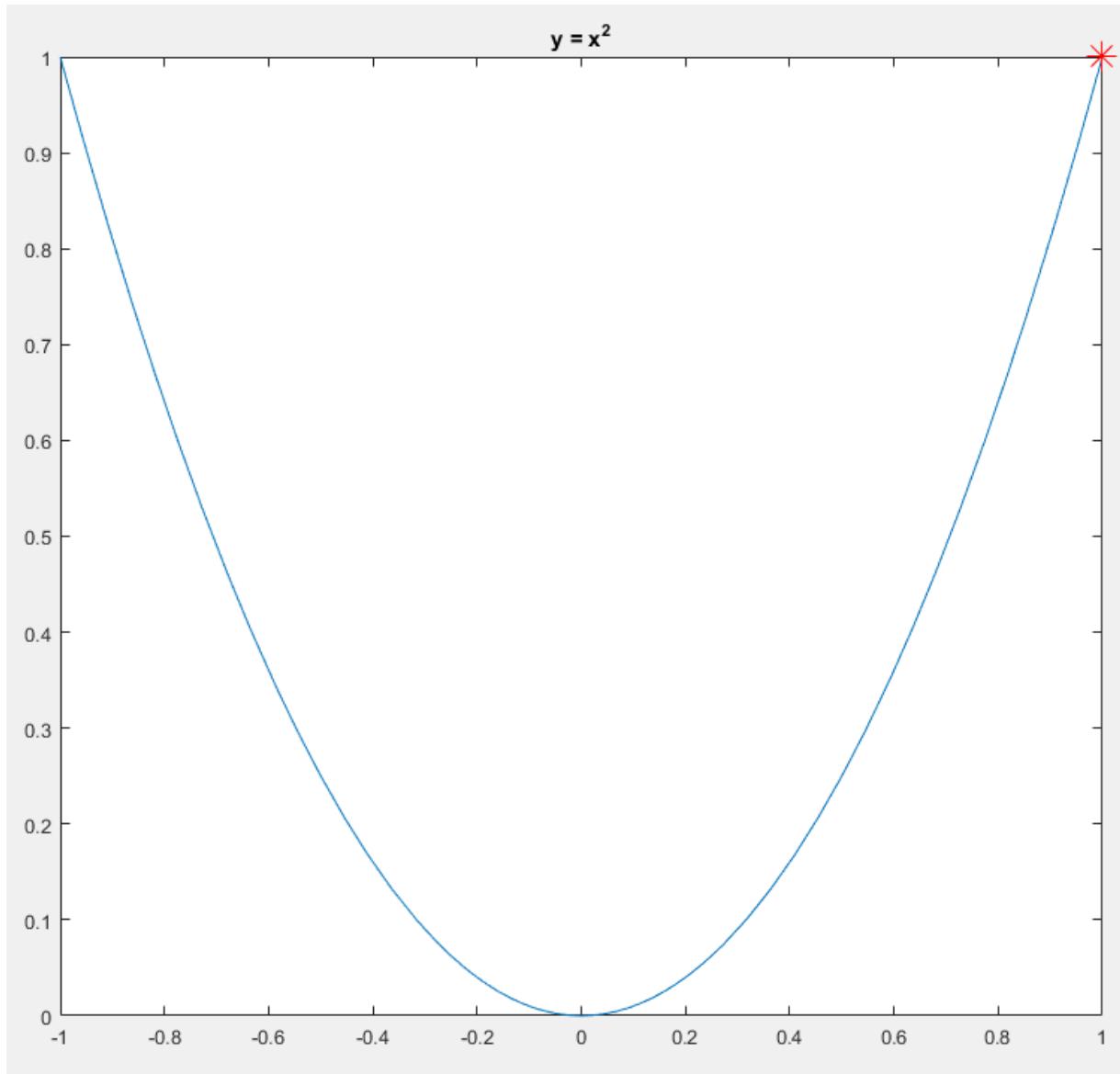
$$w(\text{new}) \leftarrow w(\text{old}) - \Delta \text{sign} \left(\frac{\partial E}{\partial w} \Big|_{w=w(\text{old})} \right)$$



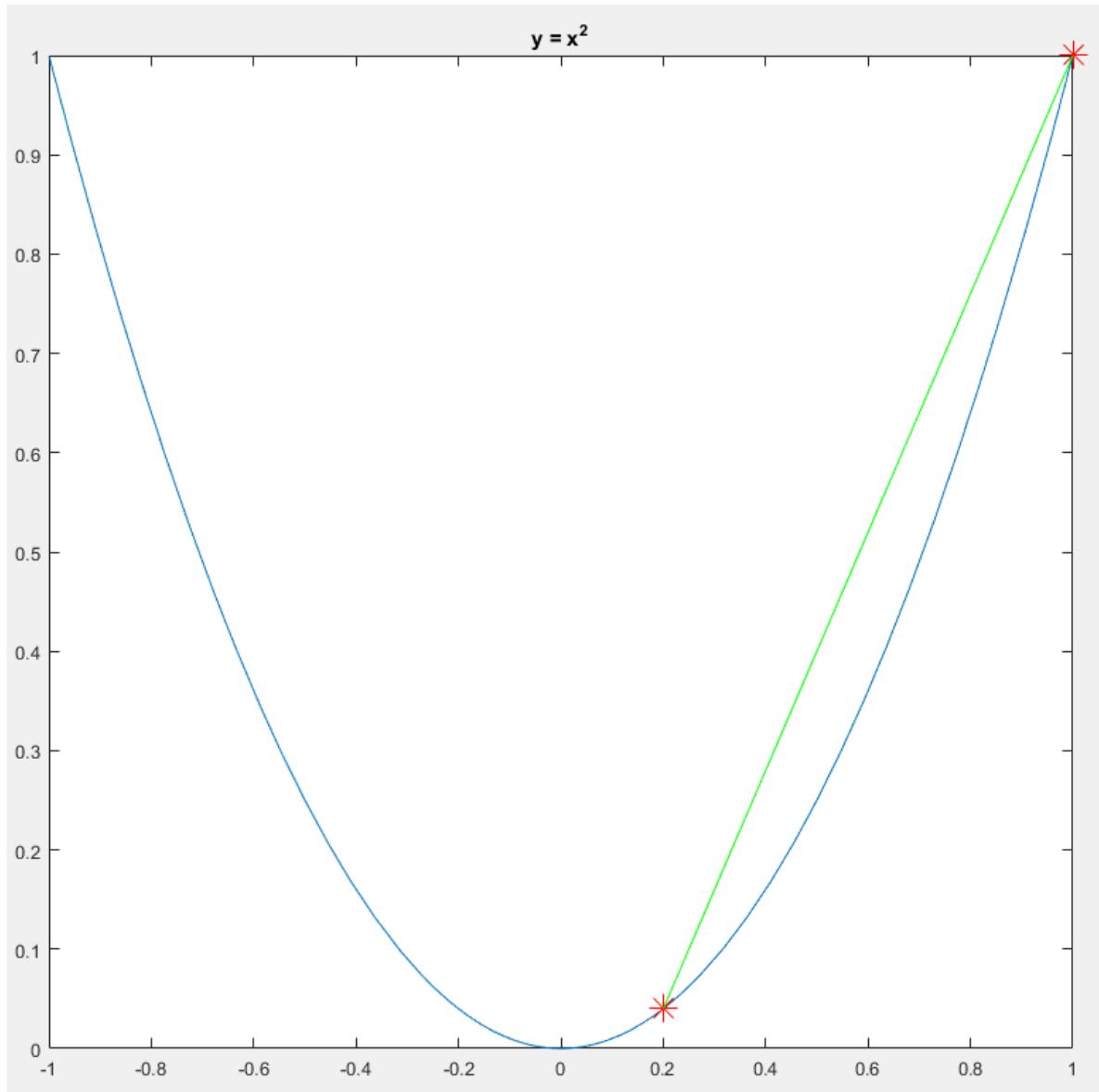
Gradient Descent Example



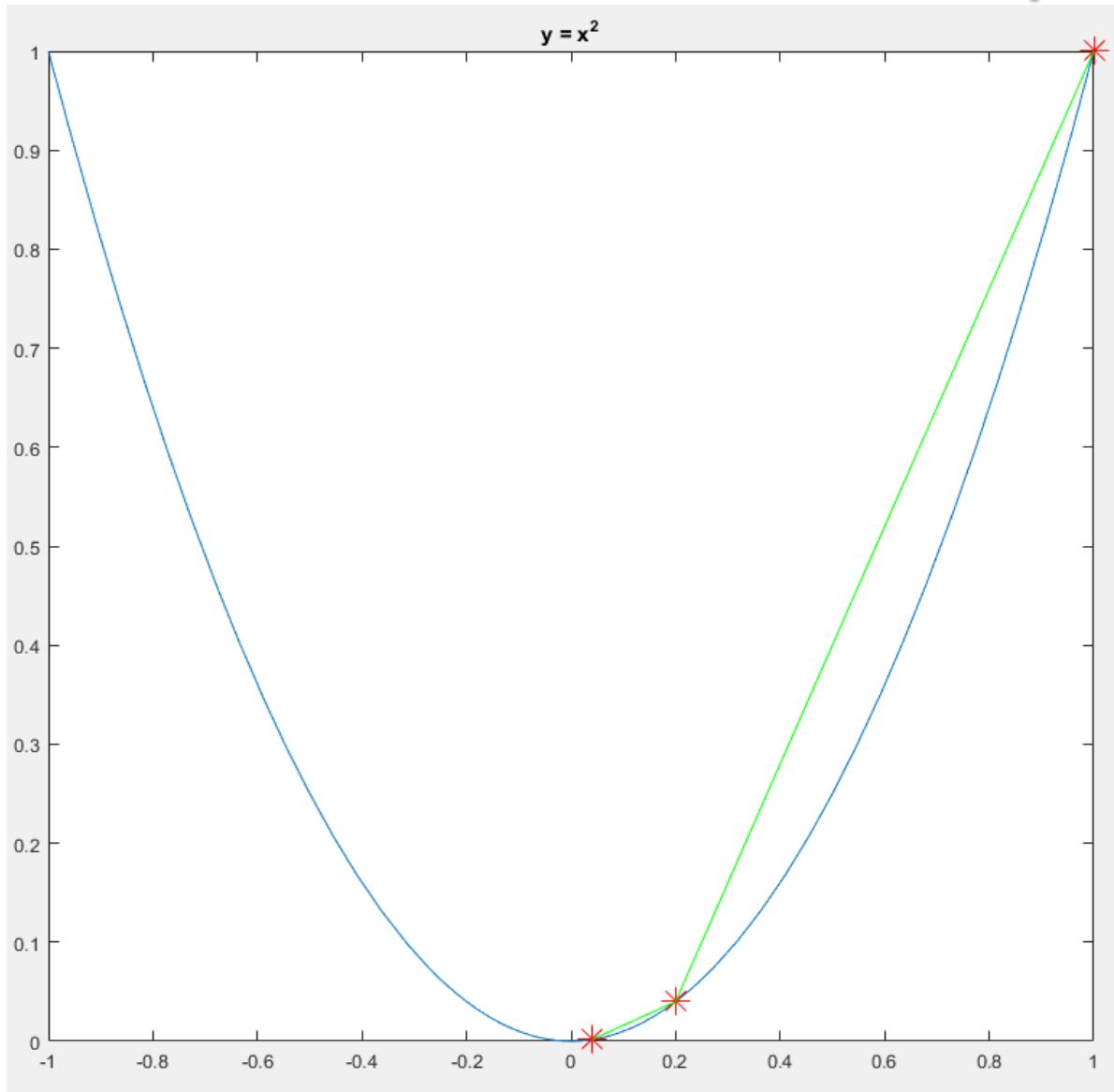
Gradient Descent Example



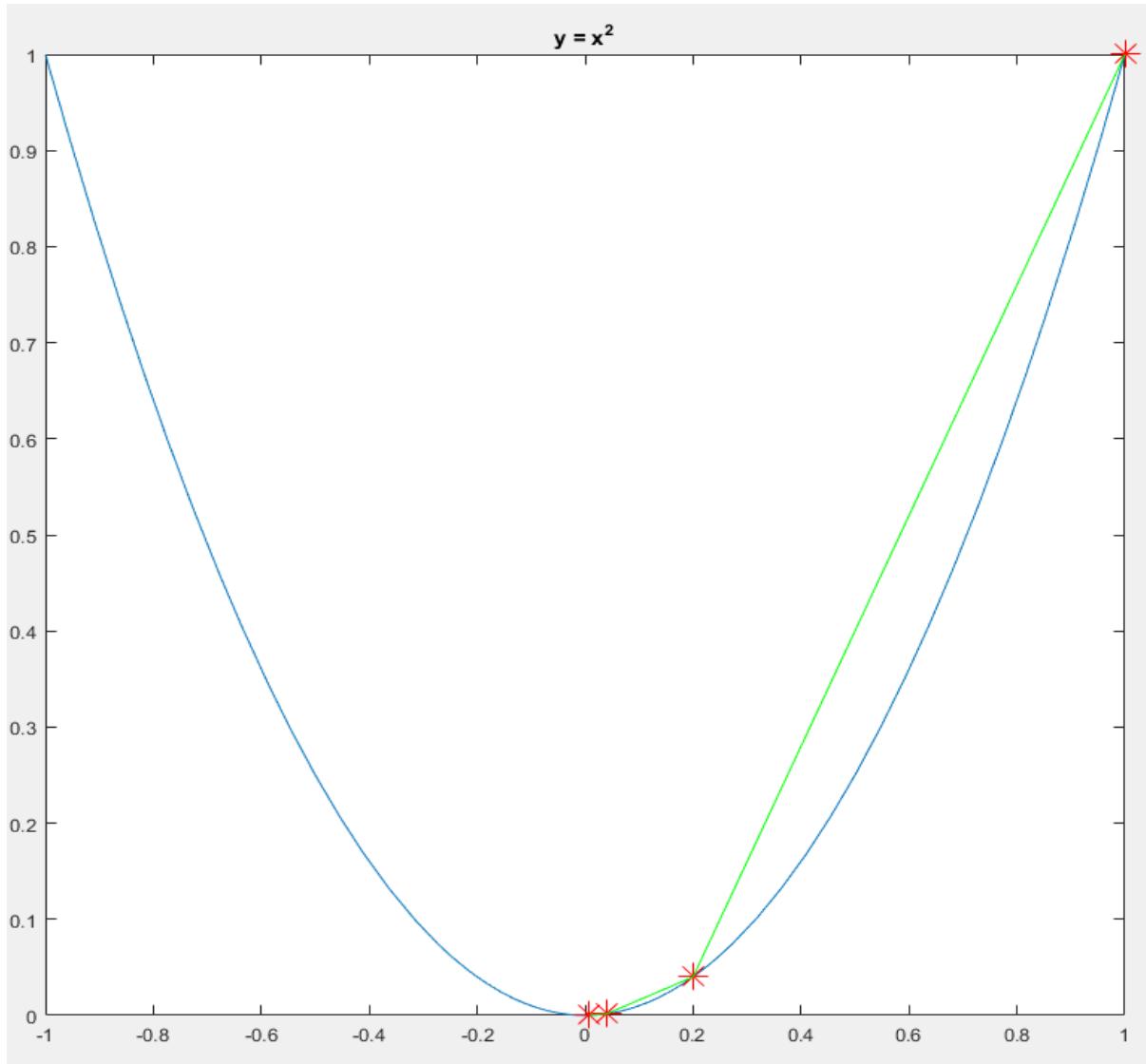
Gradient Descent Example



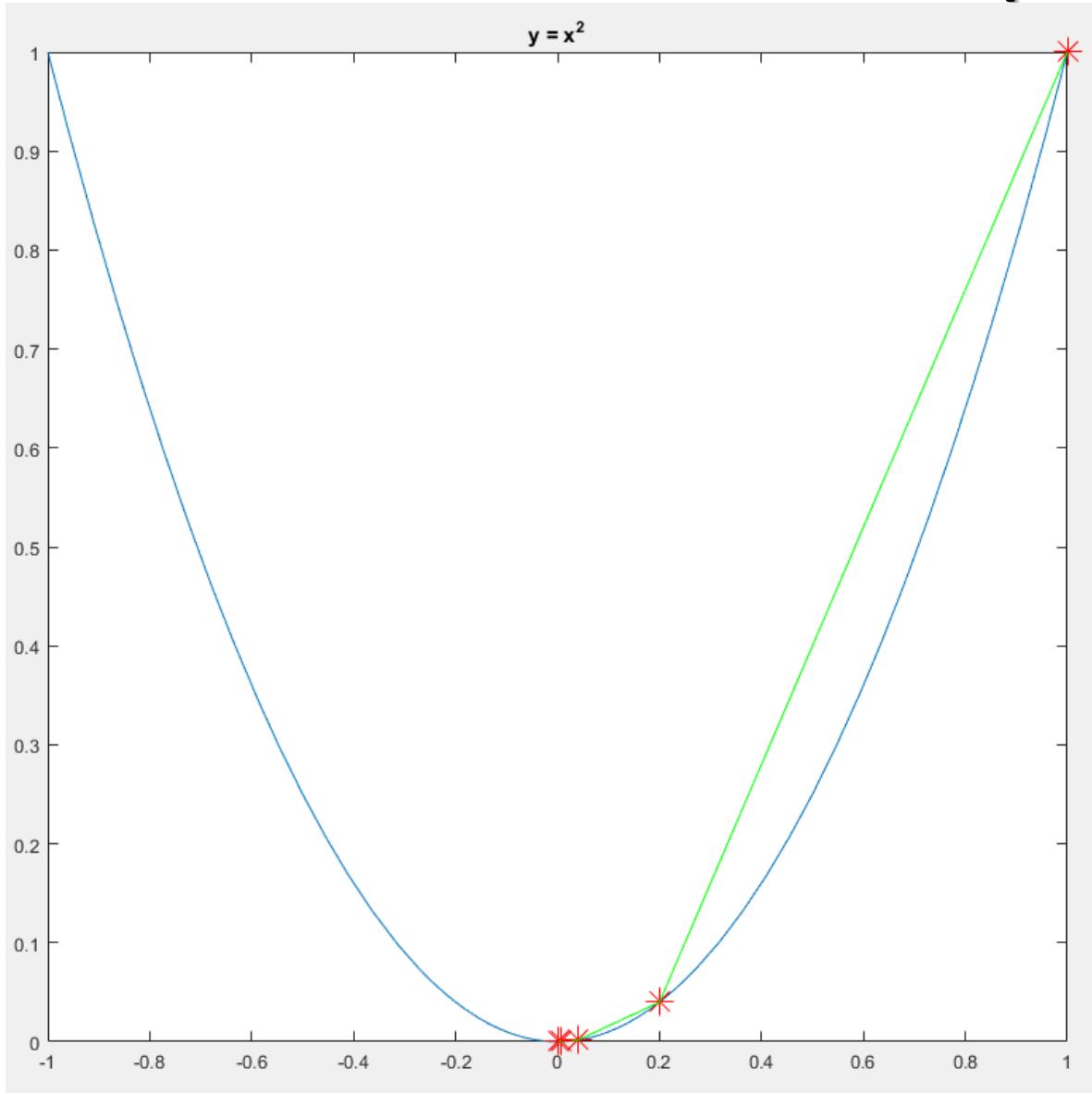
Gradient Descent Example



Gradient Descent Example



Gradient Descent Example



The Gradient Descent Rule

Formal Derivation of Gradient Descent

$$\nabla E(W) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_n} \right]$$

- The gradient of E is a vector, whose components are the partial derivatives of E with respect to each of the w_i 's
- The gradient specifies the direction that produces the steepest increase in E . (i.e., speed of change (slope) of the function in one direction.)
- Negative of the vector gives the direction of steepest decrease in E .

The Gradient Descent Rule

The gradient training rule is

$$W \leftarrow W - \eta \nabla E(W)$$

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

η is the training rate

The Gradient Descent Rule

Gradient of ADLINE Error Functions

$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial w_i} \left(\frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2 \right) \\ &= \frac{1}{2} \sum_{k=1}^K \left(\frac{\partial E}{\partial w_i} (d(k) - o(k))^2 \right) \\ &= \frac{1}{2} \sum_{k=1}^K \left(2(d(k) - o(k)) \frac{\partial E}{\partial w_i} (d(k) - o(k)) \right) \\ &= \sum_{k=1}^K \left((d(k) - o(k)) \frac{\partial E}{\partial w_i} (d(k) - w_0 - \sum_{i=1}^n w_i x_i(k)) \right) \\ &= \sum_{k=1}^K ((d(k) - o(k))(-x_i(k))) \\ &= - \sum_{k=1}^K (d(k) - o(k))x_i(k)\end{aligned}$$

The Gradient Descent Rule

ADLINE weight updating using **gradient descent rule**

$$w_i \leftarrow w_i + \eta \sum_{k=1}^K (d(k) - o(k))x_i(k)$$

The Gradient Descent Rule

Gradient descent training procedure

- Initialise w_i to small values, e.g., in the range of $(-1, 1)$, choose a learning rate, e.g., $\eta = 0.2$
- Until the termination condition is met, Do
 - For all training sample pair $(X(k), d(k))$, input the instance $X(k)$ and compute

$$\delta_i = - \sum_{k=1}^K (d(k) - o(k))x_i(k)$$

- For each weight w_i , Do

$$w_i \leftarrow w_i - \eta \delta_i$$

Batch Mode:

gradients accumulated
over **ALL** samples first

Then update the weights

Stochastic (Incremental) Gradient Descent

Also called online mode, Least Mean Square (LMS), Widrow-Hoff, and Delta Rule

- Initialise w_i to small values, e.g., in the range of (-1, 1), choose a learning rate, e.g., $\eta = 0.01$ (should be smaller than batch mode)
- Until the termination condition is met, Do
 - For EACH training sample pair $(X(k), d(k))$, compute

$$\delta_i = -(d(k) - o(k))x_i(k)$$

- For each weight w_i , Do

$$w_i \leftarrow w_i - \eta \delta_i$$

Online Mode:

Calculate gradient for
**EACH (or a SUBSET
of)samples**

Then update the weights

Training Iterations, Epochs

- Training is an iterative process; training samples will have to be used repeatedly for training.
- Assuming we have K training samples $[(X(k), d(k)), k=1, 2, \dots, K]$; then an epoch is the presentation of all K sample for training once.
 - First epoch: Present training samples: $(X(1), d(1)), (X(2), d(2)), \dots (X(K), d(K))$
 - Second epoch: Present training samples: $(X(K), d(K)), (X(K-1), d(K-1)), \dots (X(1), d(1))$
 - Note the order of the training sample presentation between epochs can (and should normally) be different.
- Normally, training will take many epochs to complete.

Termination of Training

To terminate training, there are normally two ways

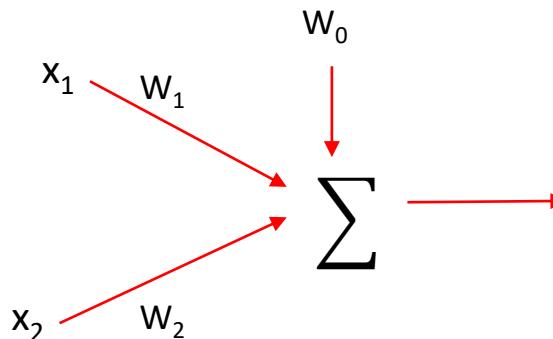
- When a pre-set number of training epochs is reached.
- When the error is smaller than a pre-set value.

$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$

Gradient Descent Training

Example

| x1 | x2 | D |
|----|----|----|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | +1 |



Initialization

$$w_0(0)=0.1; w_1(0)=0.2; w_2(0)=0.3;$$

$$\eta=0.5$$

Further Reading

Chapter 4, T. M. Mitchell, Machine Learning,
McGraw-Hill International Edition, 1997



University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

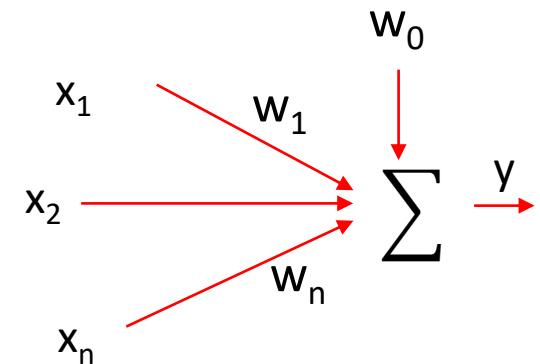
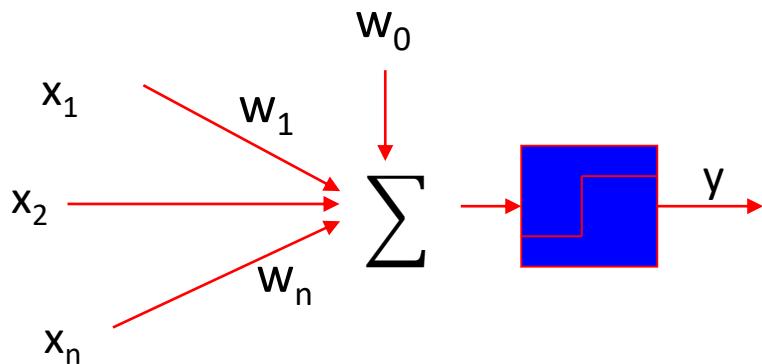
Machine Learning

Topic 12 – Multilayer Perceptrons

Zheng Lu
2023 Autumn

Limitations of Single Layer Perceptron

Only express linear decision surfaces



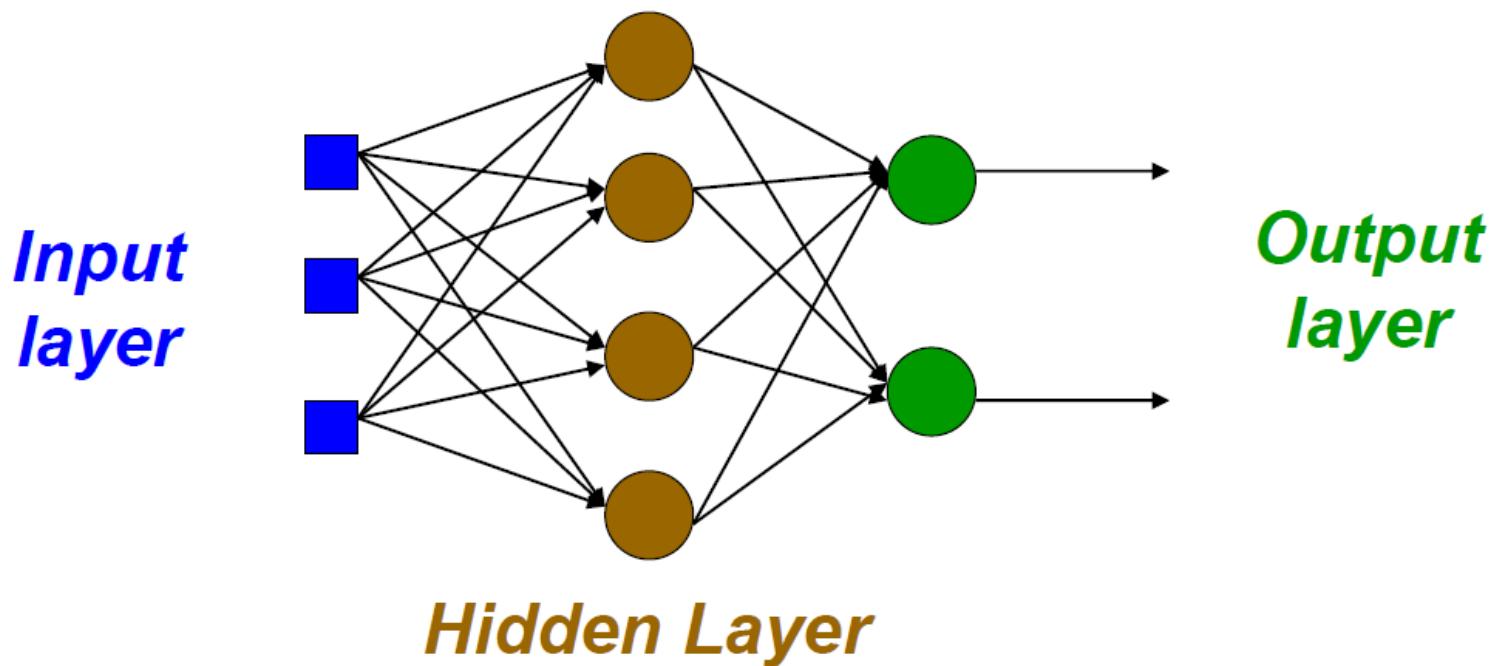
$$R = w_0 + \sum_{i=1}^n w_i x_i$$

$$o = sign(R) = \begin{cases} +1, & \text{if } R > 0 \\ -1, & \text{otherwise} \end{cases}$$

$$o = w_0 + \sum_{i=1}^n w_i x_i$$

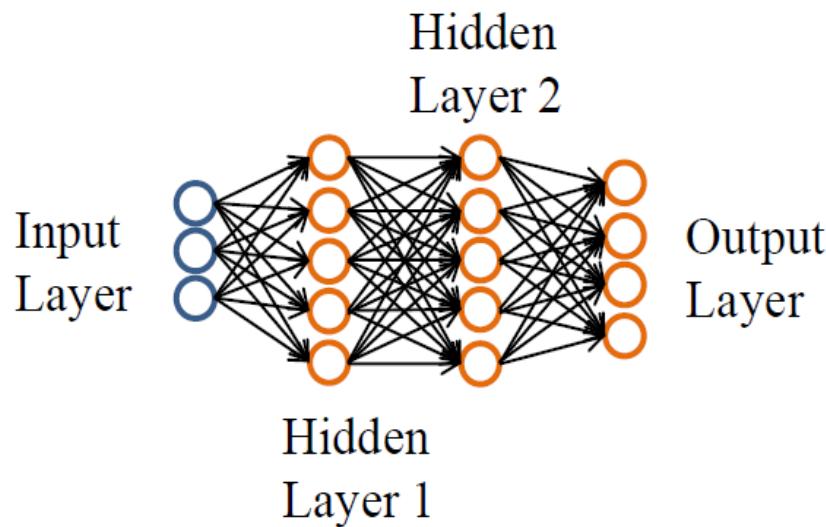
Multilayer Perceptron (MLP)

- A more general network architecture: between the input and output layers there are hidden layers
- Hidden nodes do not directly receive inputs nor send outputs to the external environment
- Fully connected between layers

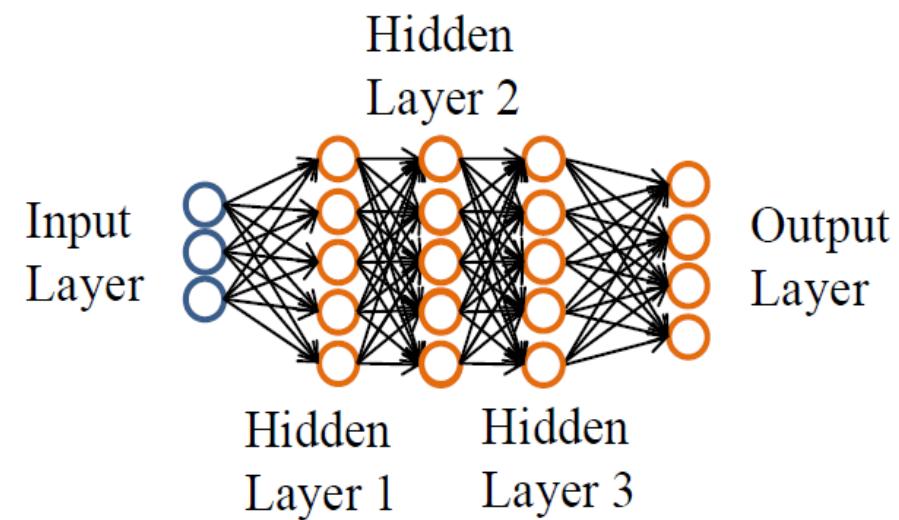


MLP Architecture

- Feedforward network: connections between the nodes do not form a cycle
- MLP usually interconnected in a feed-forward way
- The input layer does not count as a layer



3-layer feed-forward network

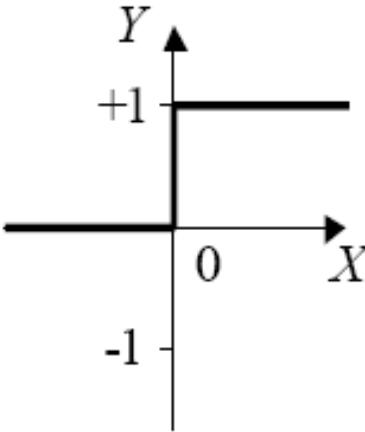
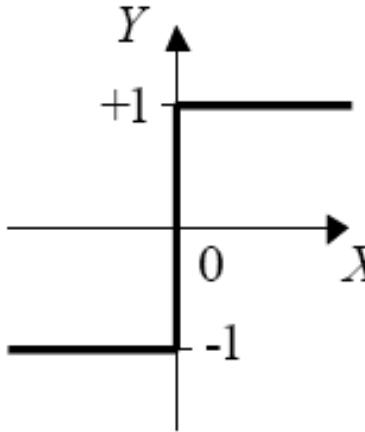
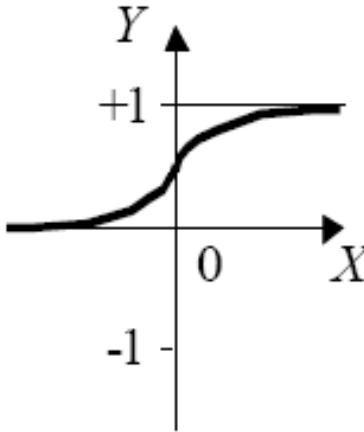
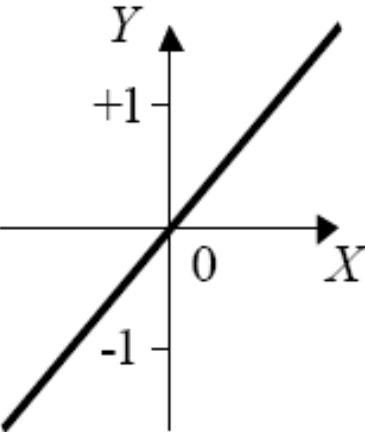


4-layer feed-forward network

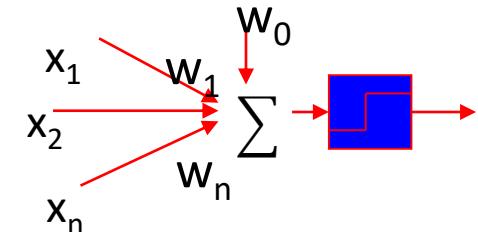
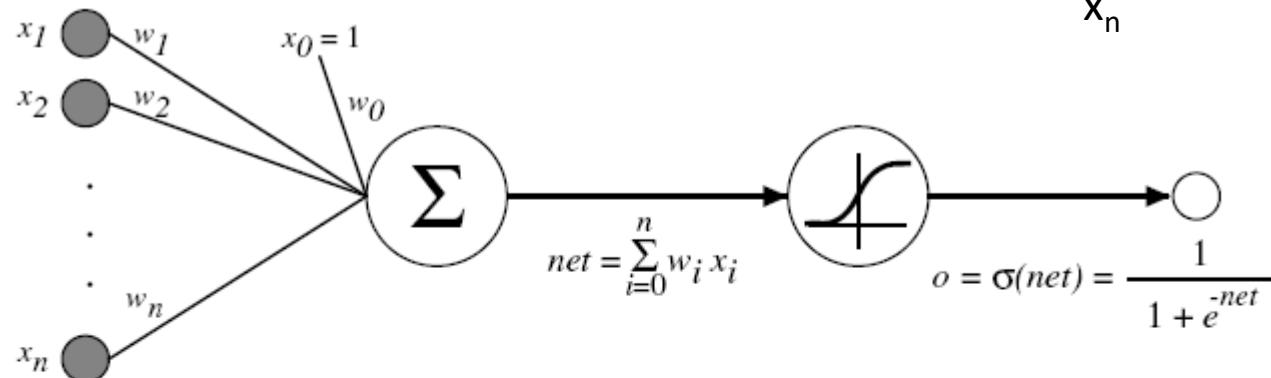
Activation Function

- Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and **determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction.** Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.
- The activation function can be considered as a mathematical “gate” in **between the input feeding the current neuron and its output going to the next layer.**

Activation Function

| Step function | Sign function | Sigmoid function | Linear function |
|--|--|--|--|
|  |  |  |  |
| $Y^{step} = \begin{cases} 1, & \text{if } X \geq 0 \\ 0, & \text{if } X < 0 \end{cases}$ | $Y^{sign} = \begin{cases} +1, & \text{if } X \geq 0 \\ -1, & \text{if } X < 0 \end{cases}$ | $Y^{sigmoid} = \frac{1}{1+e^{-X}}$ | $Y^{linear} = X$ |

Sigmoid Unit



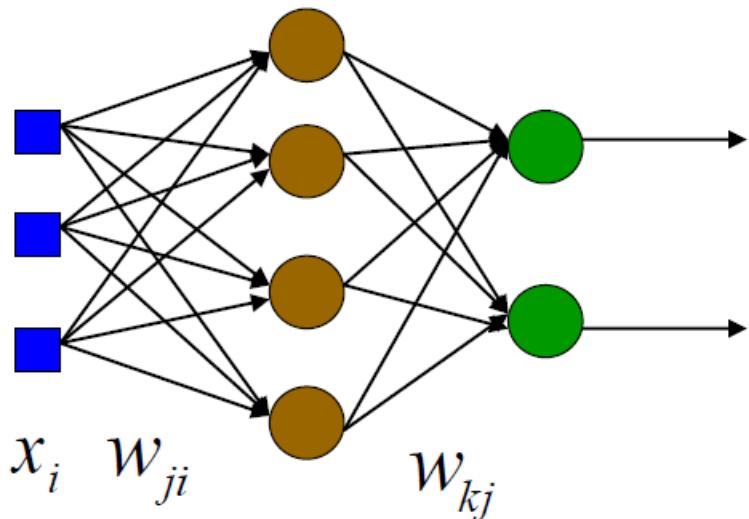
$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

We can derive gradient decent rules to train

Multilayer Perceptron



w_{ji} = weight associated with i th input to hidden unit j

w_{kj} = weight associated with j th input to output unit k

y_j = output of j th hidden unit

o_k = output of k th output unit

n = number of inputs

nH = number of hidden neurons

K = number of output neurons

$$y_j = \sigma\left(\sum_{i=0}^n x_i w_{ji}\right)$$

$$o_k = \sigma\left(\sum_{j=0}^{nH} y_j w_{kj}\right)$$

$$o_k = \sigma\left(\sum_{j=0}^{nH} \sigma\left(\sum_{i=0}^n x_i w_{ji}\right) w_{kj}\right)$$

Chain Rule

- In calculus, the chain rule is a formula to compute the derivative of a composite function.

Suppose that we have two functions $f(x)$ and $g(x)$ and they are both differentiable.

1. If we define $F(x) = (f \circ g)(x)$ then the derivative of $F(x)$ is,

$$F'(x) = f'(g(x)) \cdot g'(x)$$

2. If we have $y = f(u)$ and $u = g(x)$ then the derivative of y is,

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Chain Rule

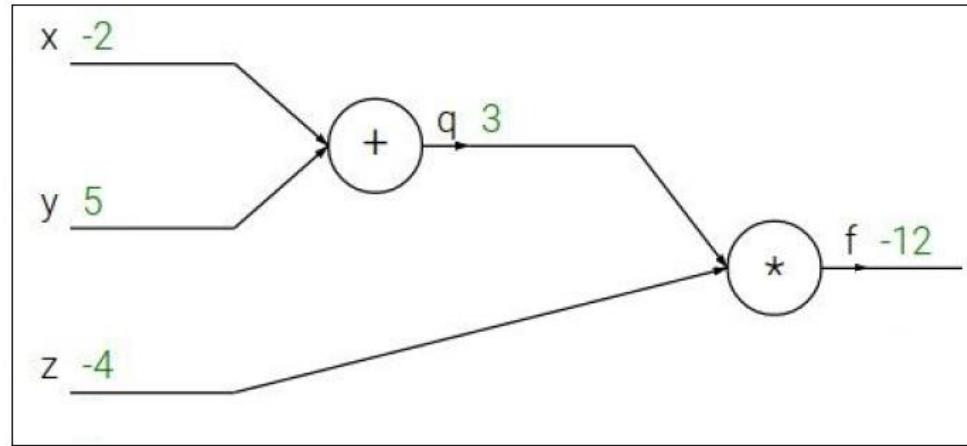
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain Rule

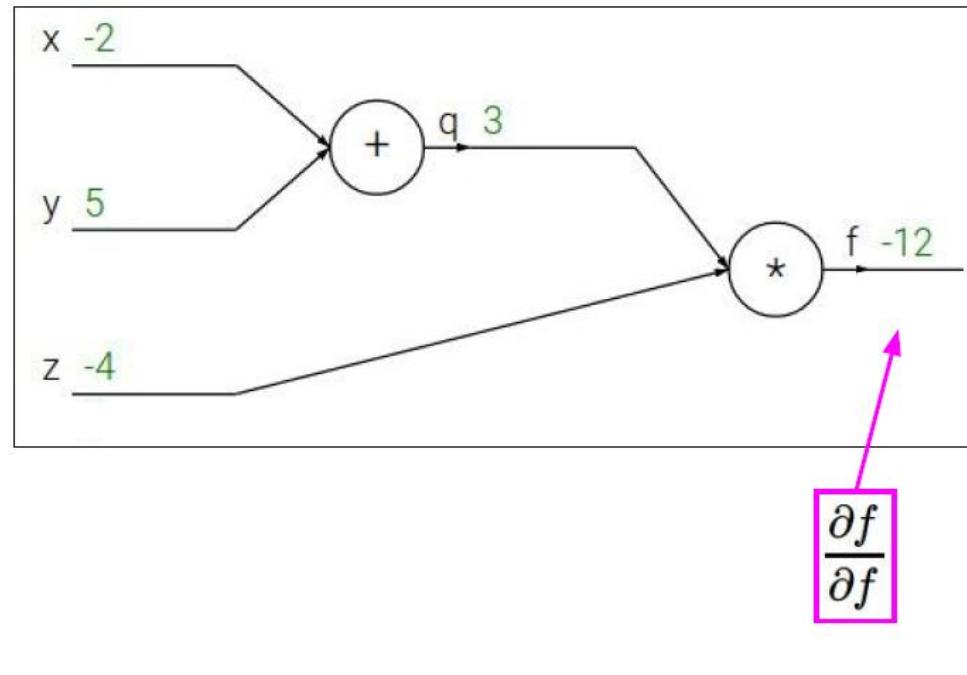
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain Rule

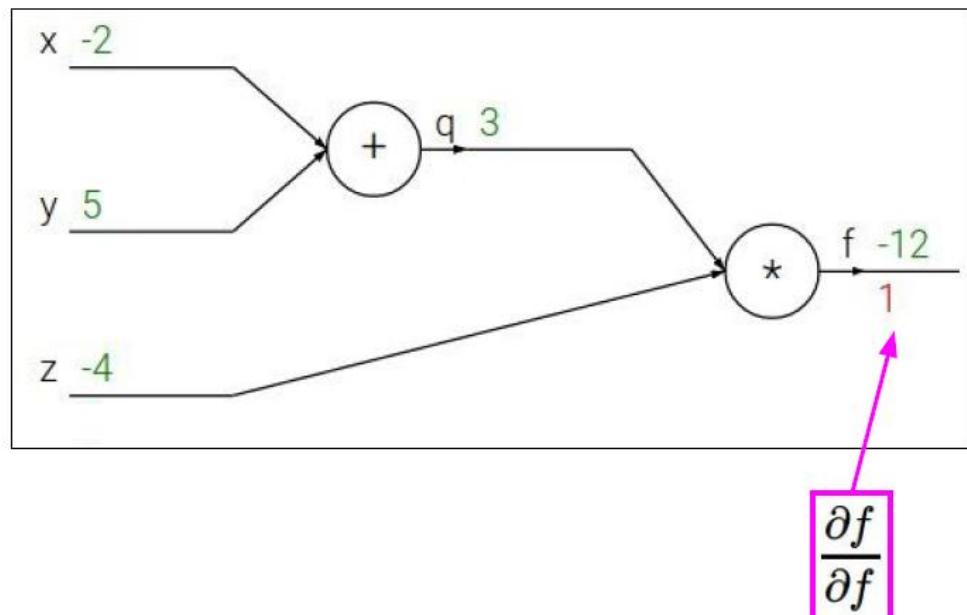
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



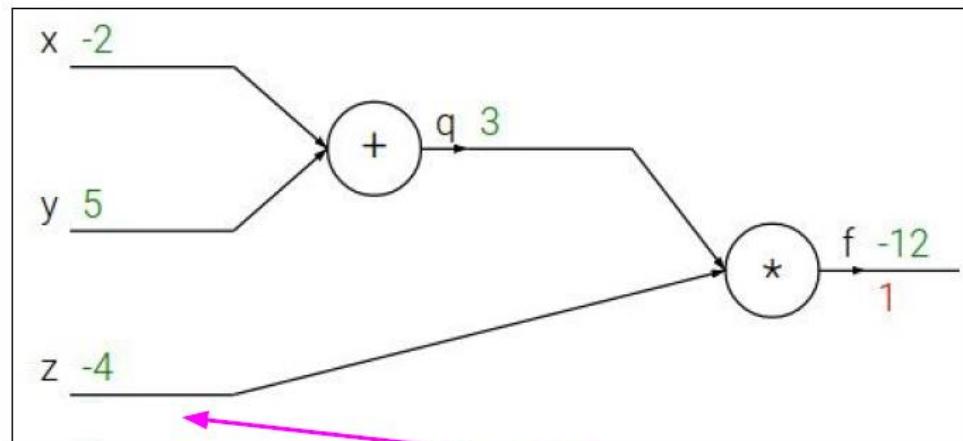
Chain Rule

$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$\frac{\partial f}{\partial z}$$

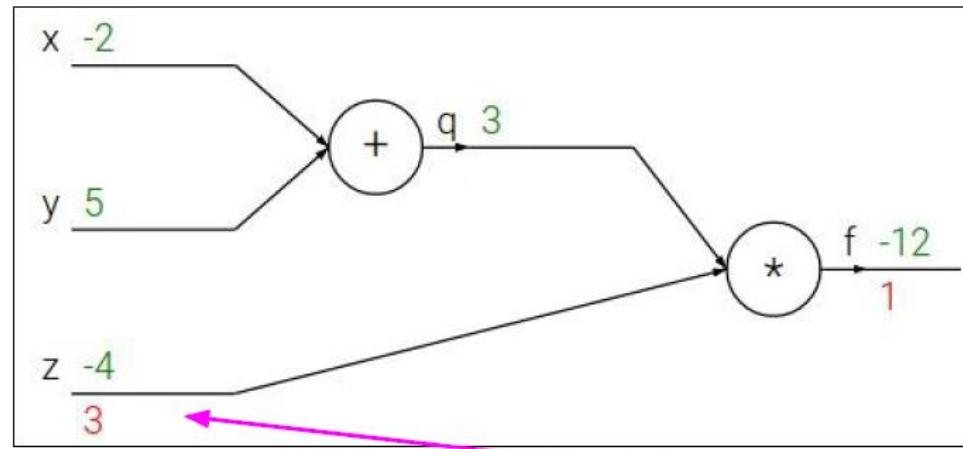
Chain Rule

$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



$$\frac{\partial f}{\partial z}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Chain Rule

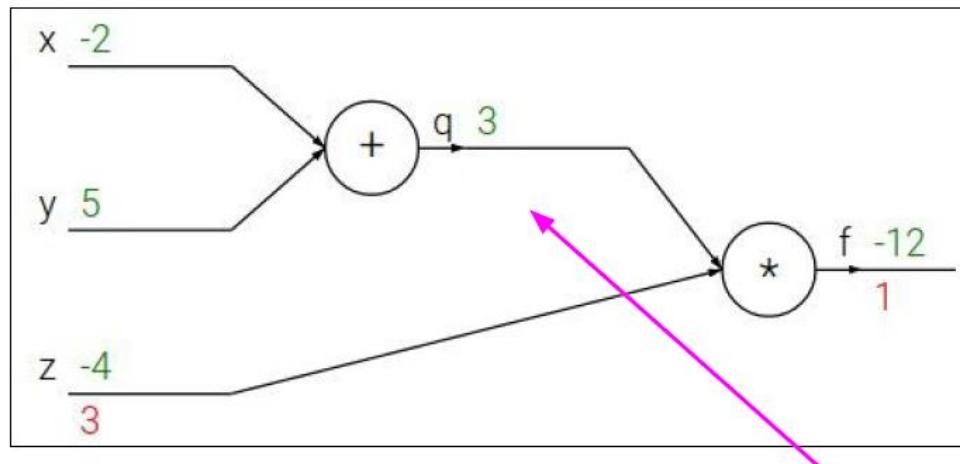
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

Chain Rule

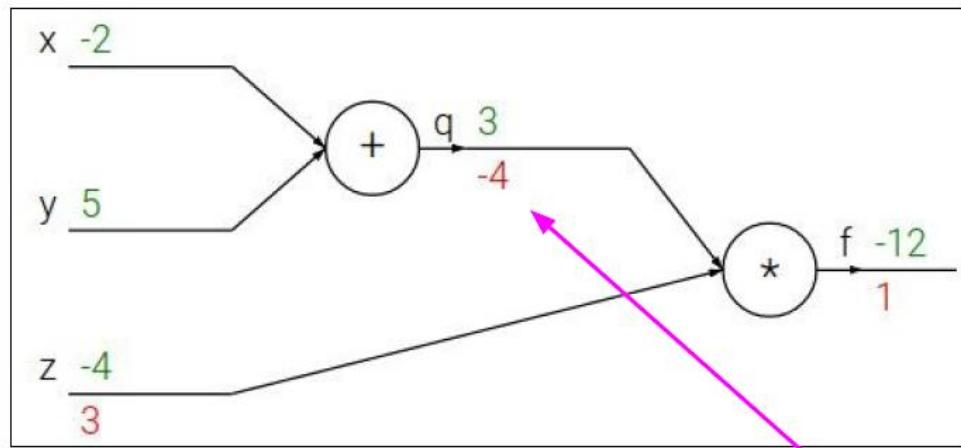
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

Chain Rule

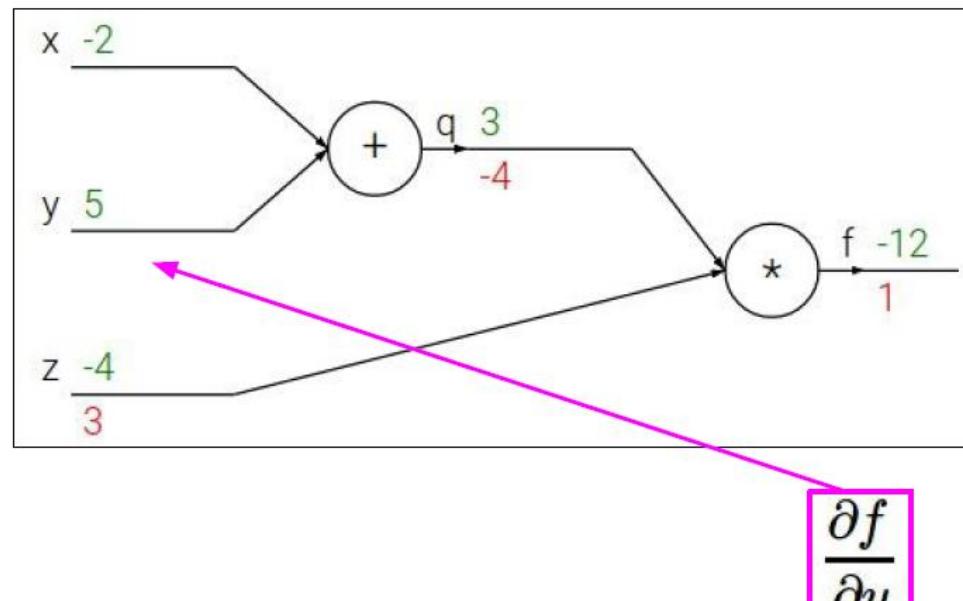
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain Rule

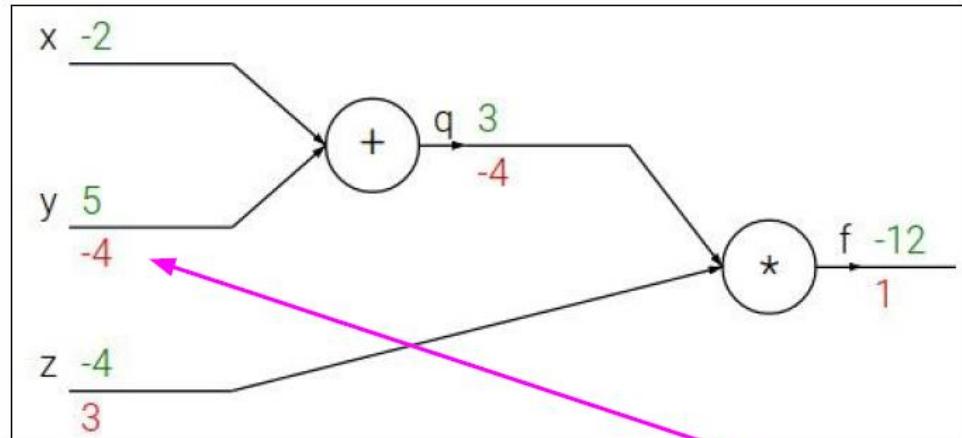
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

$$\frac{\partial f}{\partial y}$$

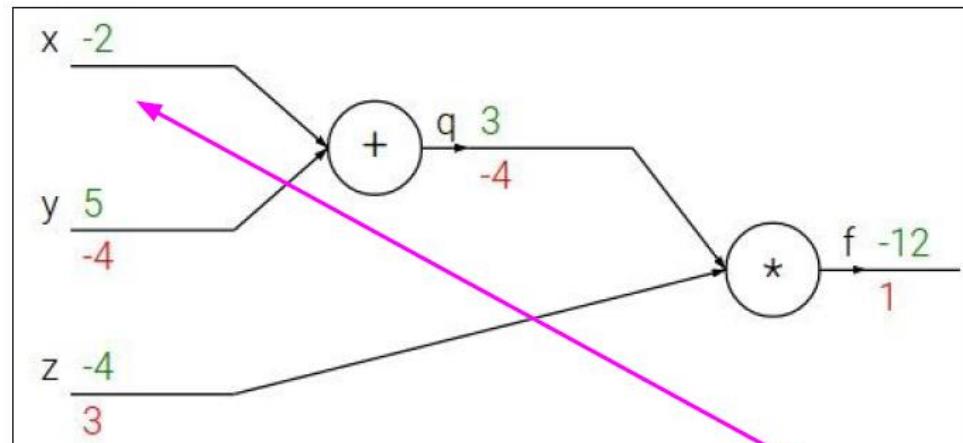
Chain Rule

$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Chain Rule

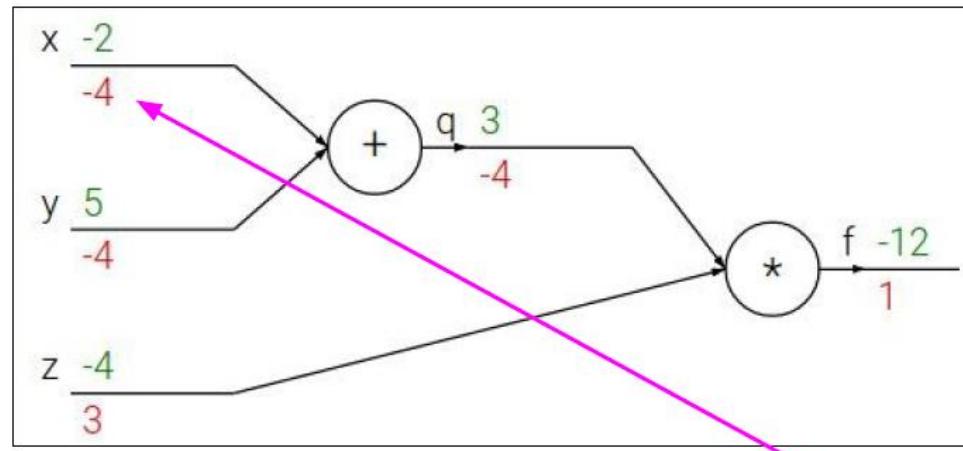
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

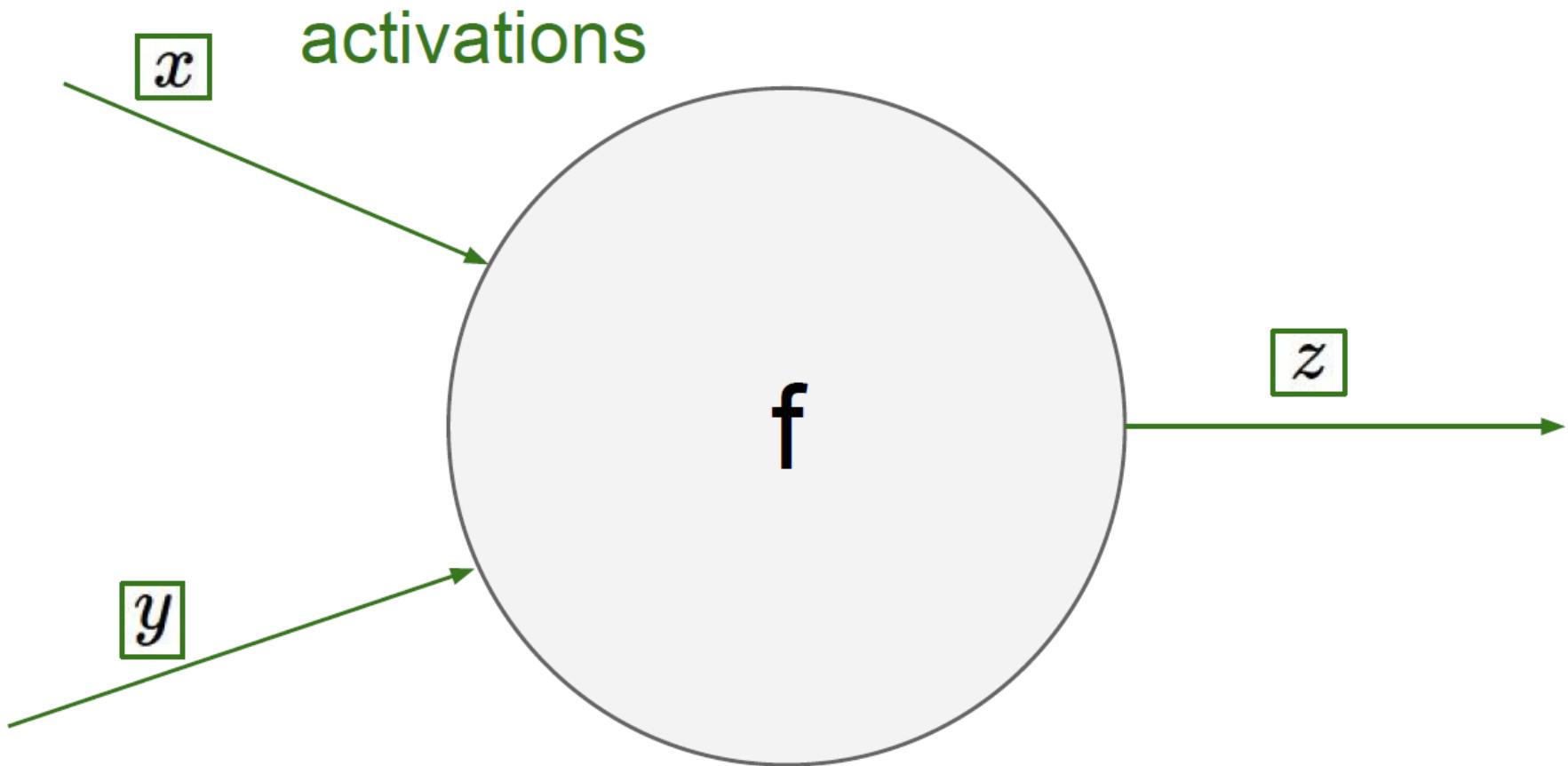


Chain rule:

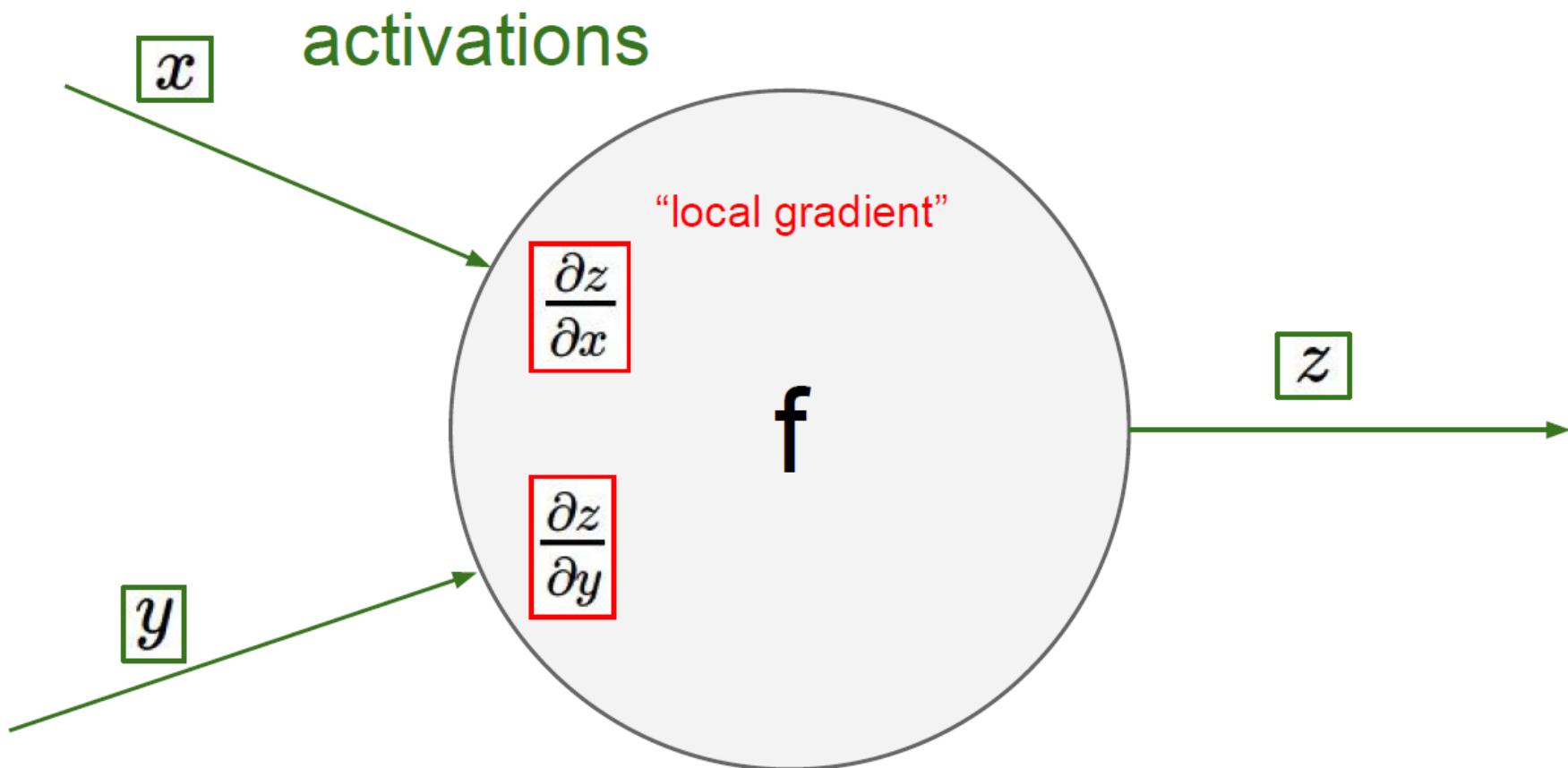
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial x}$$

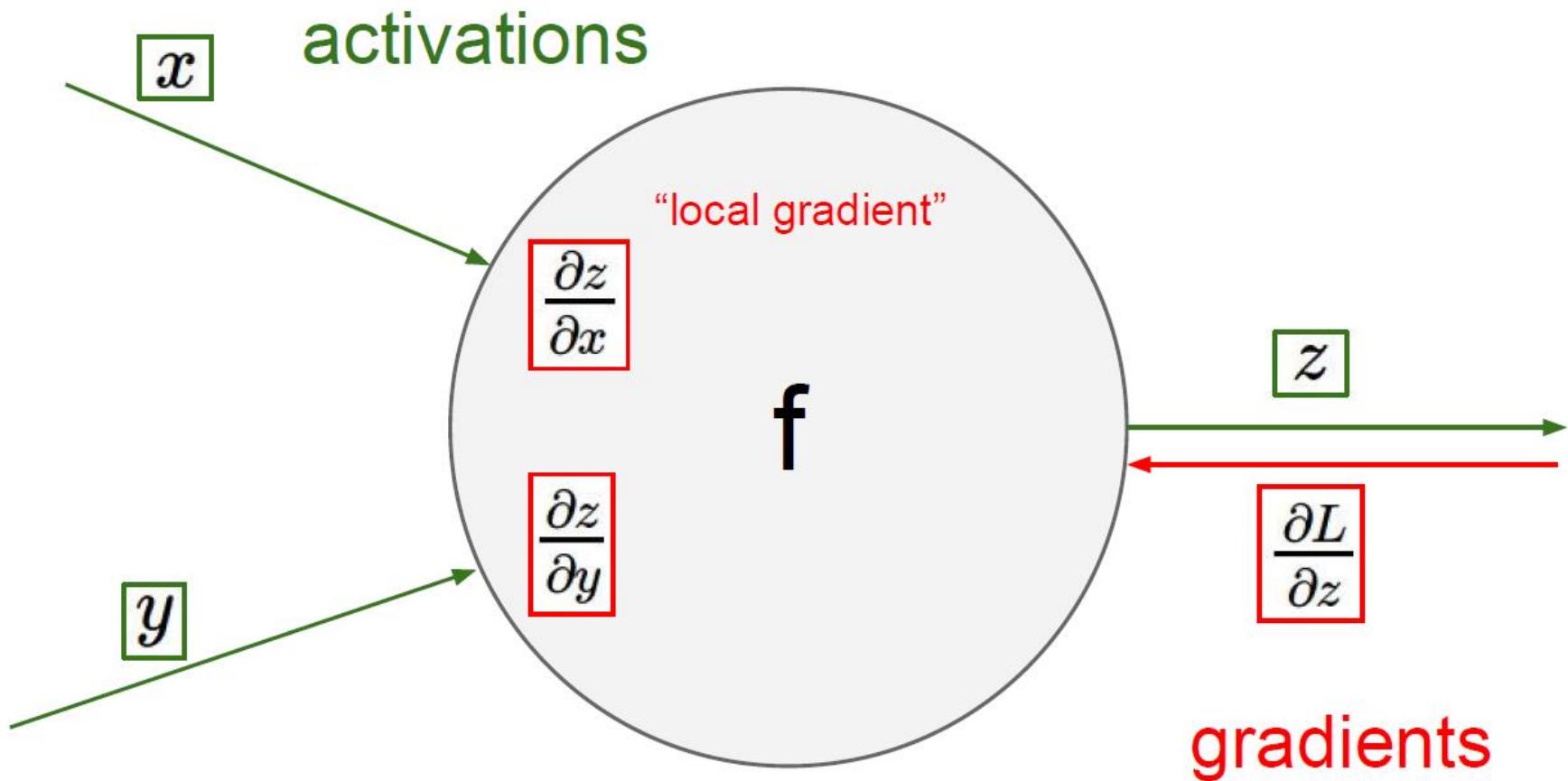
Chain Rule



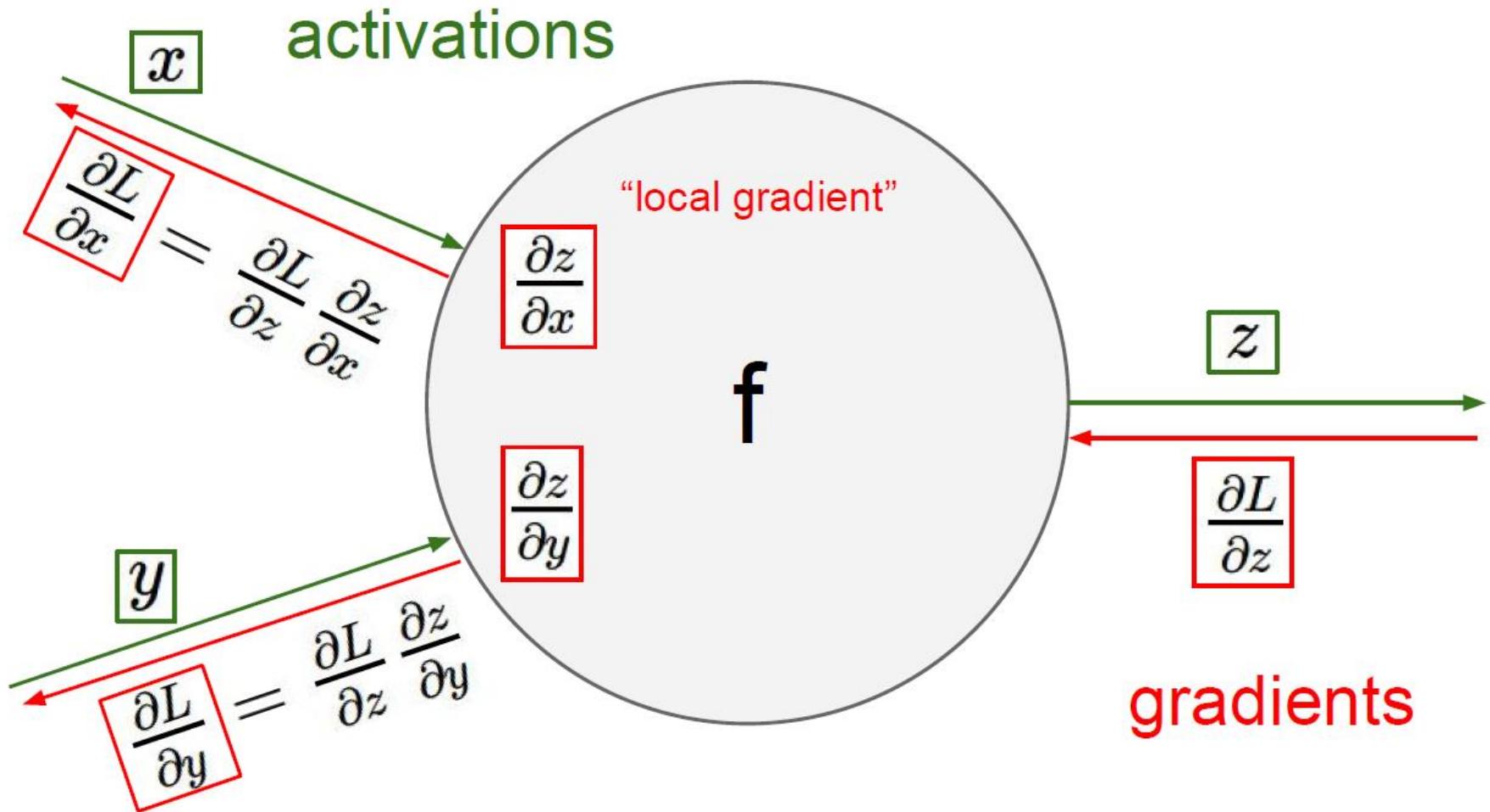
Chain Rule



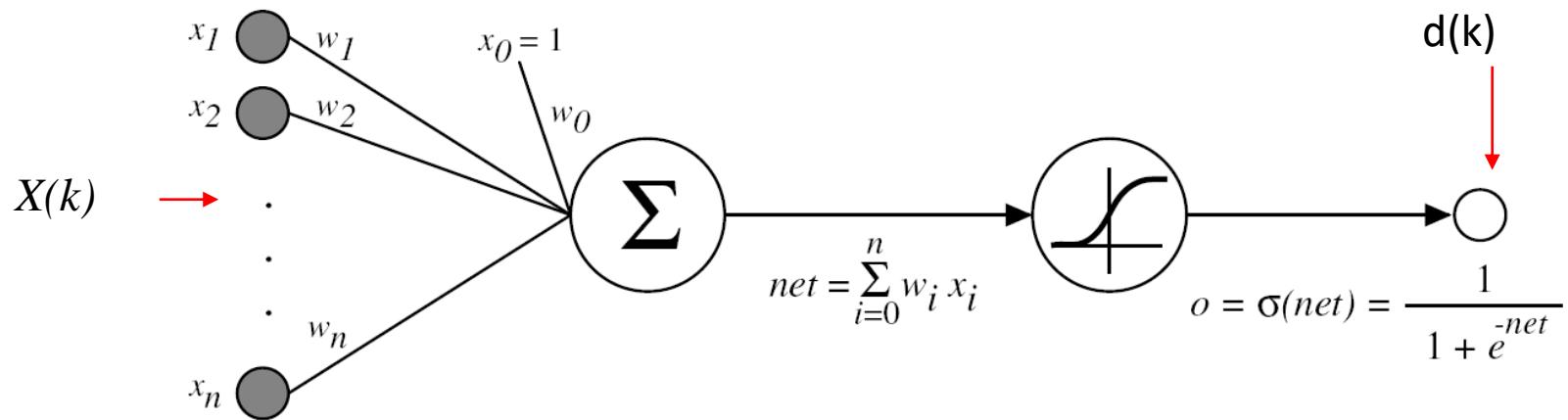
Chain Rule



Chain Rule



Error Gradient for a Sigmoid Unit



$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$

Error Gradient for a Sigmoid Unit

$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial w_i} \left(\frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2 \right) \\ &= \frac{1}{2} \sum_{k=1}^K \left(\frac{\partial E}{\partial w_i} (d(k) - o(k))^2 \right) \\ &= \frac{1}{2} \sum_{k=1}^K \left(2(d(k) - o(k)) \frac{\partial E}{\partial w_i} (d(k) - o(k)) \right) \\ &= \sum_{k=1}^K \left((d(k) - o(k)) \frac{\partial E}{\partial w_i} (-o(k)) \right) \\ &= - \sum_{k=1}^K \left((d(k) - o(k)) \frac{\partial o(k)}{\partial net(k)} \frac{\partial net(k)}{\partial w_i} \right)\end{aligned}$$

Error Gradient for a Sigmoid Unit

$$\frac{\partial o(k)}{\partial net(k)} = \frac{\partial \sigma(net(k))}{\partial net(k)} = \sigma(net(k)) (1 - \sigma(net(k))) = o(k)(1 - o(k))$$

$$\frac{\partial net(k)}{\partial w_i} = \frac{\partial (WX(k))}{\partial w_i} = x_i(k)$$

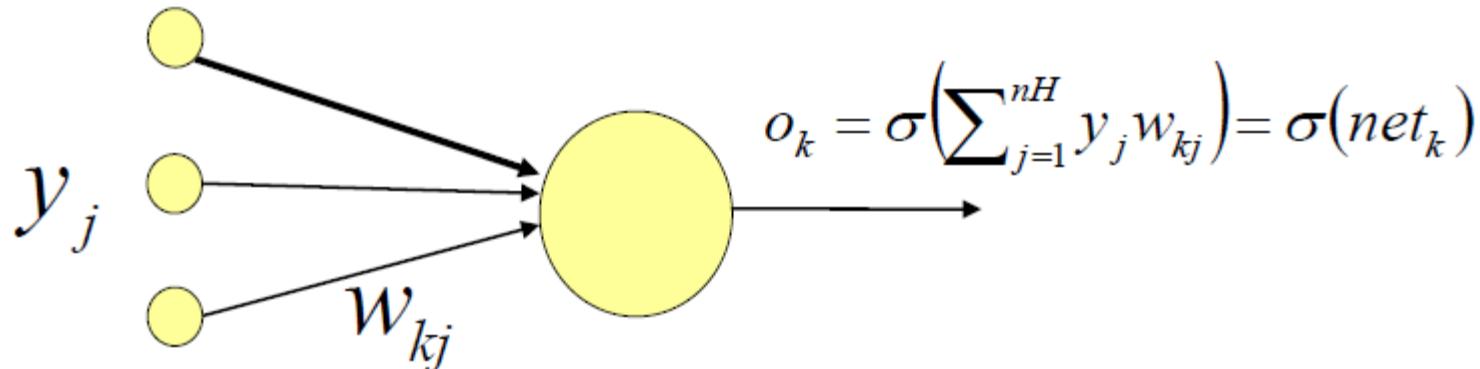
$$\begin{aligned}\frac{\partial E}{\partial w_i} &= - \sum_{k=1}^K \left((d(k) - o(k)) \frac{\partial o(k)}{\partial net(k)} \frac{\partial net(k)}{\partial w_i} \right) \\ &= - \sum_{k=1}^K \left((d(k) - o(k)) o(k) (1 - o(k)) x_i(k) \right)\end{aligned}$$

Back-propagation: Initial Steps

- Training Set: A set of input vectors x_i , $i=1 \dots D$ with the corresponding targets t_i .
- η : learning rate, controls the change rate of the weights.
- Begin with random weights.

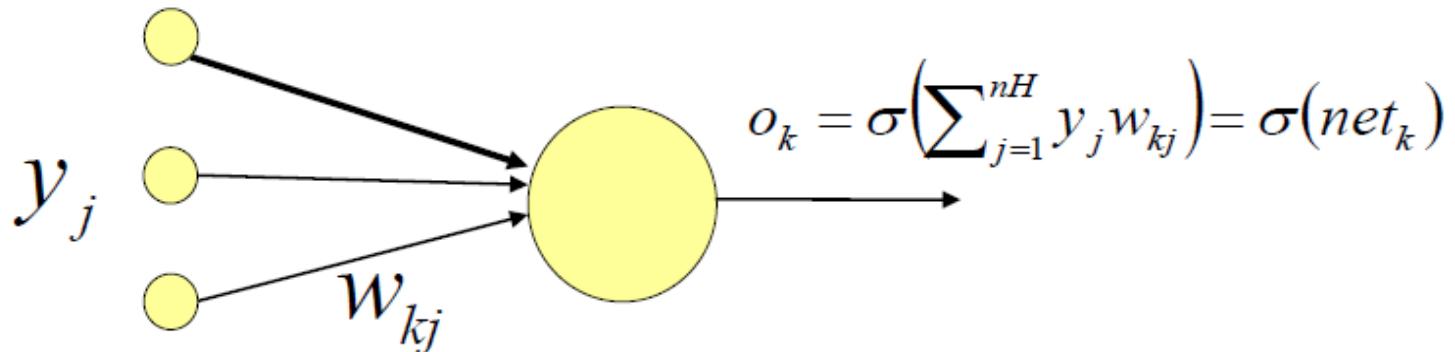
Back-propagation: Output Neurons

$$E(W) \equiv \frac{1}{2} \sum_{k=1}^K (d(k) - o(k))^2$$



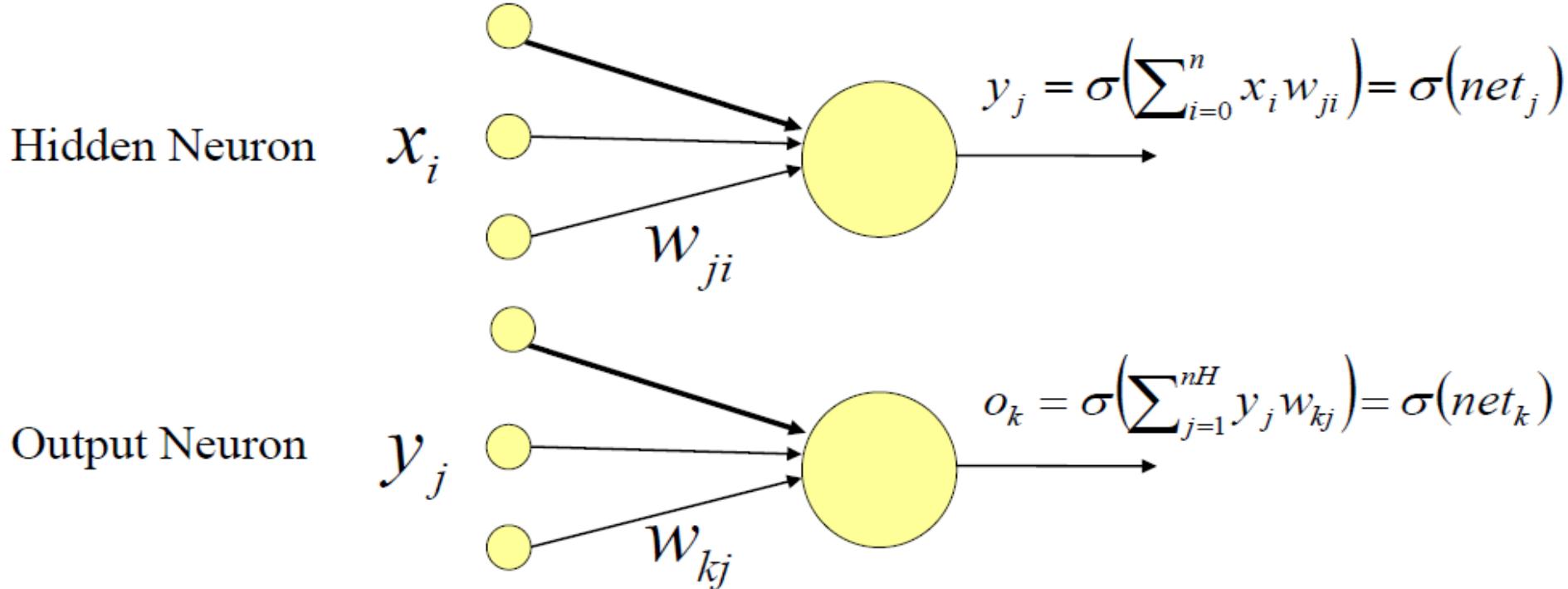
- E depends on the weights because
$$o_k = \sigma\left(\sum_{j=1}^{nH} y_j w_{kj}\right)$$
- For simplicity we assume the error of one training example

Back-propagation: Output Neurons



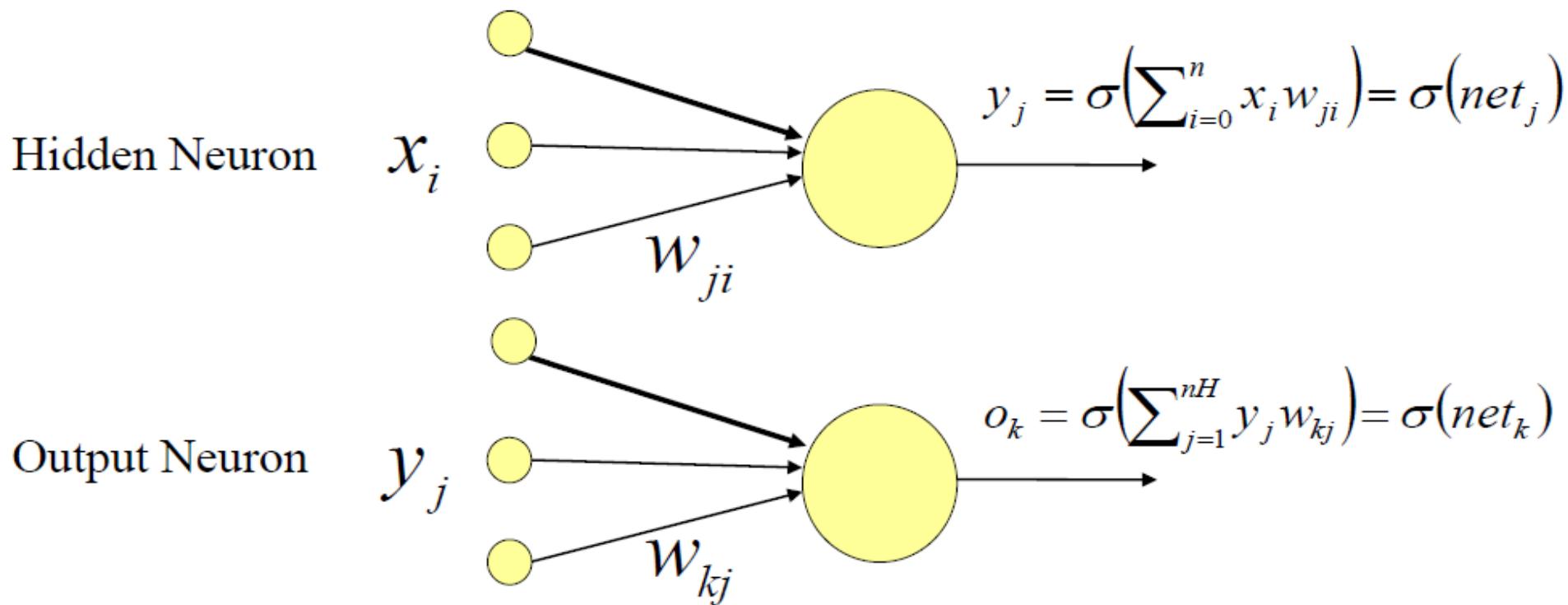
- $\frac{\partial E_k}{\partial w_{kj}} = \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial w_{kj}} = \frac{\partial E_k}{\partial o_k} \frac{\partial \sigma(\text{net}_k)}{\partial \text{net}_k} y_j$
- We define $\delta_k = \frac{\partial E_k}{\partial o_k} \frac{\partial \sigma(\text{net}_k)}{\partial \text{net}_k}$
- Update: $\Delta w_{kj} = -\eta \frac{\partial E_k}{\partial w_{kj}} = -\eta \delta_k y_j$

Back-propagation: Hidden Neurons



- $\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} \frac{\partial \sigma(net_j)}{\partial net_j} x_i$
- $\frac{\partial E}{\partial y_j} = \sum_{k=1}^K \frac{\partial E_k}{\partial y_j} = \sum_{k=1}^K \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial y_j} = \sum_{k=1}^K \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial y_j}$

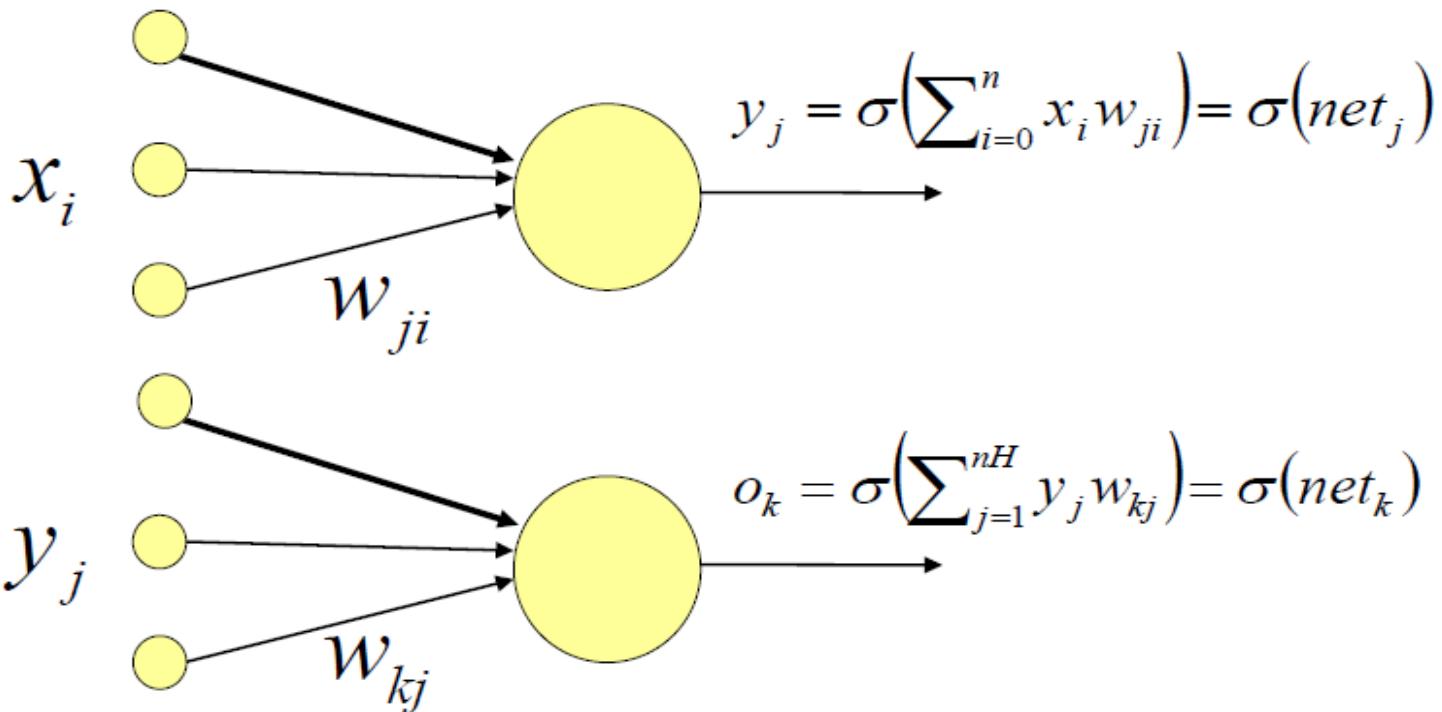
Back-propagation: Hidden Neurons



- $\frac{\partial E}{\partial y_j} = \boxed{\sum_{k=1}^K} \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial y_j} = \sum_{k=1}^K \delta_k w_{kj}$
- $\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \sum_{k=1}^K (\delta_k w_{kj}) \frac{\partial \sigma(net_j)}{\partial net_j} x_i$

Back-propagation: Hidden Neurons

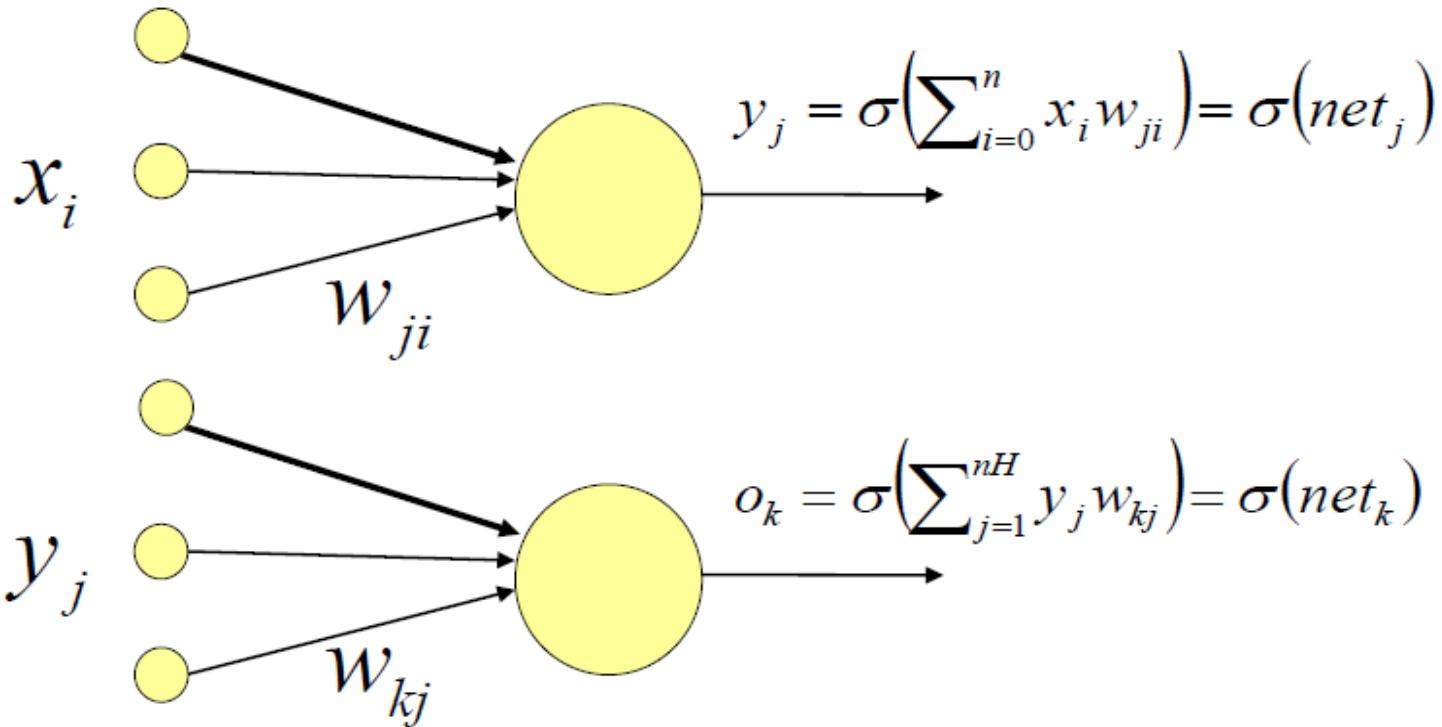
Hidden Neuron



- $\frac{\partial E}{\partial w_{ji}} = \sum_{k=1}^K (\delta_k w_{kj}) \frac{\partial \sigma(net_j)}{\partial net_j} x_i$
- We define $\delta_j = \sum_{k=1}^K (\delta_k w_{kj}) \frac{\partial \sigma(net_j)}{\partial net_j}$

Back-propagation: Hidden Neurons

Hidden Neuron



- $\frac{\partial E}{\partial w_{ji}} = \delta_j x_i$
- Update: $\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \delta_j x_i$

Back-propagation Summary

1. Initialise weights randomly
2. For each input training example x compute the outputs (**forward pass**)
3. Compute the output neurons errors and then compute the update rule for output layer weights (**backward pass**)

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = -\eta \delta_k y_j \text{ where } \delta_k = \frac{\partial E}{\partial o_k} \frac{\partial \sigma(\text{net}_k)}{\partial \text{net}_k}$$

4. Compute hidden neurons errors and then compute the update rule for hidden layer weights (**backward pass**)

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \delta_j x_i \text{ where } \delta_j = \sum_{k=1}^K (\delta_k w_{kj}) \frac{\partial \sigma(\text{net}_j)}{\partial \text{net}_j}$$

Back-propagation Summary

5. Compute the sum of all Δw , once all training examples have been presented to the network
 6. Update weights $w_i \leftarrow w_i + \Delta w_i$
 7. Repeat steps 2-6 until the stopping criterion is met
- The algorithm will converge to a weight vector with minimum error, given that the learning rate is sufficiently small

Back-propagation Summary

- Gradient descent over entire network weight vector.
- Will find a local, not necessarily a global error minimum.
- In practice, it often works well (can run multiple times).
- Minimizes error over all training samples.
 - Will it generalize to subsequent examples? i.e., will the trained network perform well on data outside the training sample.
- Training can take thousands of iterations.
- After training, use the network is fast.

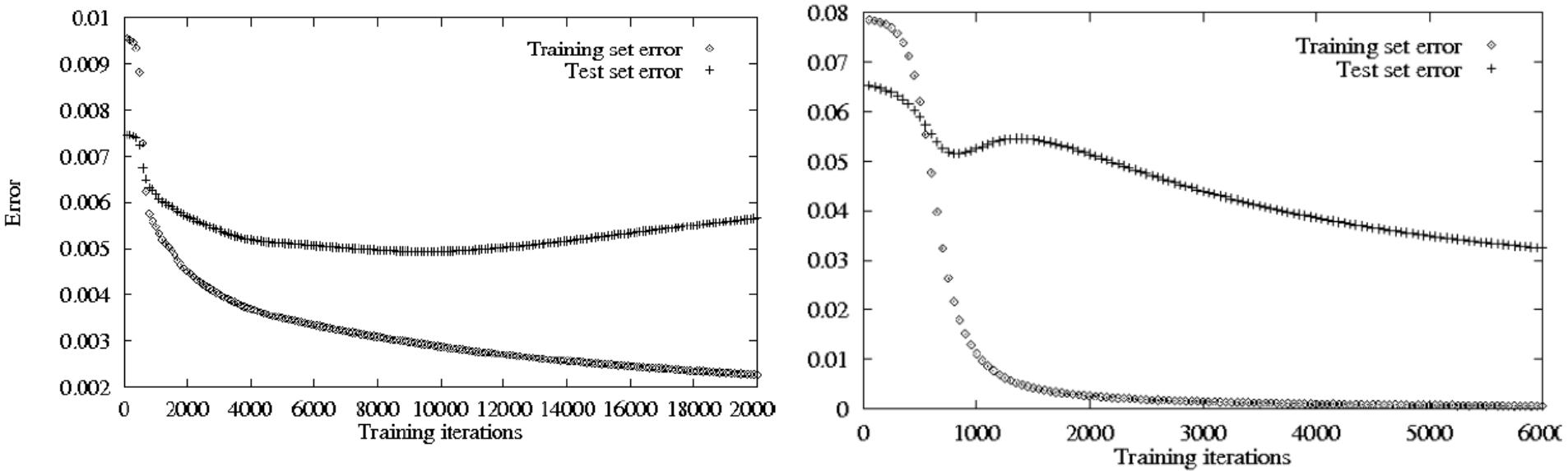
Generalization, Overfitting and Stopping Criterion

What is the appropriate condition for stopping weight update?

Continue until the error E falls below some predefined value?

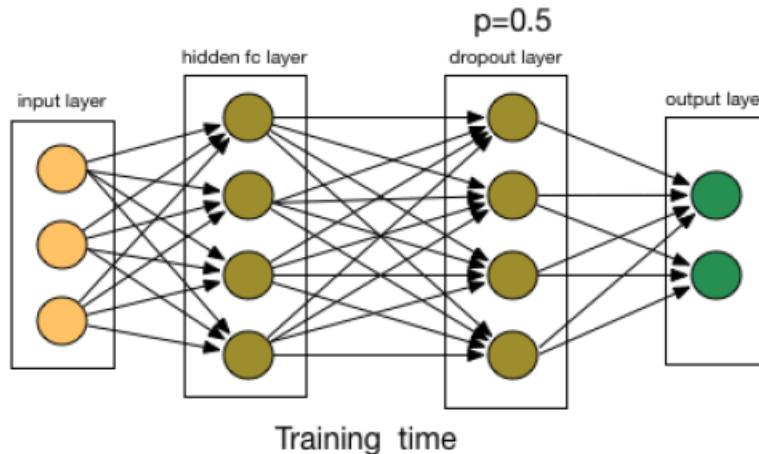
- Not a very good idea.
- Back-propagation is susceptible to overfitting the training example at the cost of decreasing generalization accuracy over other unseen examples.

Generalization, Overfitting and Stopping Criterion



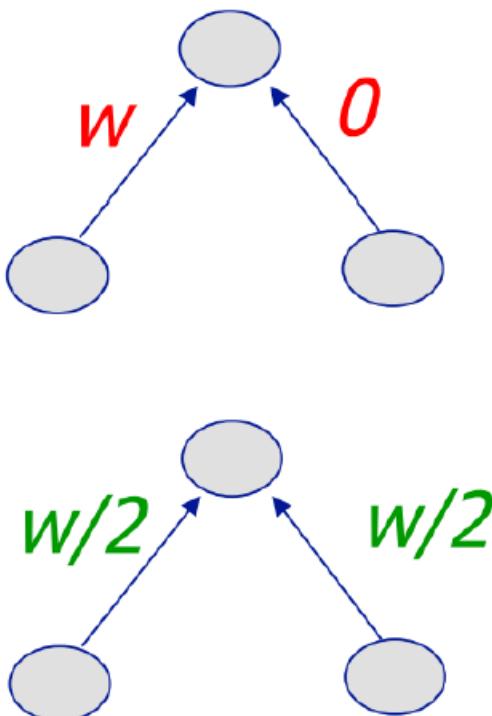
- Stop training when the validation set has the lowest error
- Error might decrease in the training set but increase in the ‘validation’ set (overfitting!)
- Early stopping: one way to avoid overfitting

Dropout



- **Dropout:** Randomly remove some nodes in the network (along with incoming and outgoing edges)
- Notes:
 - Usually $p \geq 0.5$ (p is probability of keeping node)
 - Input layers p should be much higher (and use noise instead of dropout)
 - Most deep learning frameworks come with a dropout layer

Weight Decay



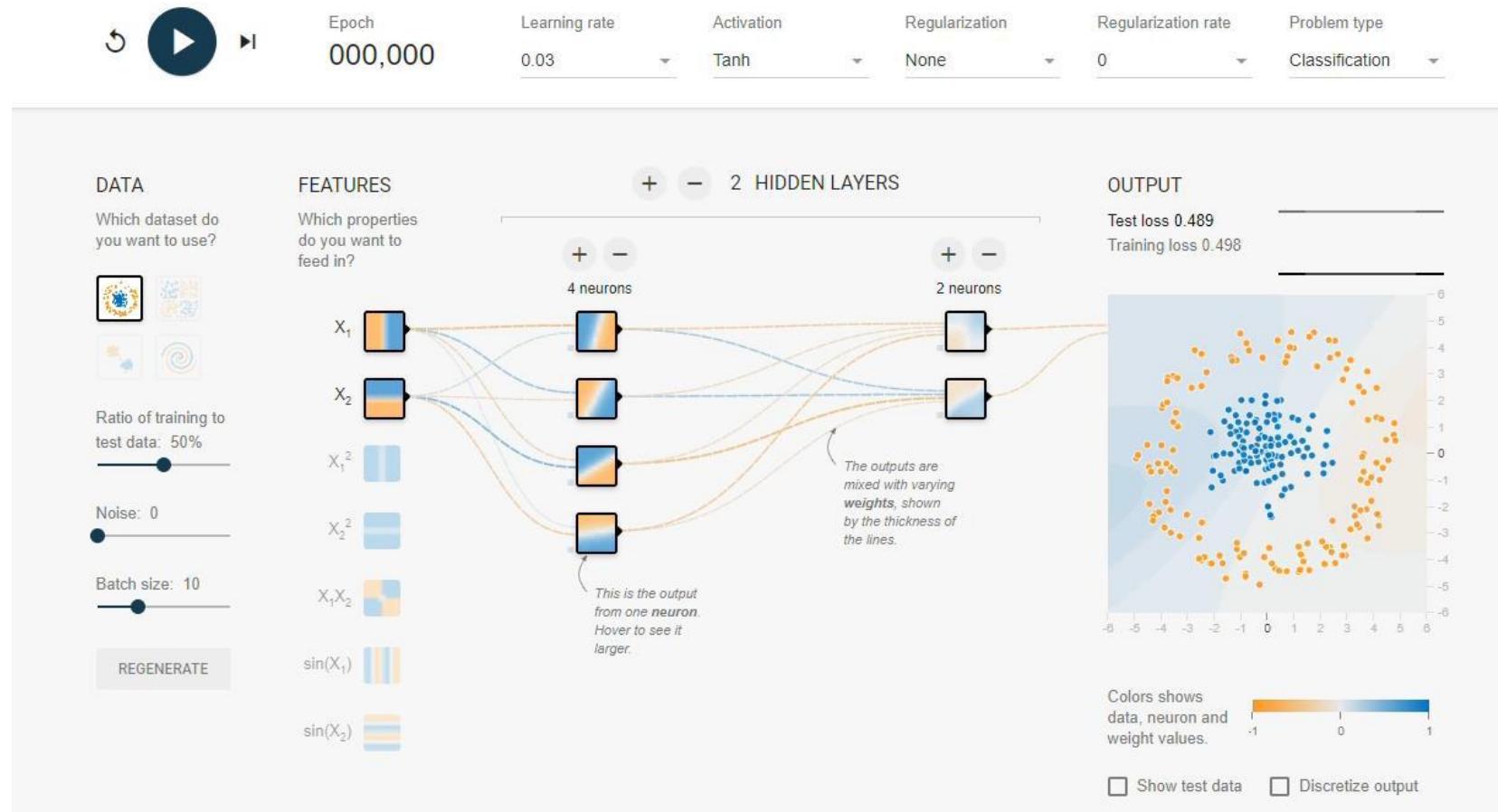
- **L2 Penalty:** Penalize squared weights. Result:
 - Keeps weight small unless error derivative is very large.
 - Prevent from fitting sampling error.
 - Smoother model (output changes slower as the input change).
 - If network has two similar inputs, it prefers to put half the weight on each rather than all the weight on one.
- **L1 Penalty:** Penalize absolute weights. Result:
 - Allow for a few weights to remain large.

Normalization

- Network Input Normalization
 - *Example:* Pixel to [0, 1] or [-1, 1] or according to mean and std.
- Batch Normalization (BatchNorm, BN)
 - Normalize hidden layer inputs to mini-batch mean & variance
 - Reduces impact of earlier layers on later layers
- Batch Renormalization (BatchRenorm, BR)
 - Fixes difference b/w training and inference by keeping a moving average asymptotically approaching a global normalization.

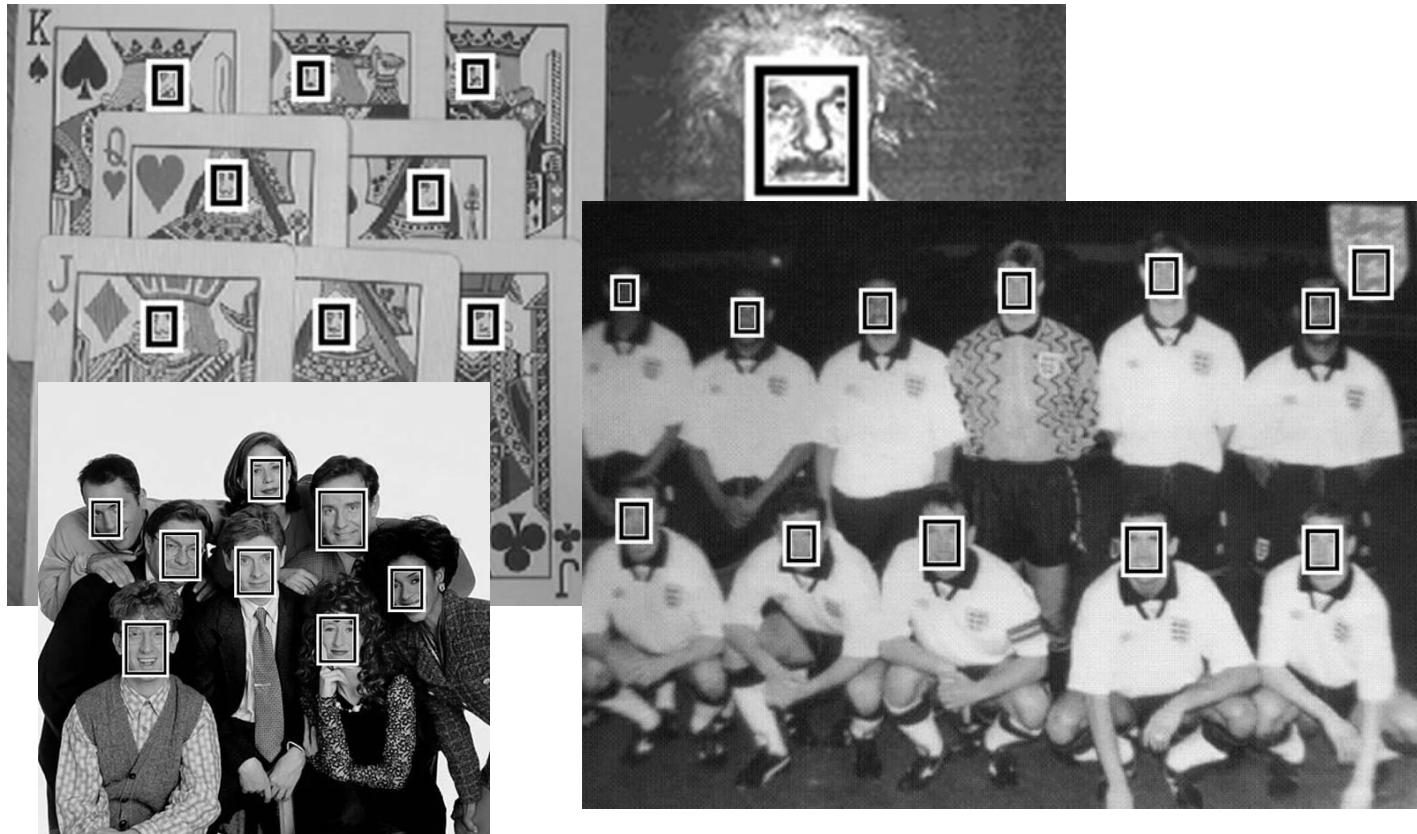
Neural Network Playground

<http://playground.tensorflow.org>



Application Example

Neural Network-based Face Detection



https://ri.cmu.edu/pub_files/pub1/rowley_henry_1996_3/rowley_henry_1996_3.pdf

Application Example

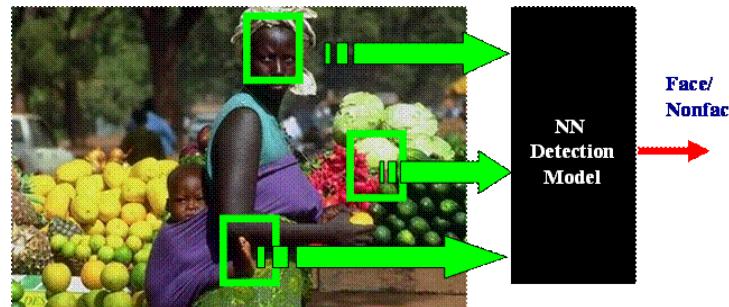
Neural Network-based Face Detection



Application Example

Neural Network-based Face Detection

- It takes 20×20 pixel window, feeds it into a NN, which outputs a value ranging from -1 to $+1$ signifying the presence or absence of a face in the region.
- The window is applied at every location of the image.
- To detect faces larger than 20×20 pixel, the image is repeatedly reduced in size.



Application Example

Neural Network-based Face Detection

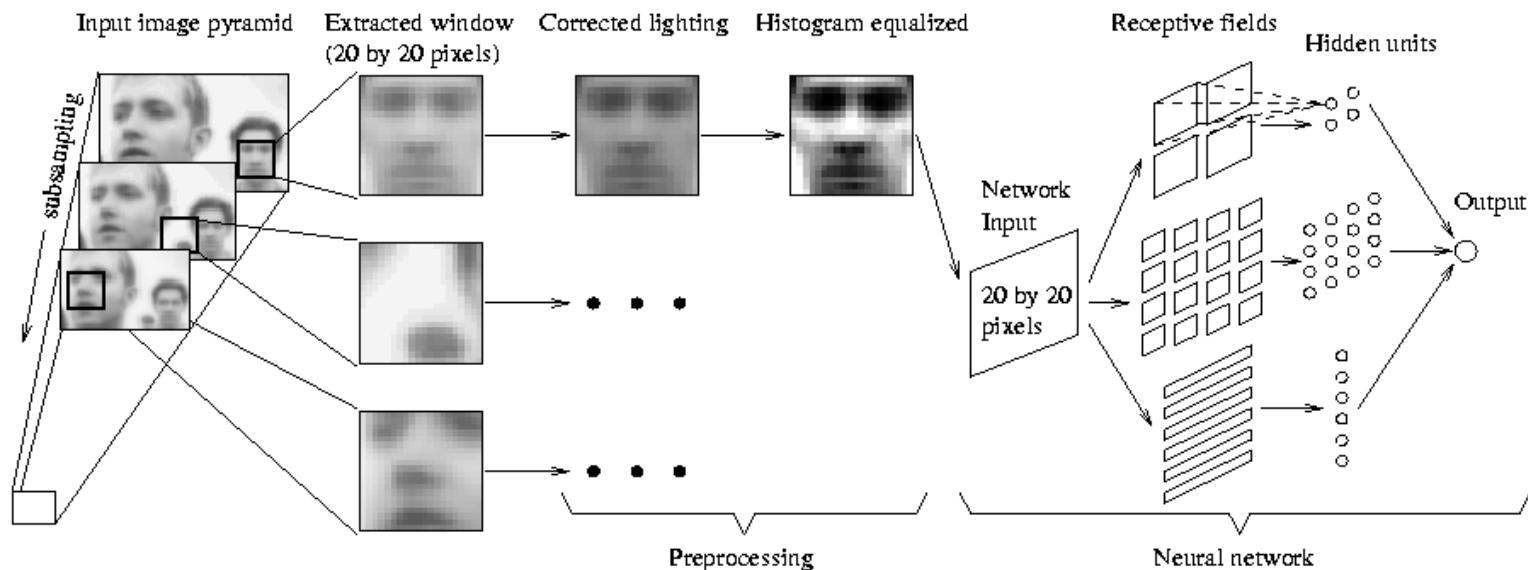


Figure 1: The basic algorithm used for face detection.

Application Example

Neural Network-based Face Detection

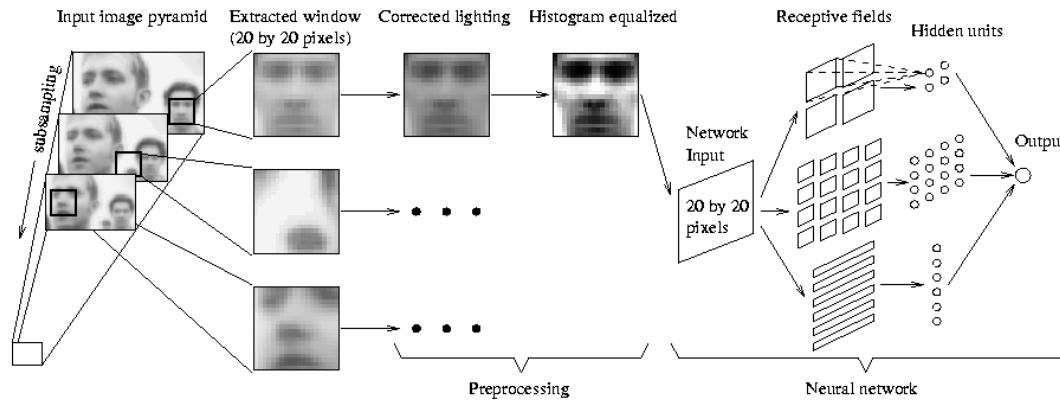


Figure 1: The basic algorithm used for face detection.

- 4 look at 10×10 subregions
- 16 look at 5×5 subregions
- 6 look at 20×5 horizontal stripes of pixels

Application Example

Neural Network-based Face Detection

- Training samples
 - 1050 initial face images. More face example are generated from this set by rotation and scaling. Desired output +1
 - Non-face training samples: Use a bootstrapping technique to collect 8000 non-face training samples from 146,212,178 subimage regions! Desired output -1

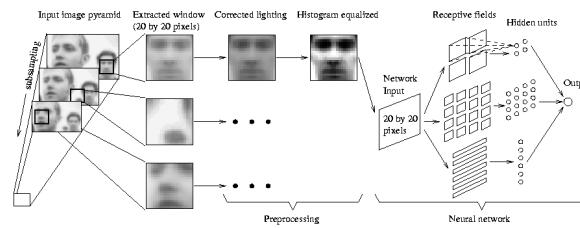


Figure 1: The basic algorithm used for face detection.

Application Example

Neural Network-based Face Detection

- Training samples: Non-face training samples

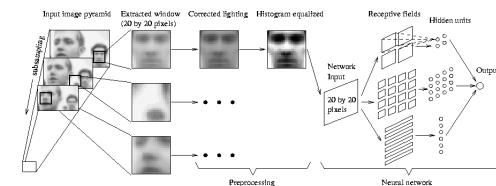
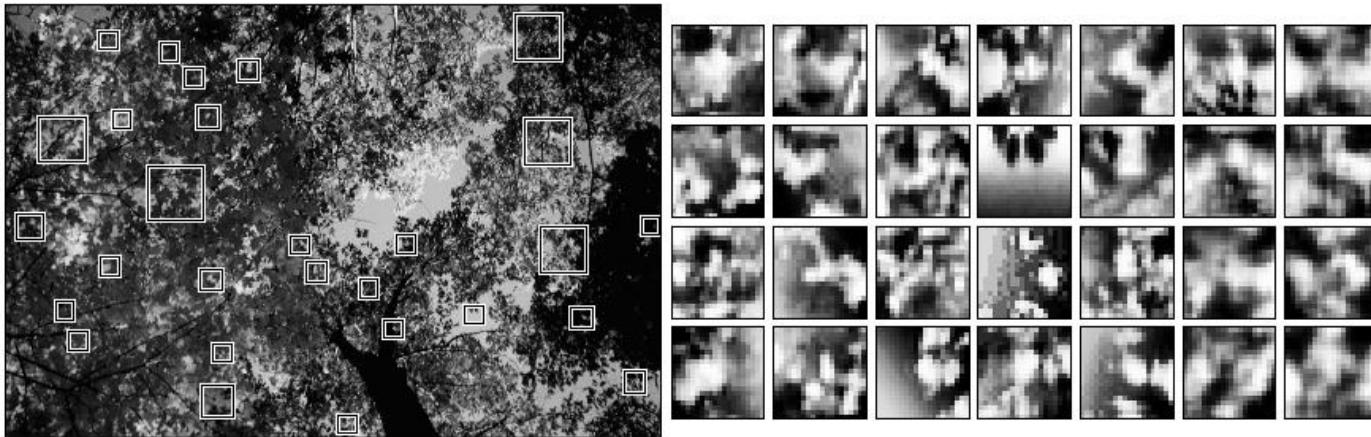


Figure 1: The basic algorithm used for face detection.



Application Example

Neural Network-based Face Detection

- Post-processing and face detection

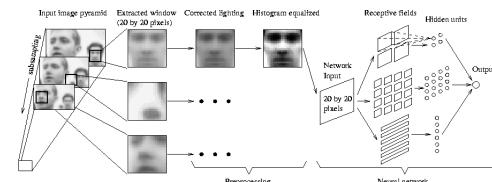
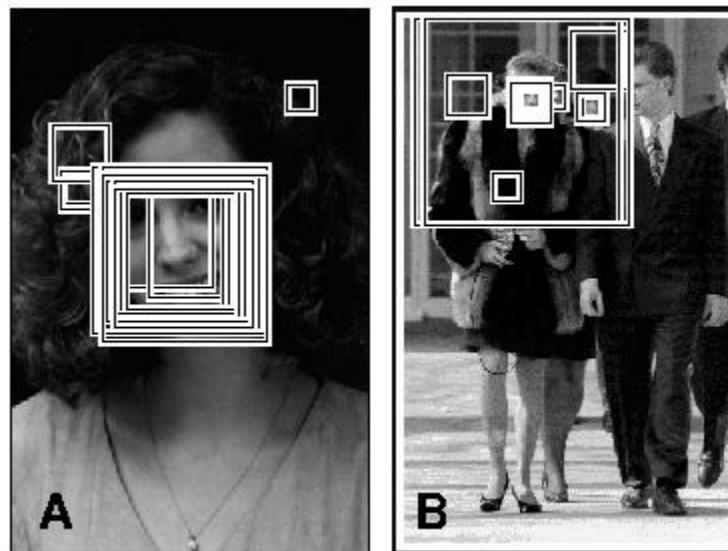


Figure 1: The basic algorithm used for face detection.



Application Examples

Neural Network-based Face Detection

- Results and Issues
- 77.% ~ 90.3% detection rate (130 test images)
- Process 320x240 image in 2 – 4 seconds on a 200MHz R4400 SGI Indigo 2

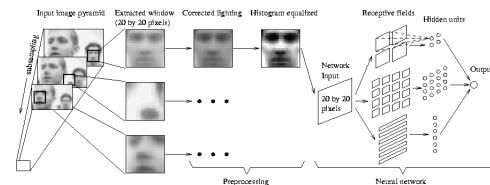


Figure 1: The basic algorithm used for face detection.

Further Reading

Chapter 4, T. M. Mitchell, Machine Learning,
McGraw-Hill International Edition, 1997



University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

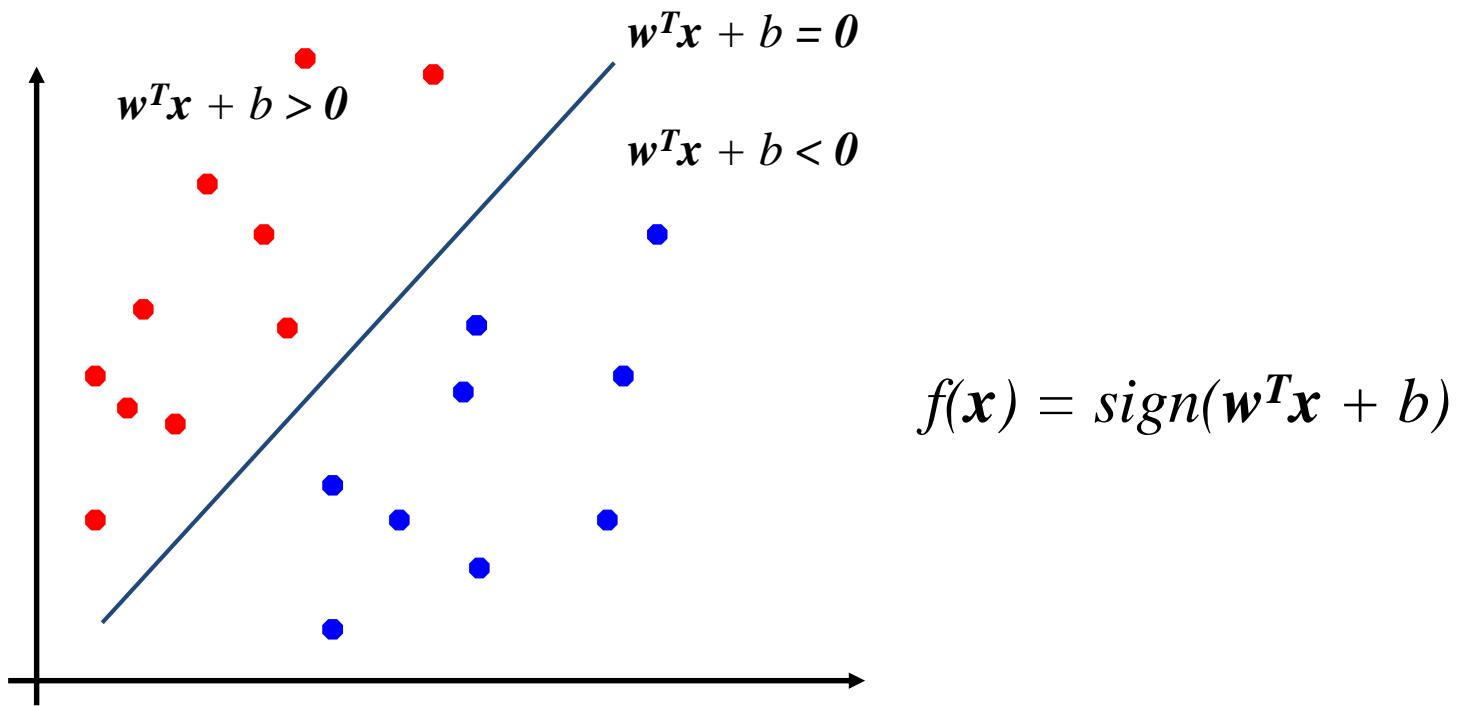
Machine Learning

Topic 13 – Support Vector Machine

Zheng Lu
2024 Autumn

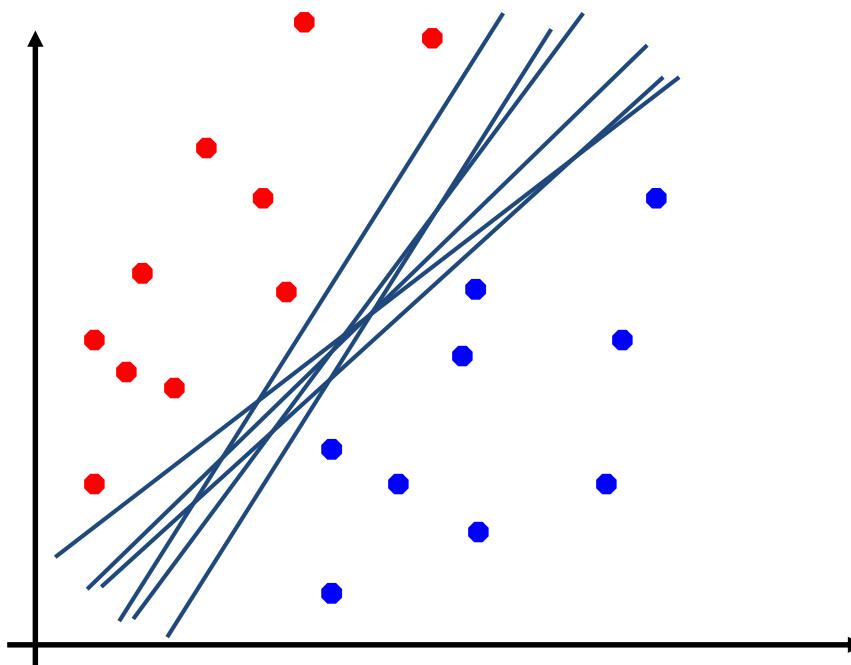
Perceptron Revisited: Linear Separators

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



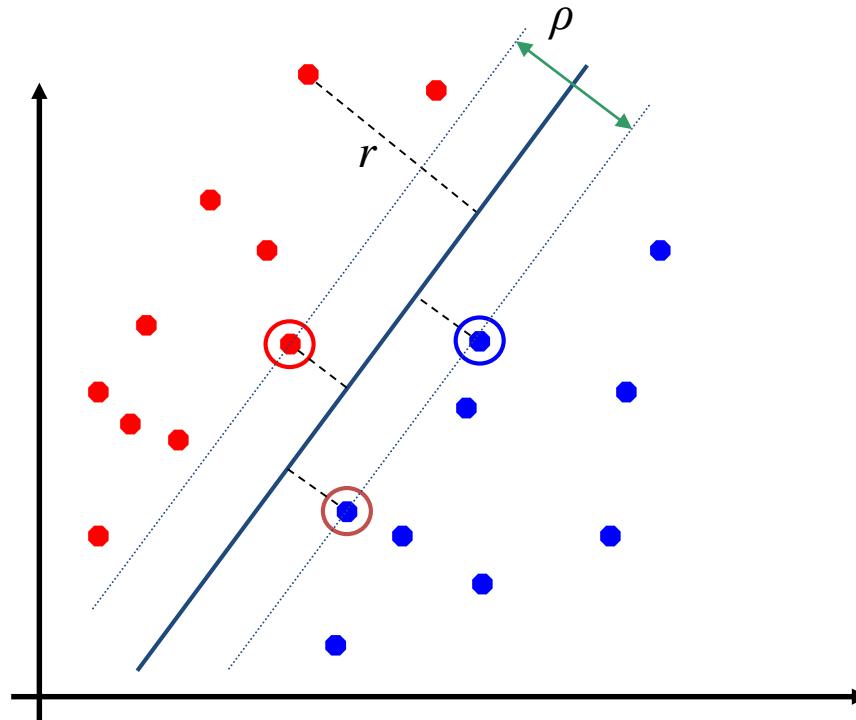
Linear Separator

Which of the linear separators is optimal?



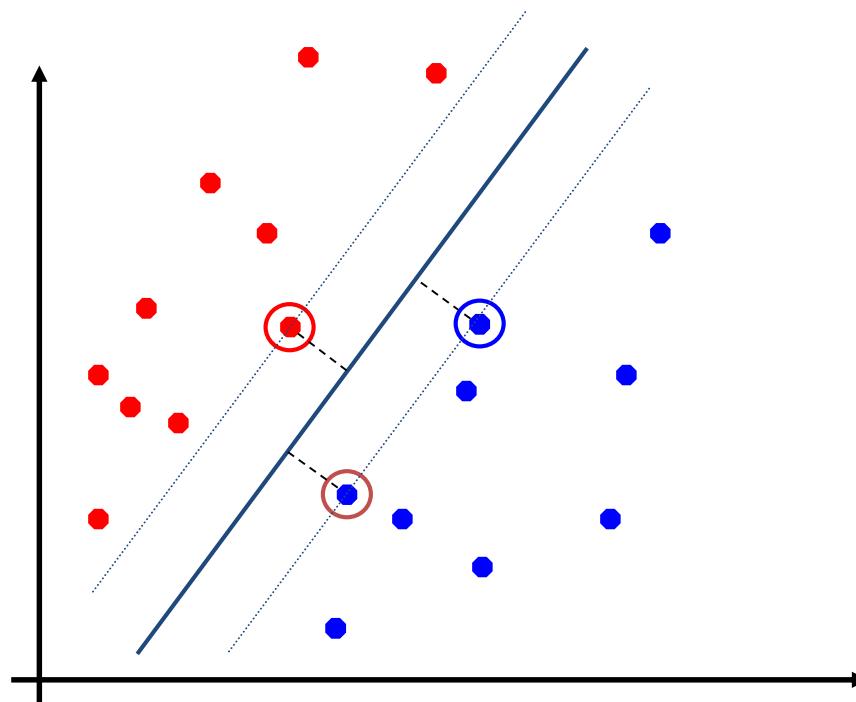
Classification Margin

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



Maximum Margin Classification

- Maximizing the margin is good according to intuition.
- Implying that only support vectors matter; other training examples are ignorable.



Linear SVM Mathematically

- Let training set $\{(x_i, y_i)\}_{i=1..w}, x_i \in R^d, y_i \in \{-1, 1\}$ be separated by a hyperplane with margin $r=2d$. Then for each training example (x_i, y_i) :

$$\begin{aligned} \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} &\leq -d & \text{if } y_i = -1 \\ \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} &\geq d & \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i \left(\frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right) \geq d$$

- For every support vector x_s the above inequality is an equality.
After rescaling, we can obtain

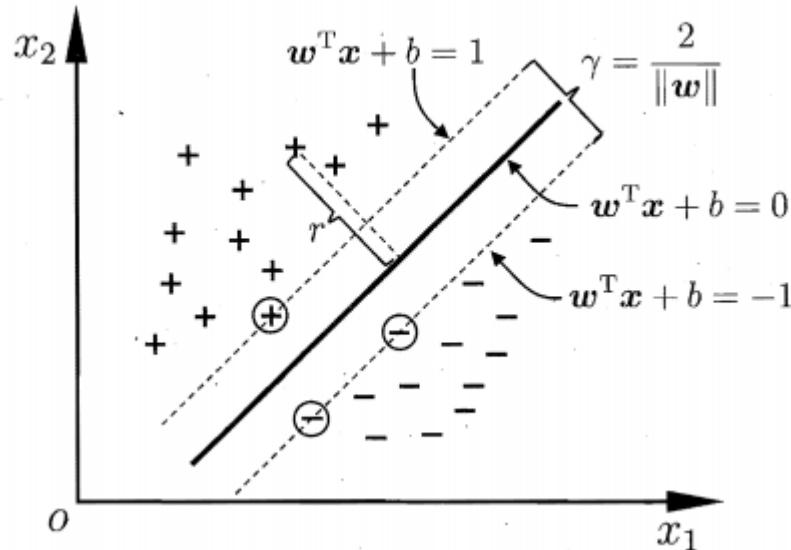
$$y_s(\mathbf{w}^T \mathbf{x}_s + b) = 1$$

- Alternatively, we have

$$(\mathbf{w}^T \mathbf{x}_s + b) = 1 \quad \text{if } y_s = 1$$

$$(\mathbf{w}^T \mathbf{x}_s + b) = -1 \quad \text{if } y_s = -1$$

Linear SVM Mathematically



- Then the margin can be expressed as:

$$d = \frac{|\mathbf{w}^T \mathbf{x}_s + b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

$$r = 2d = \frac{2}{\|\mathbf{w}\|}$$

Linear SVM Mathematically

- Then we can formulate the quadratic optimization problem:

Find w and b such that

$r = \frac{2}{\|w\|}$ is maximized,

and for all $(x_i, y_i), i = 1..n$: $y_i(w^T x_i + b) \geq 1$

Which can be reformulated as:

Find w and b such that

$\Phi(w) = \frac{1}{2} \|w\|^2$ is minimized,

and for all $(x_i, y_i), i = 1..n$: $y_i(w^T x_i + b) \geq 1$

Solving the Optimization Problem

Find w and b such that

$$\Phi(w) = \frac{1}{2} \|w\|^2 \text{ minimized,}$$

and for all $(x_i, y_i), i = 1..n$: $y_i(w^T x_i + b) \geq 1$

- Need to optimize a **quadratic function subject to linear constraints**.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a **dual problem** where a **Lagrange multiplier** α_i is associated with every inequality constraint in the primal (original) problem:

Find α_i such that

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum \alpha_i (y_i(w^T \cdot x_i + b) - 1) \text{ is maximized for all } \alpha_i \geq 0$$

The Optimization Problem Solution

- Given a solution $\alpha_1 \dots \alpha_n$ to the dual problem, solution to the primal is:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i, \quad b = y_j - \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j, \quad \text{for any } \alpha_j > 0$$

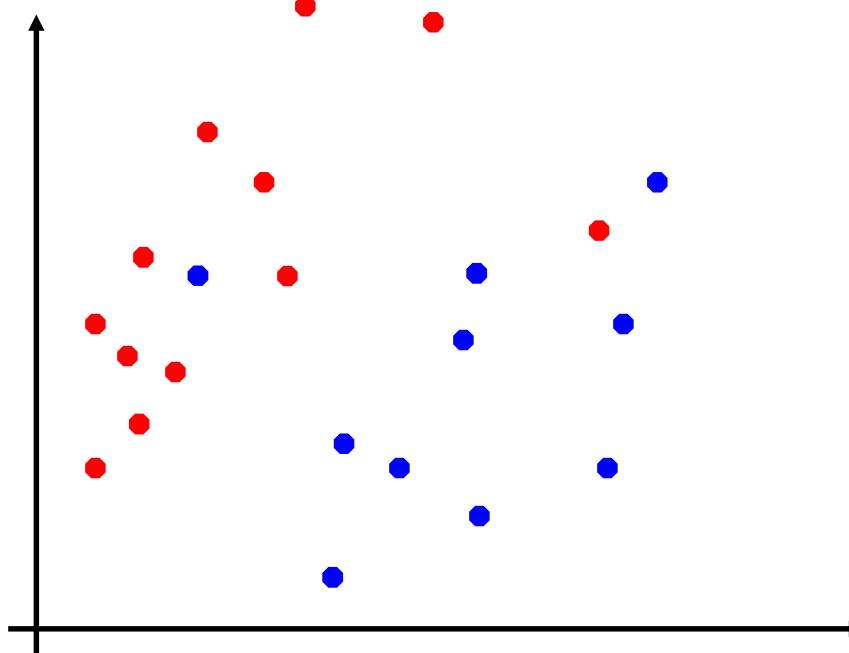
- Each non-zero α_i indicates that corresponding \mathbf{x}_i is a support vector.
- Then the classifying function is (note that we don't need \mathbf{w} explicitly):

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i – we will return to this later.

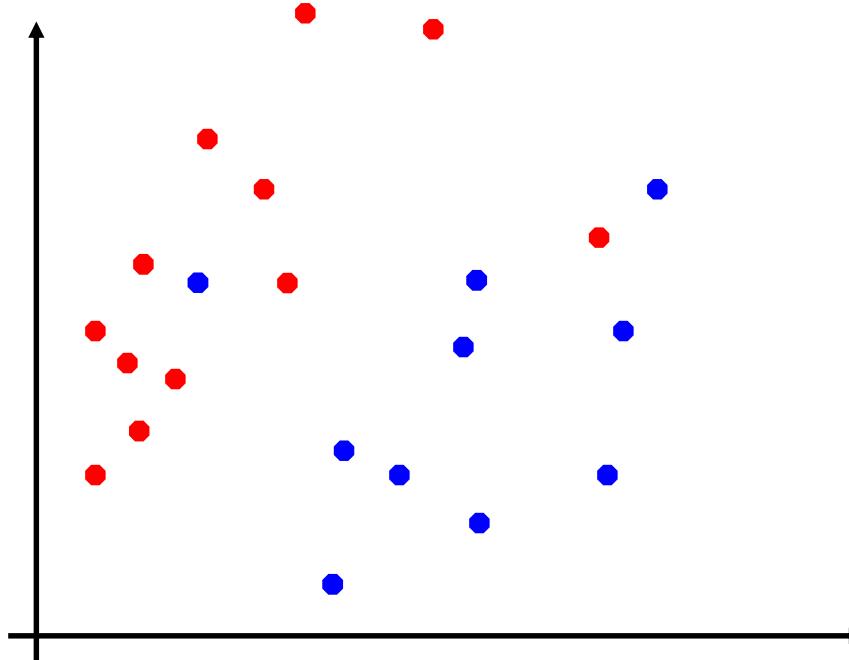
Soft Margin Classification

- What if the training set is not linearly separable?



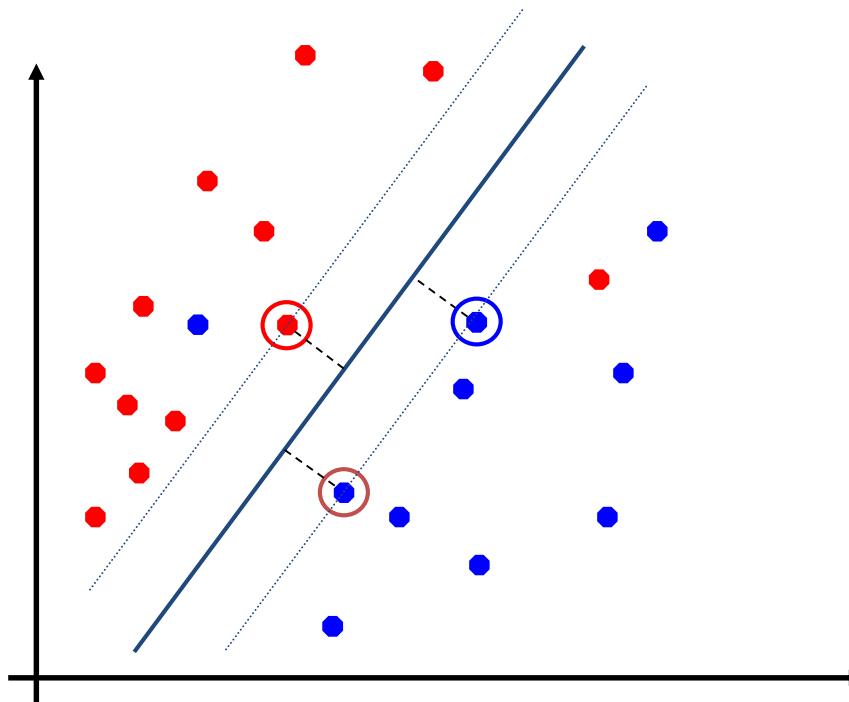
Soft Margin Classification

- What if the training set is not linearly separable?
- **Slack variables** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called **soft**.



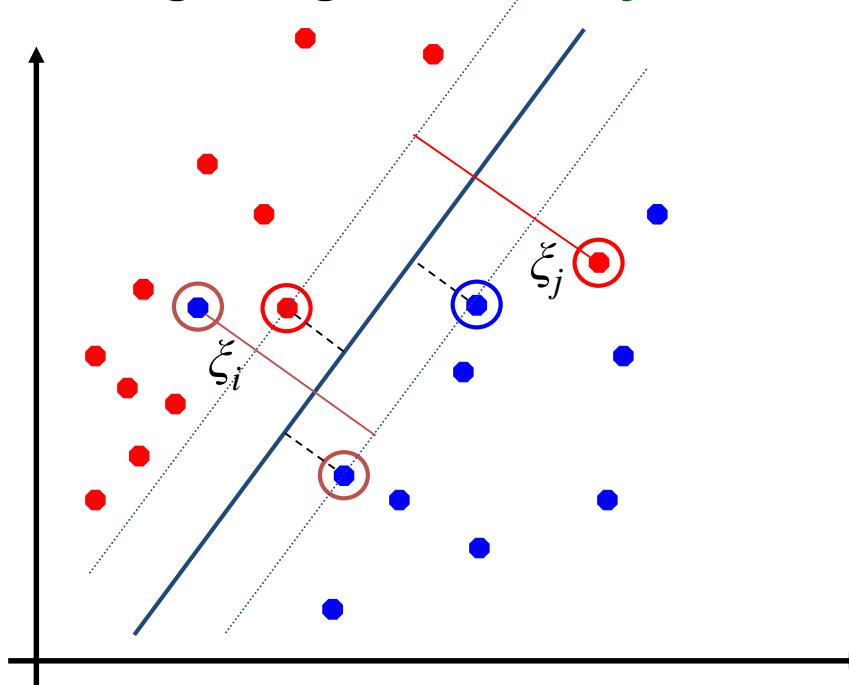
Soft Margin Classification

- What if the training set is not linearly separable?
- **Slack variables** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called **soft**.



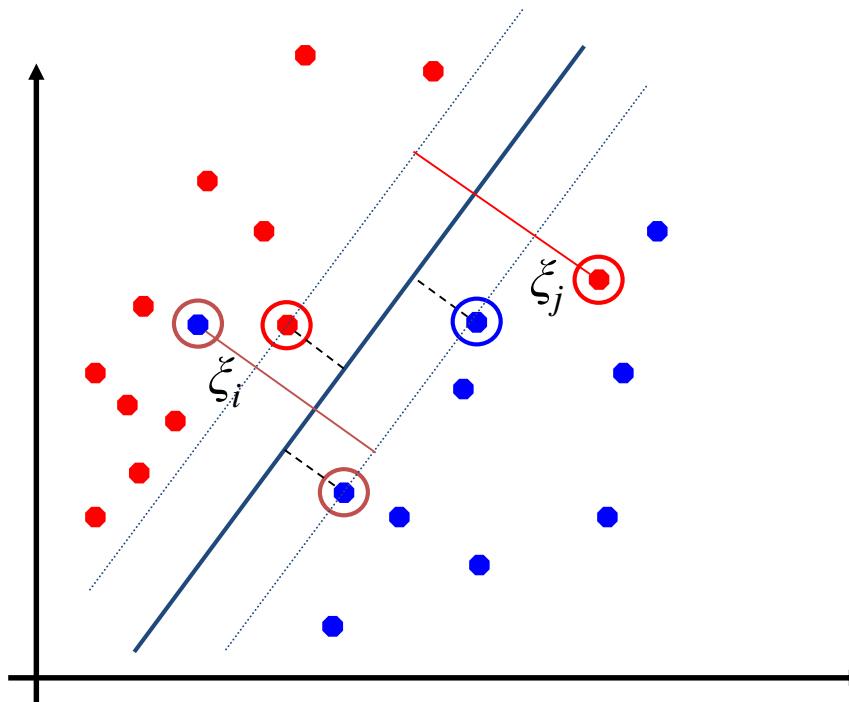
Soft Margin Classification

- What if the training set is not linearly separable?
- **Slack variables** ξ_i which measures the distance of the point to its marginal hyperplane if it is on the wrong side, otherwise 0, can be added to allow misclassification of difficult or noisy examples, resulting margin called **soft**.



Soft Margin Classification

- Applying Soft Margin, SVM tolerates a few dots to get misclassified and tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification.



Soft Margin Classification

Mathematically

- The old formulation:

Find w and b such that

$$\Phi(w) = \frac{1}{2} \|w\|^2 \text{ is minimized,}$$

and for all $(x_i, y_i), i = 1..n$: $y_i(w^T x_i + b) \geq 1$

- Modified formulation incorporates slack variables:

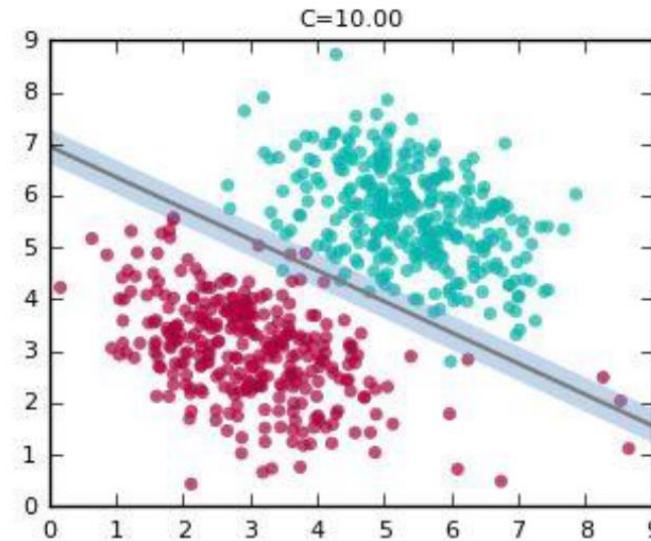
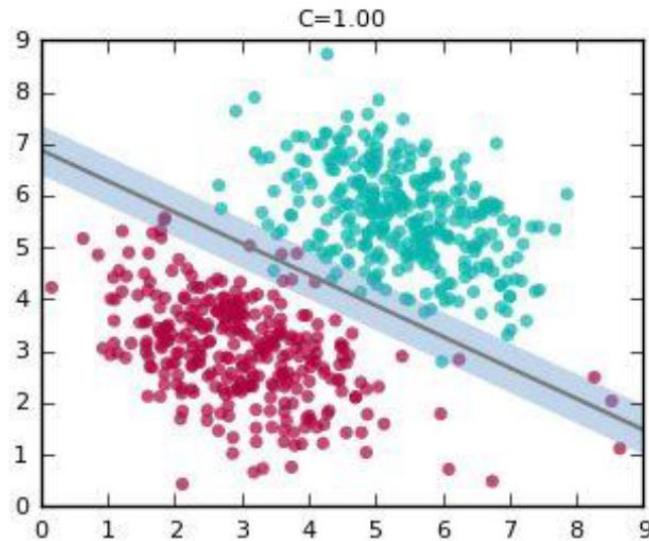
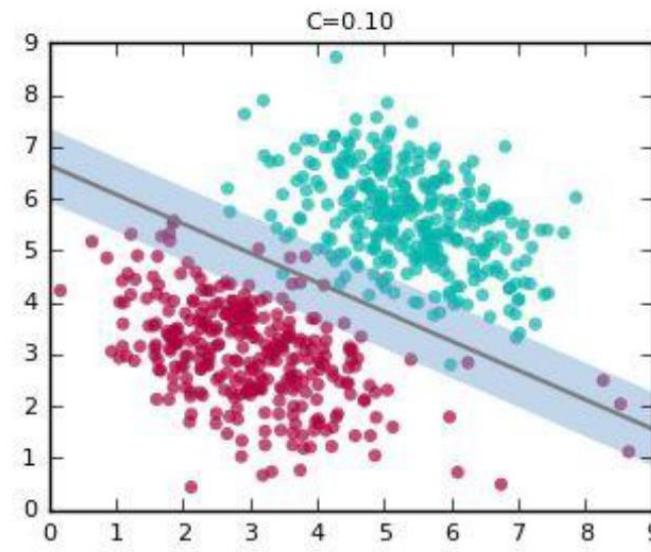
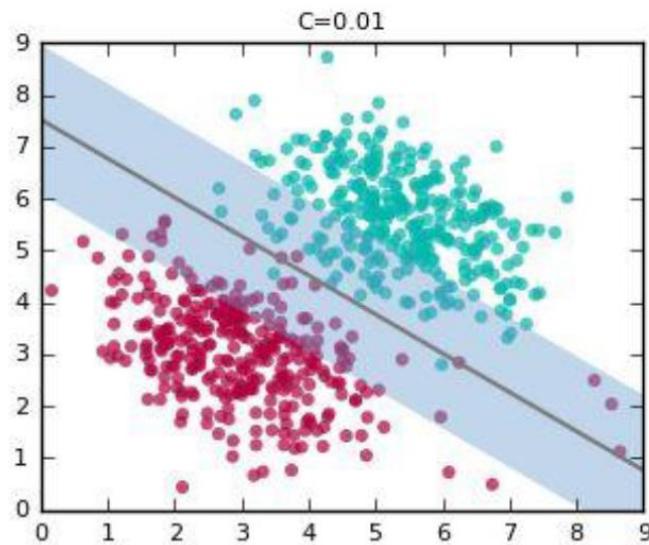
Find w and b such that

$$\Phi(w) = \frac{1}{2} \|w\|^2 + C \sum \xi_i \text{ is minimized,}$$

and for all $(x_i, y_i), i = 1..n$: $y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i > 0$

- Parameter C can be viewed as a way to control overfitting: it “trades off” the relative importance of maximizing the margin and fitting the training data.

SVM Parameter Tuning



Soft Margin Classification

Solution

- Dual problem:

Find α_i such that

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum \alpha_i (y_i (w^T \cdot x_i + b) - \xi_i), \xi_i > 0 \text{ is maximized for all } 0 \leq \alpha_i \leq C$$

- Again, x_i with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$$w = \sum \alpha_i y_i x_i, \quad b = y_j (1 - \xi_j) - \sum \alpha_i y_i x_i^T x_j, \quad \text{for any } \alpha_j > 0$$

- Again, we do not need to compute w explicitly for classification:

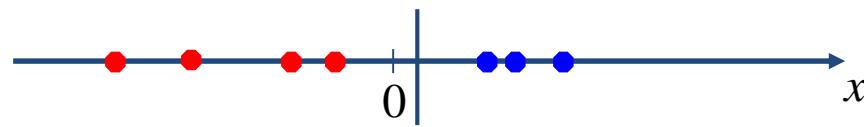
$$f(x) = \sum \alpha_i y_i x_i^T x + b$$

Linear SVM - Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i .

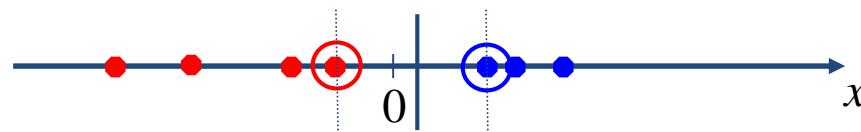
Non-Linear SVM

- Datasets that are linearly separable with some noise work out great:



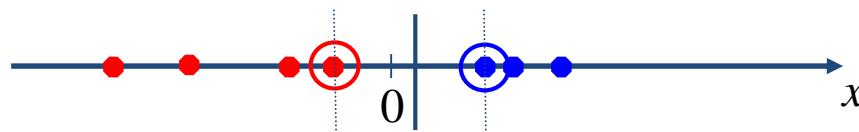
Non-Linear SVM

- Datasets that are linearly separable with some noise work out great:



Non-Linear SVM

- Datasets that are linearly separable with some noise work out great:

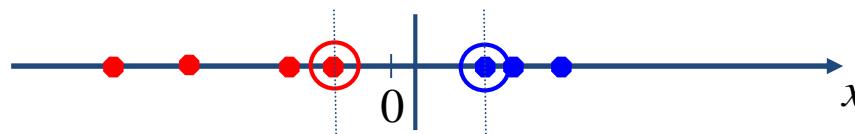


- But what are we going to do if the dataset is just too hard?



Non-Linear SVM

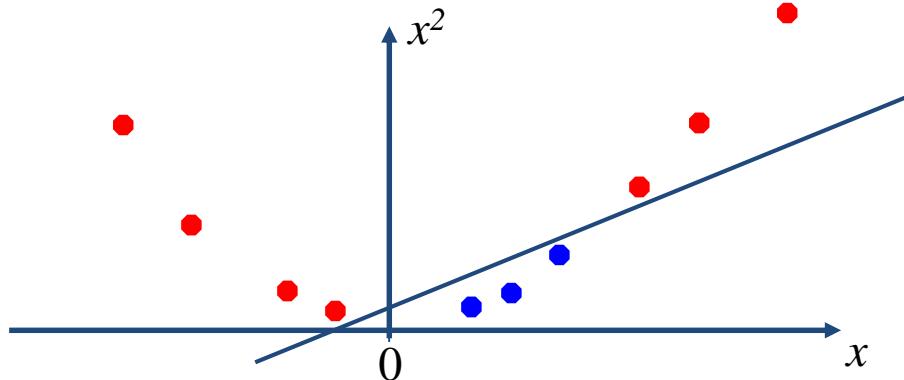
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

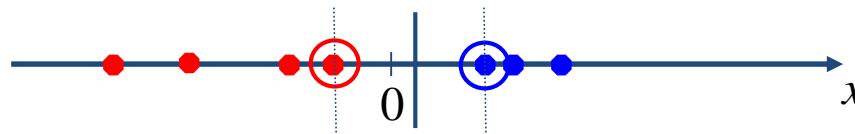


- How about... mapping data to a higher-dimensional space:



Non-Linear SVM

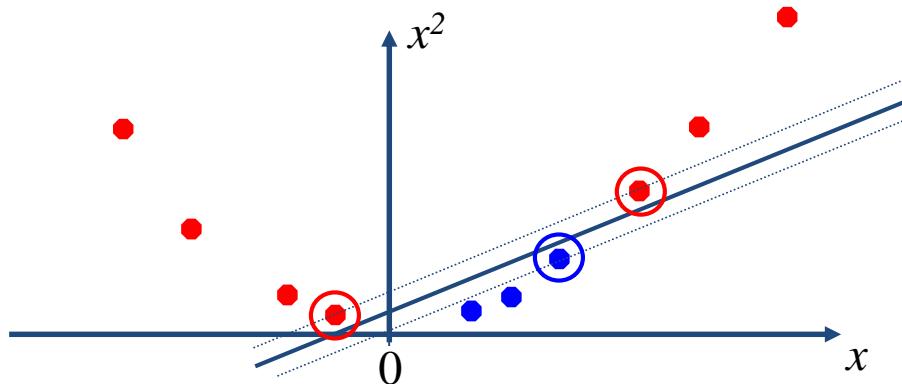
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

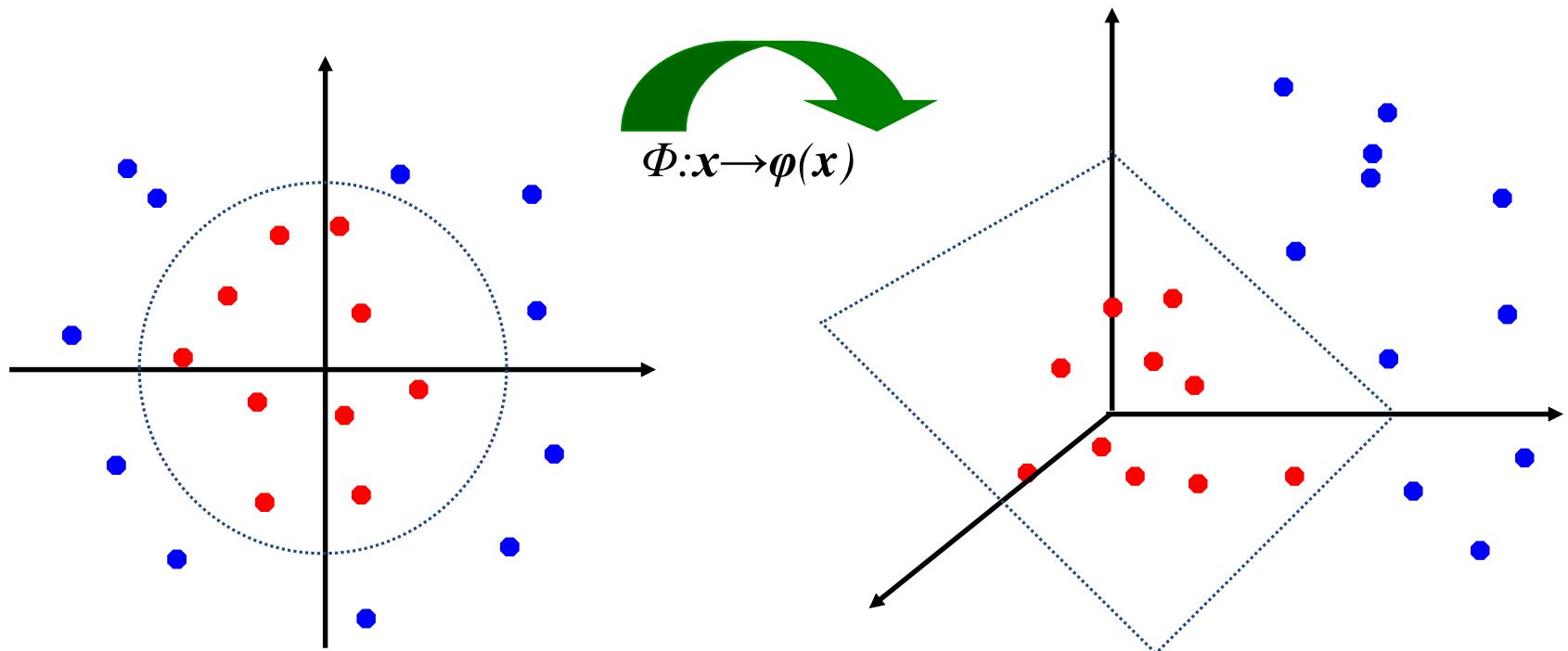


- How about... mapping data to a higher-dimensional space:



Non-Linear SVM: Feature Spaces

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



The “Kernel Trick”

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

- A ***kernel function*** is a function that is equivalent to an inner product in some feature space.
- Thus, a kernel function ***implicitly*** maps data to a high-dimensional space (without the need to compute each $\varphi(\mathbf{x})$ explicitly).

Kernel Functions

- What functions are kernel functions?
 - For some functions $K(\mathbf{x}_i, \mathbf{x}_j)$ checking that $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$ can be cumbersome.

Kernel Functions

- What functions are kernel functions?
 - For some functions $K(\mathbf{x}_i, \mathbf{x}_j)$ checking that $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$ can be cumbersome.
- Can we use any function?
 - No! A function $K(\mathbf{x}_i, \mathbf{x}_j)$ is a valid kernel if it corresponds to an inner product in some (perhaps infinite dimensional) feature space.
- Mercer's theorem
 - *Every semi-positive definite symmetric function is a kernel.*



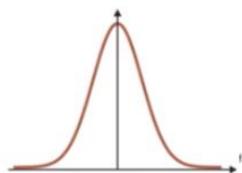
Examples of Kernel Function

$K(x_i, x_j) = (x_i \cdot x_j + 1)^p$; polynomial kernel. (1.22)

$K(x_i, x_j) = e^{\frac{-1}{2\sigma^2} (x_i - x_j)^2}$; Gaussian kernel; Special case of Radial Basis Function.

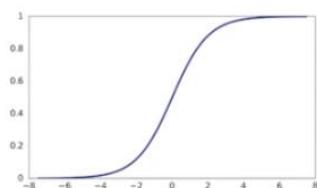
$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}$; RBF Kernel

$K(x_i, x_j) = \tanh(\eta x_i \cdot x_j + \nu)$; Sigmoid Kernel; Activation function for NN.



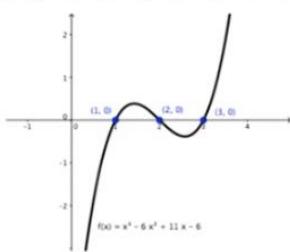
Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$$



Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

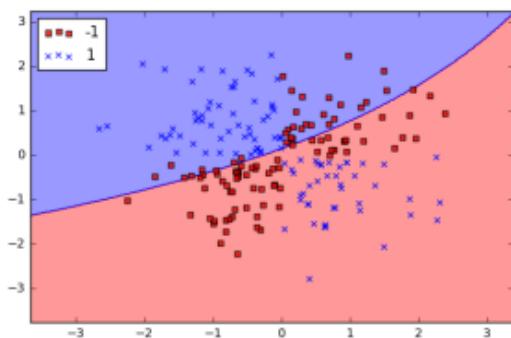


Polynomial Kernel

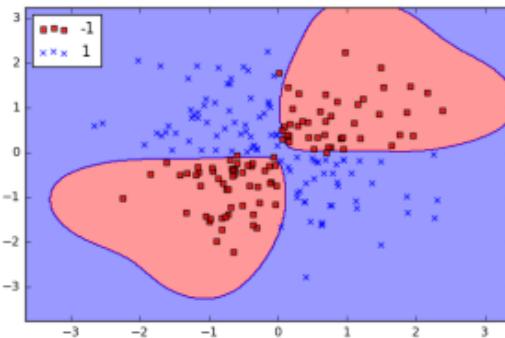
$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

SVM Parameter Tuning

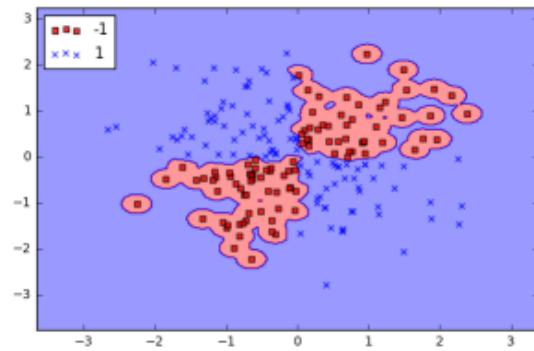
- The gamma parameter is the inverse of the standard deviation of the RBF kernel (Gaussian function), which is used as similarity measure between two points.
- Small gamma, large variance and vice versa.
- Large may cause complicated decision boundary or give rise to over-fitting.



$\gamma=0.01$



$\gamma=1$



$\gamma=100$

Non-Linear SVM

Mathematically

- Dual problem formulation:

Find α_i such that

$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum \alpha_i (y_i (w^T \cdot \varphi(x_i) + b) - 1)$ is maximized for all $\alpha_i \geq 0$

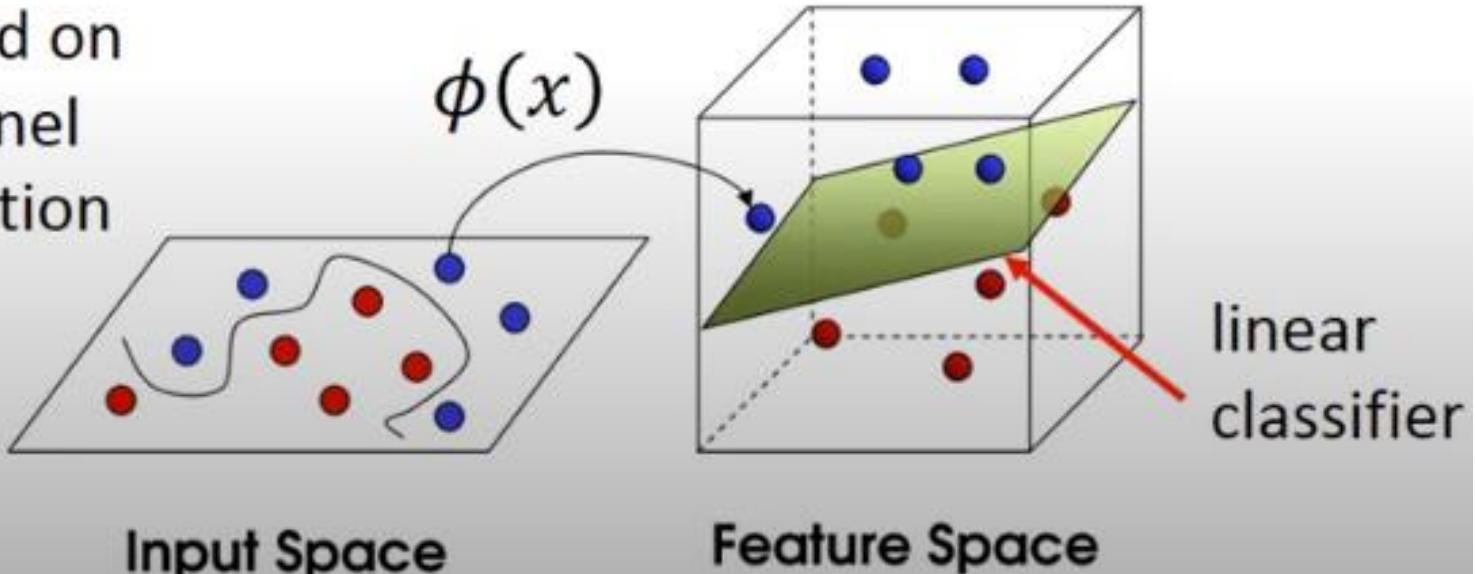
- The solution is:

$$f(x) = \sum \alpha_i y_i K(x_i, x) + b$$

- Optimization for finding α_i remains the same!

SVM and Kernel Methods

Based on
kernel
function



- Mapping data points from low dimensional space to a higher dimensional space can make it possible to apply SVM even for non-linear data sample.
- We don't need to know the mapping function itself, as long as we know the Kernel function (***Kernel Trick***)
- How the tuning parameter gamma can lead to over fitting or bias in RBF kernel.

Multi-Class Classification

- Some algorithms are designed for binary classification problems:
 - Logistic Regression
 - Perceptron
 - Support Vector Machines
- Instead, heuristic methods can be used to split a multi-class classification problem into multiple binary classification datasets and train a binary classification model each
 - One-vs-All (OVA)
 - One-vs-One (OvO)

One-vs-All (OVA)

- Every class is paired with the remaining classes
- The base classifier needs to produce a real-valued confidence score for its decision, rather than just a class label
- Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score

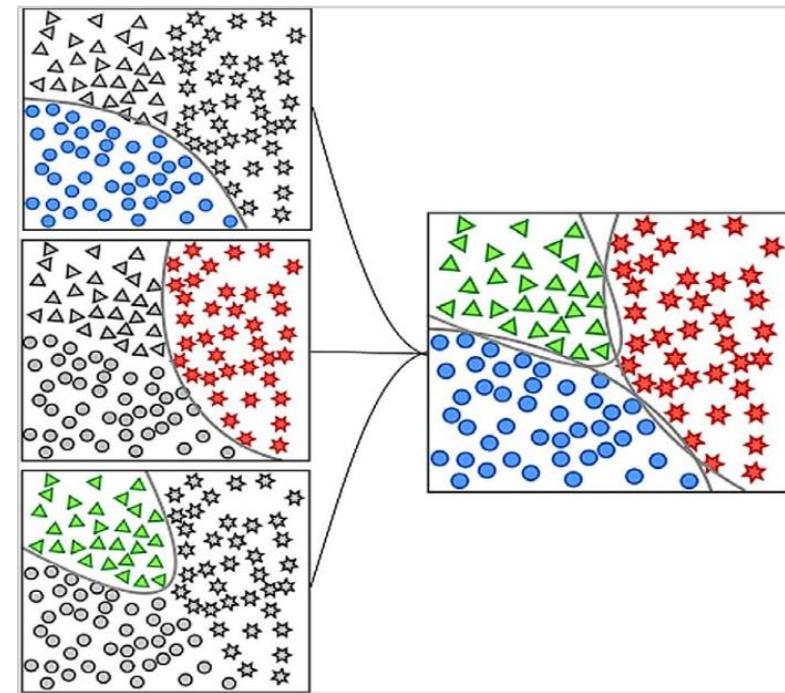


Figure 3. One vs all Strategy using 3 class problem

One-vs-One (OVO)

- Every class is paired with every other class
- At prediction time, a voting scheme is applied: all classifiers are applied to an unseen sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier

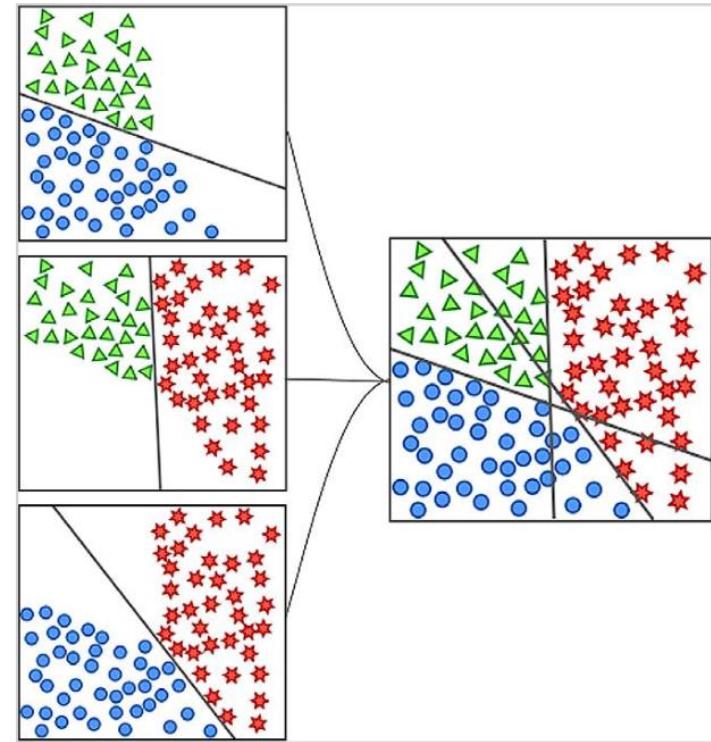


Figure 2. One vs one strategy using 3 class problem

SVM Software and Resources

- <http://www.svms.org/tutorials/>
- <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
 - LIBSVM -- A Library for Support Vector Machines by Chih-Chung Chang and Chih-Jen Lin



University of
Nottingham
UK | CHINA | MALAYSIA

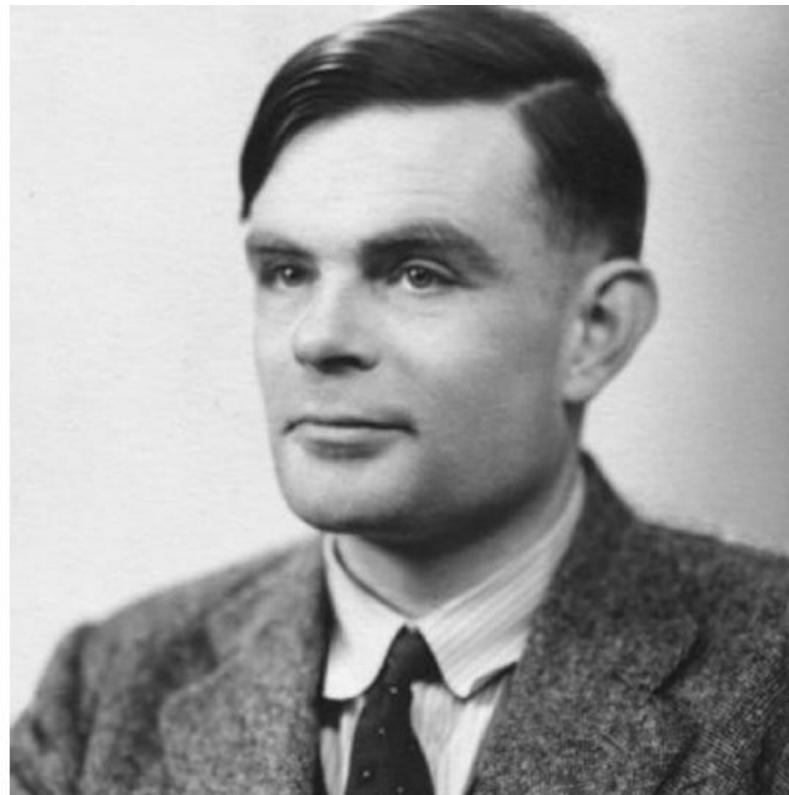
COMP3055

Machine Learning

Topic 14 – Deep Learning - Basics

Zheng Lu
2024 Autumn

History of Deep Learning Ideas



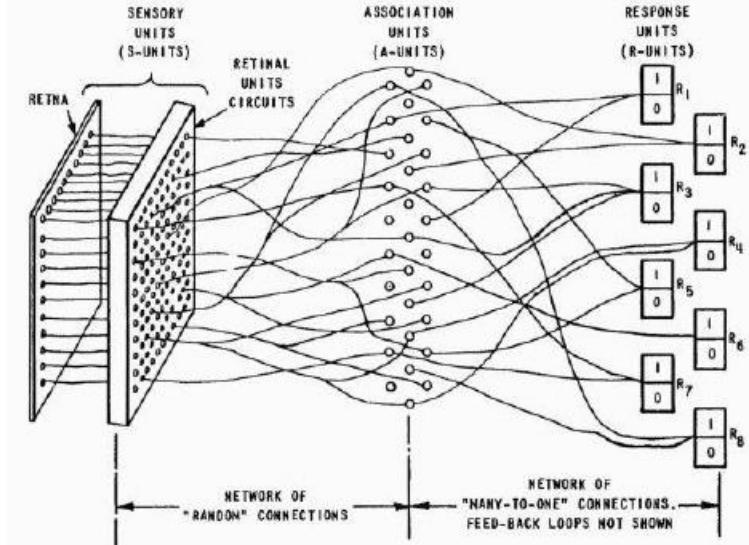
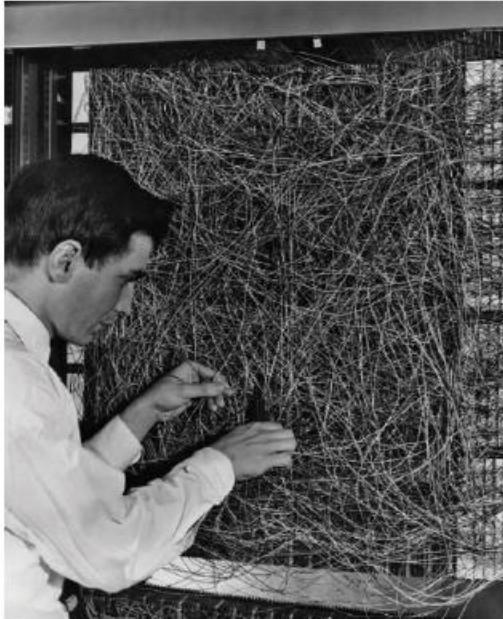
Dreams, mathematical foundations, and engineering in reality.

Alan Turing, 1951: "It seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers. They would be able to converse with each other to sharpen their wits. At some stage therefore, we should have to expect the machines to take control."

Milestones

- 1943: Neural networks
- 1957-62: Perceptron
- 1974-86: Backpropagation, RBM, RNN
- 1989-98: CNN, MNIST, LSTM, Bidirectional RNN
- 2006: “Deep Learning”, DBN
- 2009-12: ImageNet + AlexNet
- 2014: GANs
- 2016-17: AlphaGo, AlphaZero
- 2017-19: BERT, Transformers

Milestones



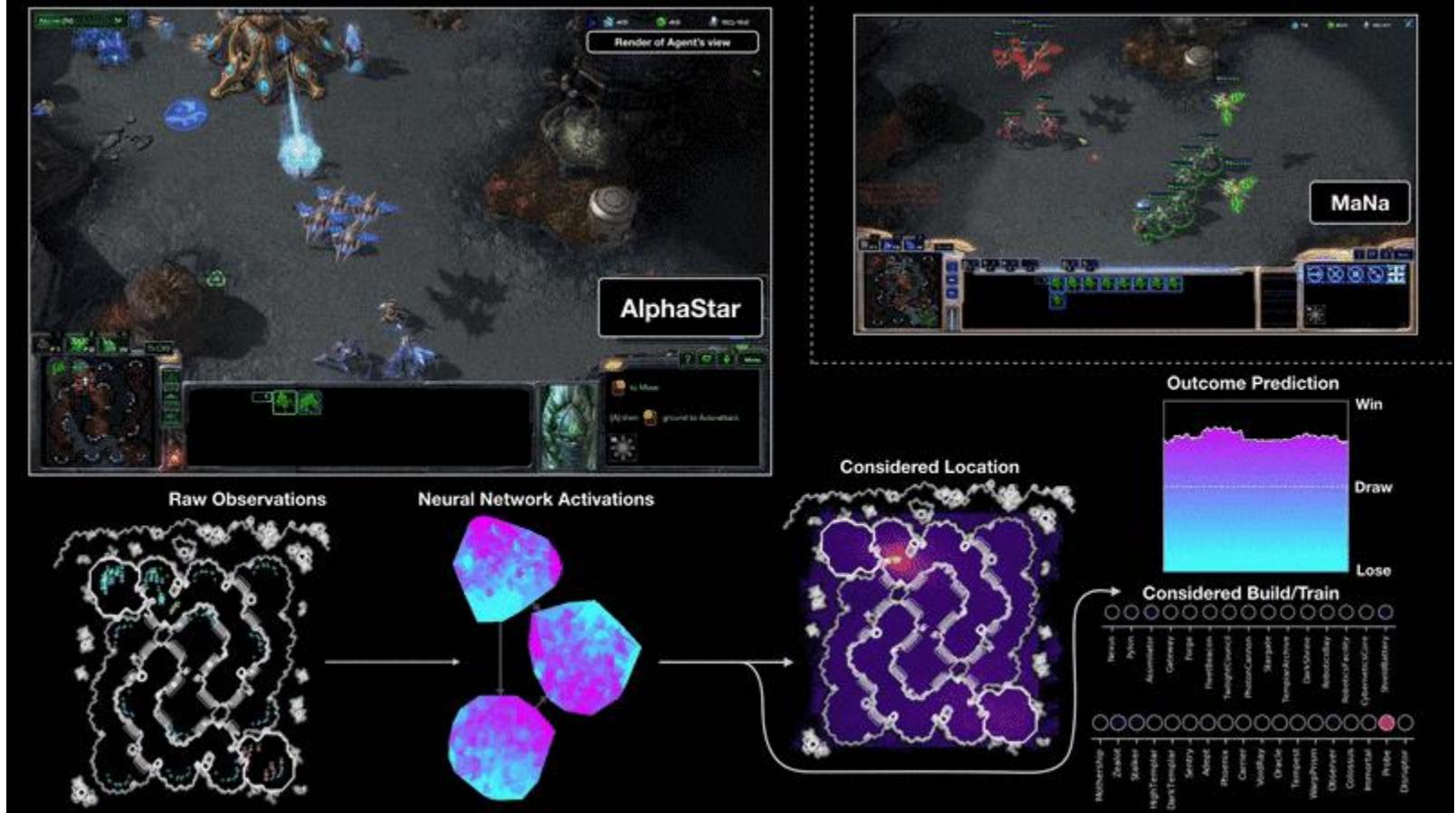
Frank Rosenblatt, Perceptron (1957, 1962): Early description and engineering of single-layer and multi-layer artificial neural networks.

Milestones



Lee Sedol vs AlphaGo, 2016

Milestones



- AlphaStar beats MaNa , one of the world's strongest professional StarCraft players, 5:0.

Turing Award for Deep Learning



- Yann LeCun
- Geoffrey Hinton
- Yoshua Bengio

Turing Award given for:

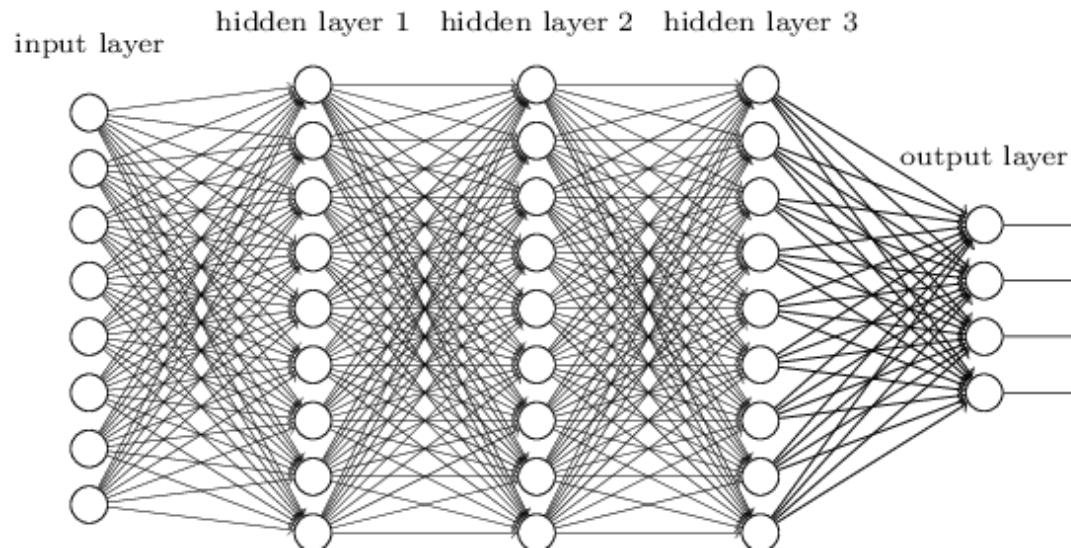
- “The conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.”

What is Deep Learning?

- Most machine learning methods work well because of human-designed input features or representations
 - Our job is to find the best features to send to learning techniques such as SVM.
- Machine learning becomes just optimizing weights to best make a final prediction.

What is Deep Learning?

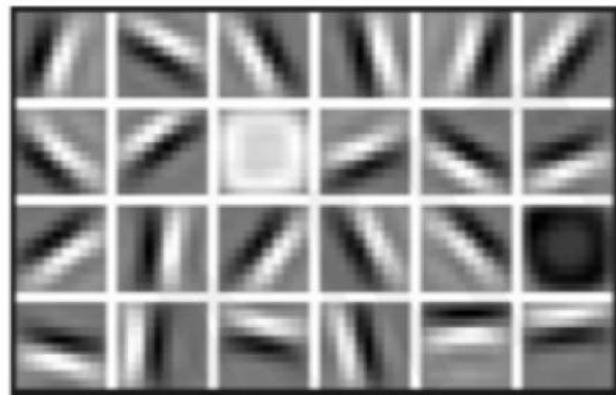
- In contrast to standard machine learning
- **Representation learning** attempts to automatically learn good features or representations
- **Deep learning algorithms** learn multiple levels of representations (here: hidden layer 1, 2, 3) and an output layer
- From “raw” inputs x (e.g. sound, pixels, characters, or words)
- **Neural networks** are the currently successful method for deep learning
- A.k.a. “**Differentiable Programming**”



Why Deep Learning?

- To learn the underlying features directly

Low Level Features



Lines & Edges

Mid Level Features



Eyes & Nose & Ears

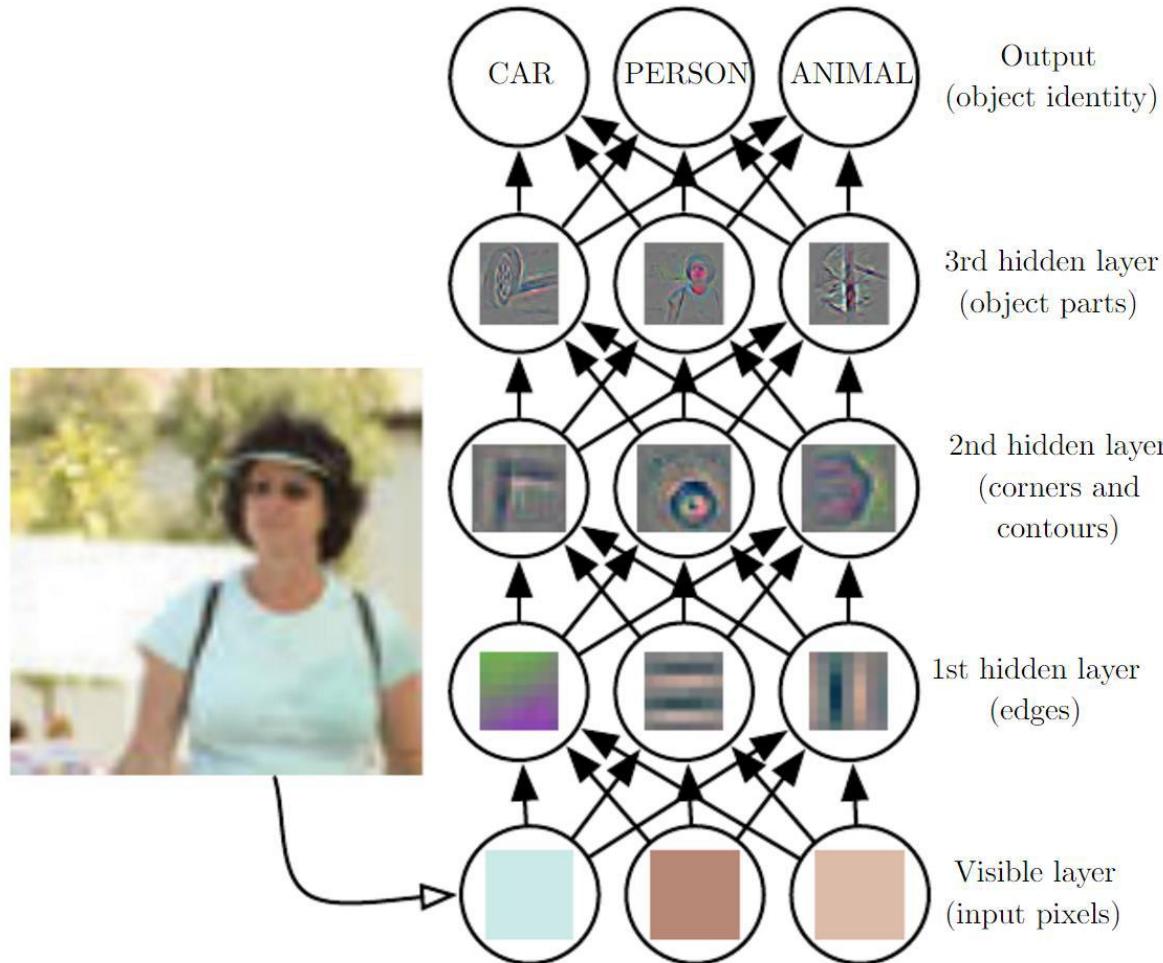
High Level Features



Facial Structure

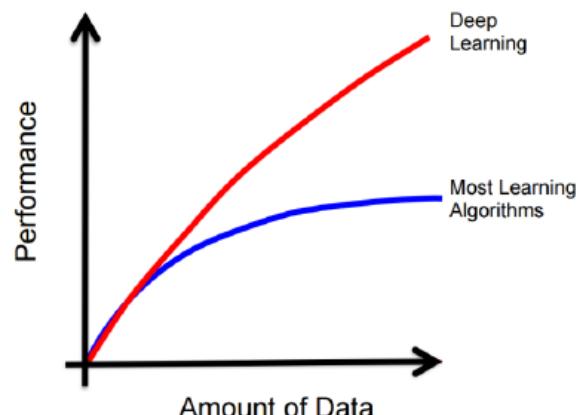
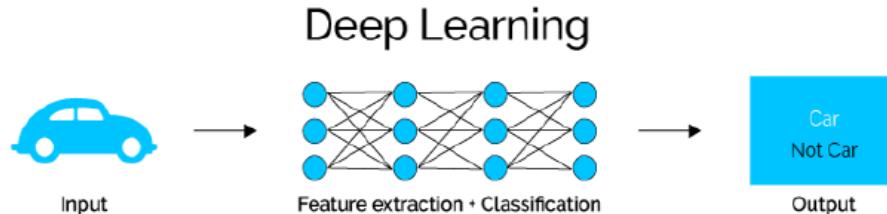
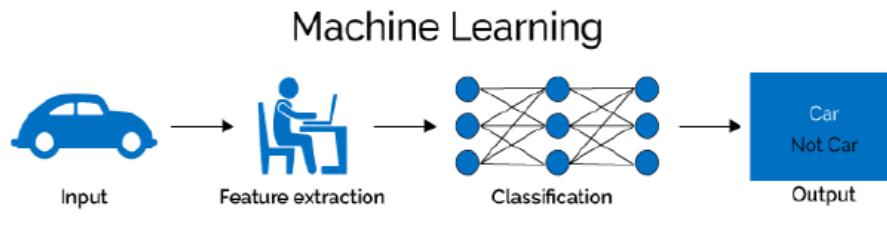
Why Deep Learning?

- To learn the underlying features directly



Why Deep Learning?

- Scalable machine learning



Why Now?

- Big data
 - Easier collection and storage
 - Larger datasets (ImageNet, COCO)



- Hardware
 - Graphics Processing Units (GPUs)
 - Massively parallelizable



- Software
 - Toolboxes
 - Powerful framework



Why Deep

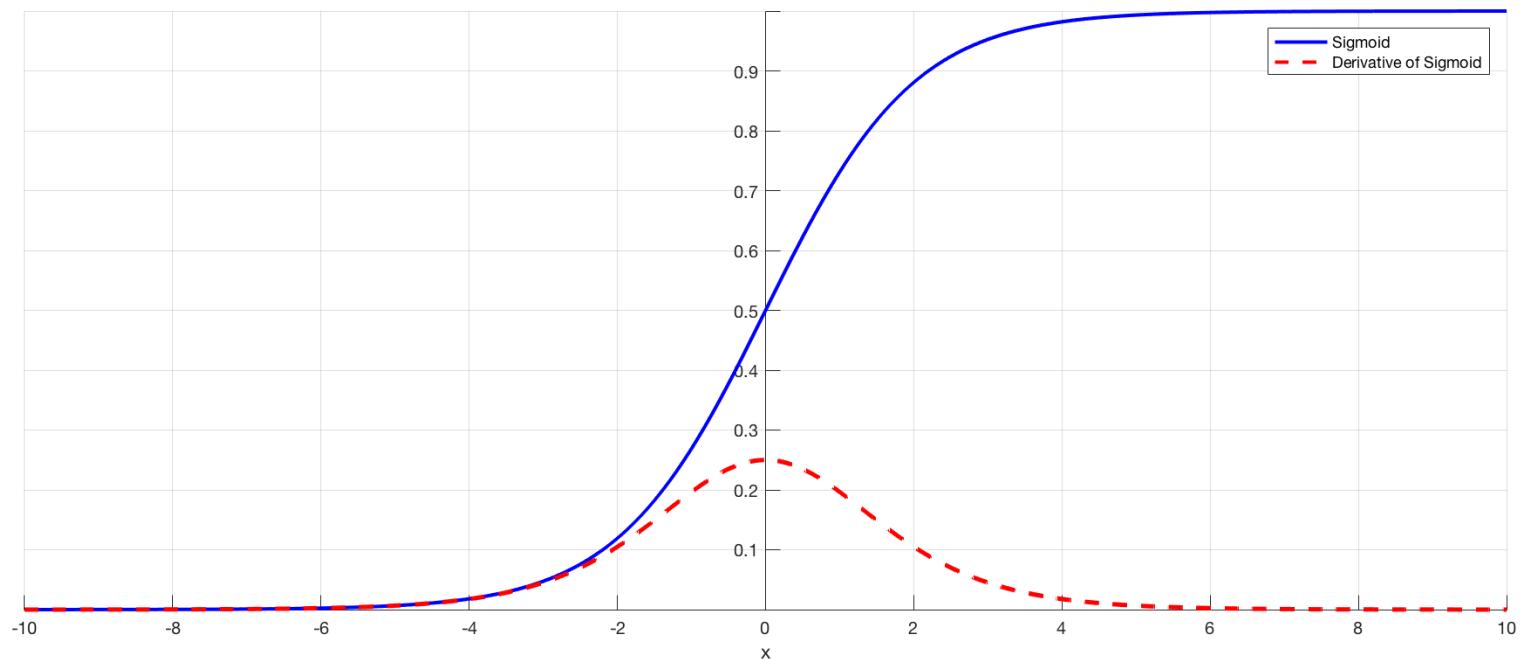
- More sophisticated models
 - Learn more good features (representations)
 - So better classification results



Problems with Going Deeper

- Vanishing gradients
 - As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train
 - Certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small

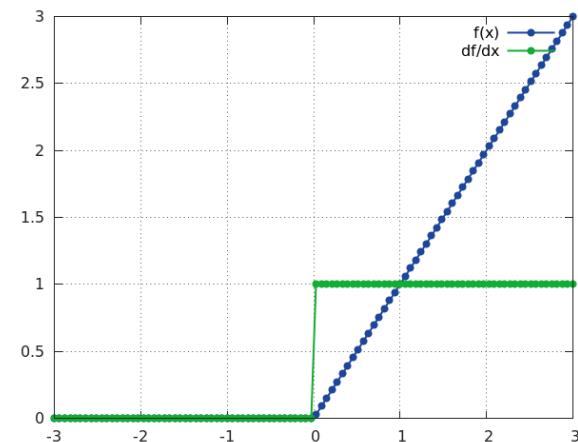
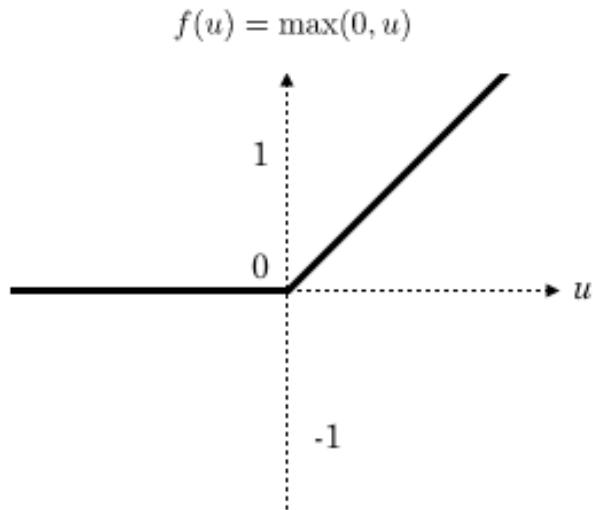
Vanishing Gradients



Vanishing Gradients

Use better activation function

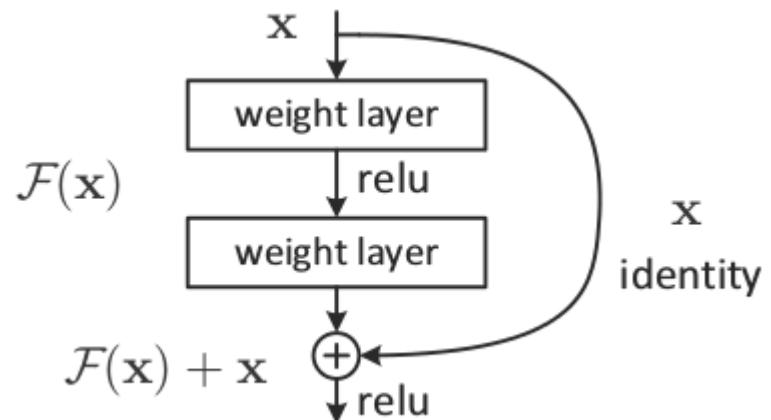
- Rectified Linear Units (ramp)
 - $f(x) = \max(0, x)$
 - Derivative: All in or all out (unit step)
 - $f'(x) = 1 \text{ if } x > 0 \text{ else } 0$
- Dead ReLUs
 - LeakyReLU: $f(x) = \max(x, 0.01x)$
 - PReLU: $f(x) = \max(x, ax)$



Vanishing Gradients

Use better architecture

- Residual networks
 - provide residual connections straight to earlier layers
 - This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block

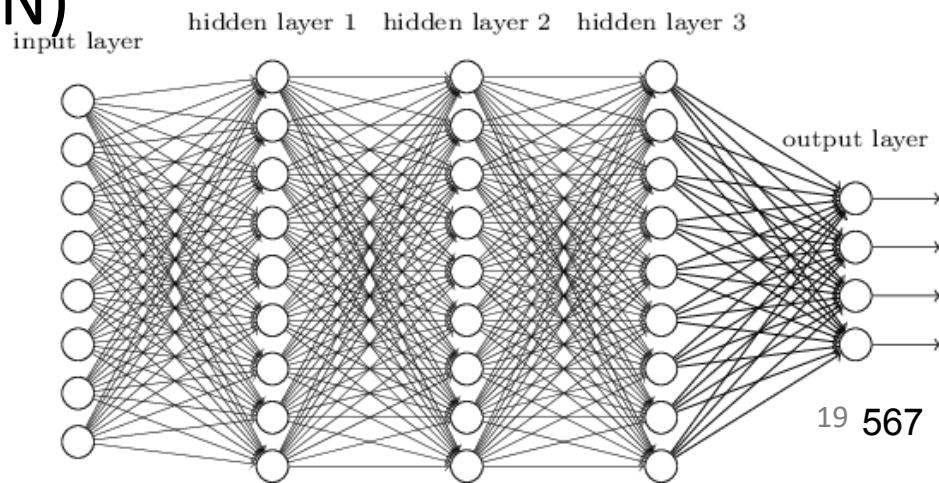


Use normalization

- Batch normalization

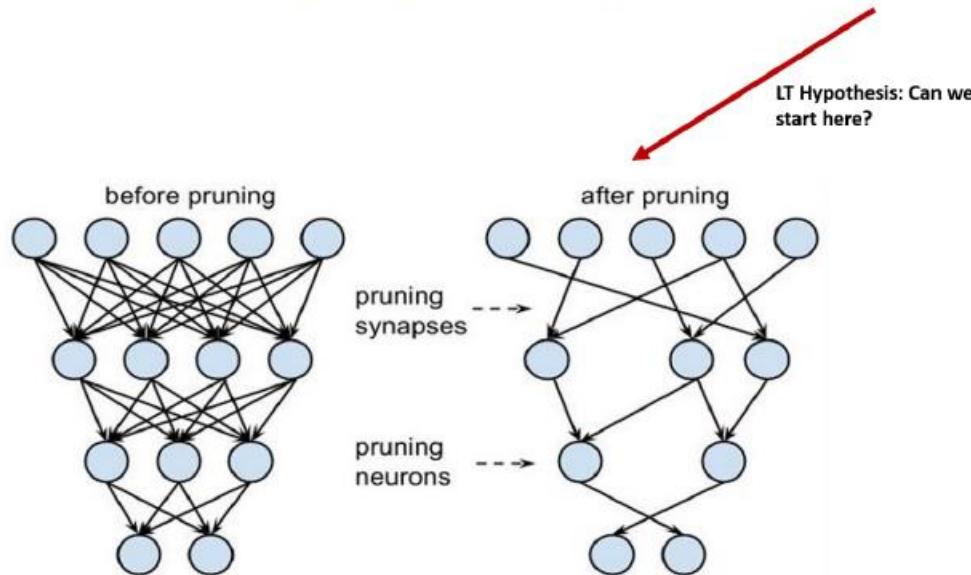
Problems with Going Deeper

- Parameter explosion
 - Too many weights to optimize as we go deeper
 - Search space is much harder to navigate
- Proposal: shared weights
 - Spatially shared (CNN)
 - Temporally shared (RNN)



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

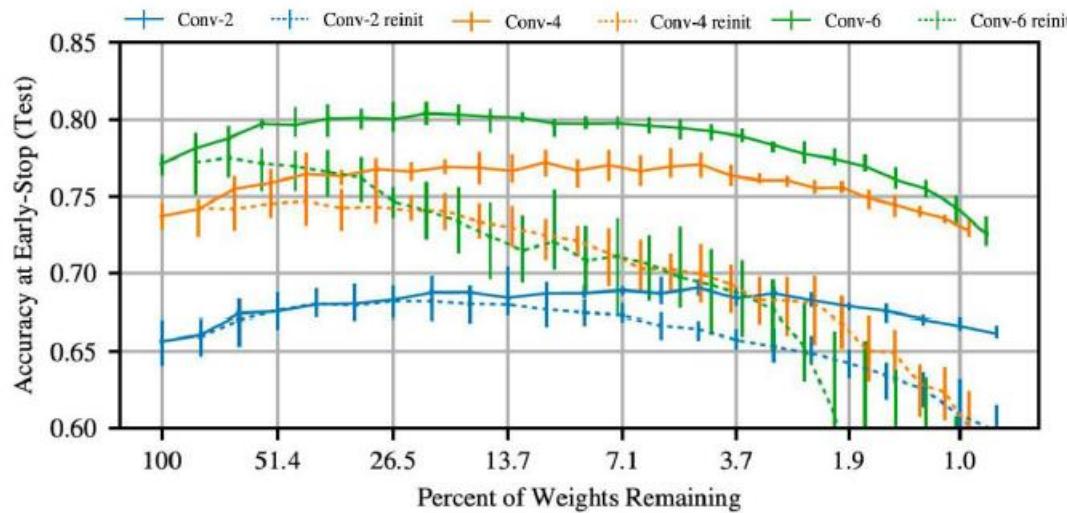
Frankle et al. (MIT) - Best Paper at ICLR 2019



1. Randomly initialize a neural network.
2. Train the network until it converges.
3. Prune a fraction of the network.
4. Reset the weights of the remaining network to initialization values from step 1
5. Train the pruned, untrained network. Observe convergence and accuracy.

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

Frankle et al. (MIT) - ICLR 2019 Best Paper

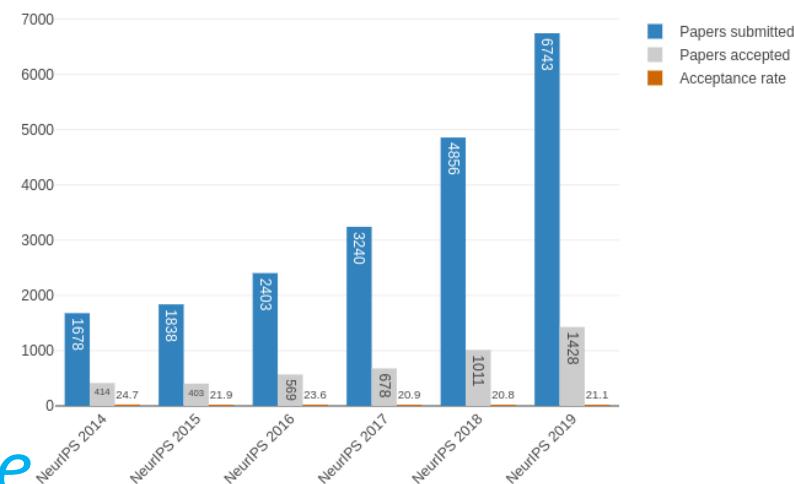


- **Idea:** For every neural network, there is a subnetwork that can achieve the same accuracy in isolation after training.
- **Iterative pruning:** Find this subset subset of nodes by iteratively training network, pruning its smallest-magnitude weights, and re-initializing the remaining connections to their original values. Iterative vs one-shot is key.
- **Inspiring takeaway:** There exist architectures that are much more efficient. Let's find them!

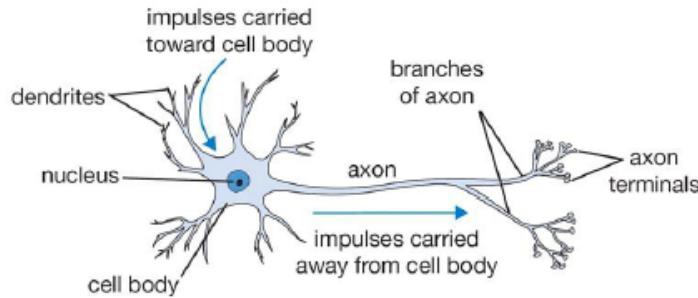
Deep Learning Future

- History of science is a story of both people and ideas
- Many brilliant people contributed to the development of AI
- *“The future depends on some graduate student who is deeply suspicious of everything I have said.”* - Geoffrey Hinton

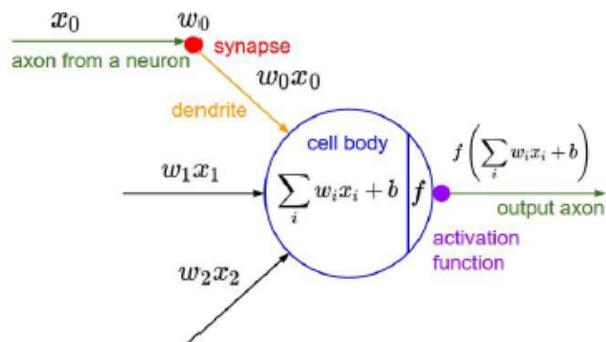
Statistics of acceptance rate NeurIPS



Biological and Artificial Neural Networks



- **Neuron:** computational building block for the brain



- **(Artificial) Neuron:** computational building block for the “neural network”

Key Difference:

- **Parameters:** Human brains have ~10,000,000 times synapses than artificial neural networks.
- **Topology:** Human brains have no “layers”. **Async:** The human brain works asynchronously, ANNs work synchronously.
- **Learning algorithm:** ANNs use gradient descent for learning. We don't know what human brains use
- **Power consumption:** Biological neural networks use very little power compared to artificial networks
- **Stages:** Biological networks usually never stop learning. ANNs first train then test.



University of
Nottingham
UK | CHINA | MALAYSIA

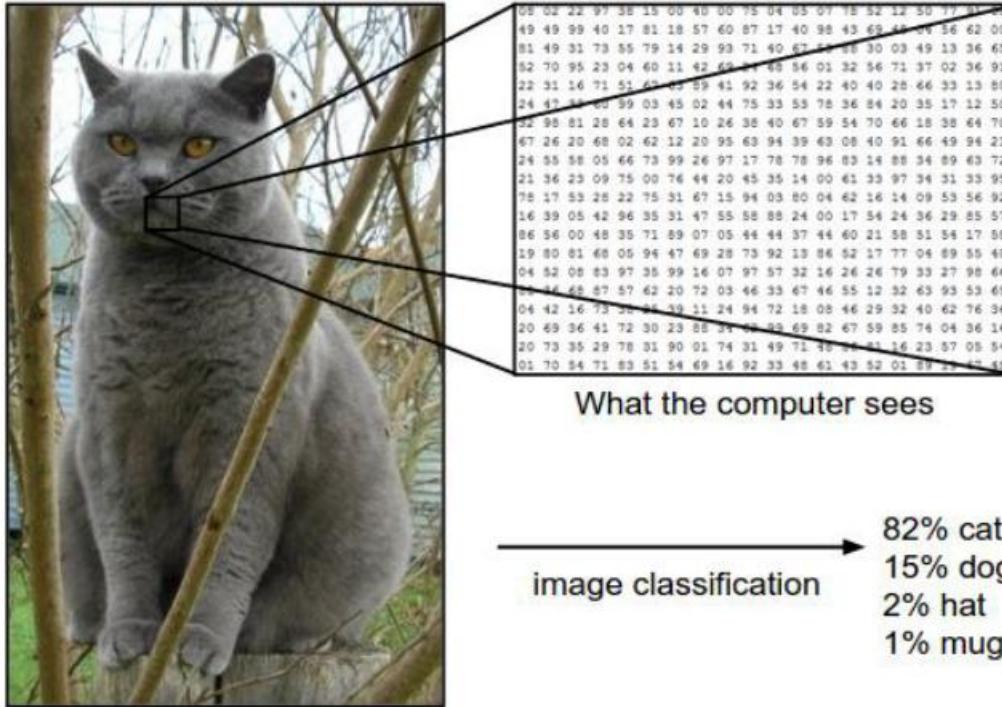
COMP3055

Machine Learning

Topic 15 – Convolutional Neural Network

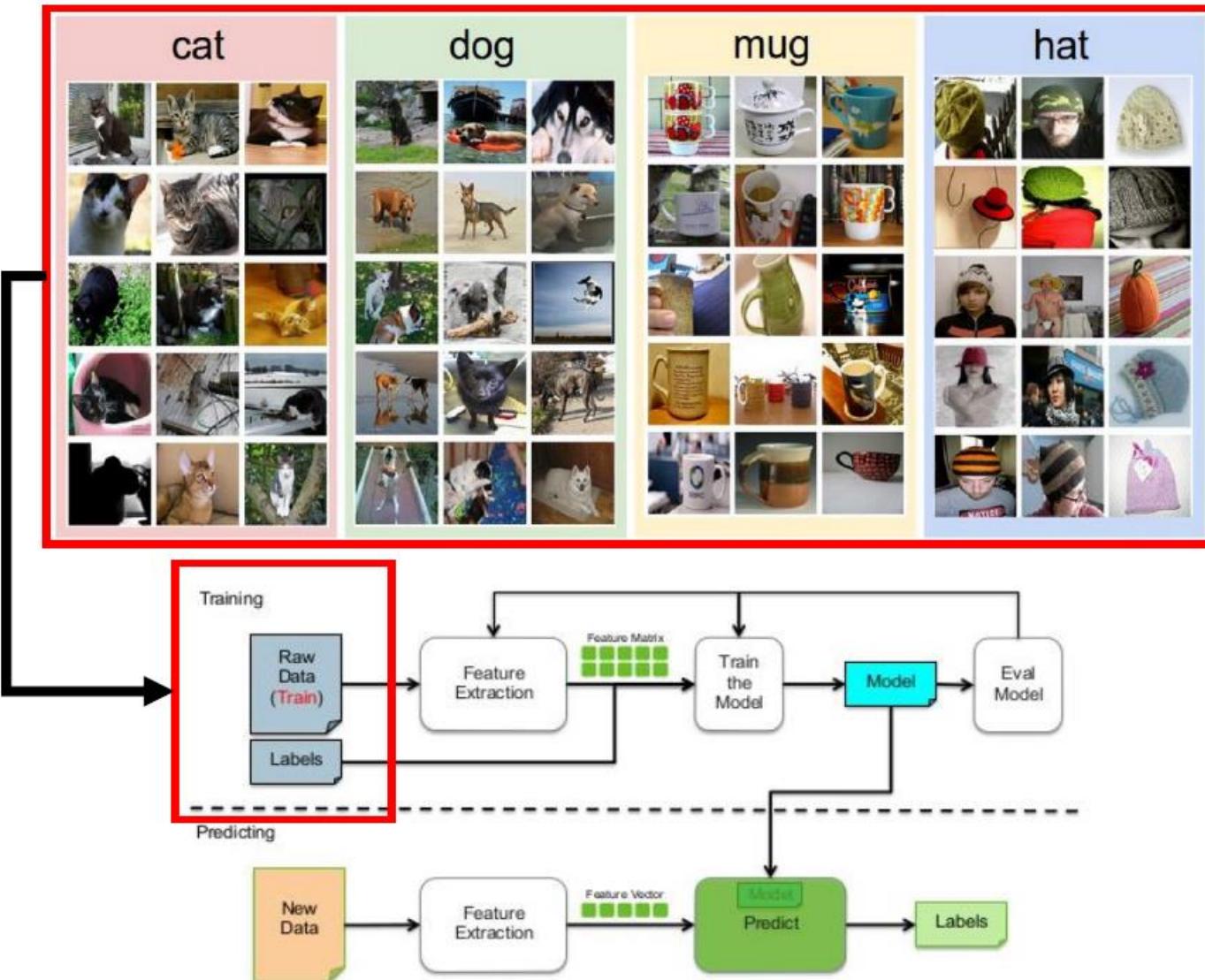
Zheng LU
2024 Autumn

Image are Numbers



- **Regression:** The output variable takes continuous values
- **Classification:** The output variable takes class labels
 - Underneath it may still produce continuous values such as probability of belonging to a particular class.

Image Classification Pipeline



Famous CV Datasets

931972451032437590349
302479483201353574685
282323824982913911199
636903603011393150496
338070569884144464533
179580437750542098124
9500511174772865189411
021617095632664715232
94321002084037936934
551662767566581687105
717592396304580040466
167964114131234815507
0161675556688177283765
646877130738691673648
797313279362492145038

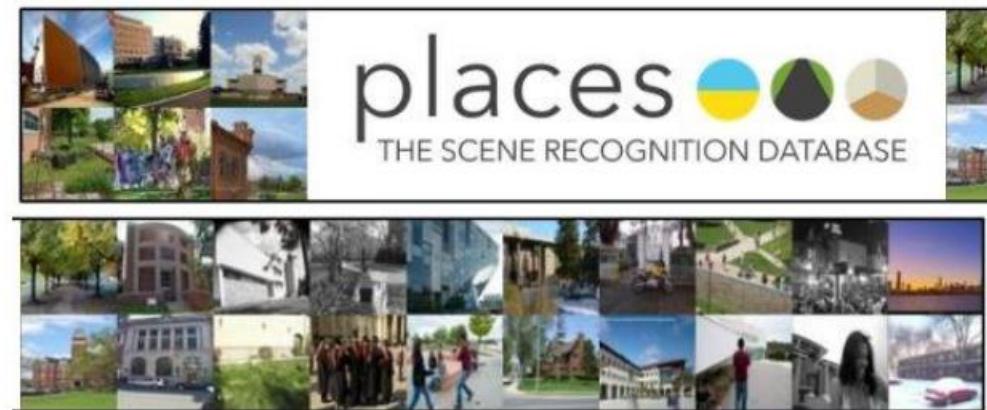
MNIST: handwritten digits



CIFAR-10(0): tiny images

offspring teacher computer printer housing animal weight drop headquarters album garage John Wayne television
register gallery court key structure light date spread king fireplace church press market lighte
resturant counts horse concert hotel road paper cup pack
sport screen wall means fan hill cabin camp fish, brown
sky plant wine fox tree tower railcar
bread table top man ear by study bird
cloud cover range leash net menu ball's button
spring range leash sun soft center seat
desert fruit dog shop train signs glass
bed kitchen train America box center goal
kitchen train America box center step
engine chain tea overall decide bar
student apple girl first bank home room office rule hall
dinner stone chair mining castle club
radio valley cross chair base library stage video food building
beach Support level line street golf
base football hospital match material player log shirt desk vehicle
short circuit bridge scale equipment cell phone mountain telephone
circuit breaker equipment go pad microphone recording crowd

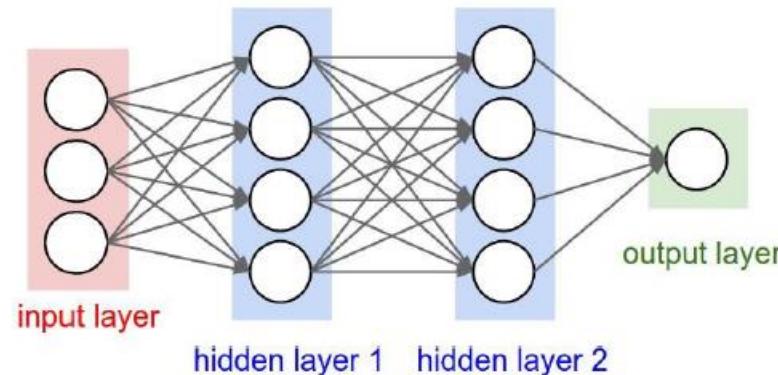
ImageNet: WordNet hierarchy



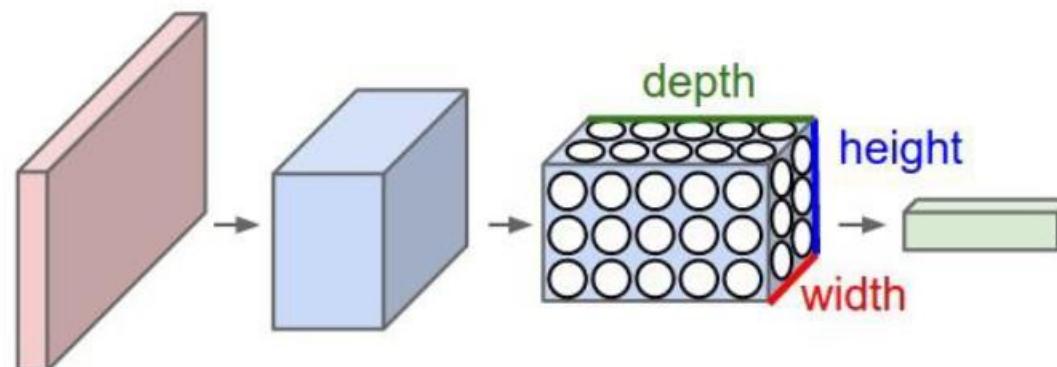
Places: natural scenes

Convolutional Neural Network

Regular neural network (fully connected):



Convolutional neural network:



Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

Why CNN for Image

- Some patterns are much smaller than the whole image

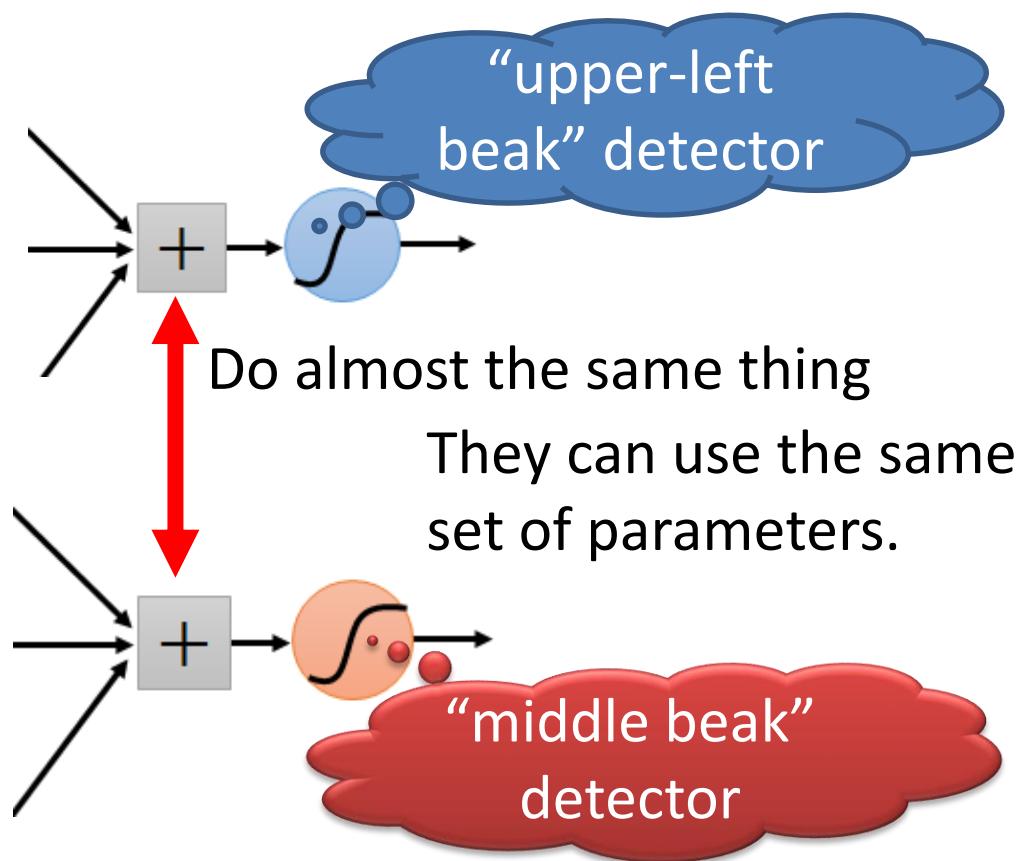
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters



Why CNN for Image

- The same patterns appear in different regions.



Why CNN for Image

- Downsampling the pixels will not change the object



downsampling



We can downsample the pixels to make image smaller

→ Less parameters for the network to process the image

CNN Layers

- **INPUT**: will hold the raw data (e.g., pixel values of the image)
- **CONV**: will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume
- **RELU**: will apply an elementwise activation function
- **POOL**: will perform a downsampling operation along the spatial dimensions (width, height)
- **FC**: will compute the class scores

Convolution Layer

Those are the network parameters to be learned.

| | | | | | |
|---|---|---|---|---|---|
| 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

⋮

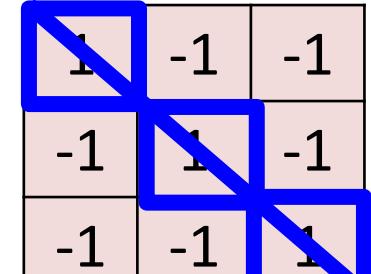
Each filter detects a small pattern (3 x 3).

CNN – Convolution

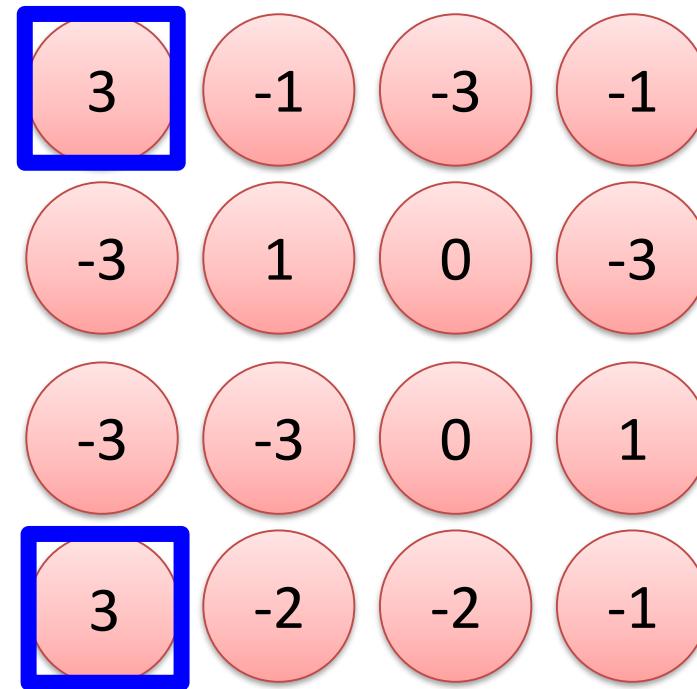
stride=1

| | | | | | |
|---|---|---|---|---|---|
| 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



Filter 1



CNN – Convolution

| | | |
|----|---|----|
| -1 | 1 | -1 |
| -1 | 1 | -1 |
| -1 | 1 | -1 |

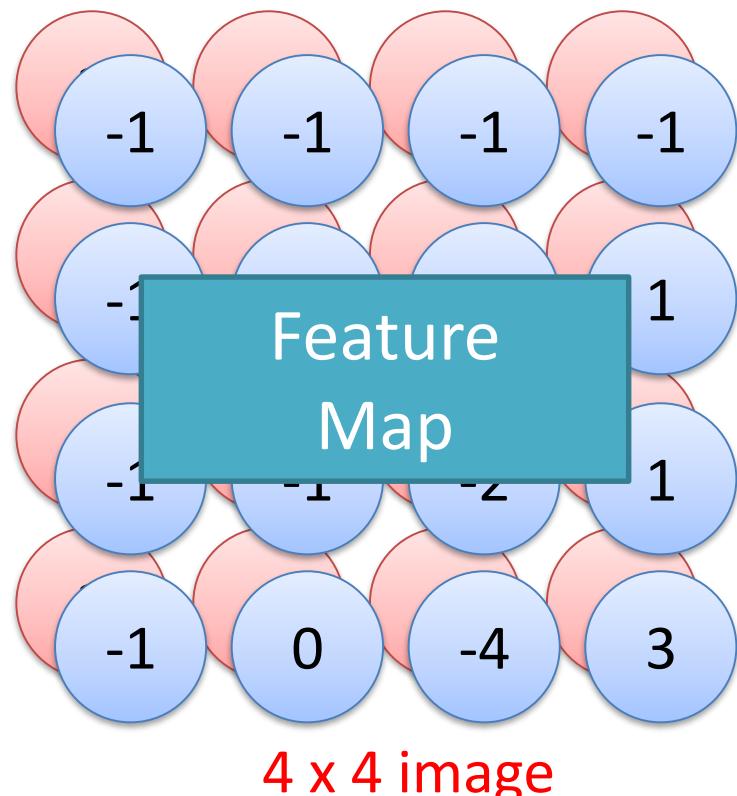
Filter 2

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 |

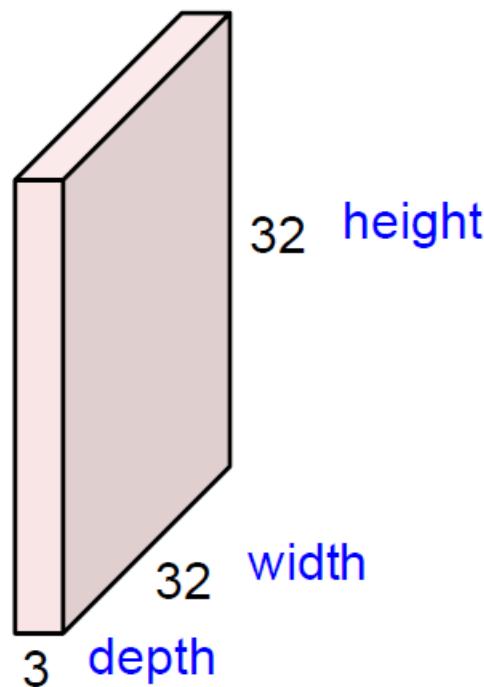
6 x 6 image

Do the same process for
every filter

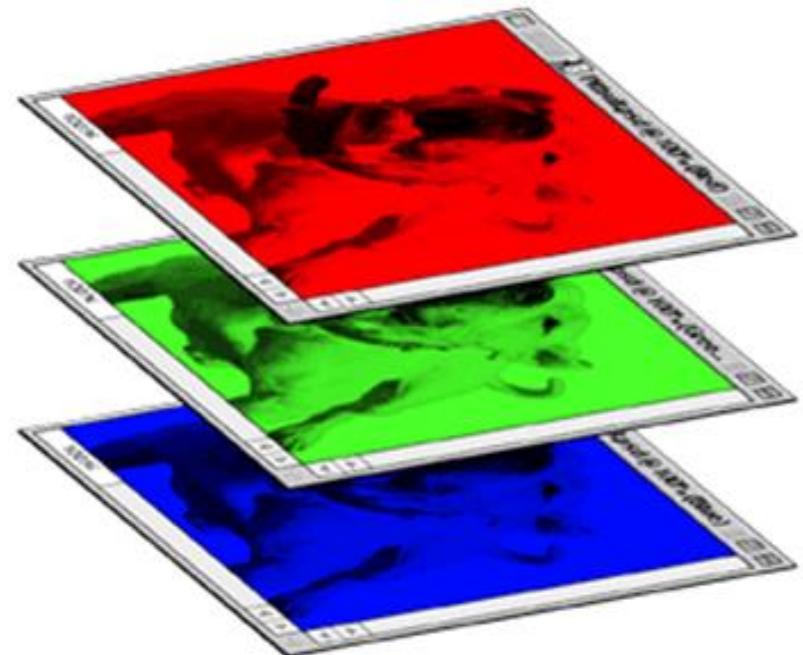


Convolution Layer

32x32x3 image -> preserve spatial structure

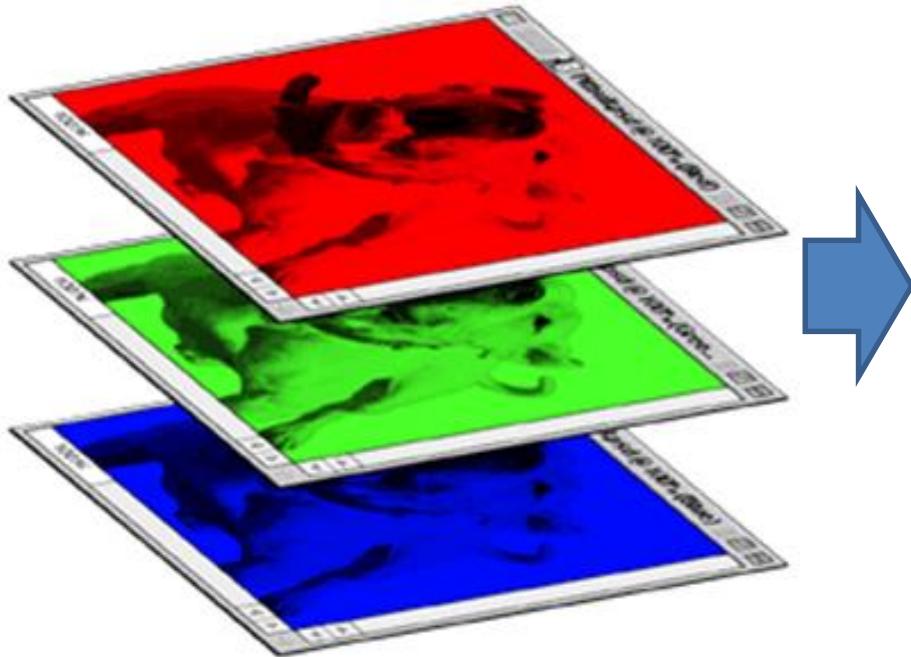


Colorful image



Convolution Layer

Colorful image



| | | |
|----|----|----|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

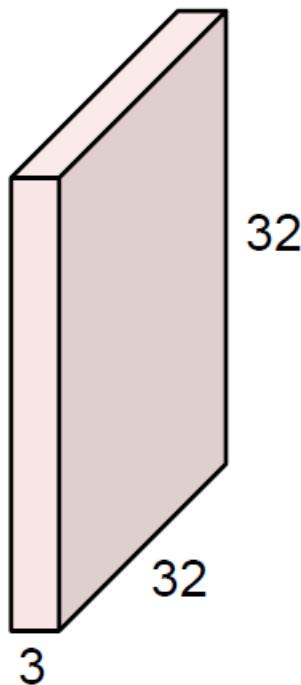
| | | |
|----|---|----|
| -1 | 1 | -1 |
| -1 | 1 | -1 |
| -1 | 1 | -1 |

Filter 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 |

Convolution Layer

32x32x3 image

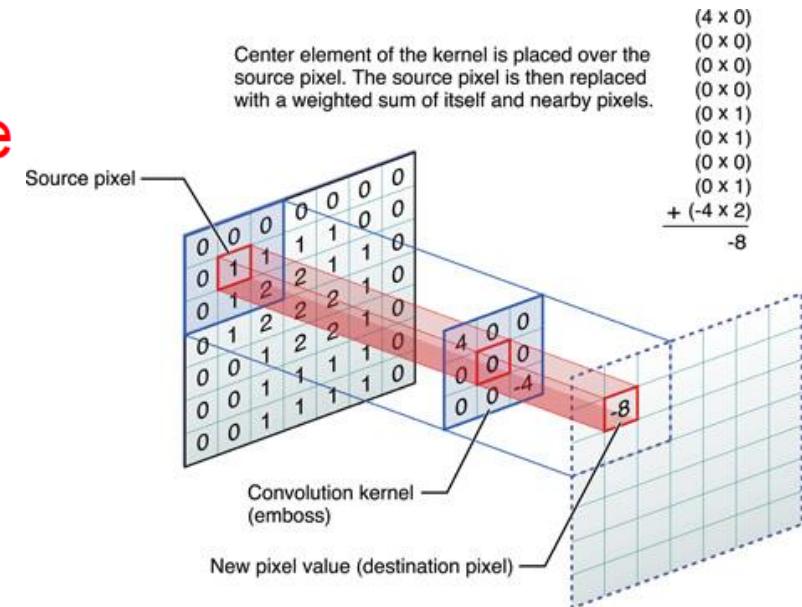
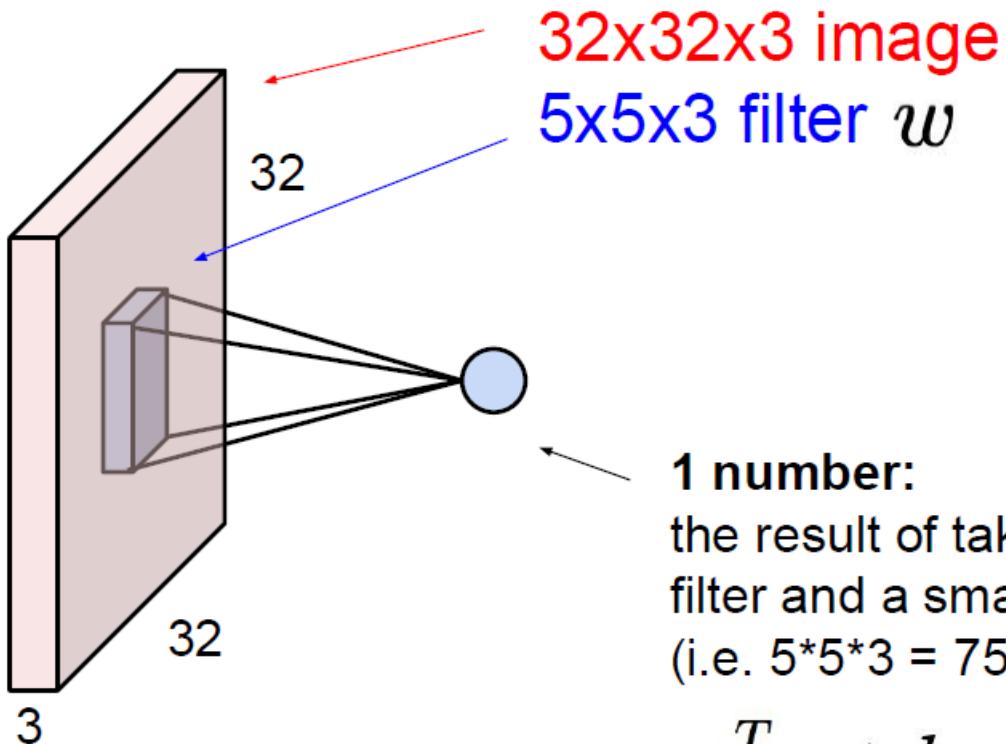


5x5x3 filter



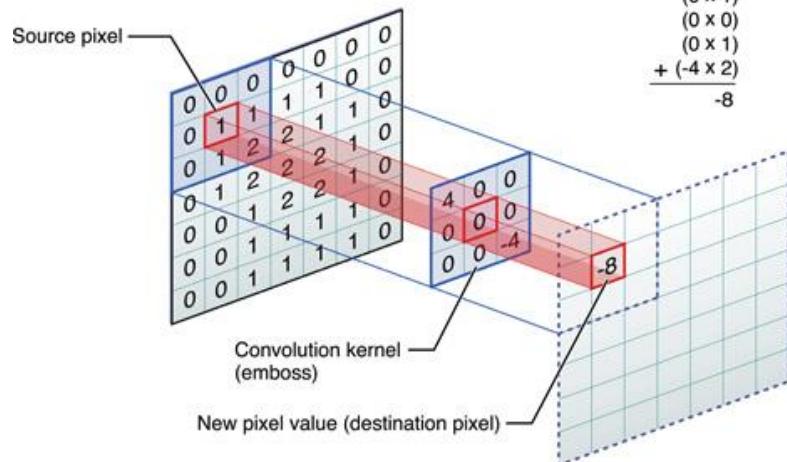
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer



Convolution Layer

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.



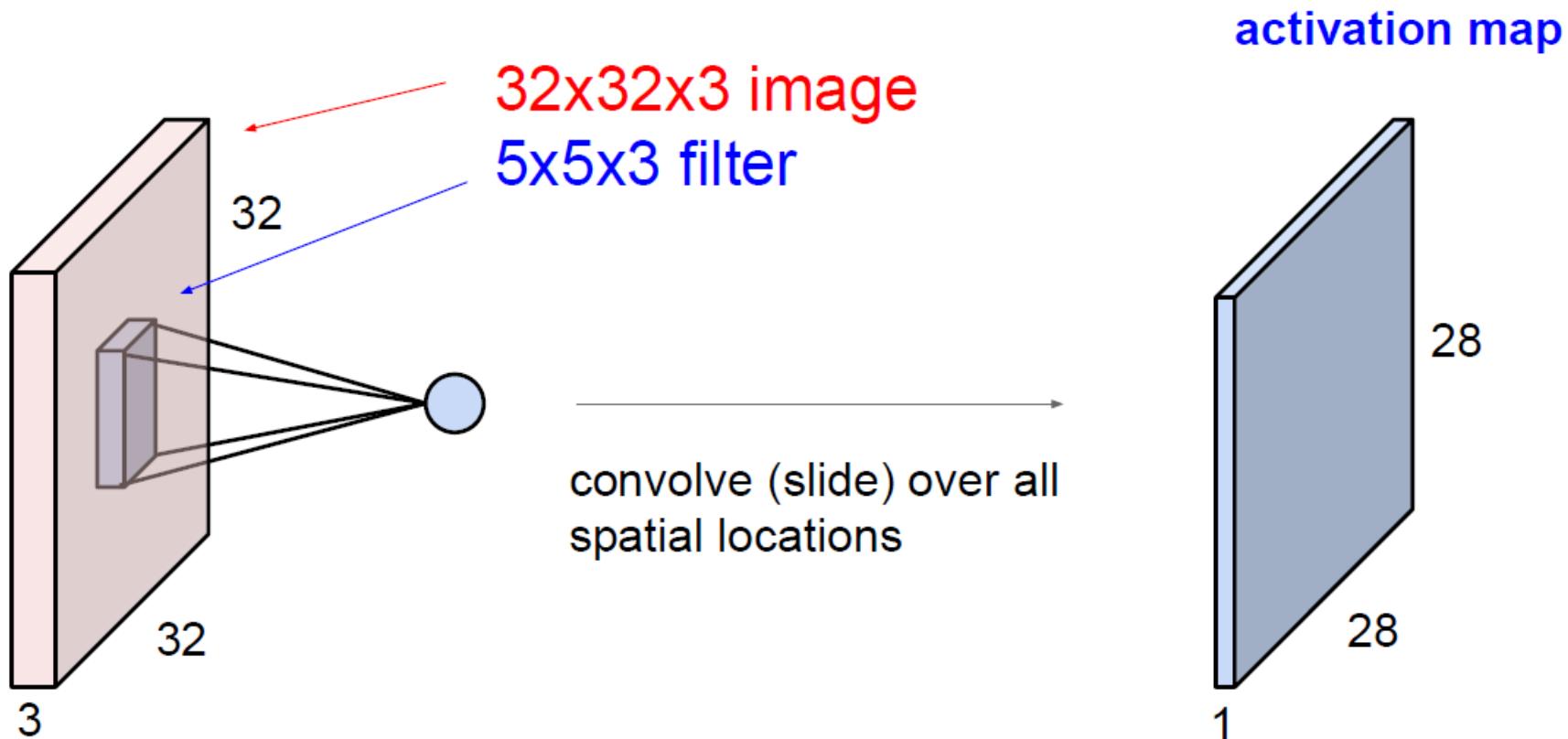
| | | | | |
|-----------------|-----------------|-----------------|---|---|
| 1 _{x1} | 1 _{x0} | 1 _{x1} | 0 | 0 |
| 0 _{x0} | 1 _{x1} | 1 _{x0} | 1 | 0 |
| 0 _{x1} | 0 _{x0} | 1 _{x1} | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Image

| | | |
|---|--|--|
| 4 | | |
| | | |
| | | |
| | | |

Convolved Feature

Convolution Layer

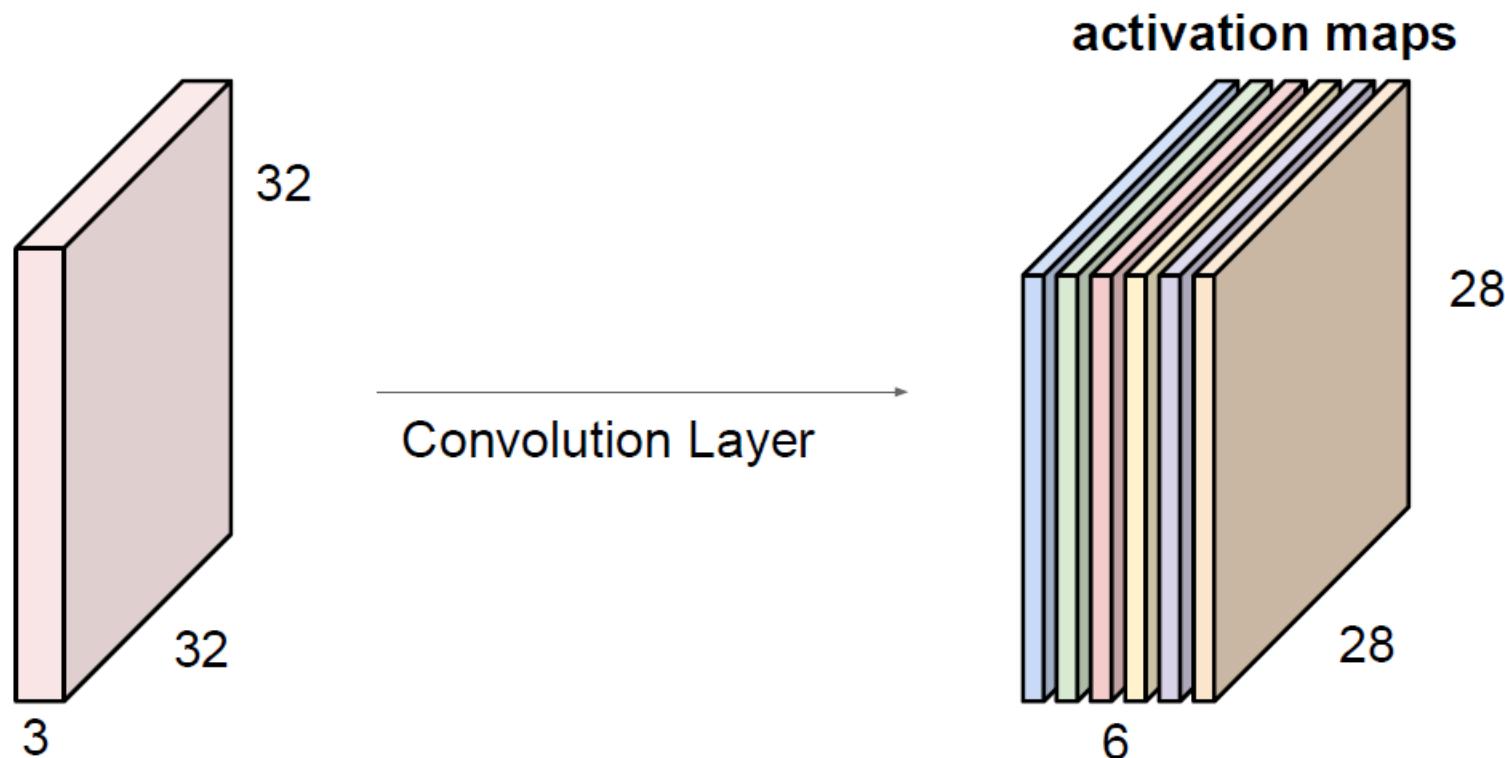


Convolution Layer



Convolution Layer

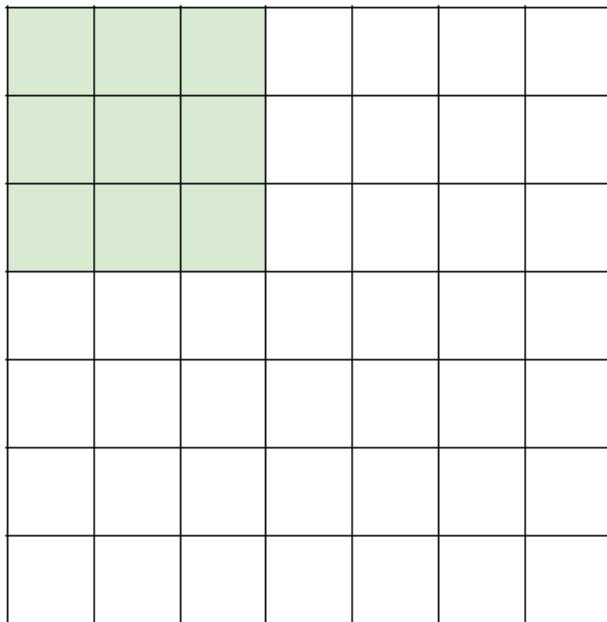
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a “new image” of size $28 \times 28 \times 6$!

Convolution Layer: Spatial Dimensions

7

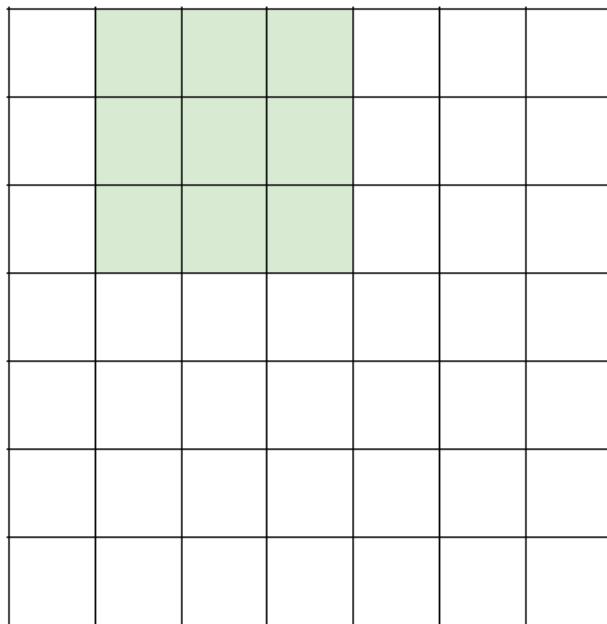


7x7 input (spatially)
assume 3x3 filter

7

Convolution Layer: Spatial Dimensions

7

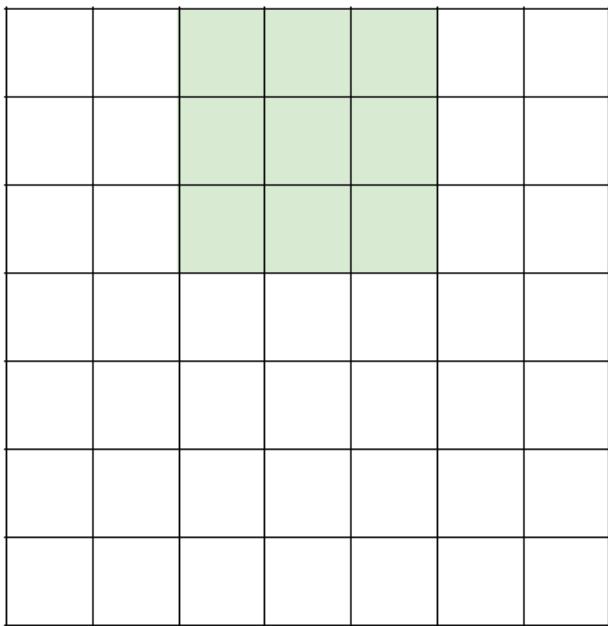


7x7 input (spatially)
assume 3x3 filter

7

Convolution Layer: Spatial Dimensions

7

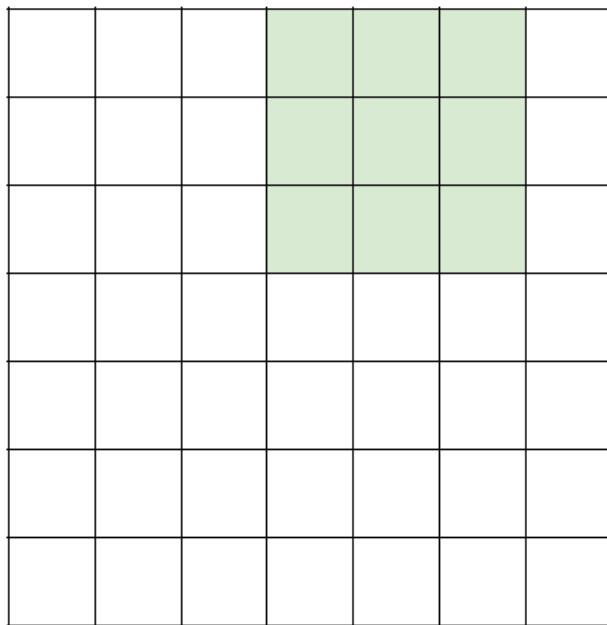


7x7 input (spatially)
assume 3x3 filter

7

Convolution Layer: Spatial Dimensions

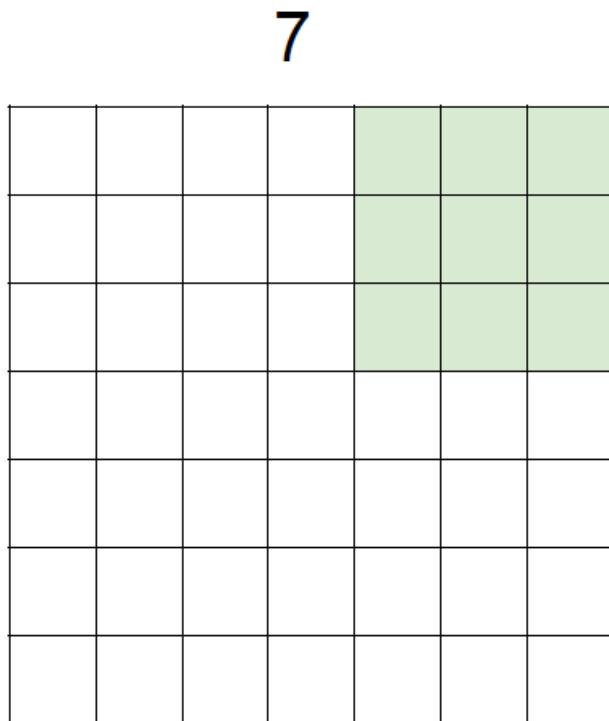
7



7x7 input (spatially)
assume 3x3 filter

7

Convolution Layer: Spatial Dimensions

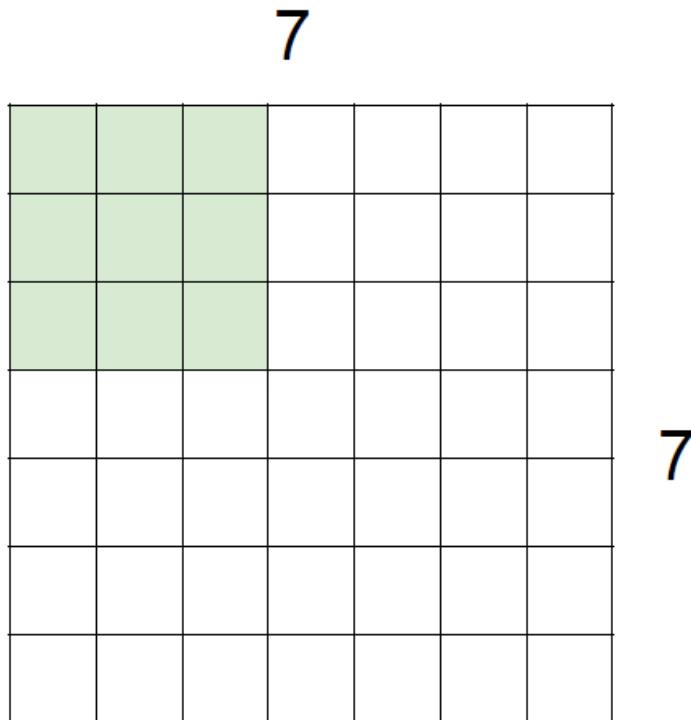


7x7 input (spatially)
assume 3x3 filter

7

=> 5x5 output

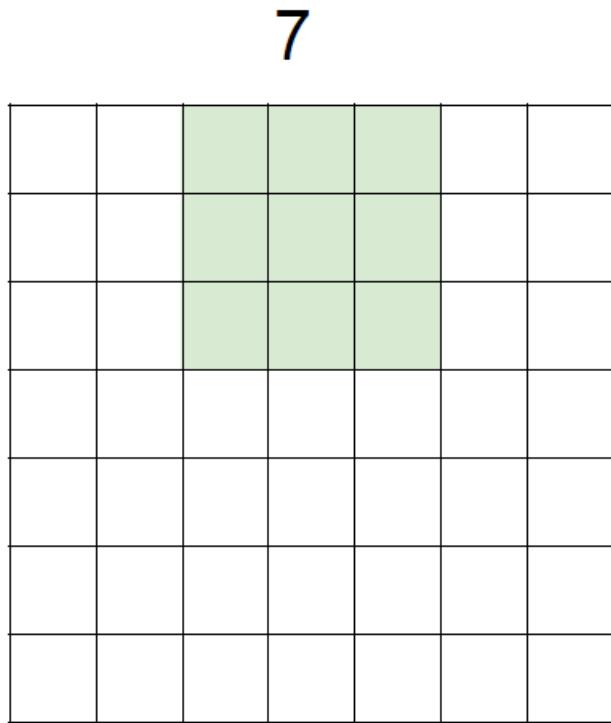
Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

7

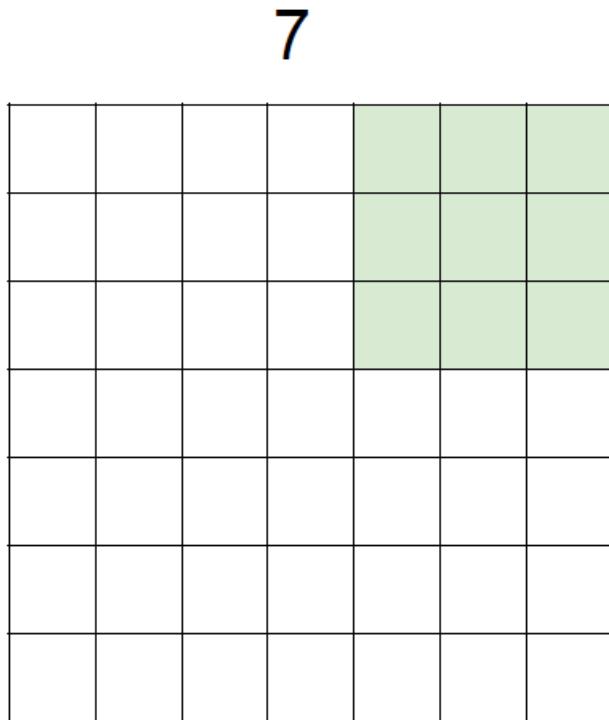
Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

7

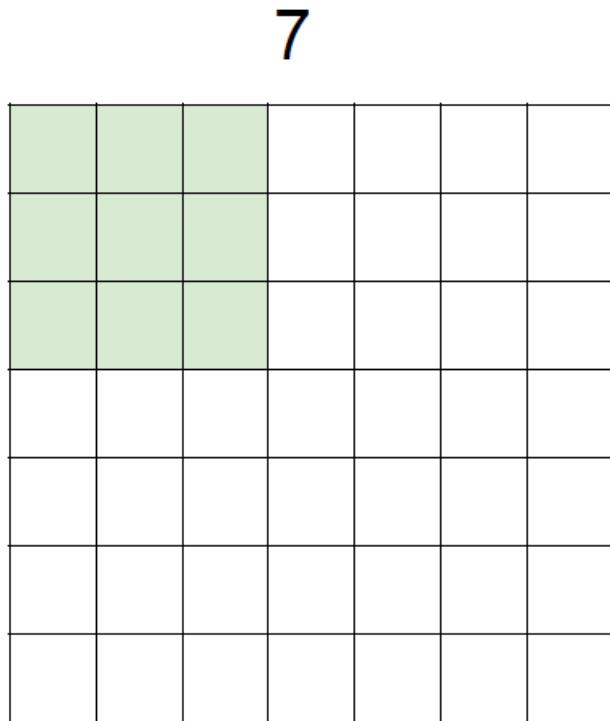
Convolution Layer: Spatial Dimensions



7

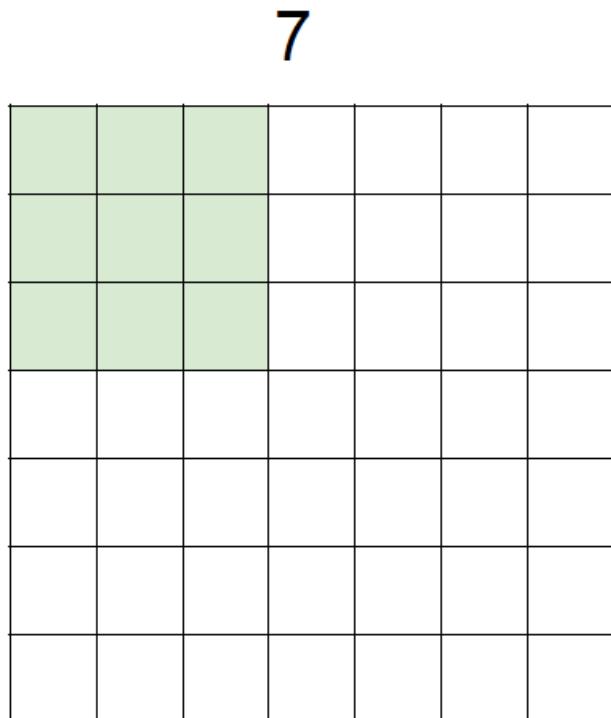
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

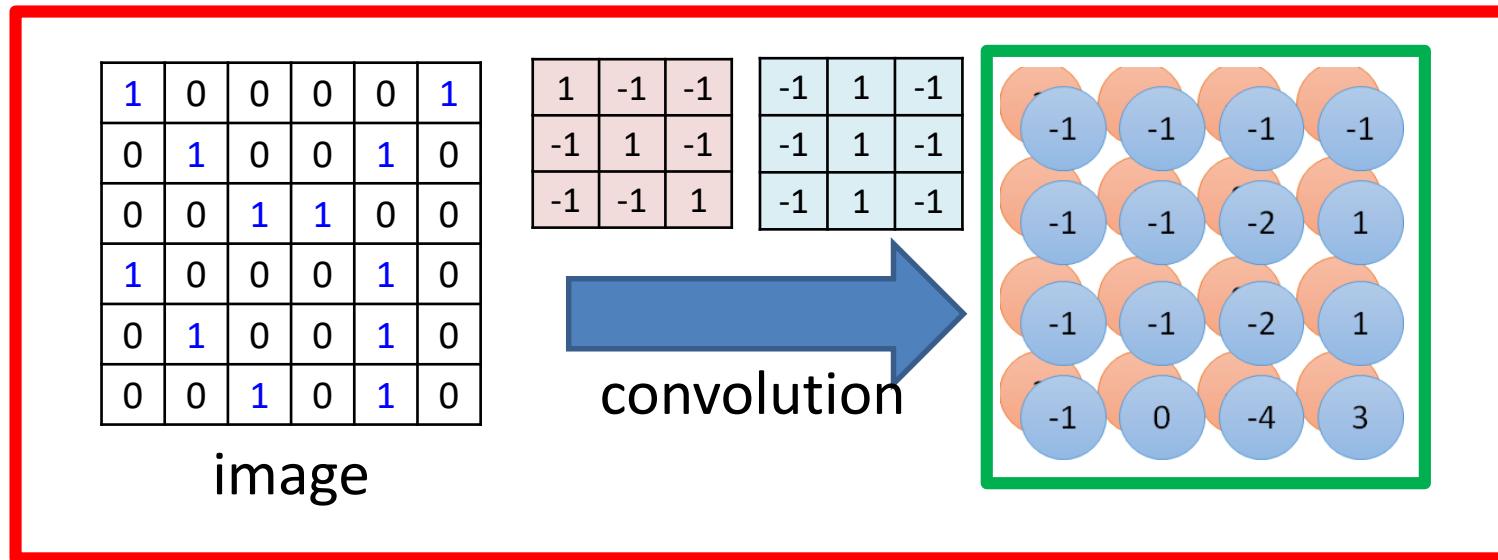
Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

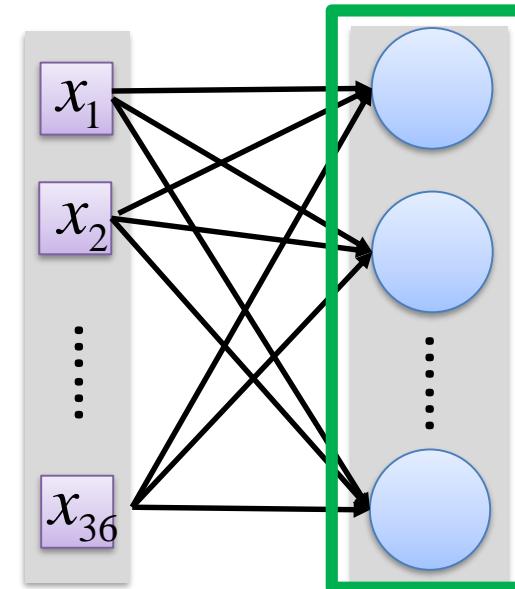
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

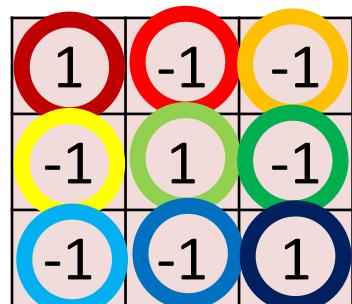
Convolution v.s. Fully Connected



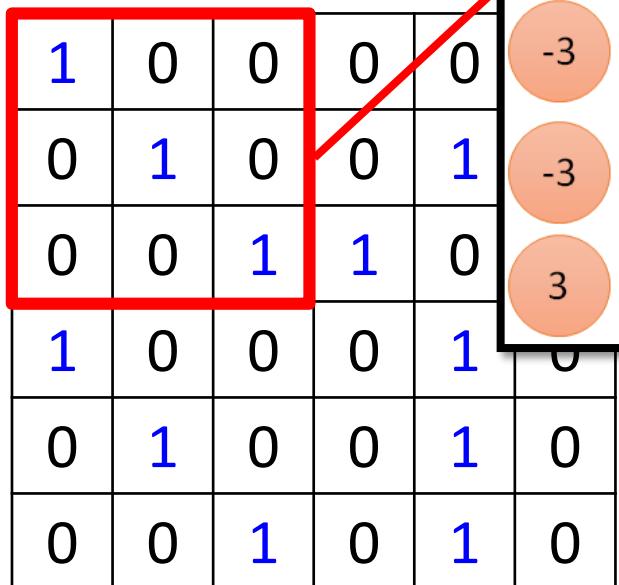
Fully-
connected

| | | | | | |
|---|---|---|---|---|---|
| 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 |



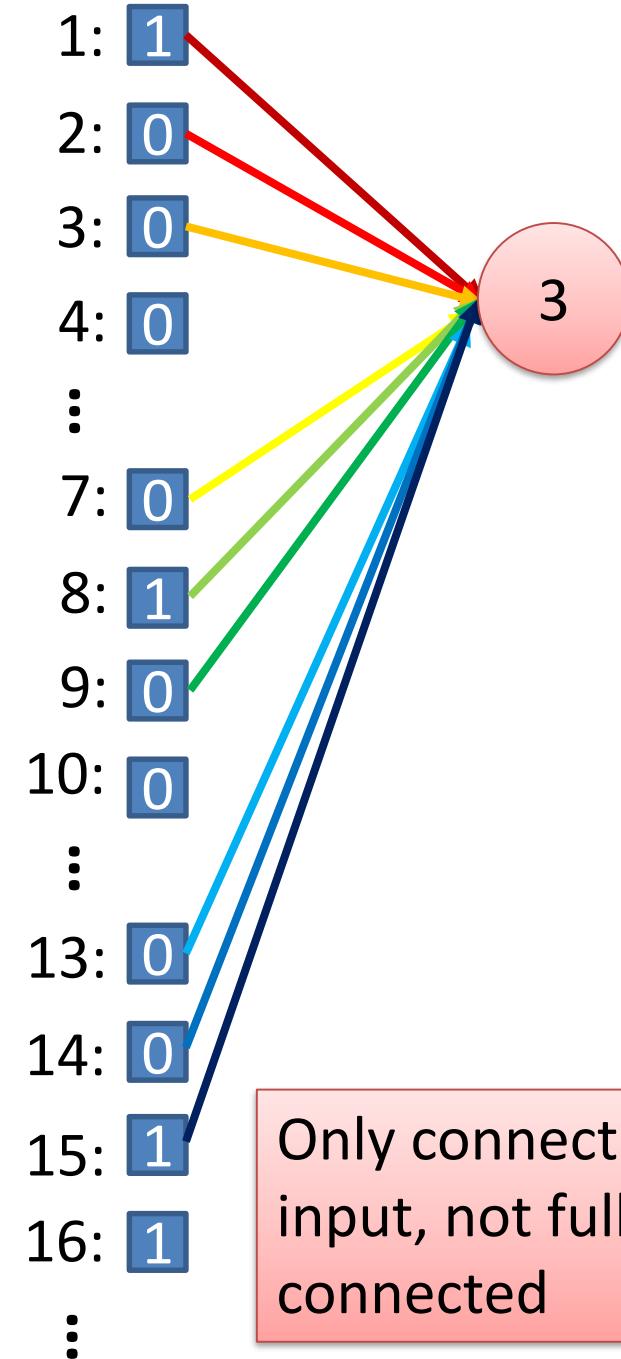
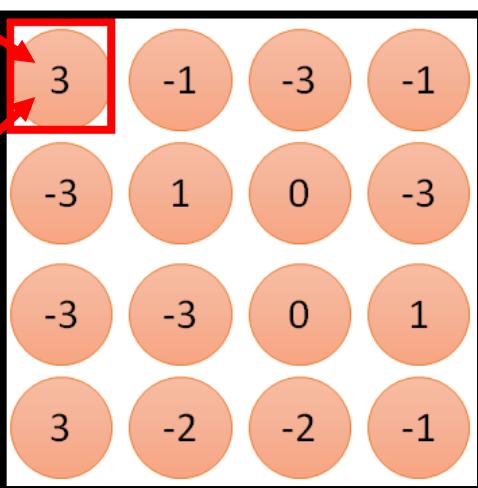


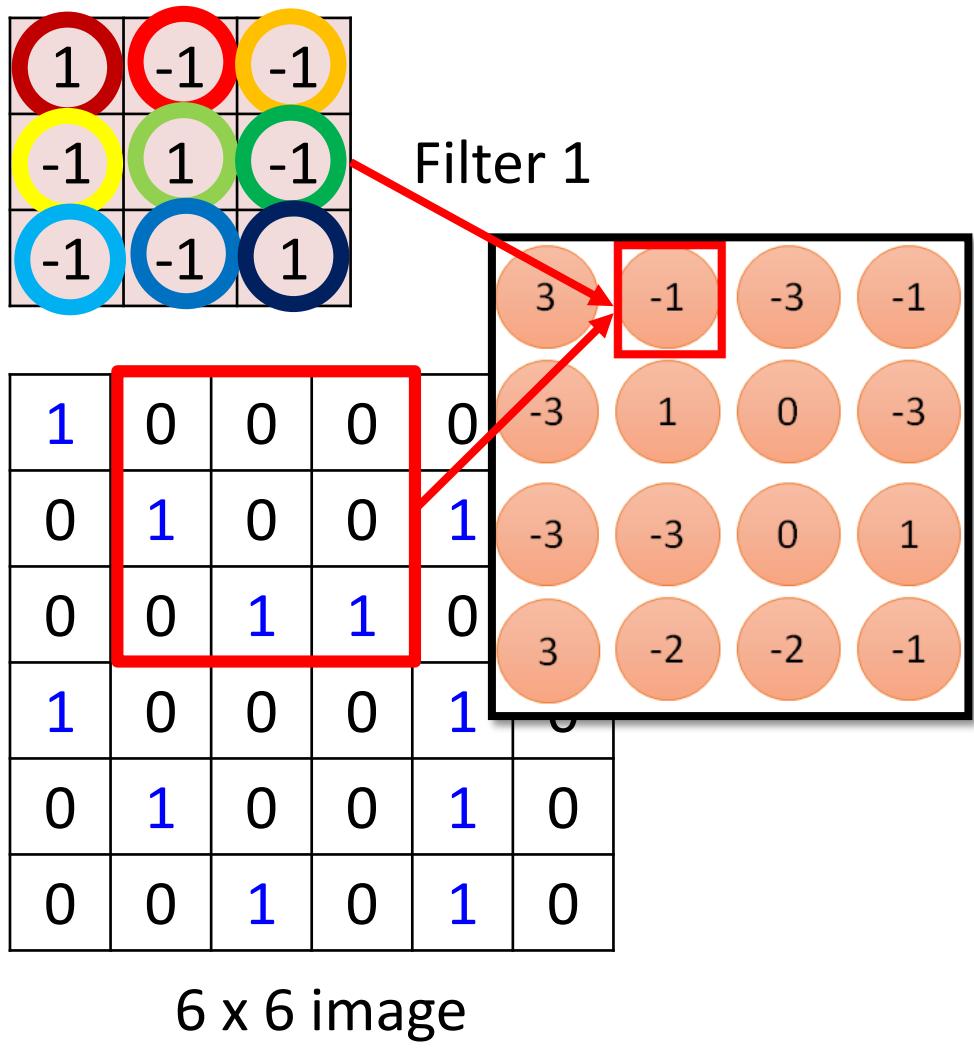
Filter 1



6×6 image

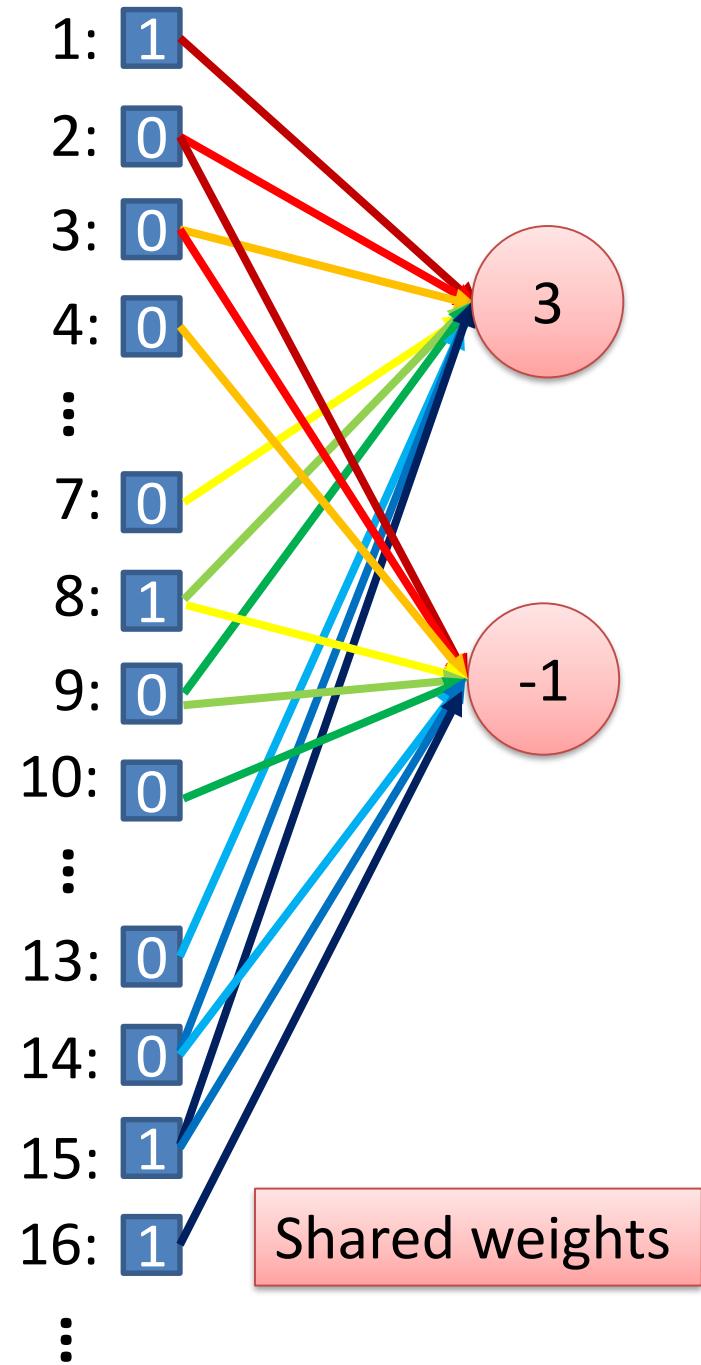
Less parameters!





Less parameters!

Even less parameters!

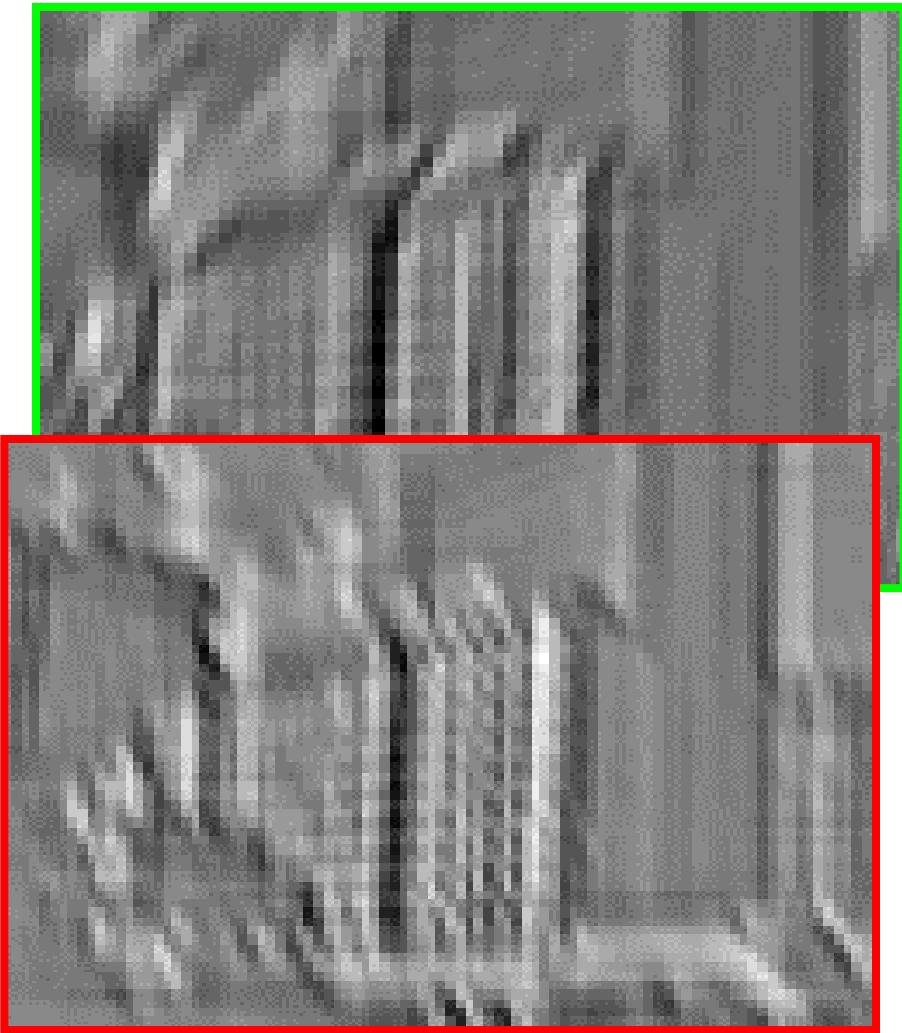


What Does Convolution Do?

- Weighted moving sum
- Find matching location between kernel and image



Input

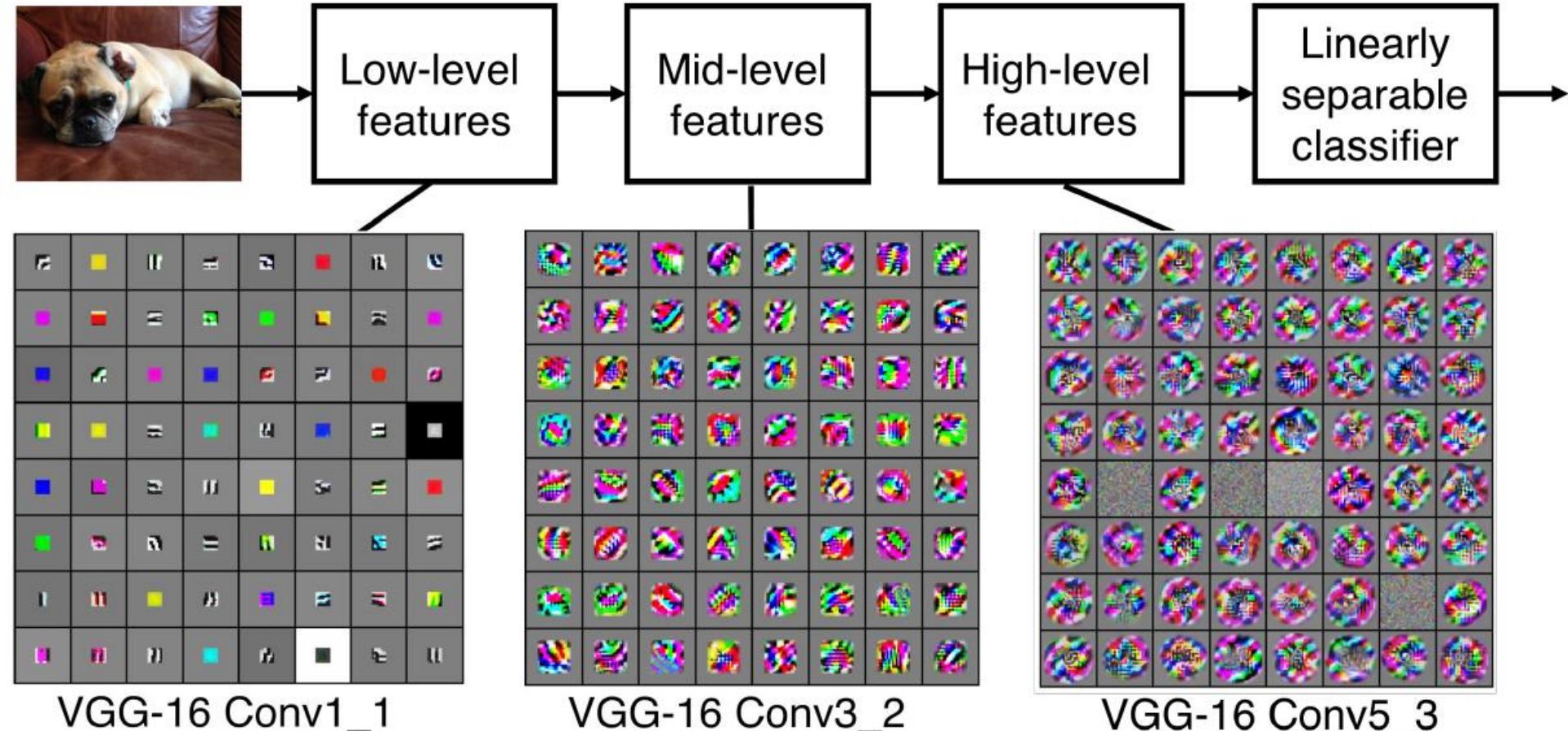


Feature Activation Map

34 605

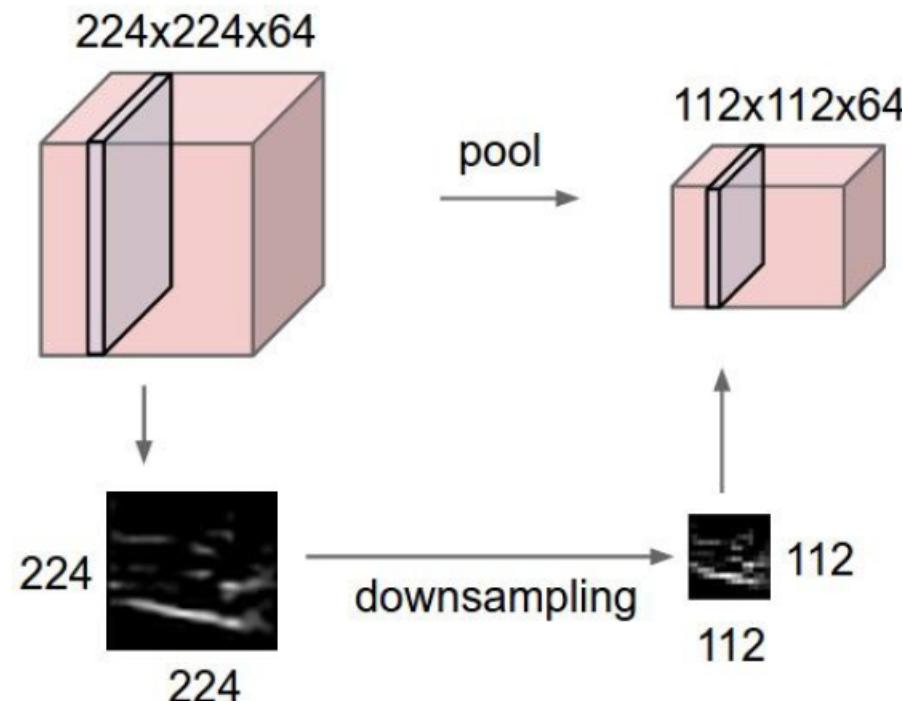
slide credit: S. Lazebnik

What Does Convolution Do?

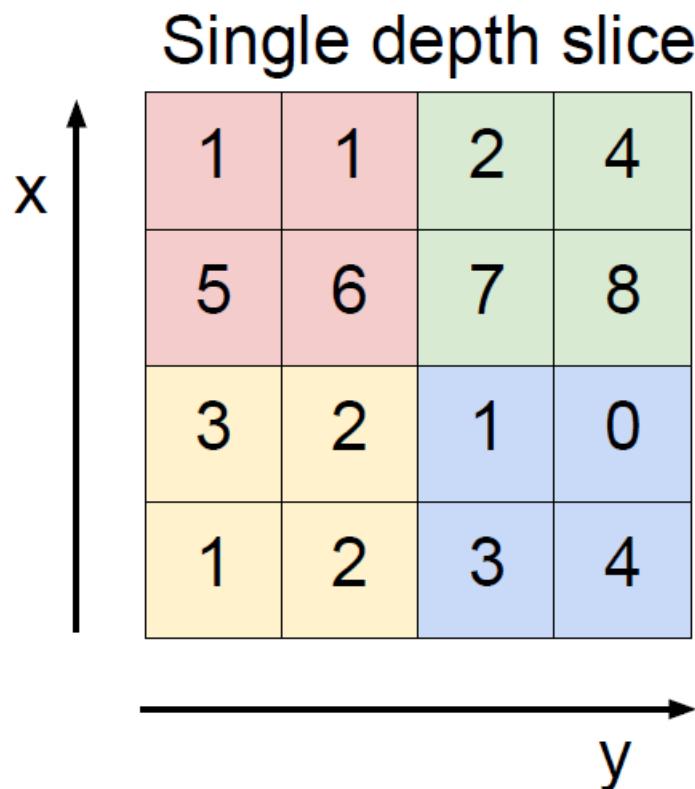


Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Pooling Layer



max pool with 2x2 filters
and stride 2

A 2x2 output matrix resulting from the max pooling operation. The top-left cell contains 6 (the max of 1,1,5,6), the top-right cell contains 8 (the max of 2,1,7,8), the bottom-left cell contains 3 (the max of 3,2,1,0), and the bottom-right cell contains 4 (the max of 0,4).

| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

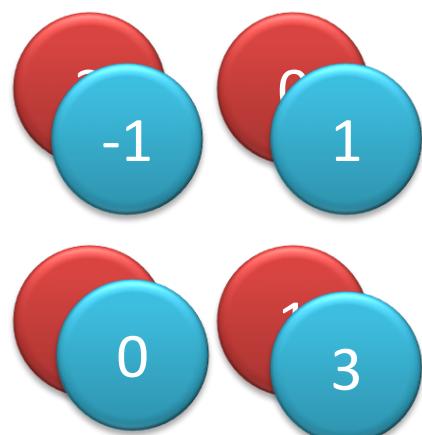
Pooling Layer

- The result of using a pooling layer and creating down sampled or pooled feature maps is a **summarized version** of the **features detected** in the input.
- They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location.
- This capability added by pooling is called the model's **invariance to local translation**.

Pooling Layer

- “*In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change*”

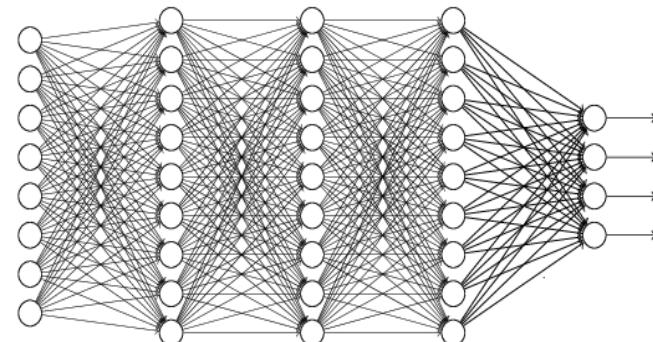
Flatten



Flatten



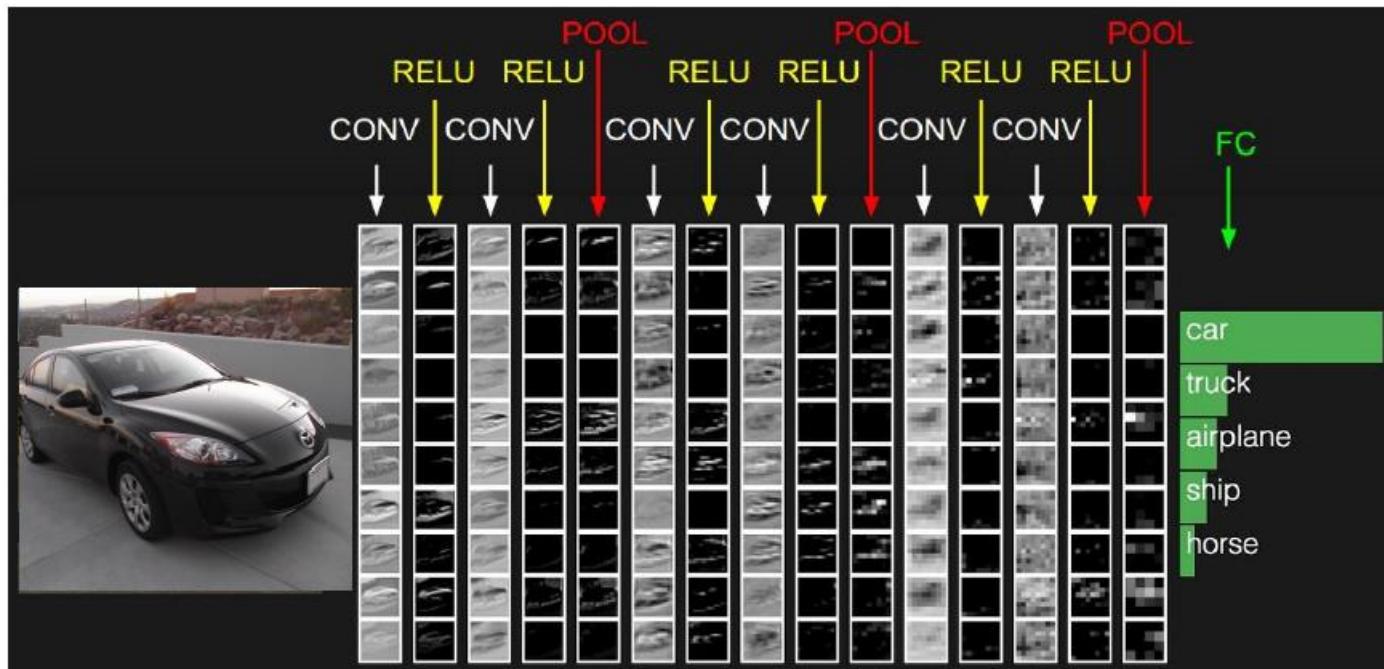
Flatten



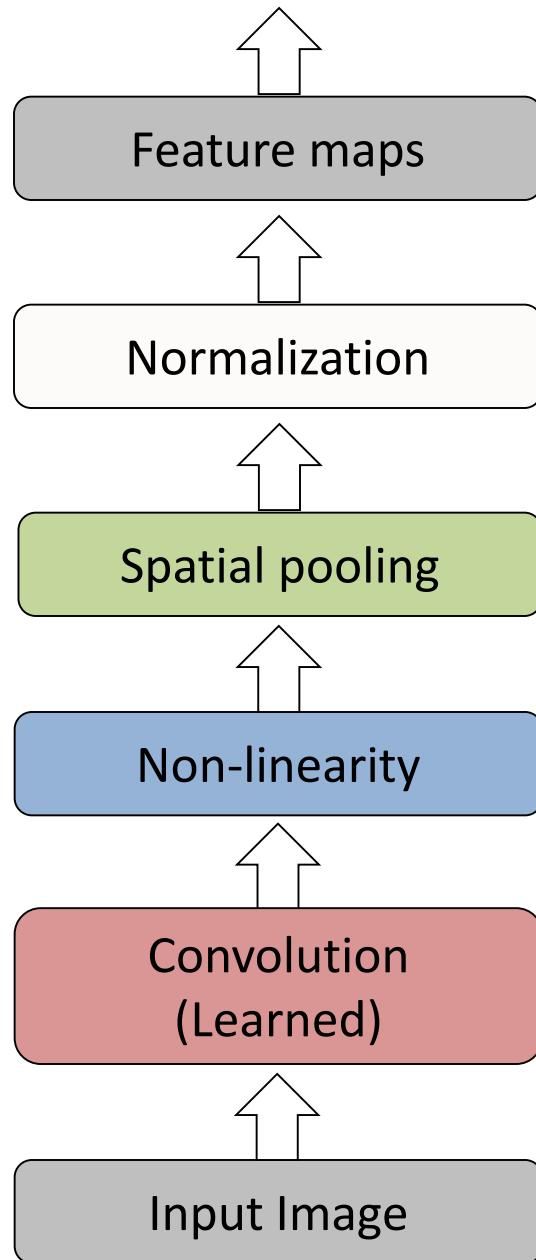
Fully Connected
Feedforward network

Fully Connected Layer

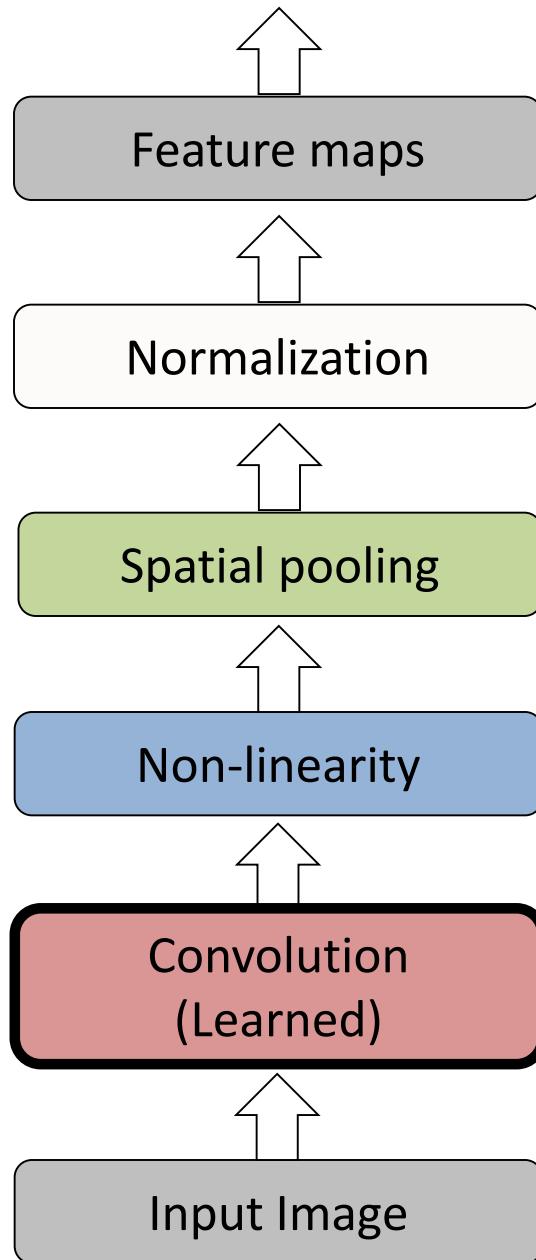
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



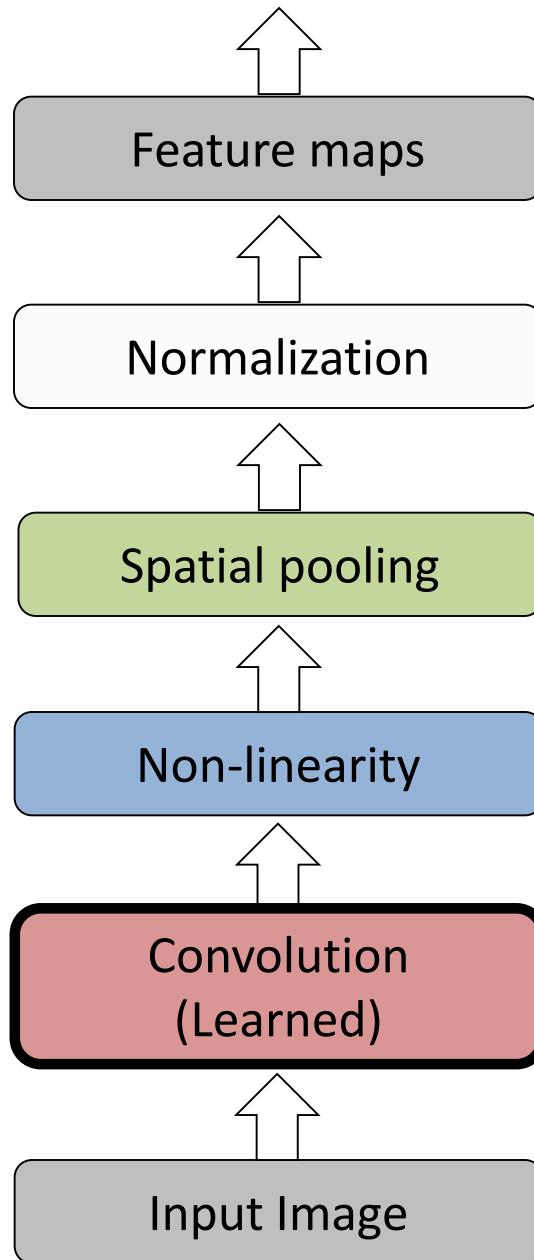
Convolutional Neural Networks



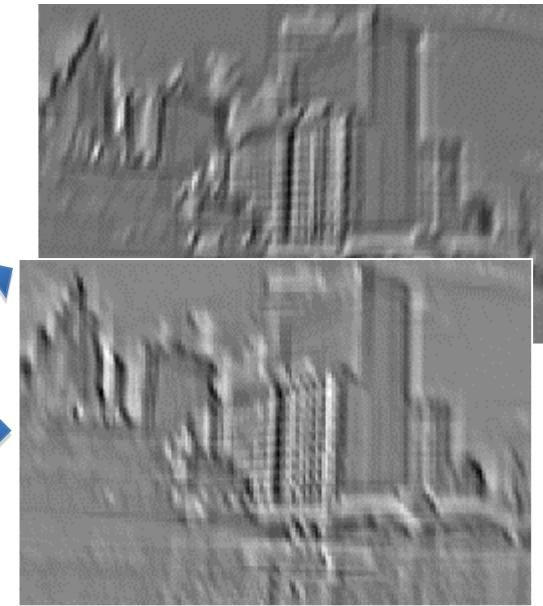
Convolutional Neural Networks



Convolutional Neural Networks

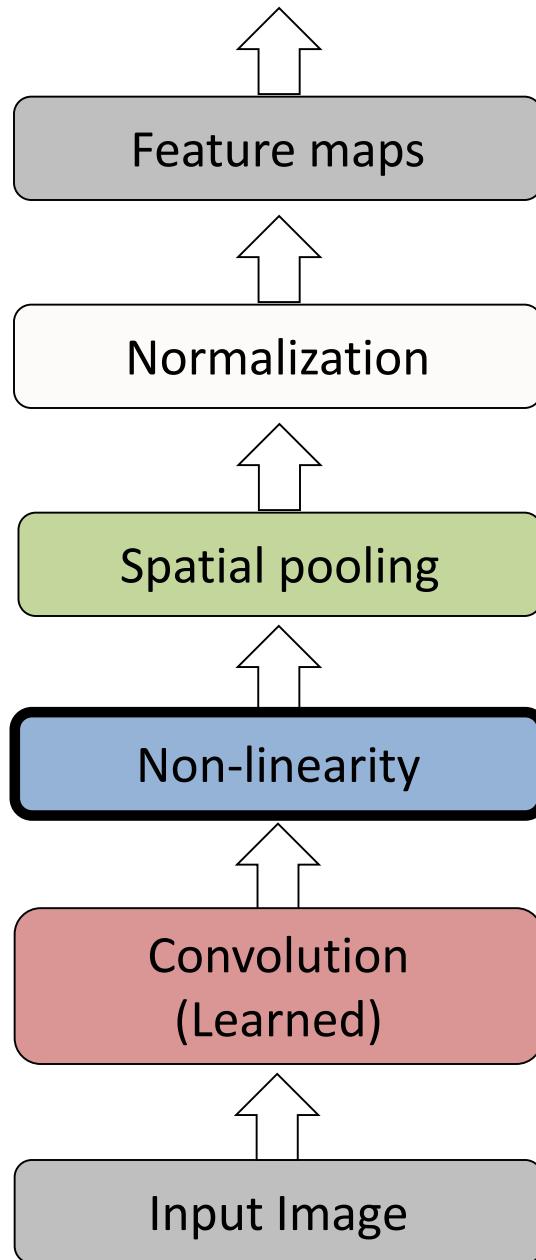


Input

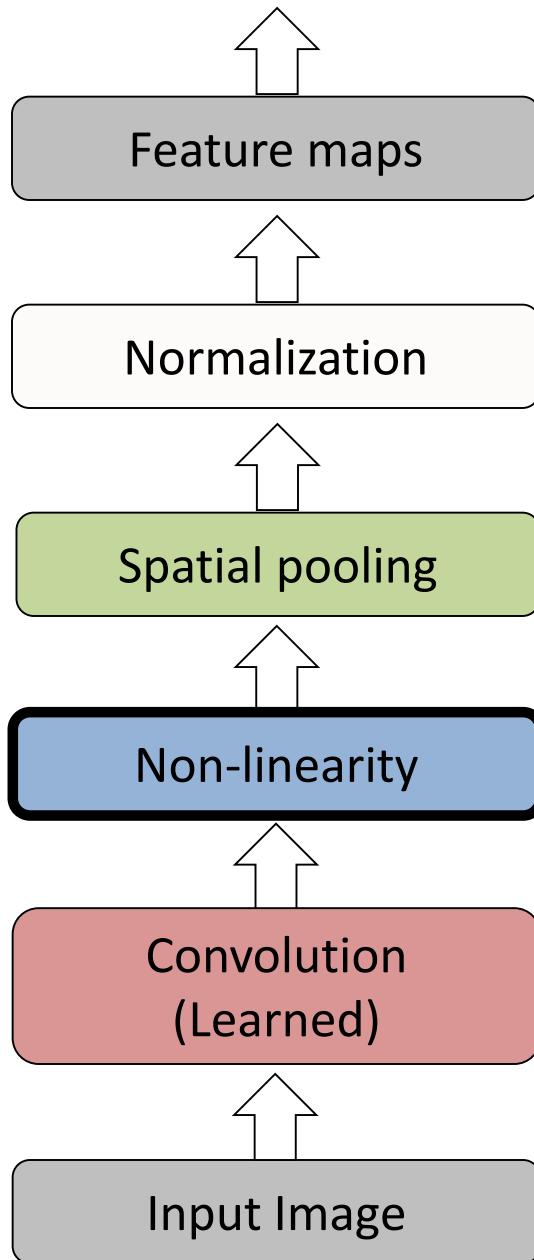


Feature Map

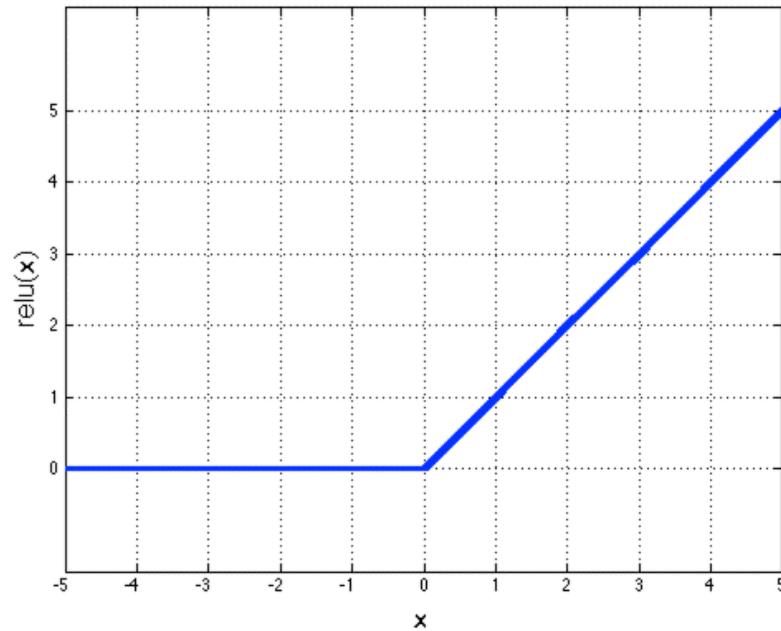
Convolutional Neural Networks



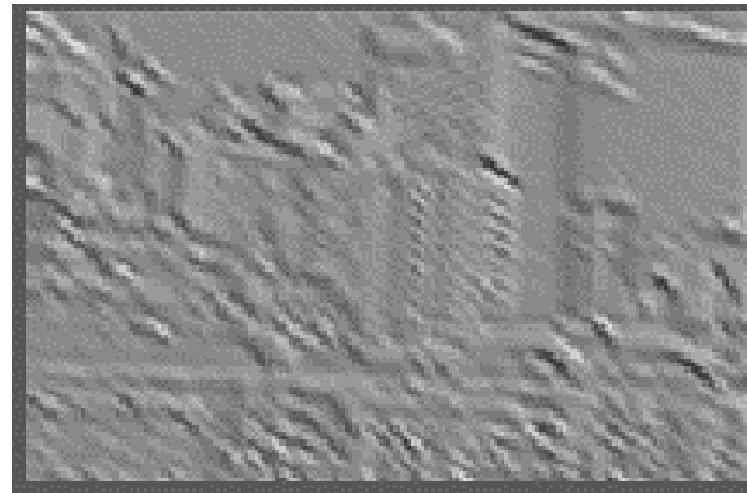
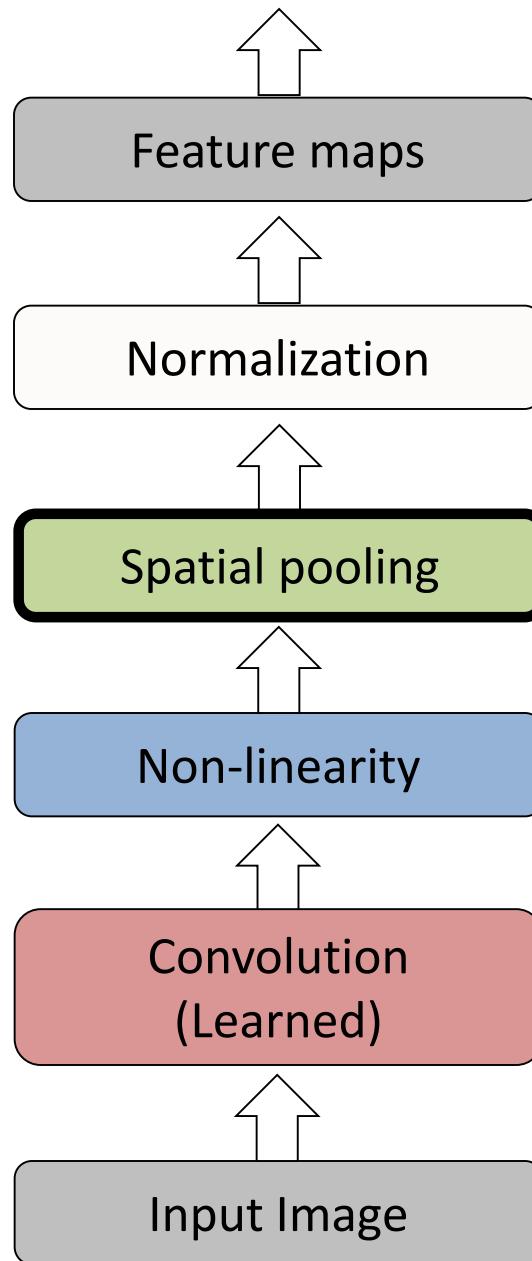
Convolutional Neural Networks



Rectified Linear Unit (ReLU)

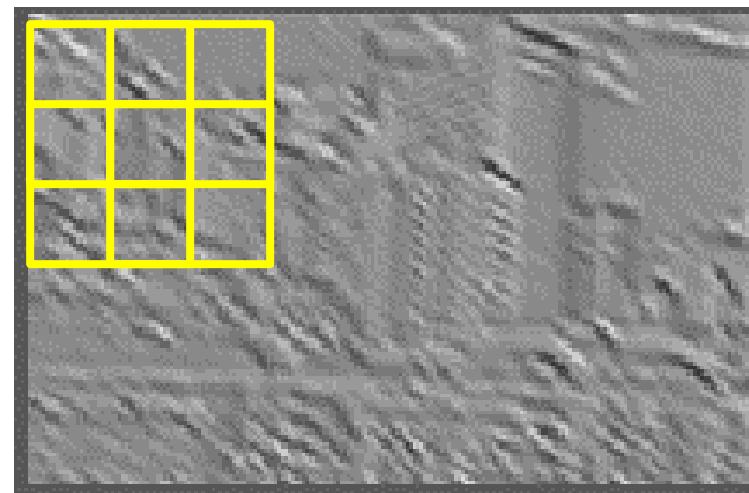
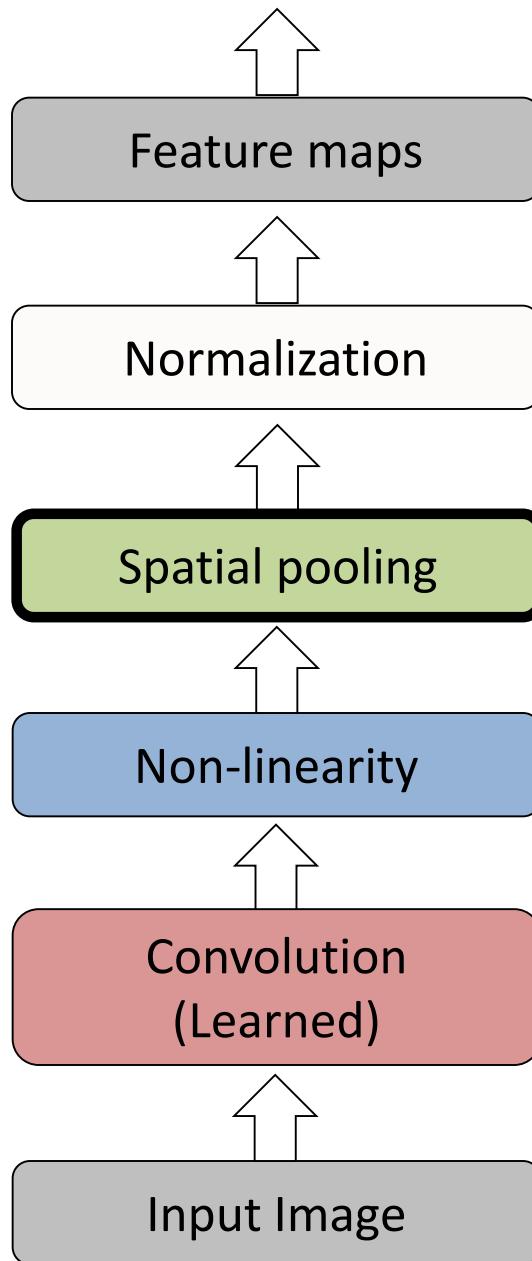


Convolutional Neural Networks

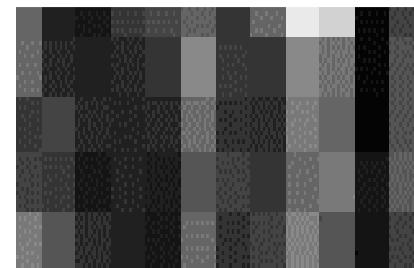


Max-pooling: a non-linear down-sampling

Convolutional Neural Networks

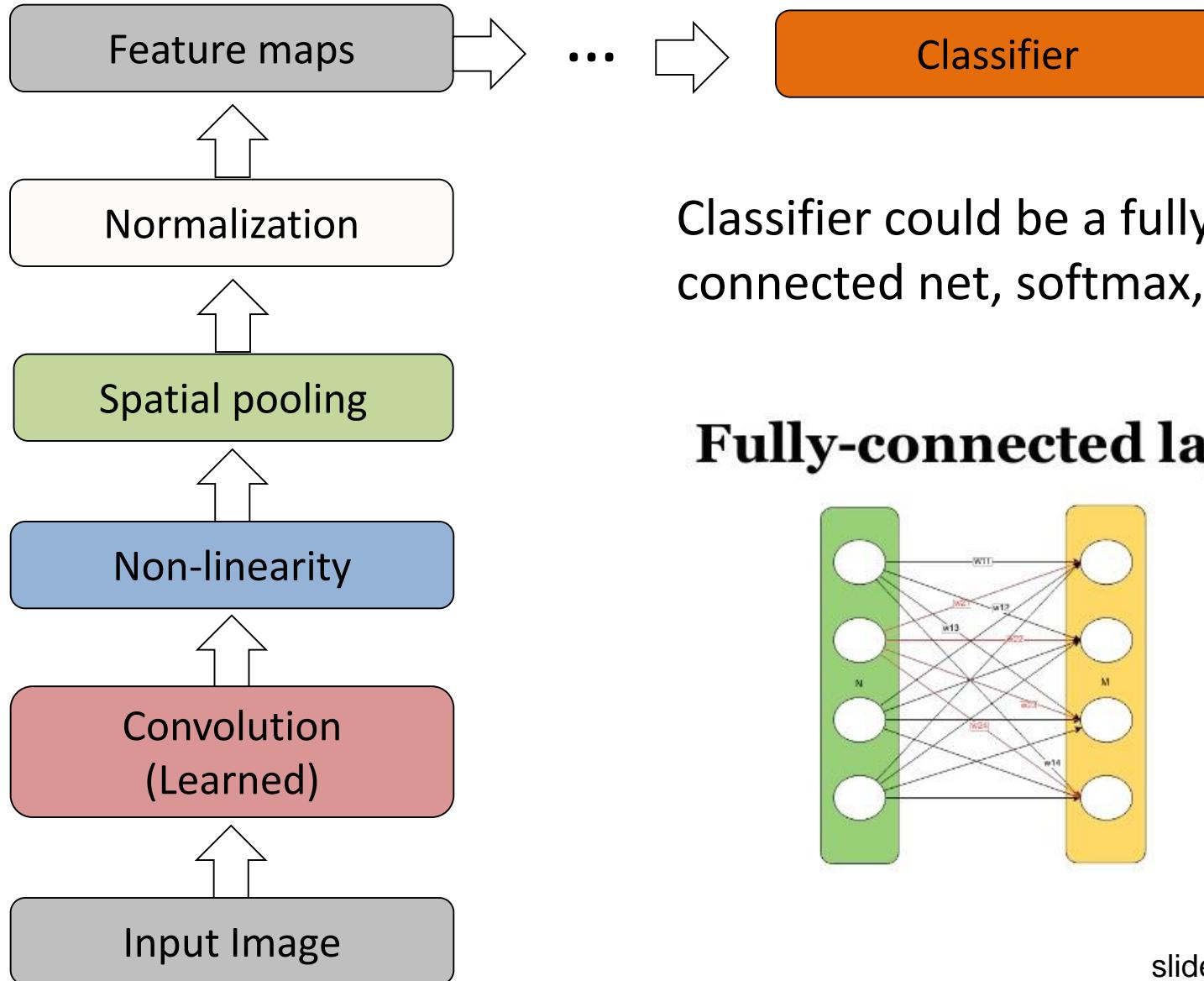


Max pooling



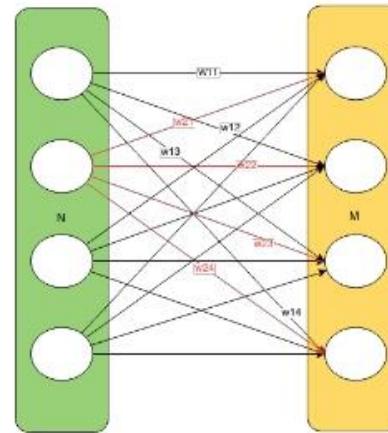
Max-pooling: a non-linear down-sampling

Convolutional Neural Networks

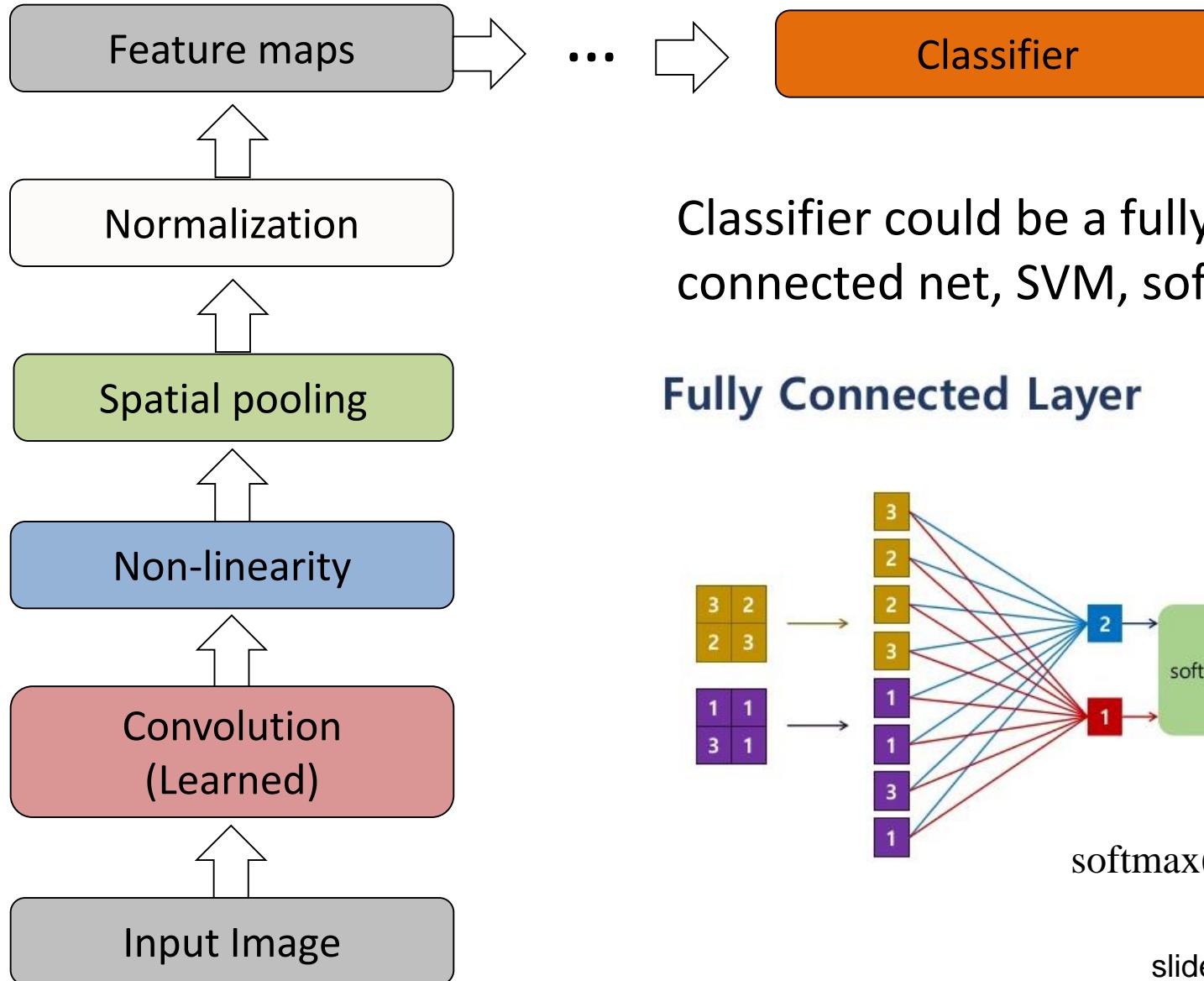


Classifier could be a fully connected net, softmax, etc.

Fully-connected layer

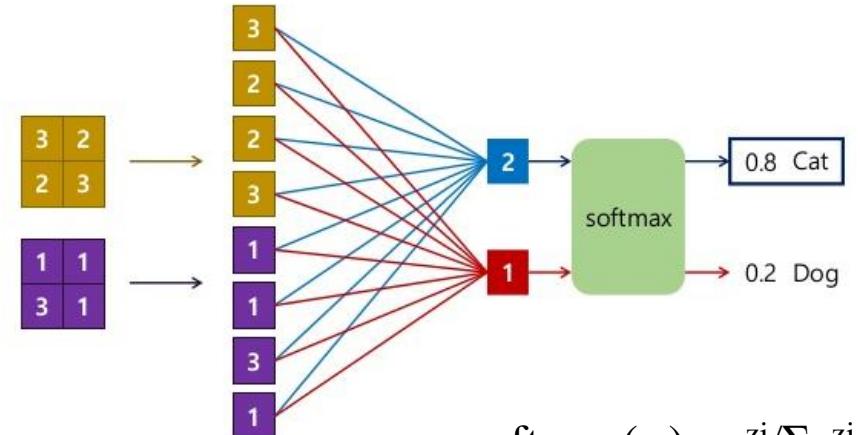


Convolutional Neural Networks



Classifier could be a fully connected net, SVM, softmax, etc.

Fully Connected Layer



$$\text{softmax}(z_i) = e^{z_i} / \sum e^{z_j}$$

LeNet (1989 - 1998)

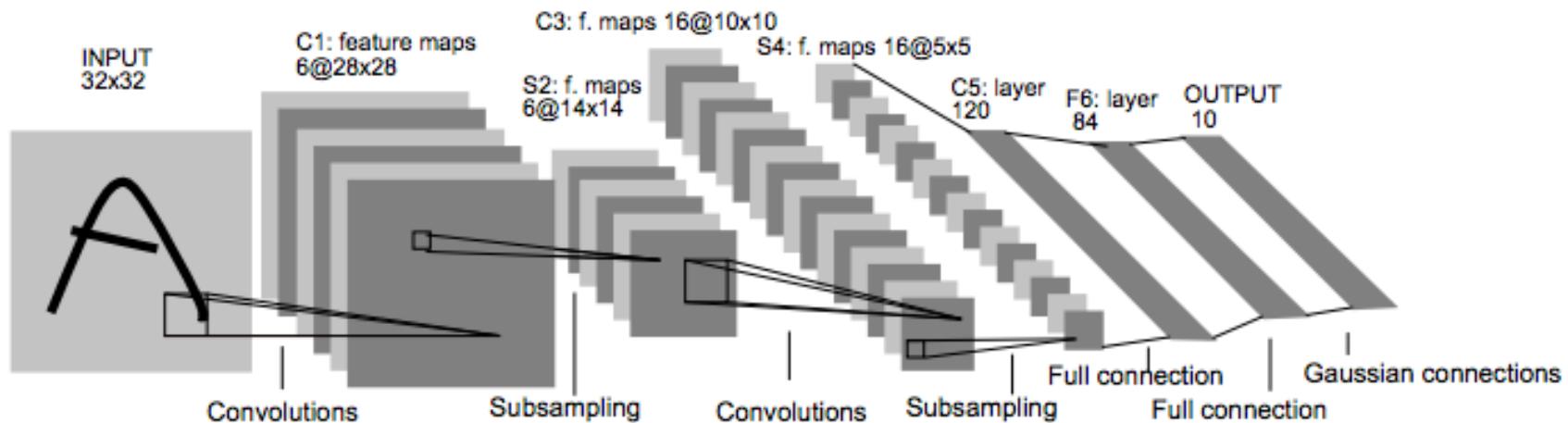


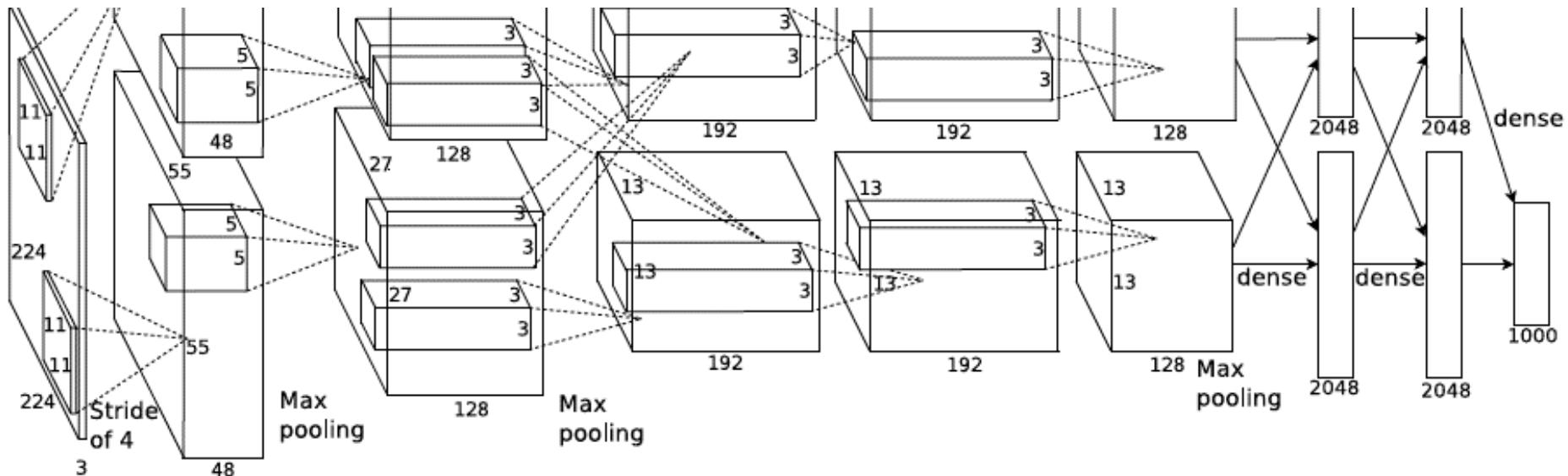
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.



<http://yann.lecun.com/exdb/lenet/>

AlexNet

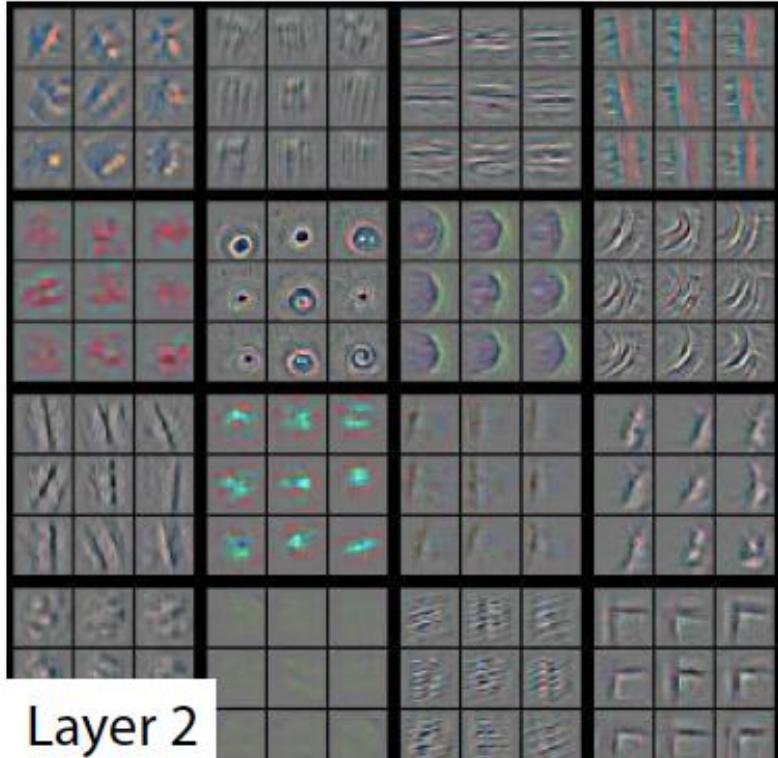
- Similar framework to LeCun'98 but:
 - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
 - More data (10^6 vs. 10^3 images)
 - GPU implementation (50x speedup over CPU)
 - Trained on two GPUs for a week



A. Krizhevsky, I. Sutskever, and G. Hinton,

[ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

Visualising Learned Filters

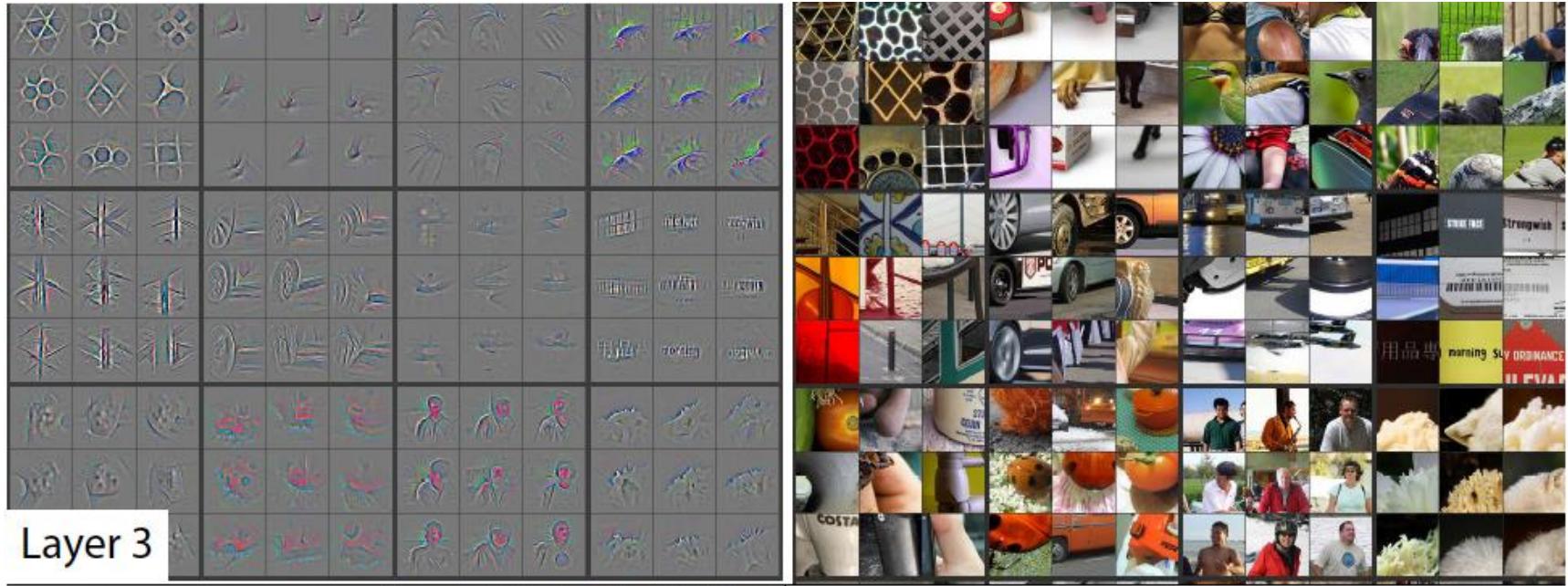


Layer 2

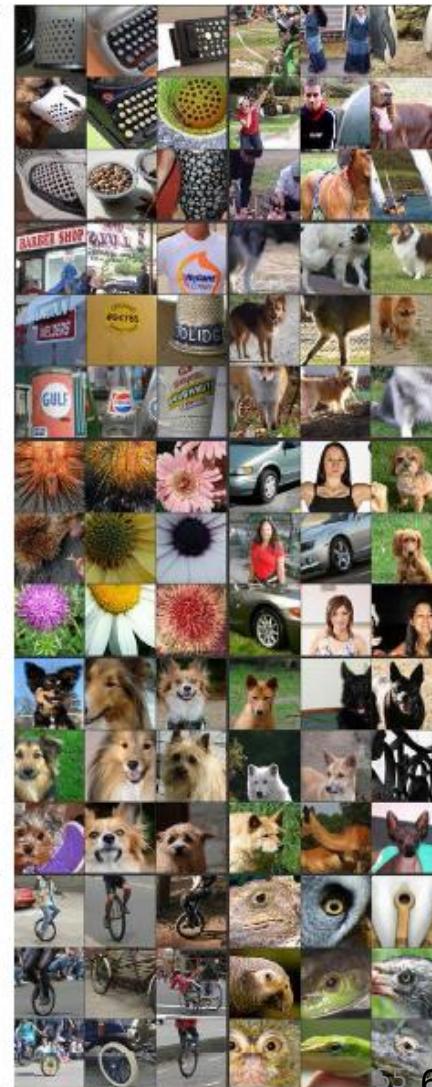
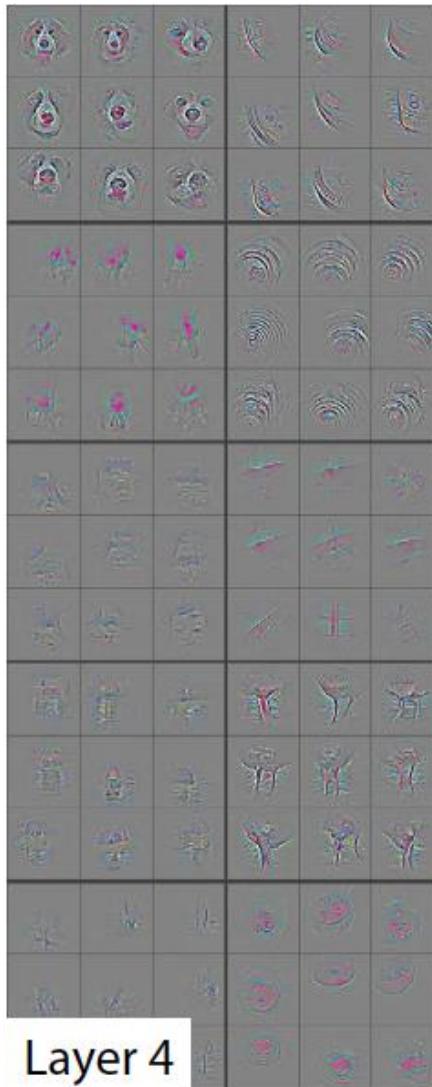


- Top 9 activations in a randomly selected set of (16) feature maps, projected down to previous level
- The image patches that generated those activations

Visualising Learned Features



Visualising Learned Features



Applications

- Handwritten text/digits
 - MNIST (0.17% error [Ciresan et al. 2011])
 - Arabic & Chinese [Ciresan et al. 2012]
- Simpler recognition benchmarks
 - CIFAR-10 (9.3% error [Wan et al. 2013])
 - Traffic sign recognition
 - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]



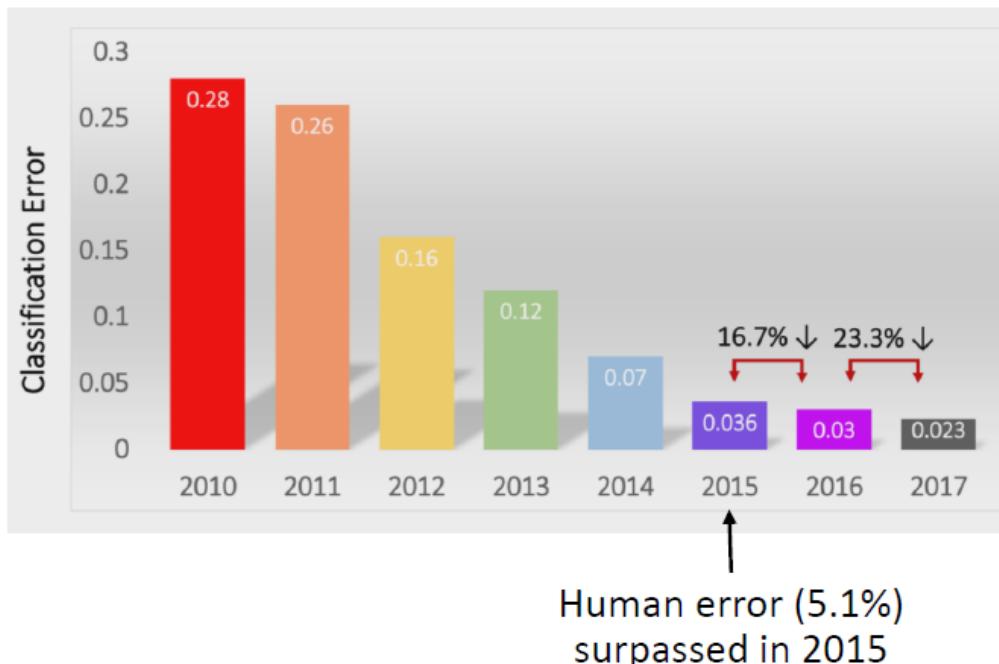
Application: ImageNet



- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

[Deng et al. CVPR 2009]

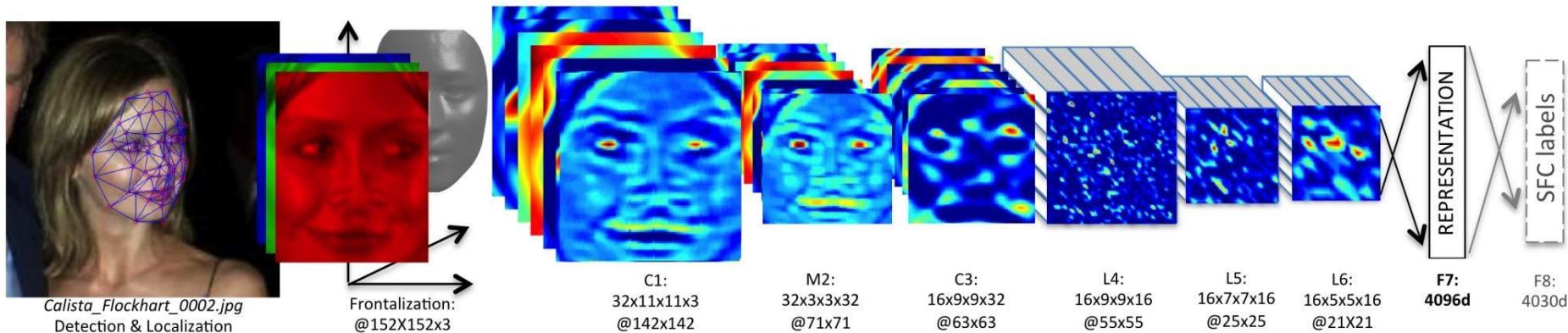
ImageNet Classification Challenge



- **AlexNet (2012): First CNN (15.4%)**
 - 8 layers
 - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
 - 8 layers
 - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
 - Beautifully uniform:
3x3 conv, stride 1, pad 1, 2x2 max pool
 - 16 layers
 - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
 - Inception modules
 - 22 layers
 - 5 million parameters
(throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
 - More layers = better performance
 - 152 layers
- **CUIimage (2016): 3.57% to 2.99%**
 - Ensemble of 6 models
- **SENet (2017): 2.99% to 2.251%**
 - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.

Industry Deployment

- Used in Facebook, Google, Microsoft
- Image Recognition, Speech Recognition,
- Fast at test time



Taigman et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR'14

R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL

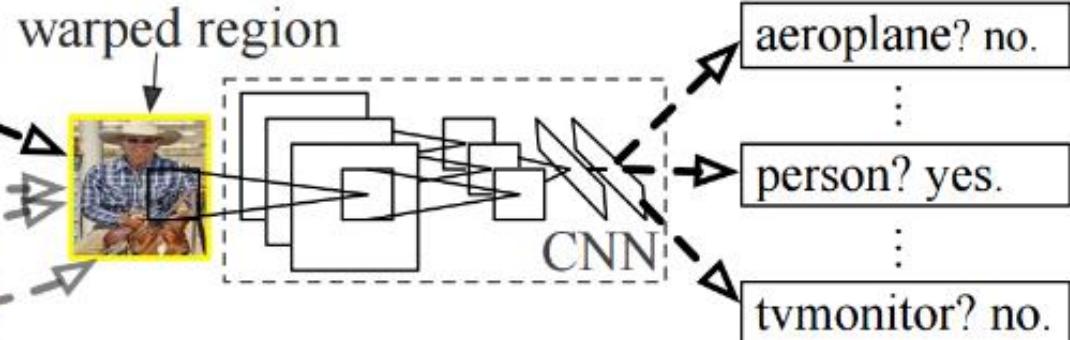
R-CNN: *Regions with CNN features*



1. Input image



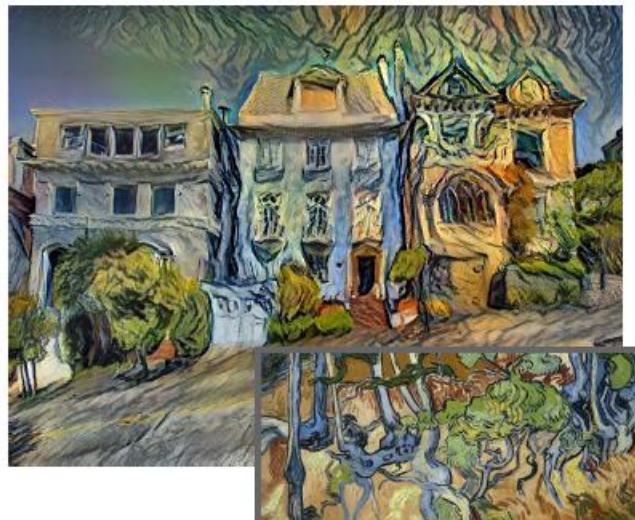
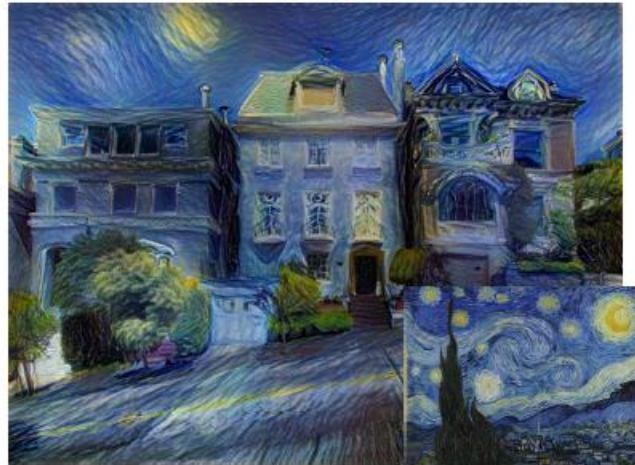
2. Extract region proposals (~2k)



3. Compute CNN features

4. Classify regions

Style Transfer



[Original image](#) is CC0 public domain

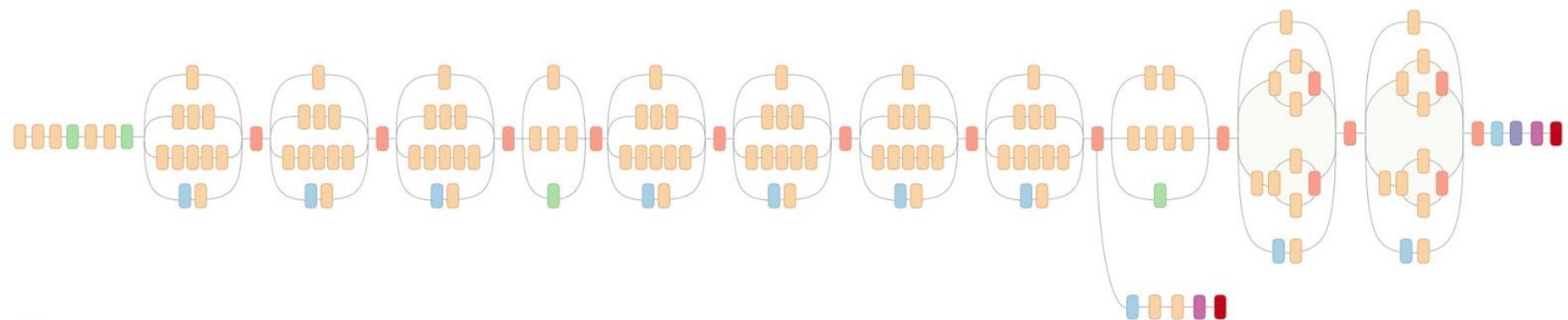
[Starry Night](#) and [Tree Roots](#) by Van Gogh are in the public domain

[Bokeh image](#) is in the public domain

Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Inception V3 (2015)

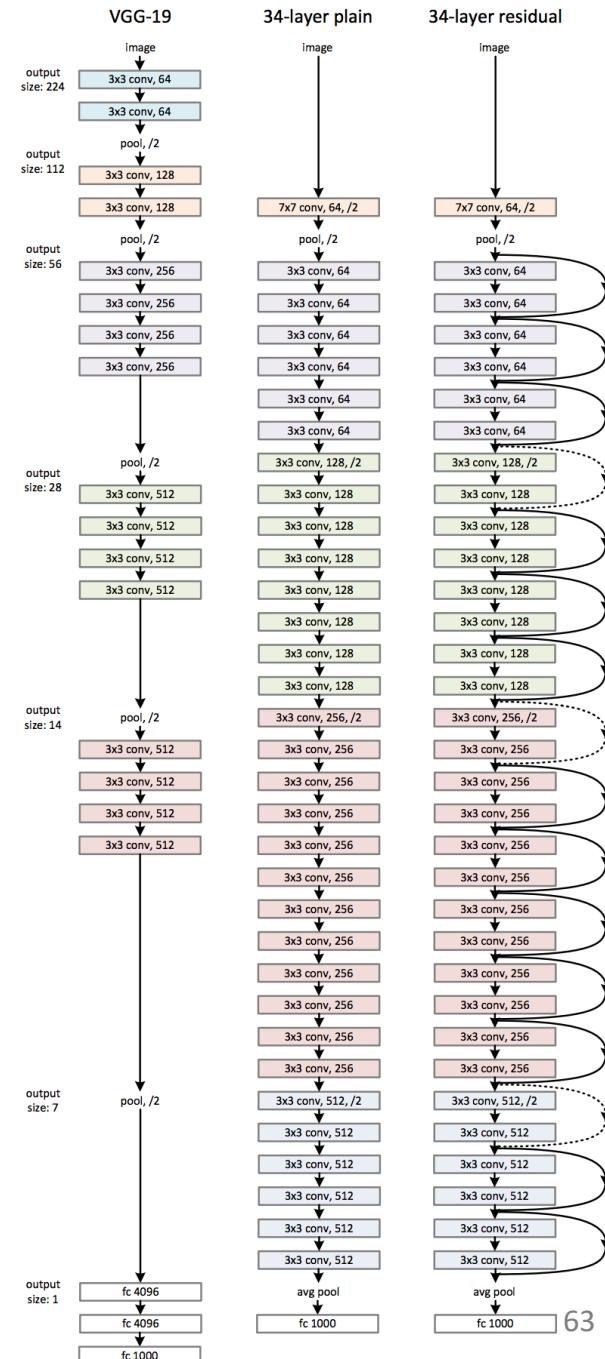


- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

Rethinking the Inception Architecture for Computer Vision, 2015
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, 2016

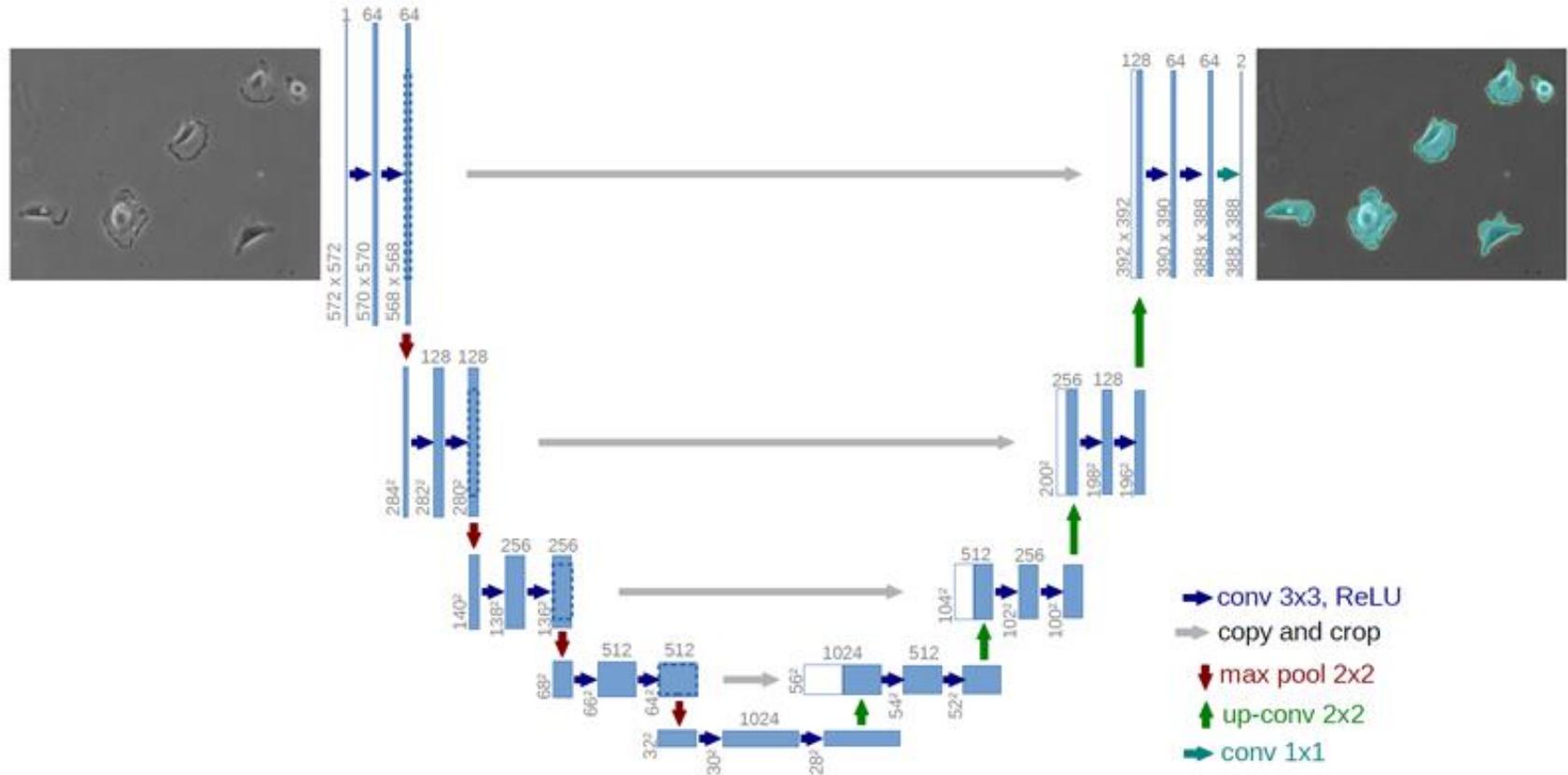
ResNets (2015)

- Gradients highways



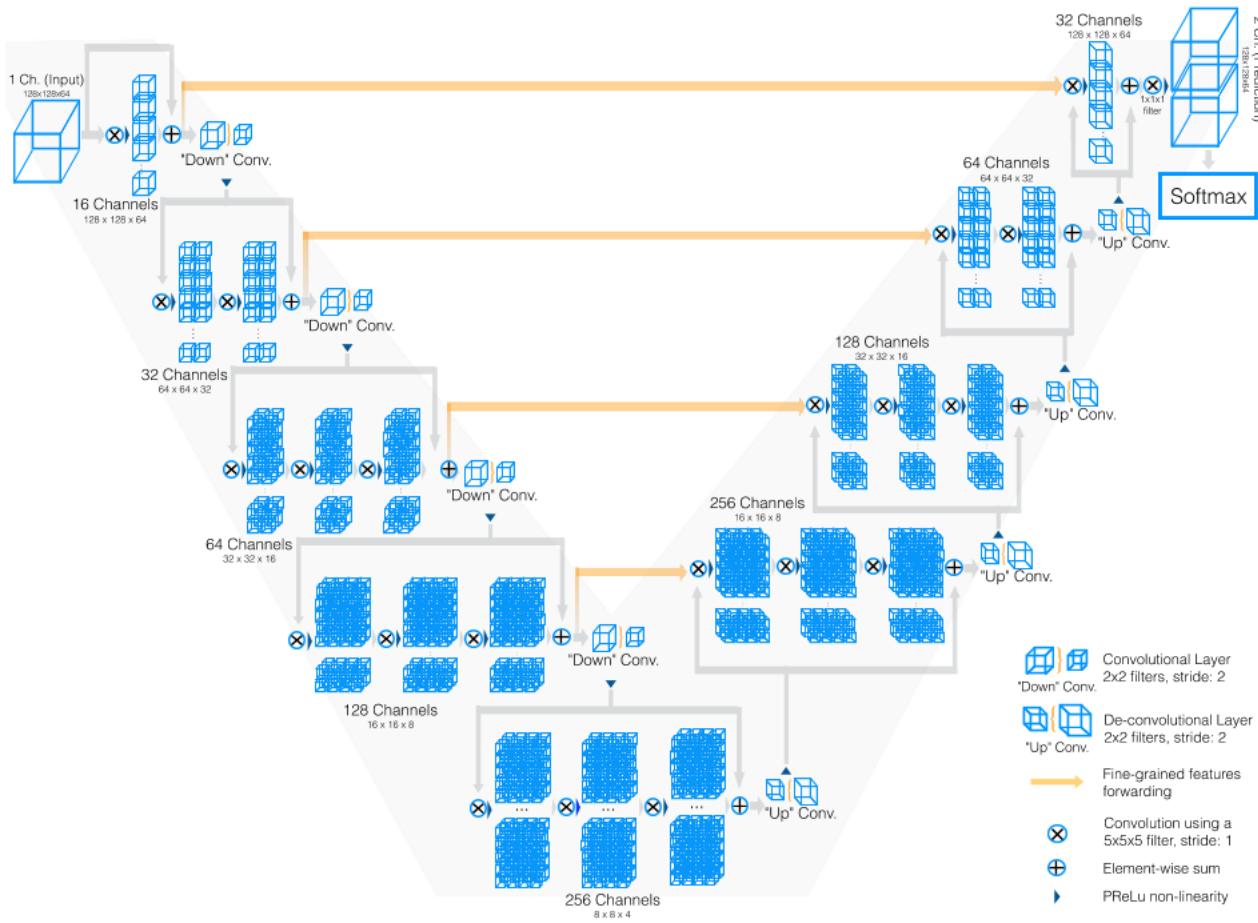
Deep Residual Learning for Image Recognition, 2015

U-net (2015)



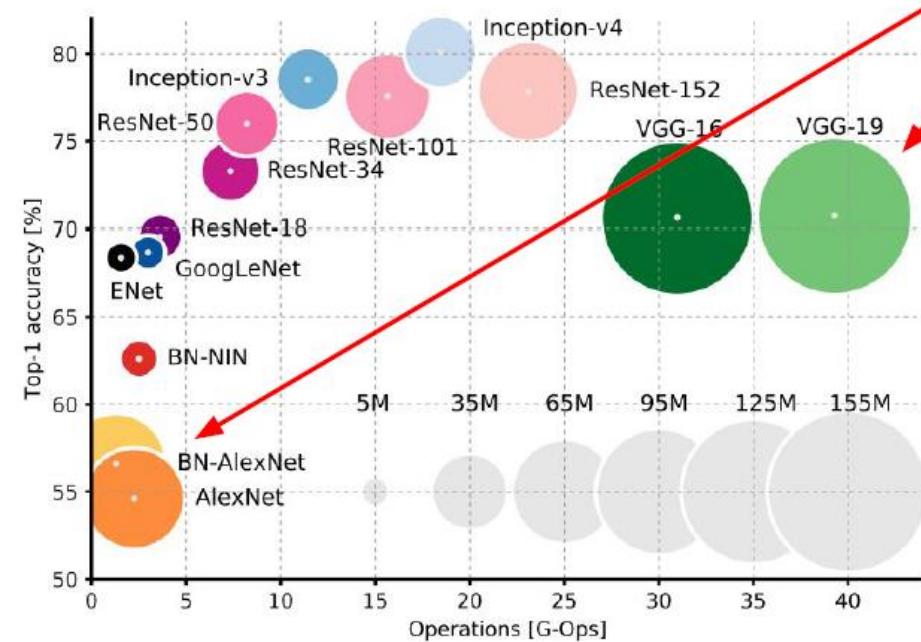
U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

V-Net (2016)



V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation, 2016

CNN Architectures



AlexNet and VGG have tons of parameters in the fully connected layers

AlexNet: ~62M parameters

FC6: 256x6x6 -> 4096: 38M params
FC7: 4096 -> 4096: 17M params
FC8: 4096 -> 1000: 4M params
~59M params in FC layers!

Summary

- CNN stack Conv, Pool, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of Pool/ FC layers



University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

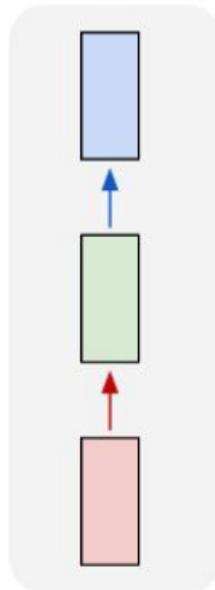
Machine Learning

Topic 16 – RNN,LSTM

Zheng Lu
2024 Autumn

“Vanilla” Neural Network

one to one



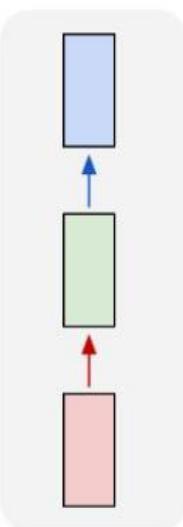
Vanilla Neural Networks

Recurrent Neural Networks

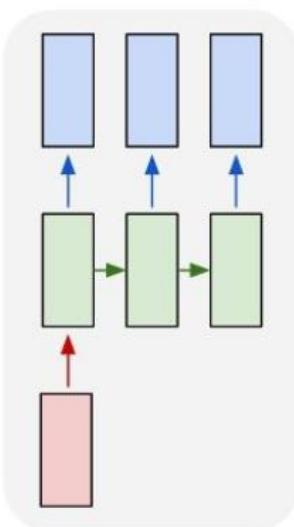
- Excellent models for problems more than one-to-one
 - Time series prediction and classification.
 - Sequence prediction and classification.
 - Simplify some problems that are difficult for multi-layer perceptron.

RNN: Process Sequences

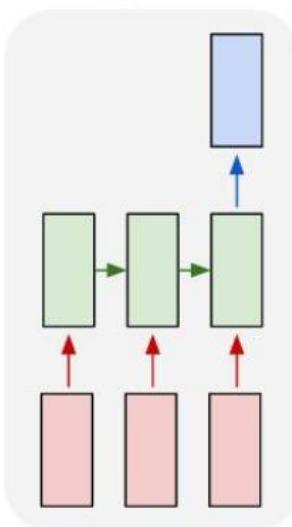
one to one



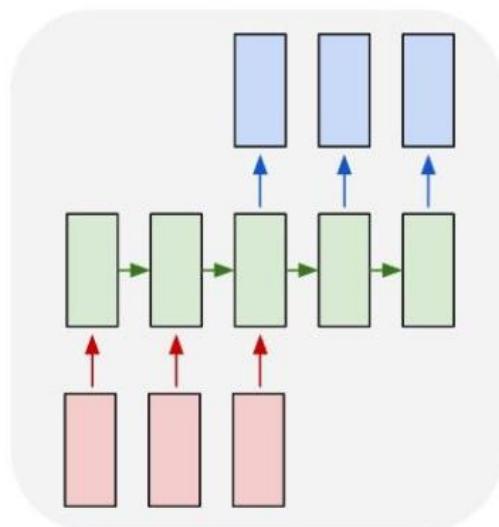
one to many



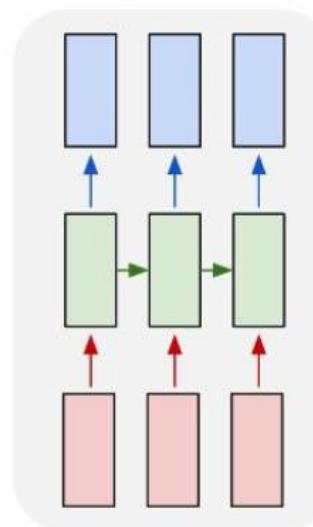
many to one



many to many



many to many



→
e.g. **Image Captioning**
image -> sequence of words

Image Caption



A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass



Two people walking on the beach with surfboards

A tennis player in action on the court

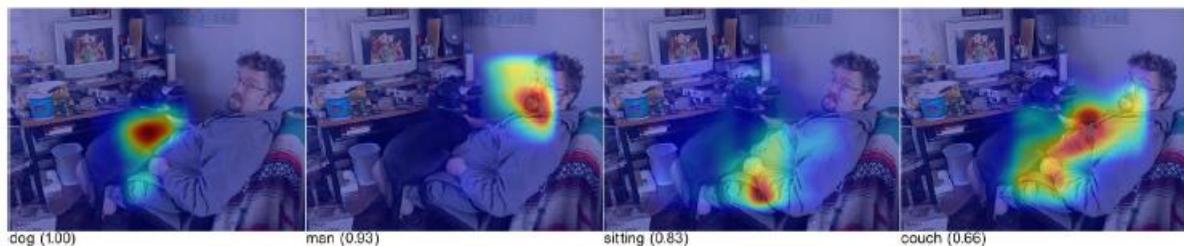
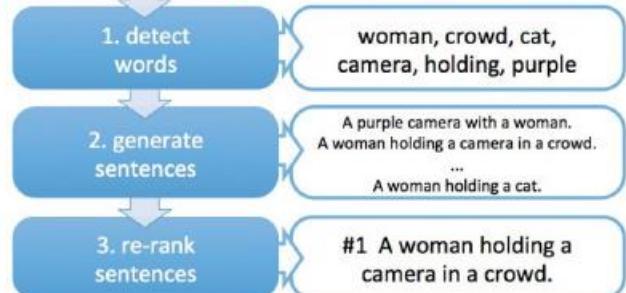
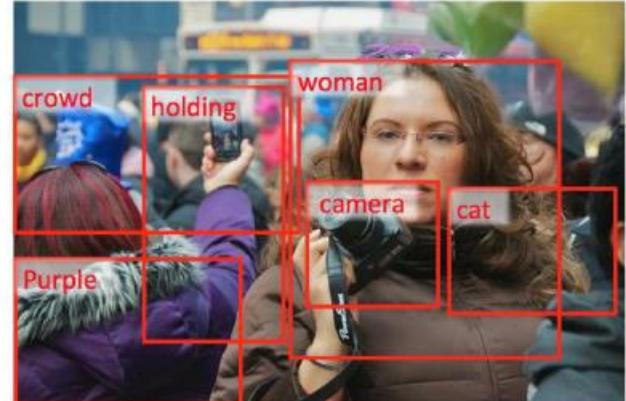
Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

Image Caption



a man sitting on a couch with a dog
a man sitting on a chair with a dog in his lap

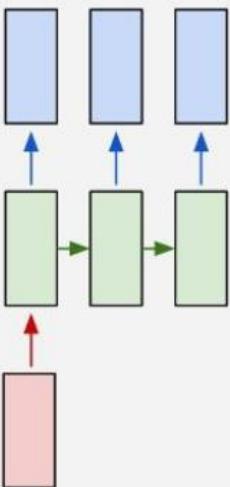


RNN: Process Sequences

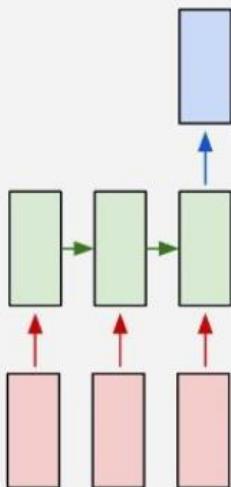
one to one



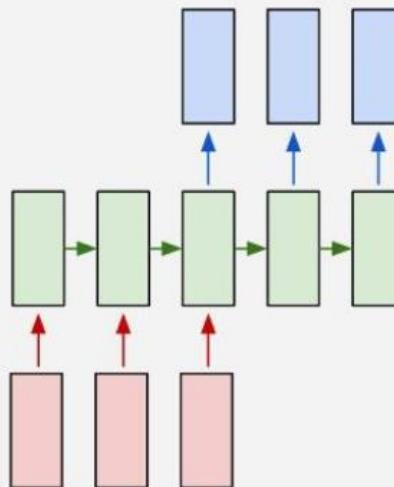
one to many



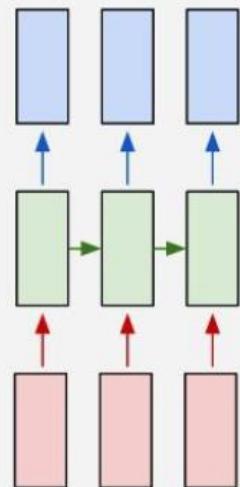
many to one



many to many



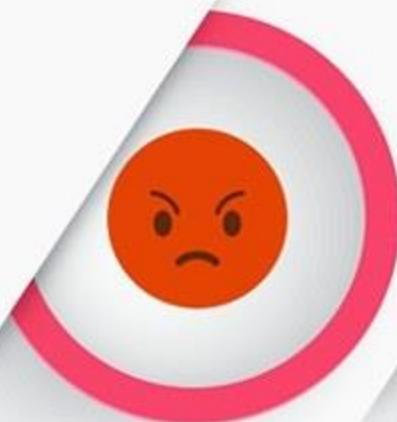
many to many



e.g. **Sentiment Classification**
sequence of words -> sentiment

Sentiment Classification

SENTIMENT ANALYSIS



NEGATIVE

Totally dissatisfied with the service. Worst customer care ever.



NEUTRAL

Good Job but I will expect a lot more in future.

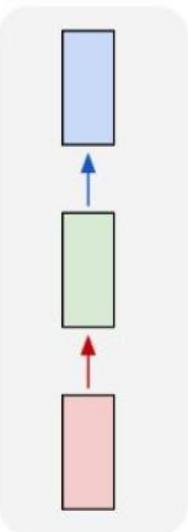


POSITIVE

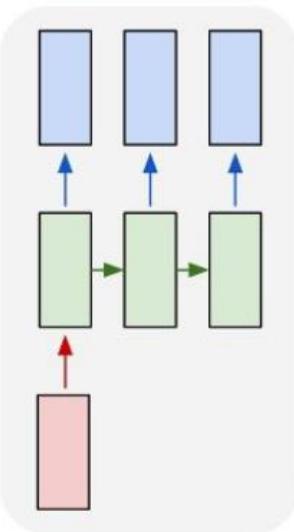
Brilliant effort guys! Loved Your Work.

RNN: Process Sequences

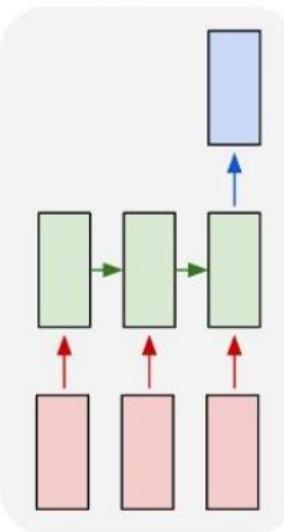
one to one



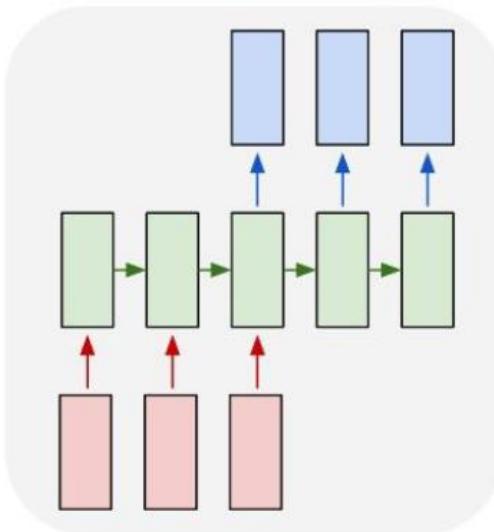
one to many



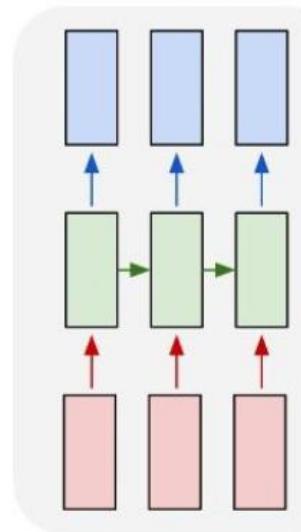
many to one



many to many

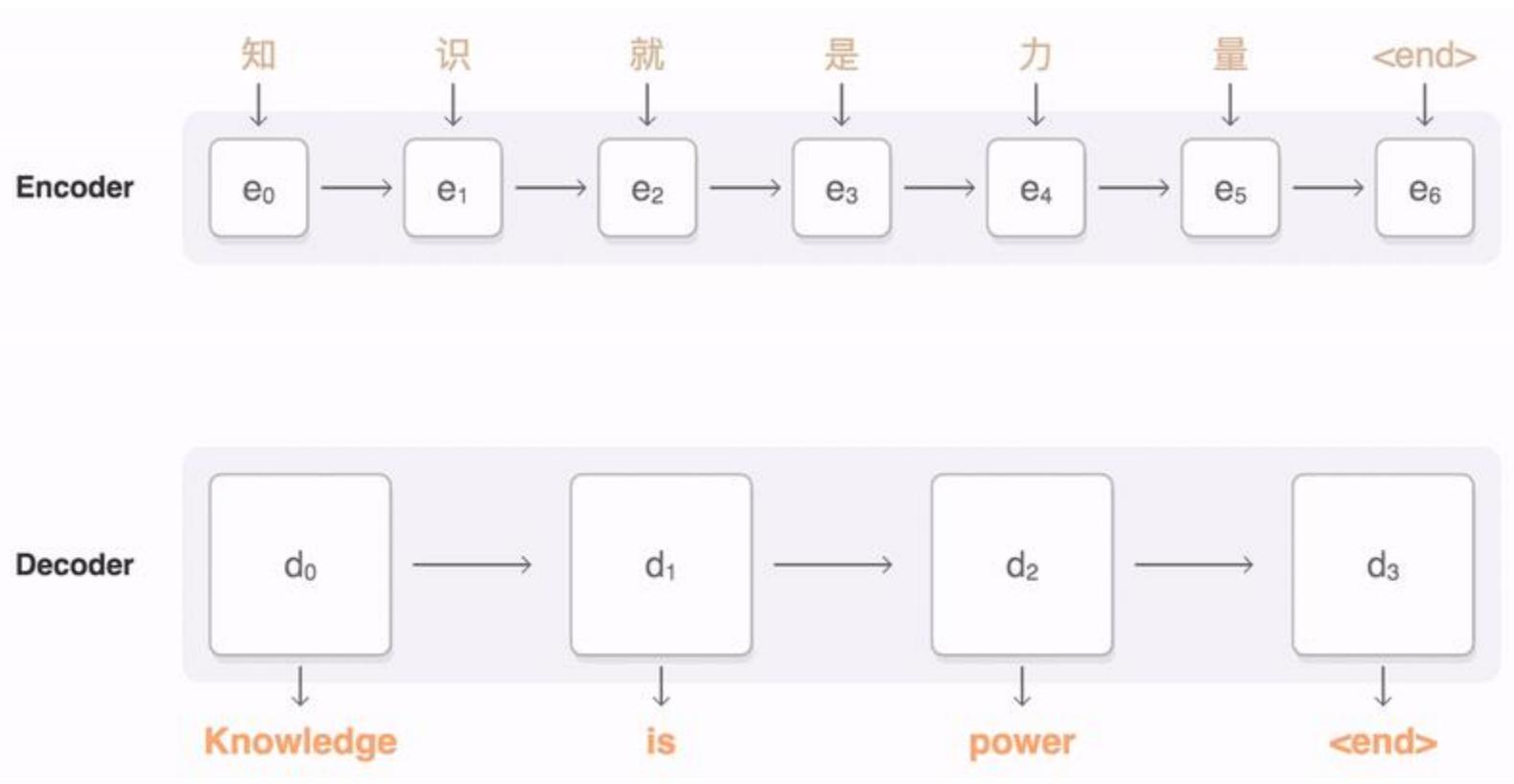


many to many



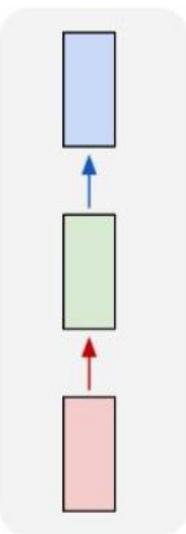
↑
e.g. **Machine Translation**
seq of words -> seq of words

Machine Translation

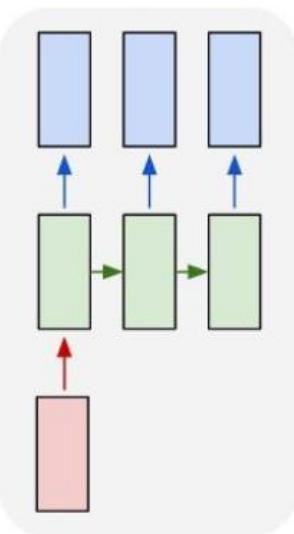


RNN: Process Sequences

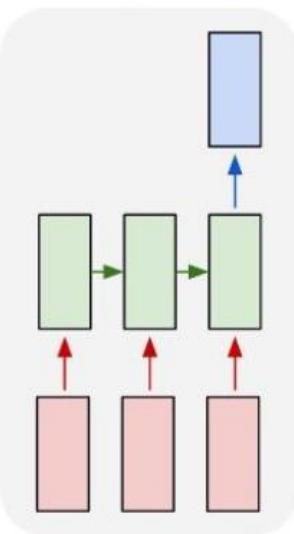
one to one



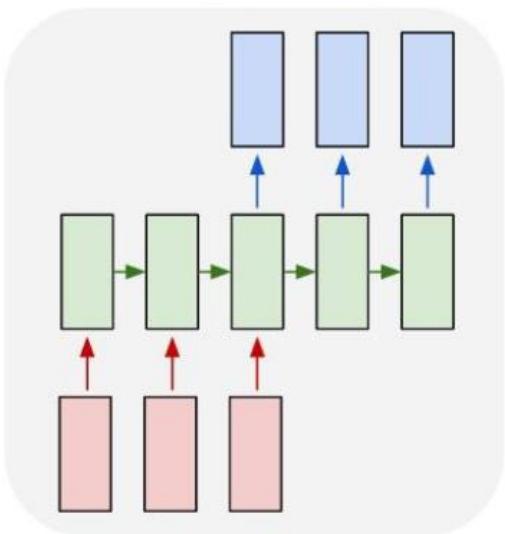
one to many



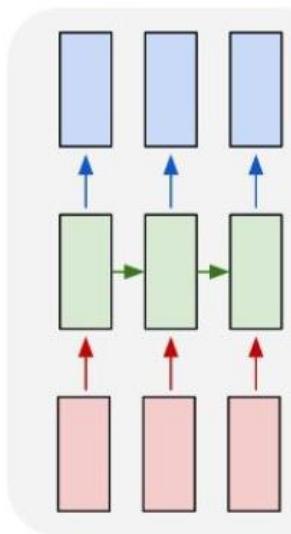
many to one



many to many



many to many



e.g. Video classification on frame level

Video Classification (frame level)

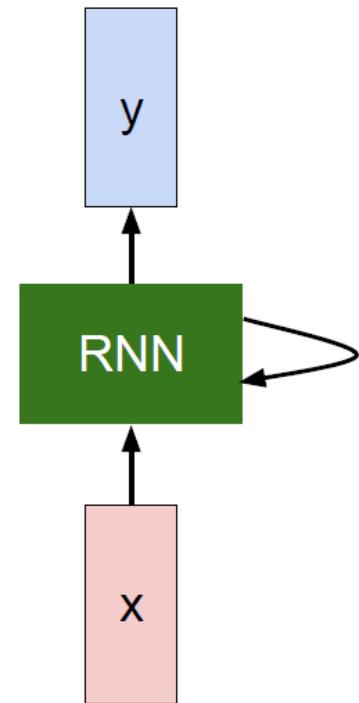


RNN

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

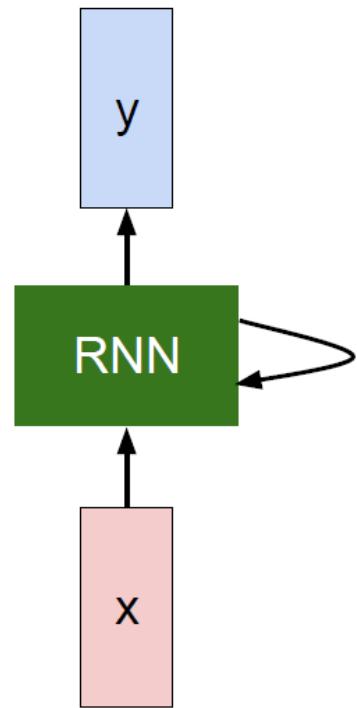
new state old state input vector at
some function some time step
with parameters W



RNN

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

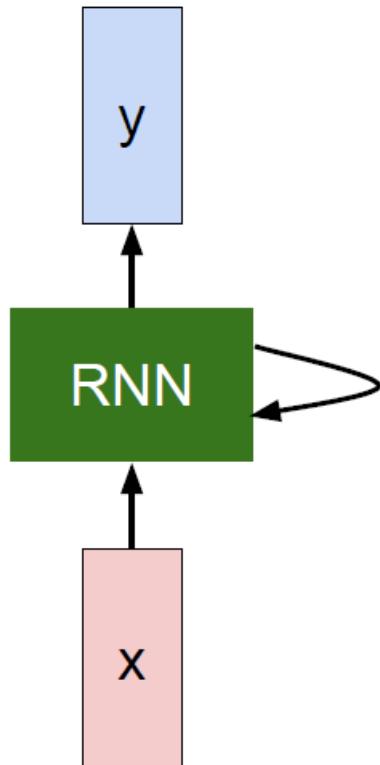
$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

RNN

The state consists of a single “*hidden*” vector \mathbf{h} :



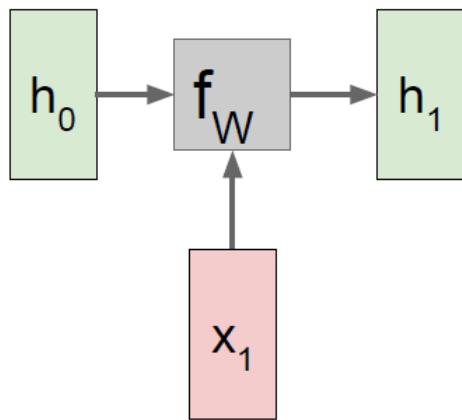
$$h_t = f_W(h_{t-1}, x_t)$$



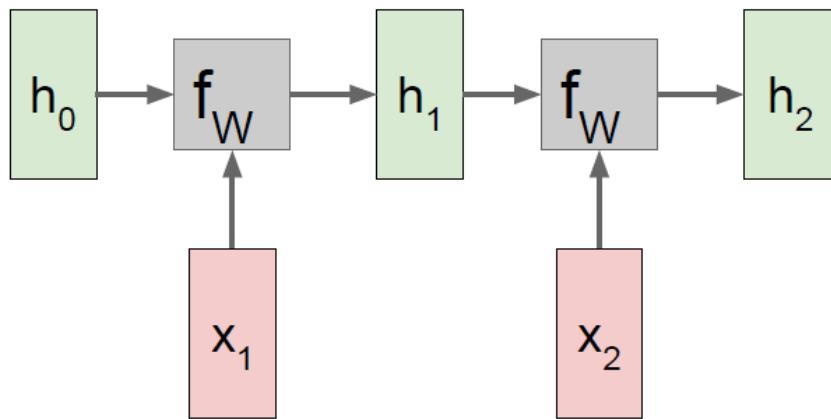
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

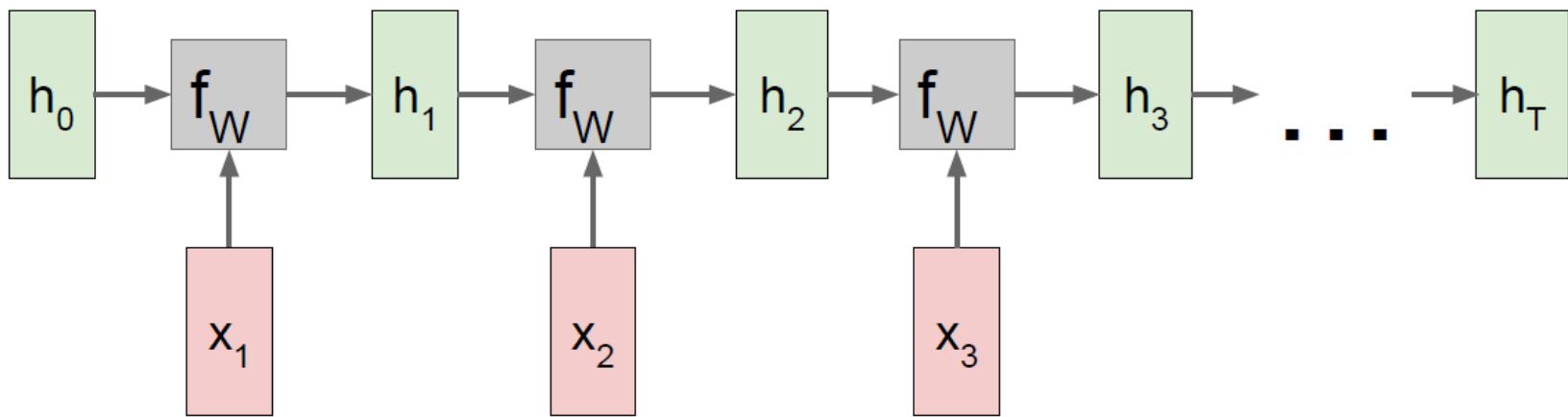
RNN: Computational Graph



RNN: Computational Graph

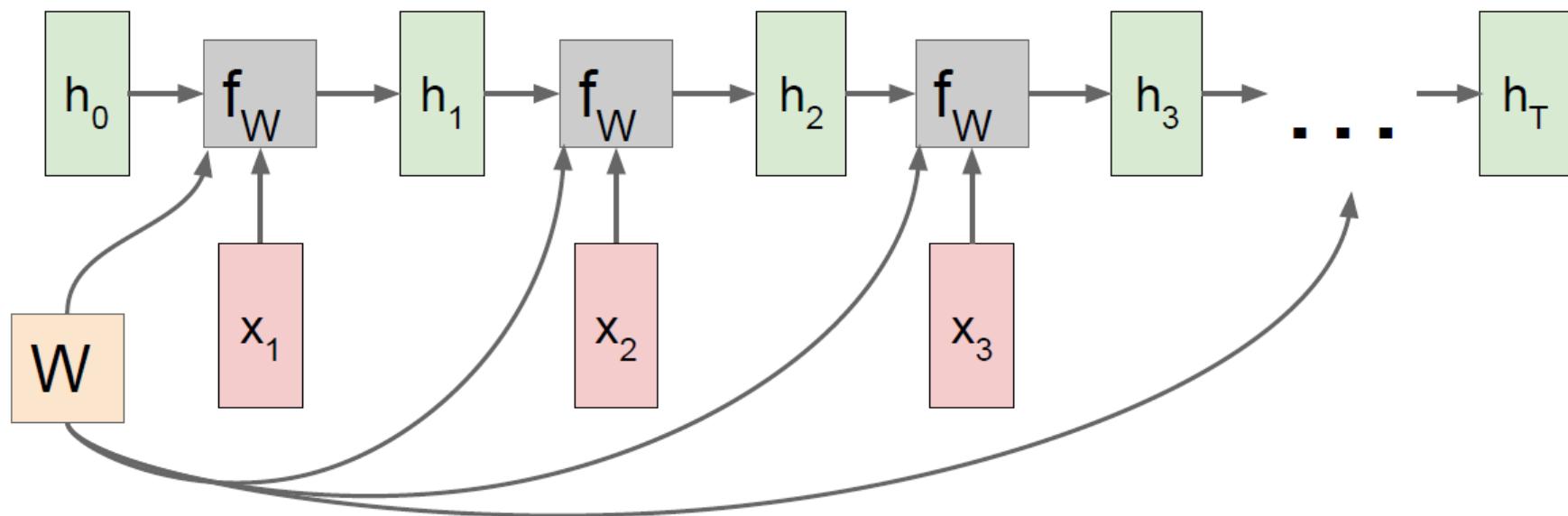


RNN: Computational Graph

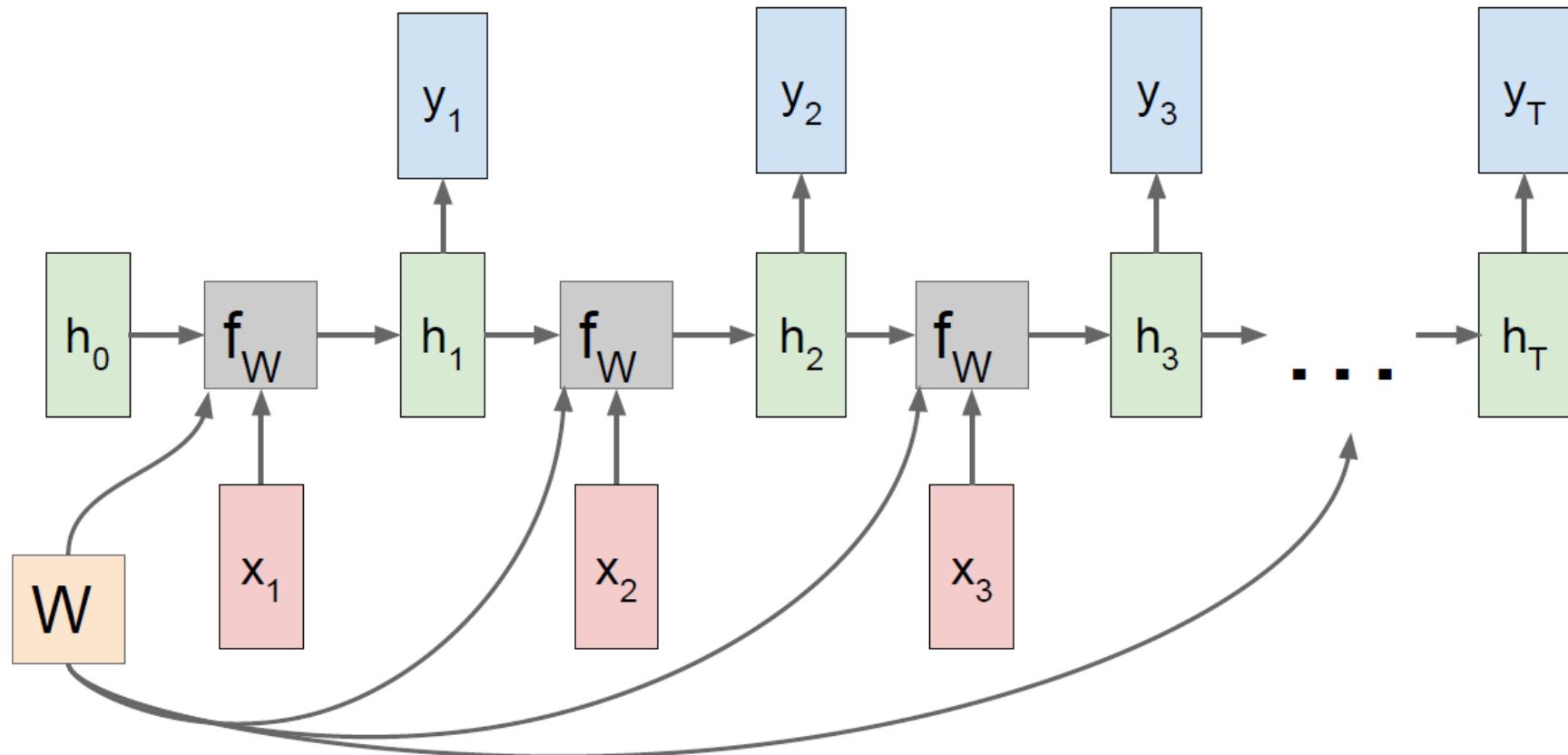


RNN: Computational Graph

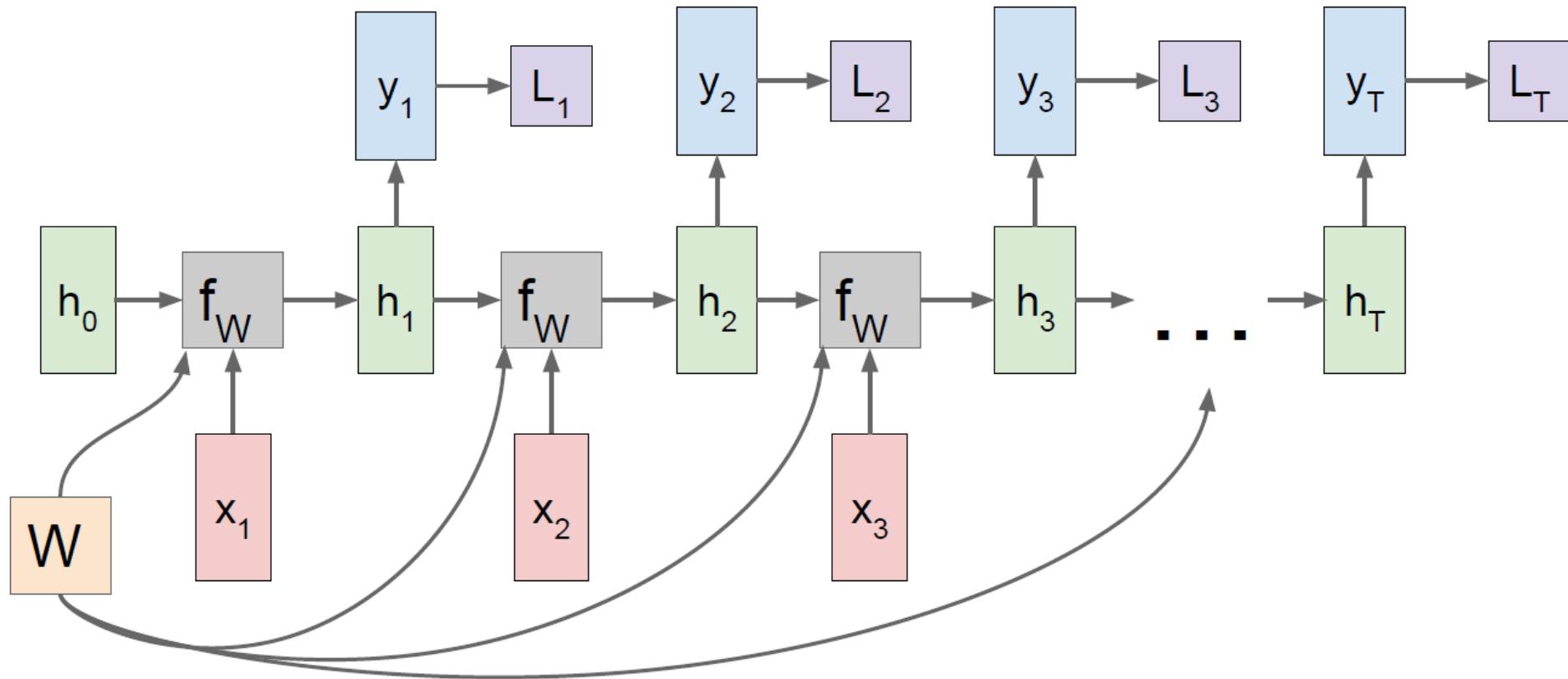
Re-use the same weight matrix at every time-step



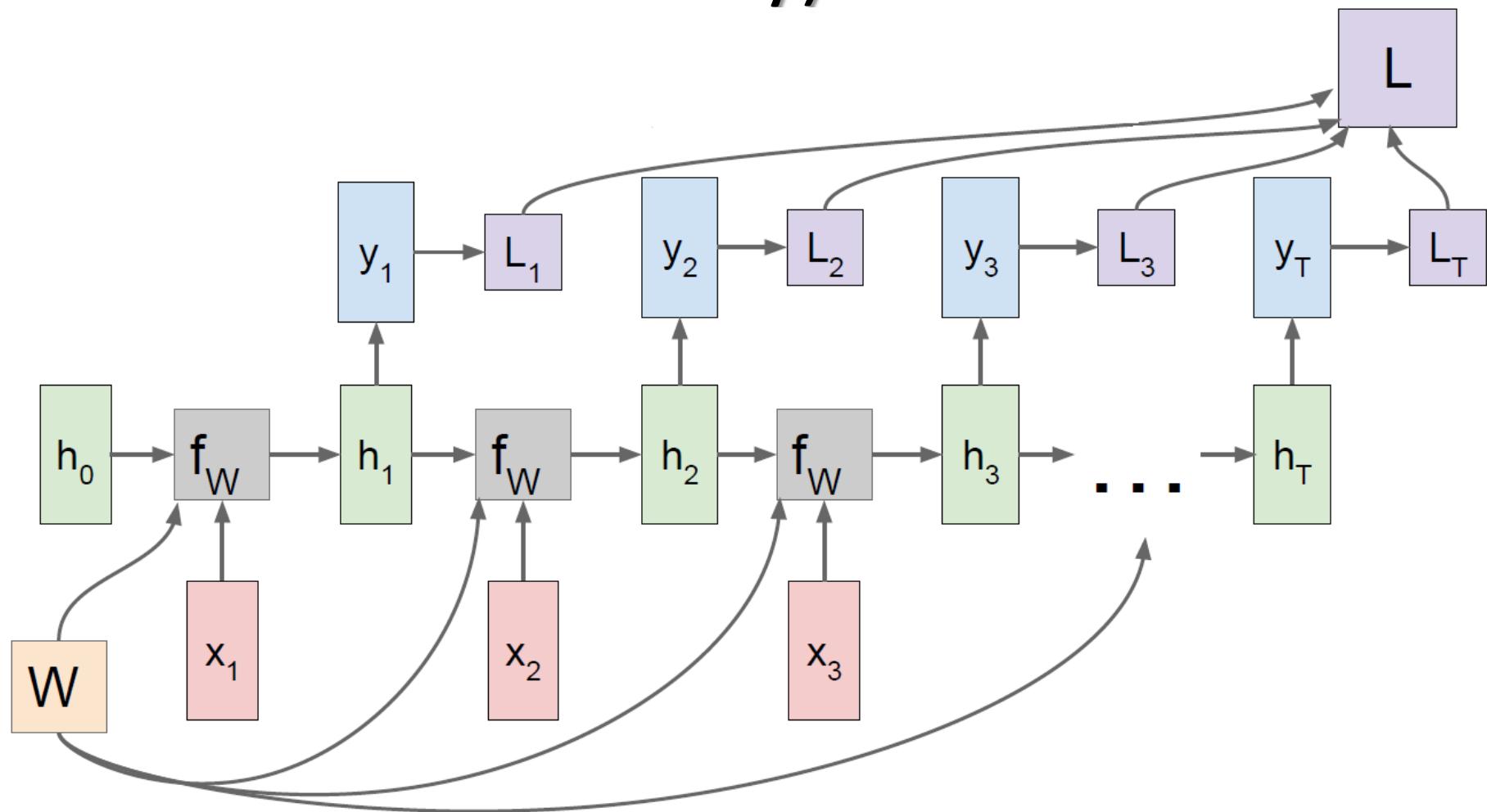
RNN: Computational Graph (Many to Many)



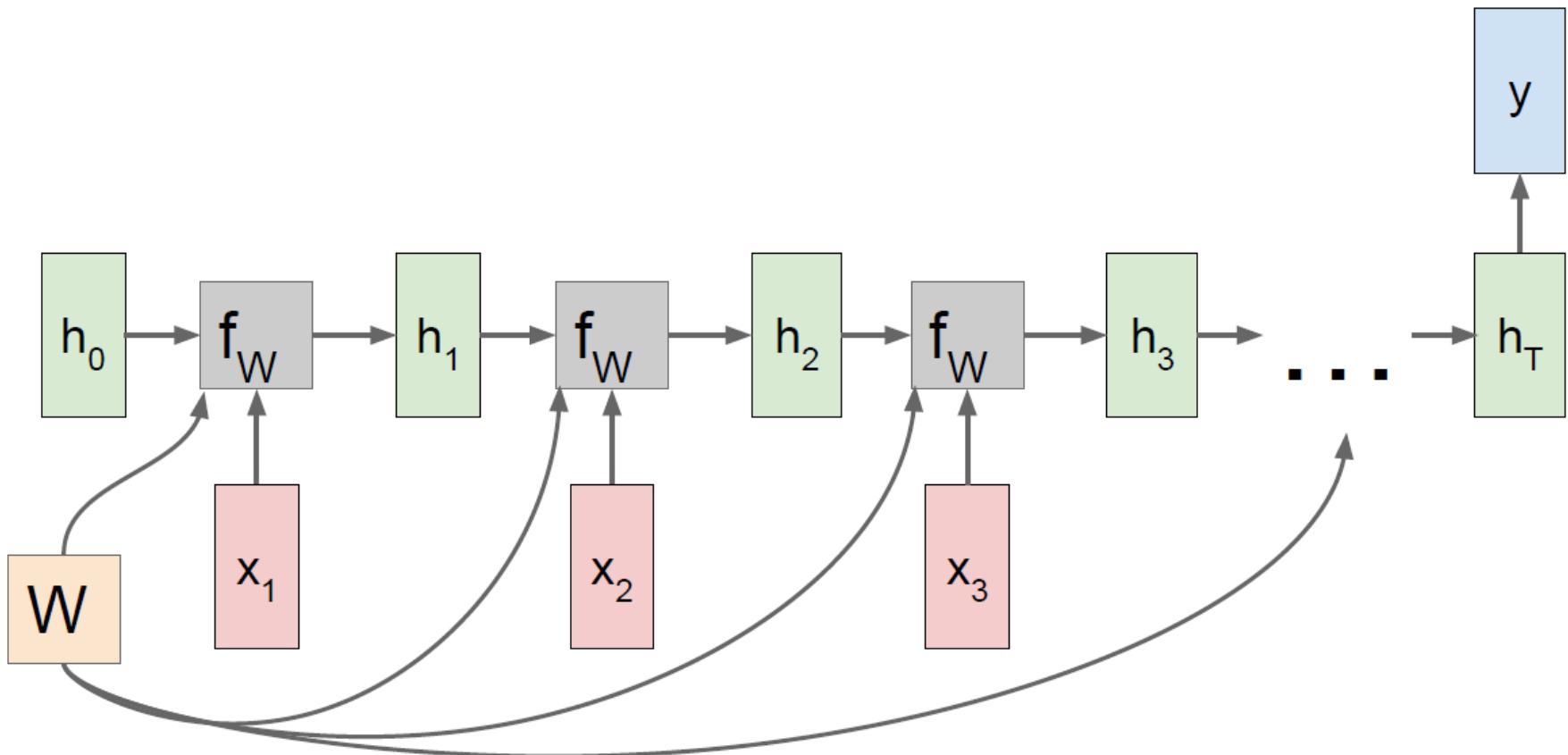
RNN: Computational Graph (Many to Many)



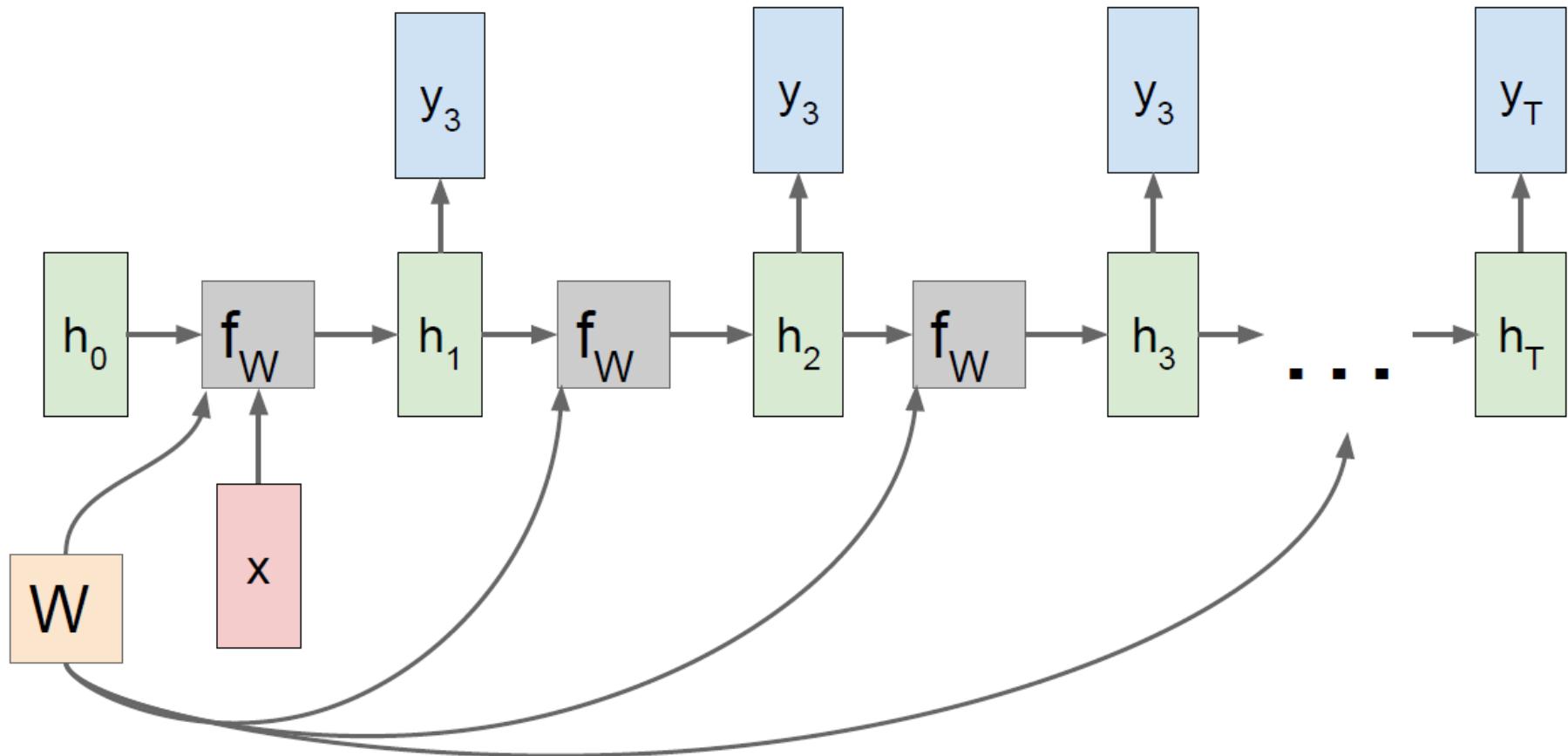
RNN: Computational Graph (Many to Many)



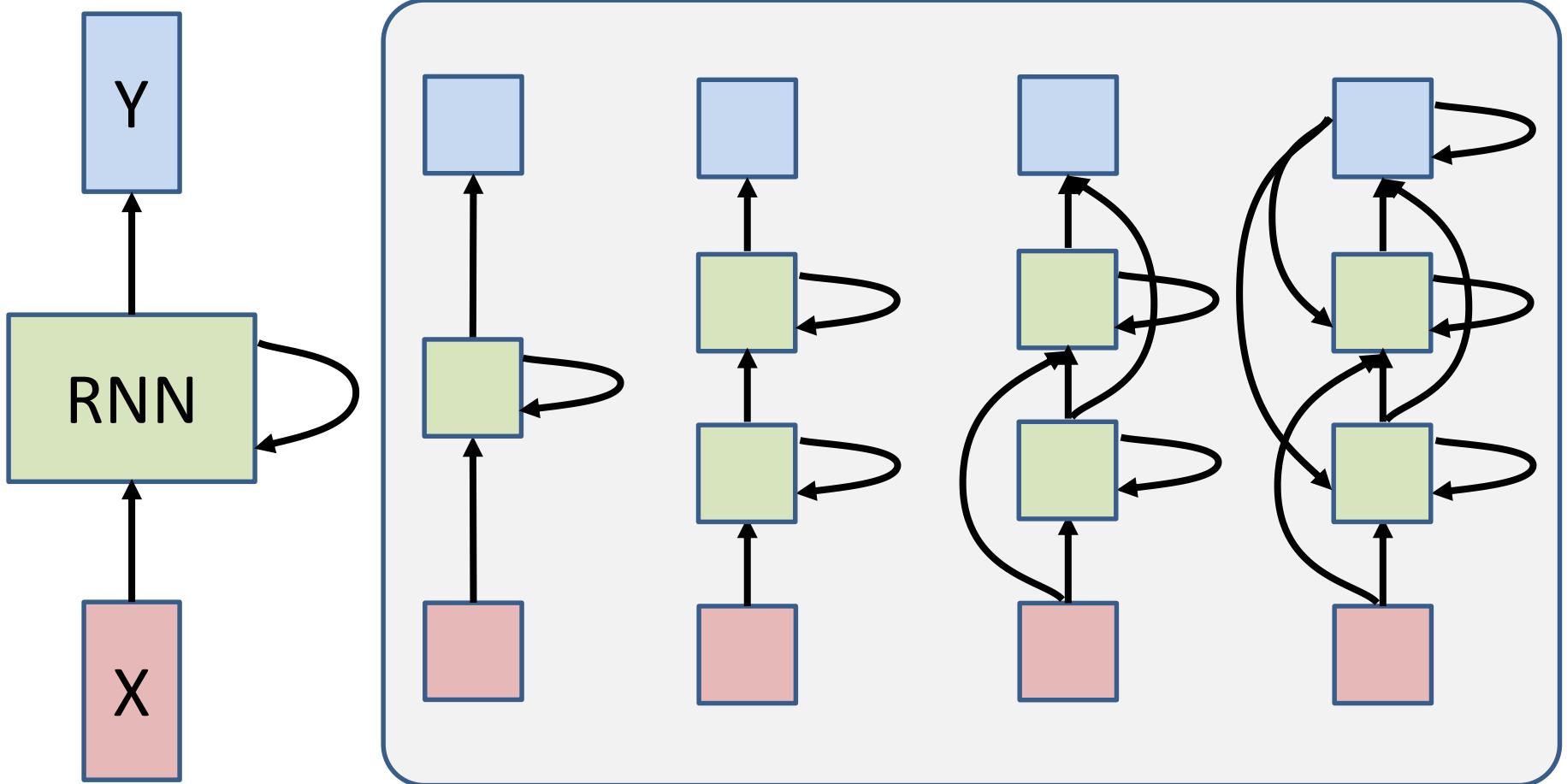
RNN: Computational Graph (Many to One)



RNN: Computational Graph (One to Many)



RNN: other design



RNN can be designed very sophisticatedly with different layers different ways of recurrency

RNN

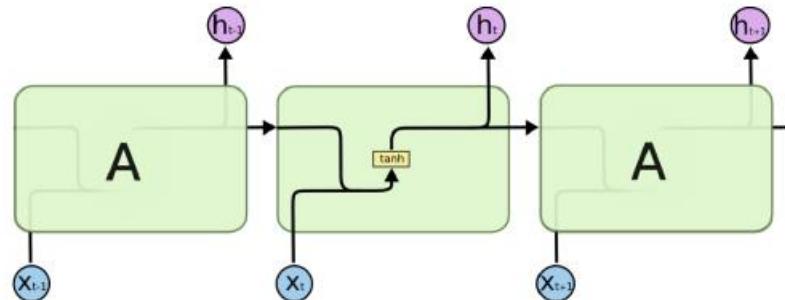
- In theory RNN retains information from the infinite past.
 - All past hidden state has influence on the future state.
- In practice RNN has little response to the early states.
 - Little memory over what seen before.
 - The hidden outputs blowup or shrink to zeros.
 - The “memory” also depends on activation functions.
 - ReLU and Sigmoid do not work well. Tanh is OK but still not “memorize” for too long.
- Vanishing gradient problem
 - Deeper layers do not have meaningful weights.

Long-Term Dependency

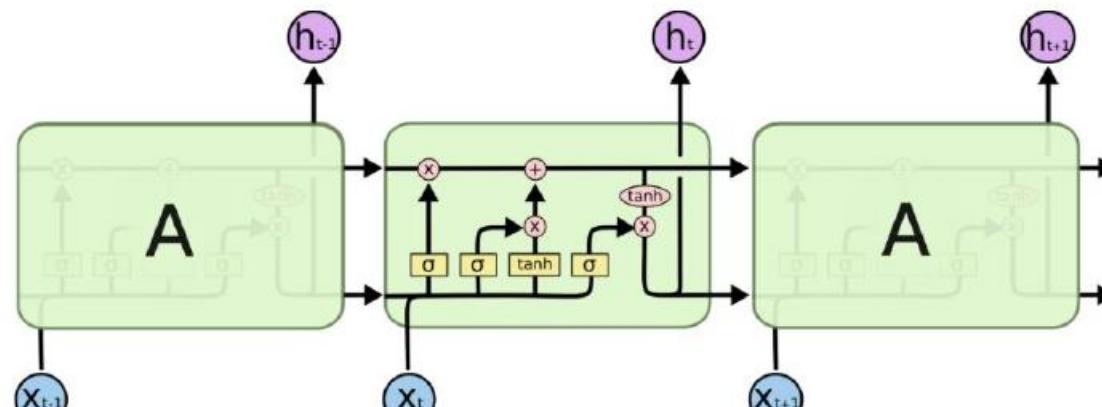
- In theory, vanilla RNNs can handle arbitrarily long term dependence
- In practice, it's difficult
- Long-term dependency:
 - **Bob** likes **apples**. He is hungry and decided to have a snack. So now he is eating an **apple**.

LSTM: Gates Regulate

Vanilla RNN:



LSTM:



Neural Network Layer

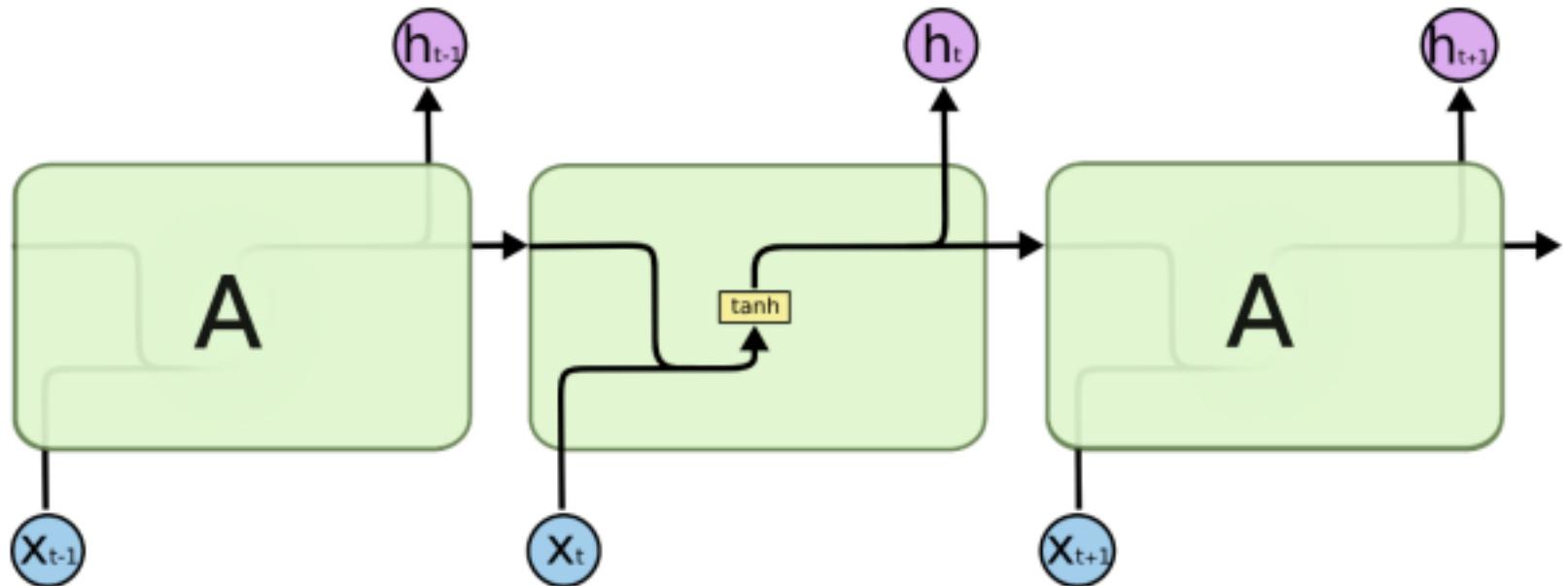
Pointwise Operation

Vector Transfer

Concatenate

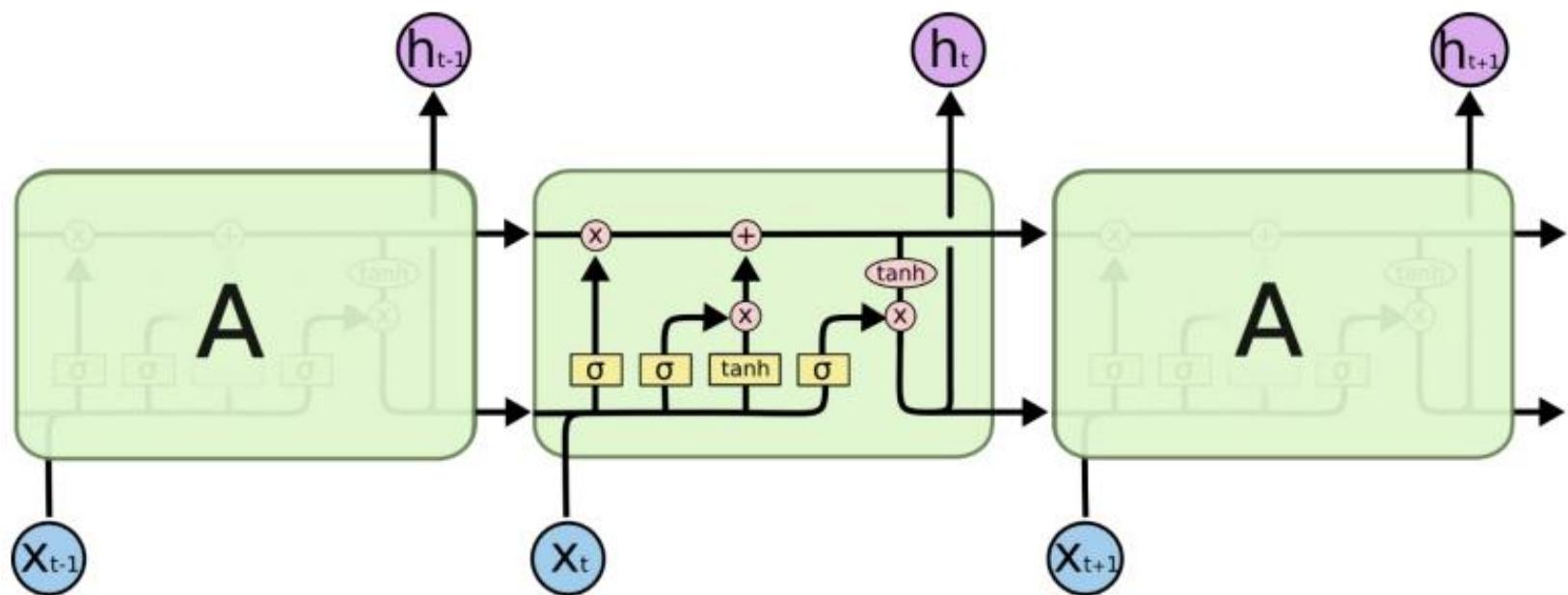
Copy

RNN VS LSTM

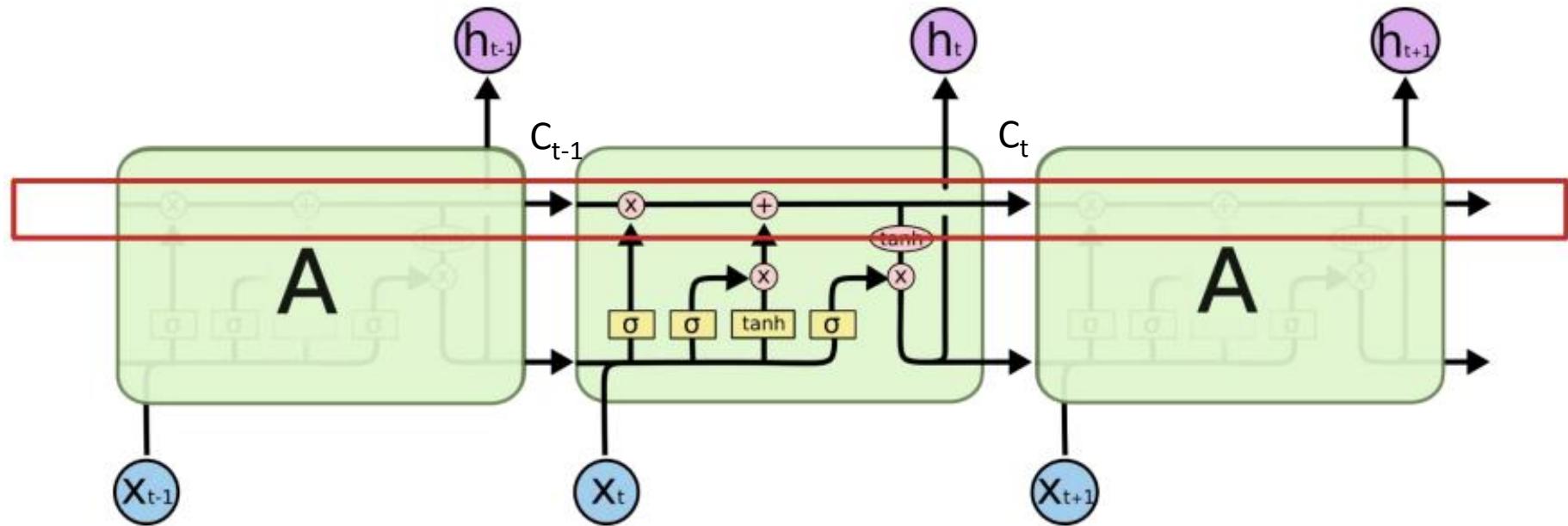


- Recurrent neurons receive past recurrent outputs and current input as inputs.
- Processed through a $\text{tanh}()$ activation function
- Current recurrent output passed to next higher layer and next time step.

LSTM



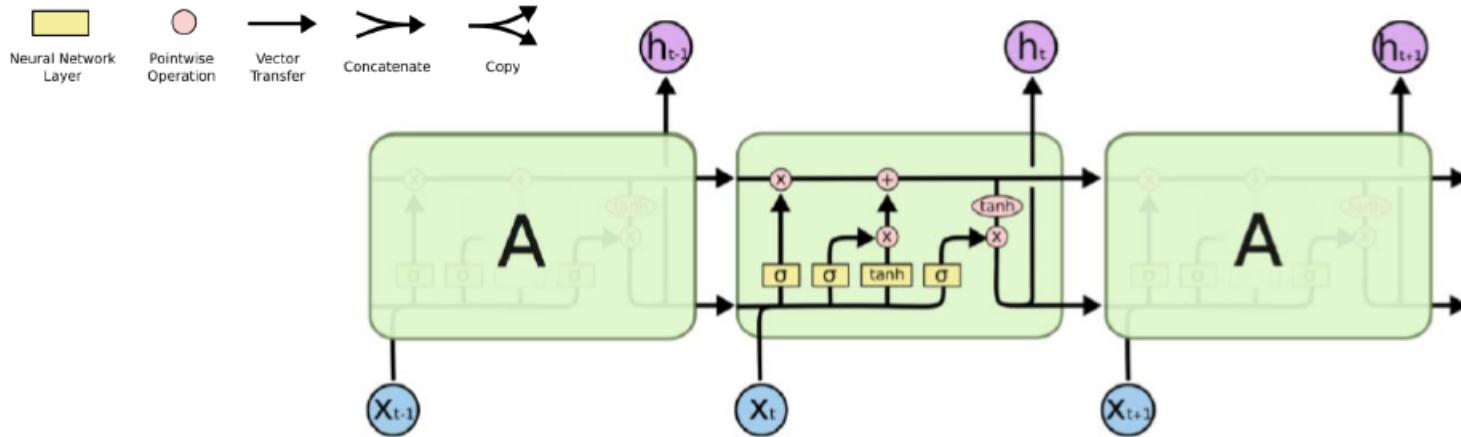
LSTM



Constant Error Carousel

- Key of LSTM: a remembered cell state
- C_t is the linear history carried by the constant error carousel.
- Carries information through and only effected by a gate
 - Addition of history (gated).

LSTM

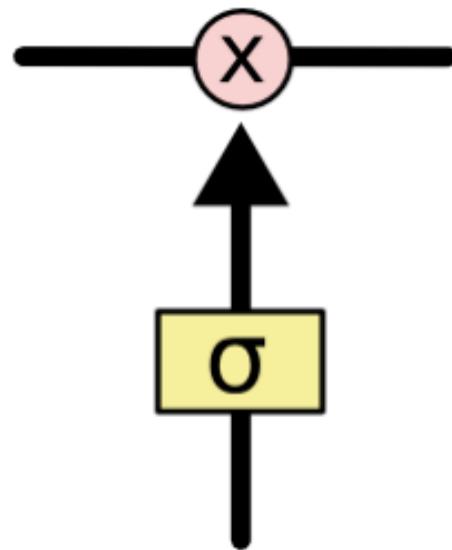


Bob and Alice are having lunch. Bob likes apples. Alice likes oranges.
She is eating an orange.

Conveyer belt for **previous state** and **new data**:

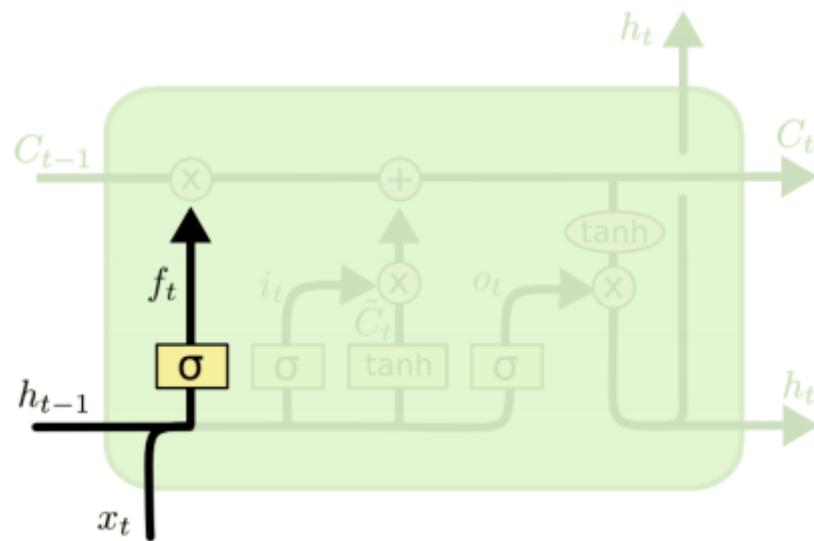
1. Decide what to forget (state)
2. Decide what to remember (state)
3. Decide what to output (if anything)

LSTM - Gate



- A simple sigmoid function to project output in range (0, 1).
 - Information is let through (~1)
 - Information is not let through (~0)
- \otimes : element-wise multiplication.

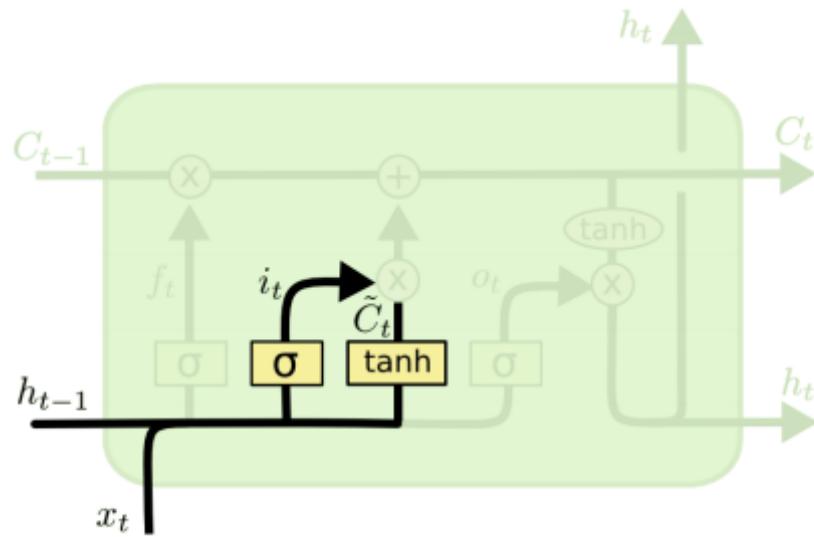
LSTM – Forget Gate



$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

- The first gate determines whether to carry over the history or forget it
 - Called “forget” gate.
 - Actually, determine how much history to carry over.
 - The memory C and hidden state h are distinguished.

LSTM – Input Gate



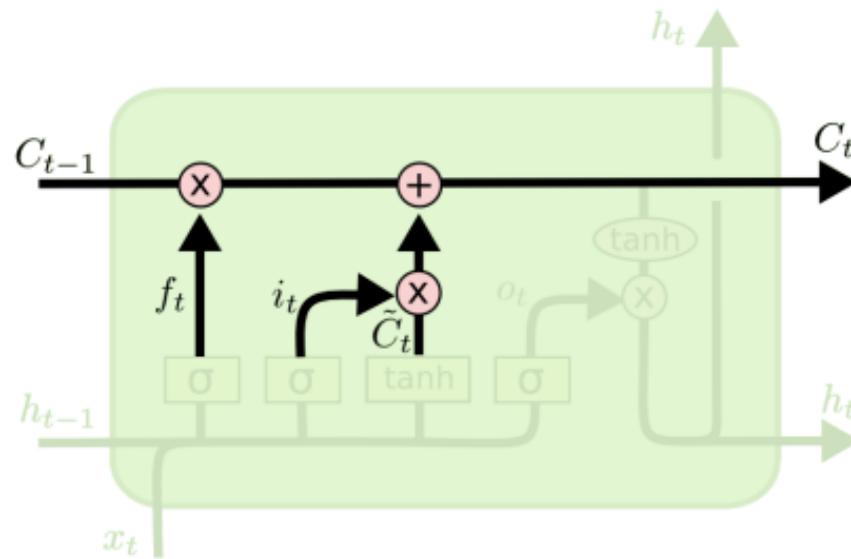
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

The second gate has two parts

- A \tanh unit determines if there is something new or interesting in the input.
- A gate decides if it is worth remembering.

LSTM – Memory Cell Update

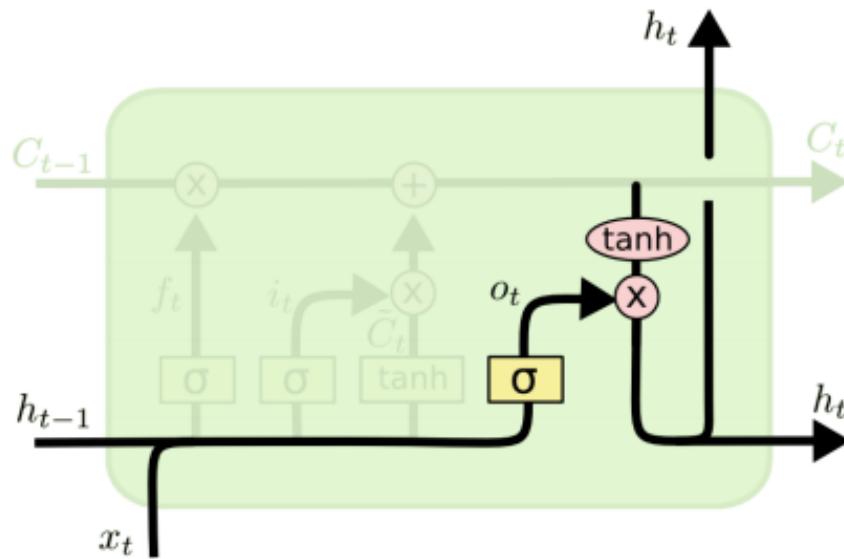


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Add the output of input gate to the current memory cell

- After the forget gate.
- \oplus : Element-wise addition.
- Perform the forgetting and the state update

LSTM – Output and Output Gate

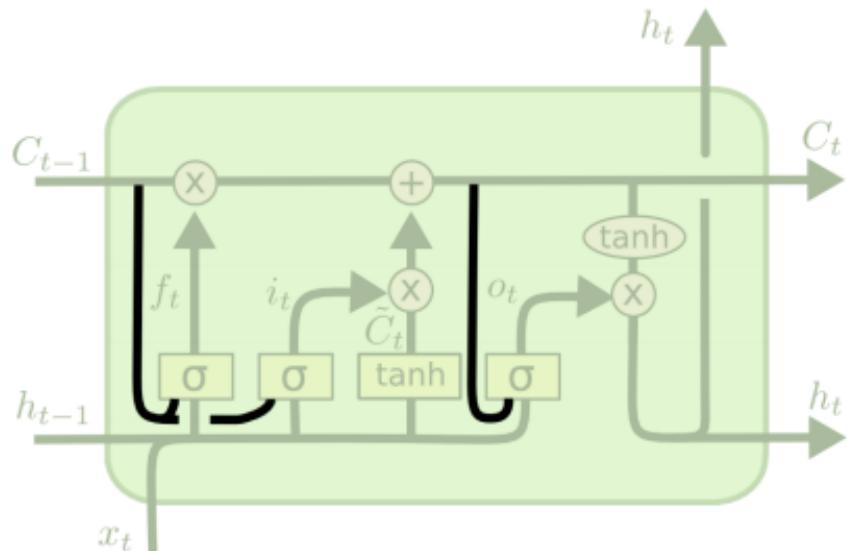


$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh (C_t)$$

The output of the memory cell

- Similar to input gate.
- A *tanh* unit over the memory to output in range [-1, 1].
- A *sigmoid* unit [0,1] decide the filtering.
- Note the memory is carried through without *tanh*.

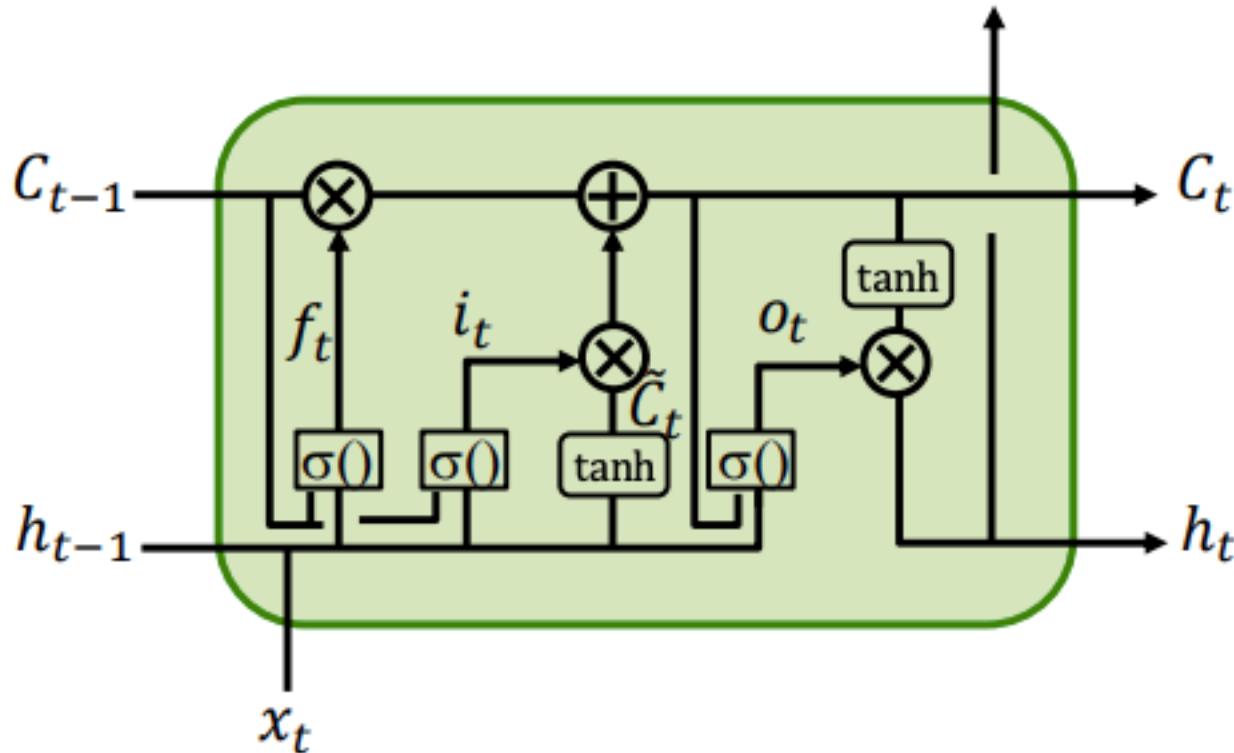
LSTM – the “Peephole” Connection



$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$

Let the memory cell directly influence the gates!

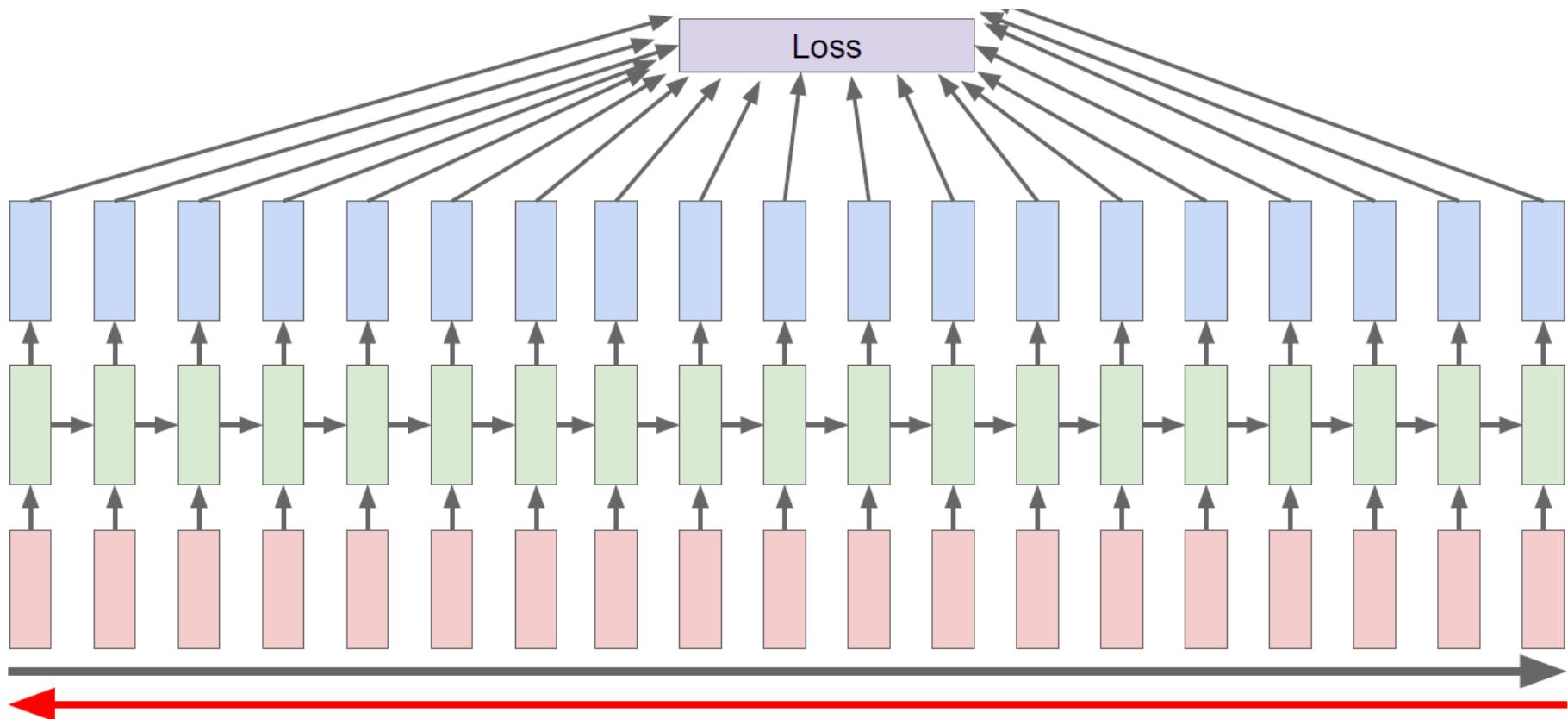
The Complete LSTM Unit



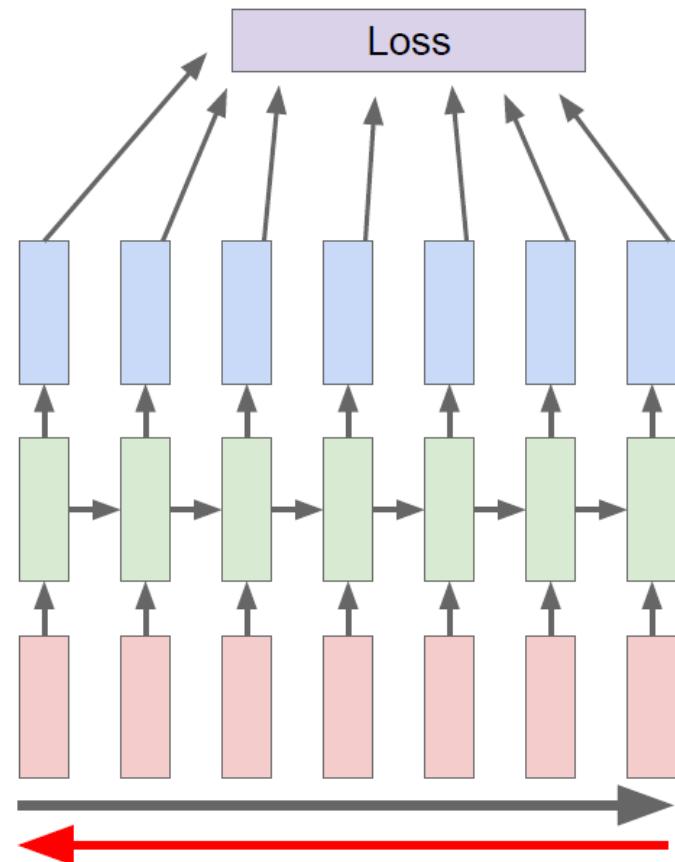
Input, output, forget gates with peephole connection

Back Propagation Through Time (BPTT)

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

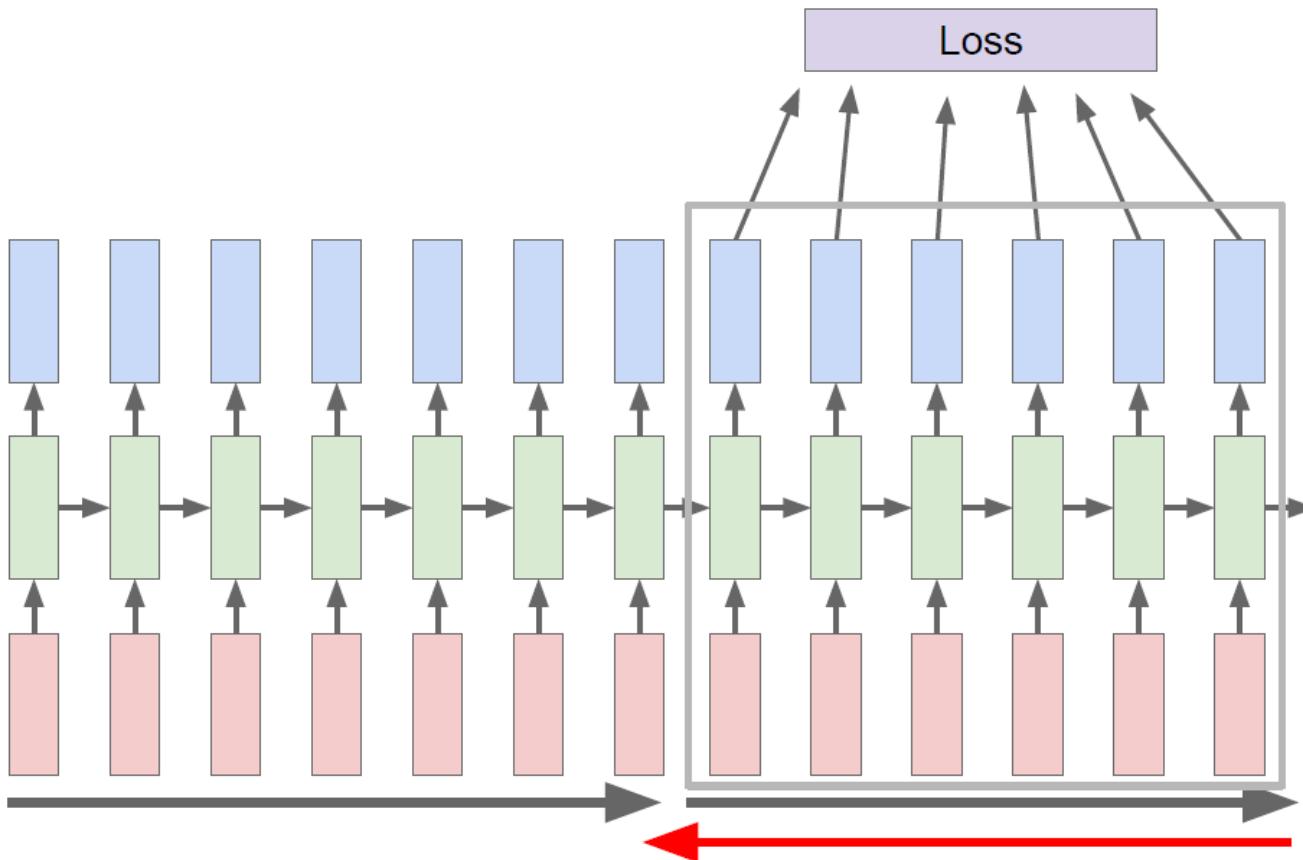


Truncated BPTT



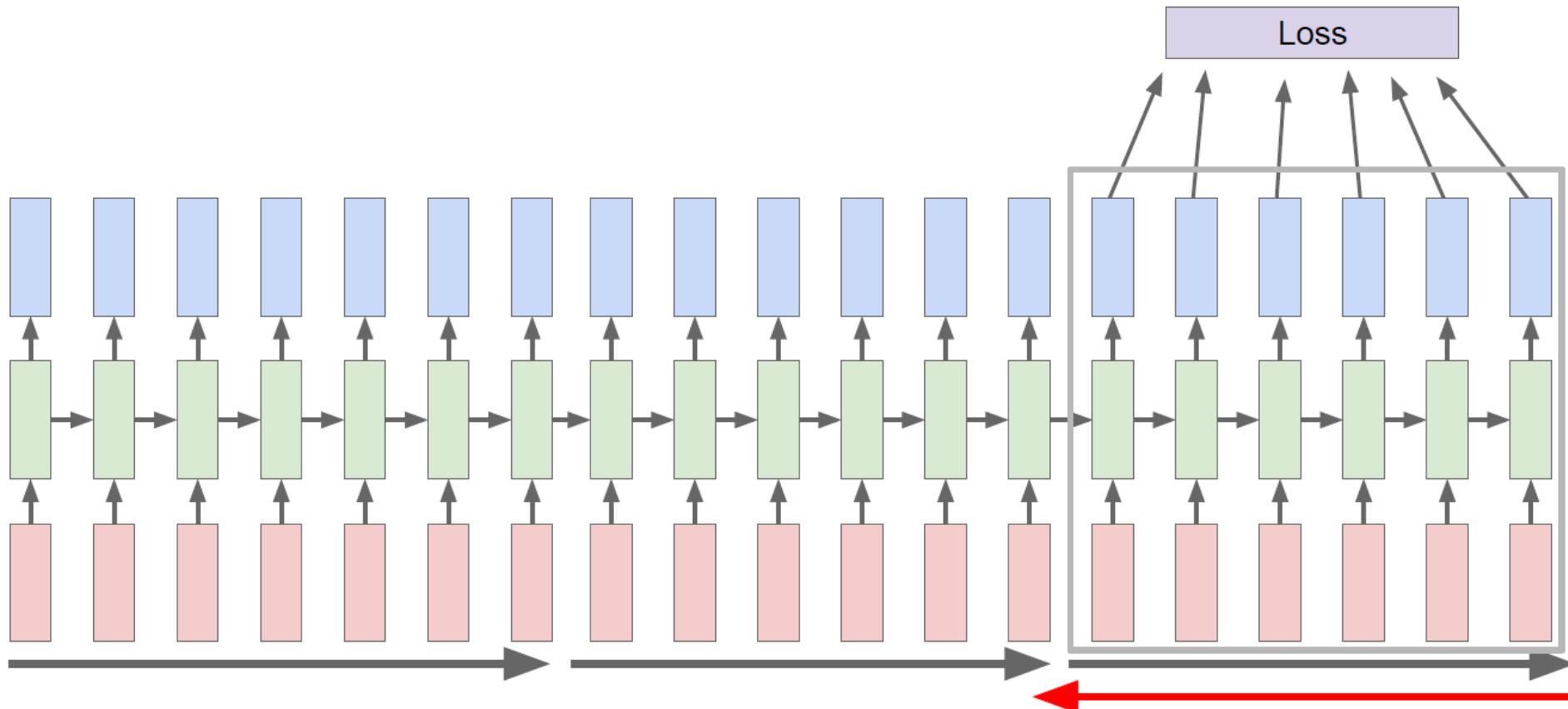
Run forward and backward
through chunks of the
sequence instead of whole
sequence

Truncated BPTT



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

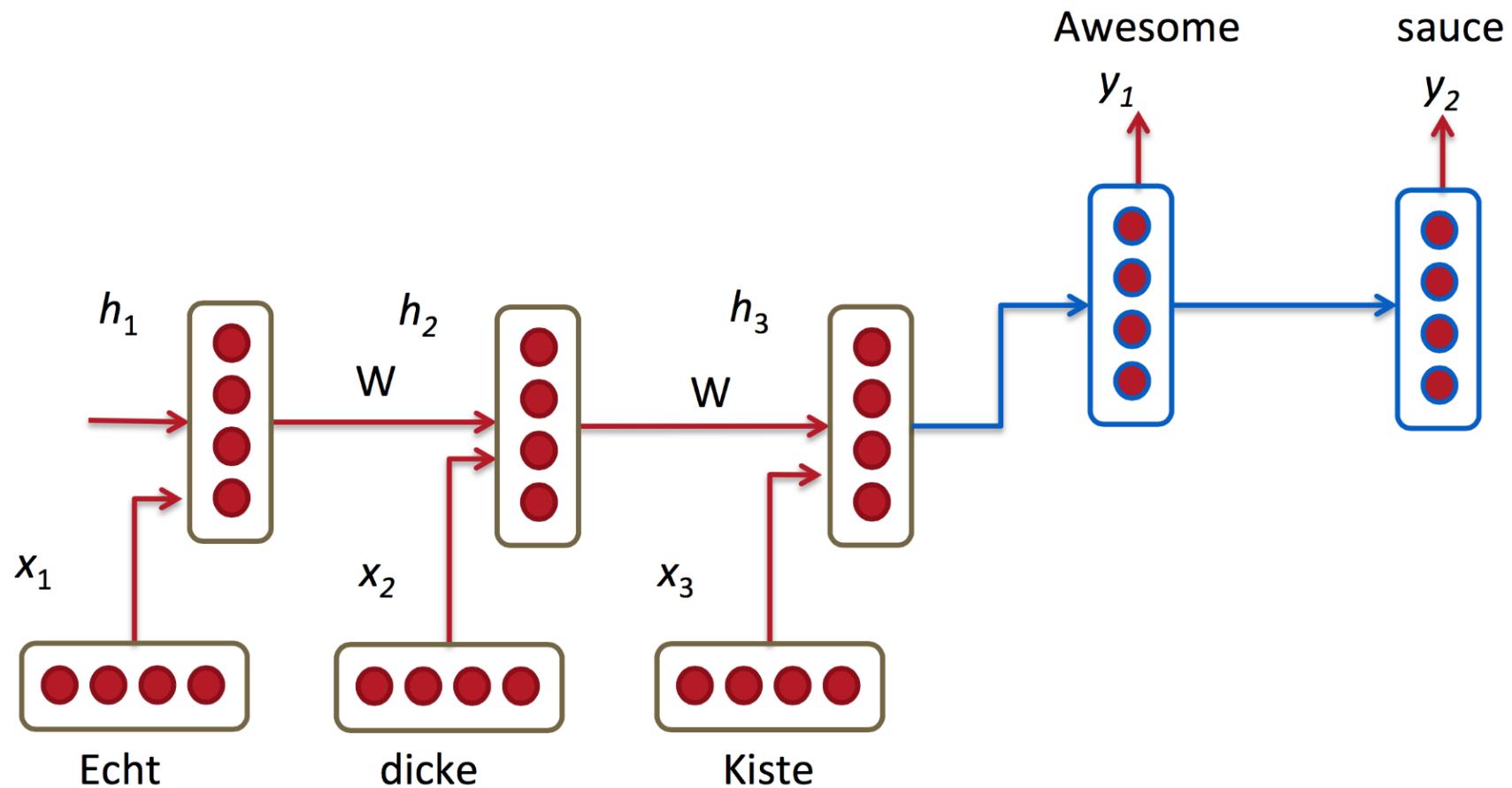
Truncated BPTT



Applications of LSTM

- Nowadays, considered as the default models for sequence labeling tasks.
- Does not suffer from Vanishing Gradient problem.
- Very powerful, especially in deeper networks.
- Very useful when you have a lot of data.

Machine Translation



Machine Translation

| Method | test BLEU score (ntst14) |
|--|--------------------------|
| Bahdanau et al. [2] | 28.45 |
| Baseline System [29] | 33.30 |
| Single forward LSTM, beam size 12 | 26.17 |
| Single reversed LSTM, beam size 12 | 30.59 |
| Ensemble of 5 reversed LSTMs, beam size 1 | 33.00 |
| Ensemble of 2 reversed LSTMs, beam size 12 | 33.27 |
| Ensemble of 5 reversed LSTMs, beam size 2 | 34.50 |
| Ensemble of 5 reversed LSTMs, beam size 12 | 34.81 |

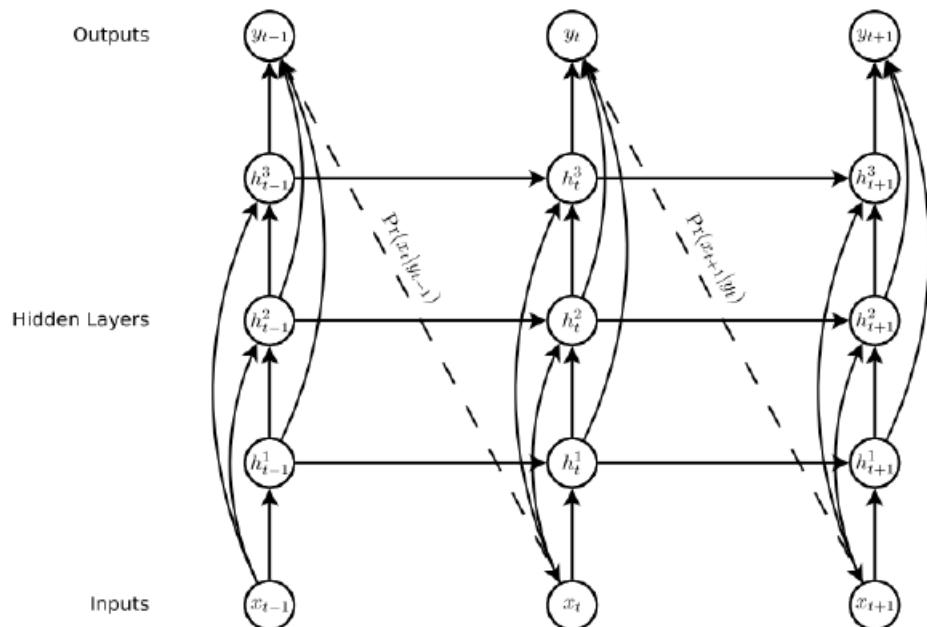
Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

| Method | test BLEU score (ntst14) |
|---|--------------------------|
| Baseline System [29] | 33.30 |
| Cho et al. [5] | 34.54 |
| Best WMT'14 result [9] | 37.0 |
| Rescoring the baseline 1000-best with a single forward LSTM | 35.61 |
| Rescoring the baseline 1000-best with a single reversed LSTM | 35.85 |
| Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs | 36.5 |
| Oracle Rescoring of the Baseline 1000-best lists | ~45 |

Handwriting Generation from Text

- Input: Machine Learning UNNC

Machine learning unnc



Alex Graves. "Generating sequences with recurrent neural networks." (2013).

Applications of LSTM

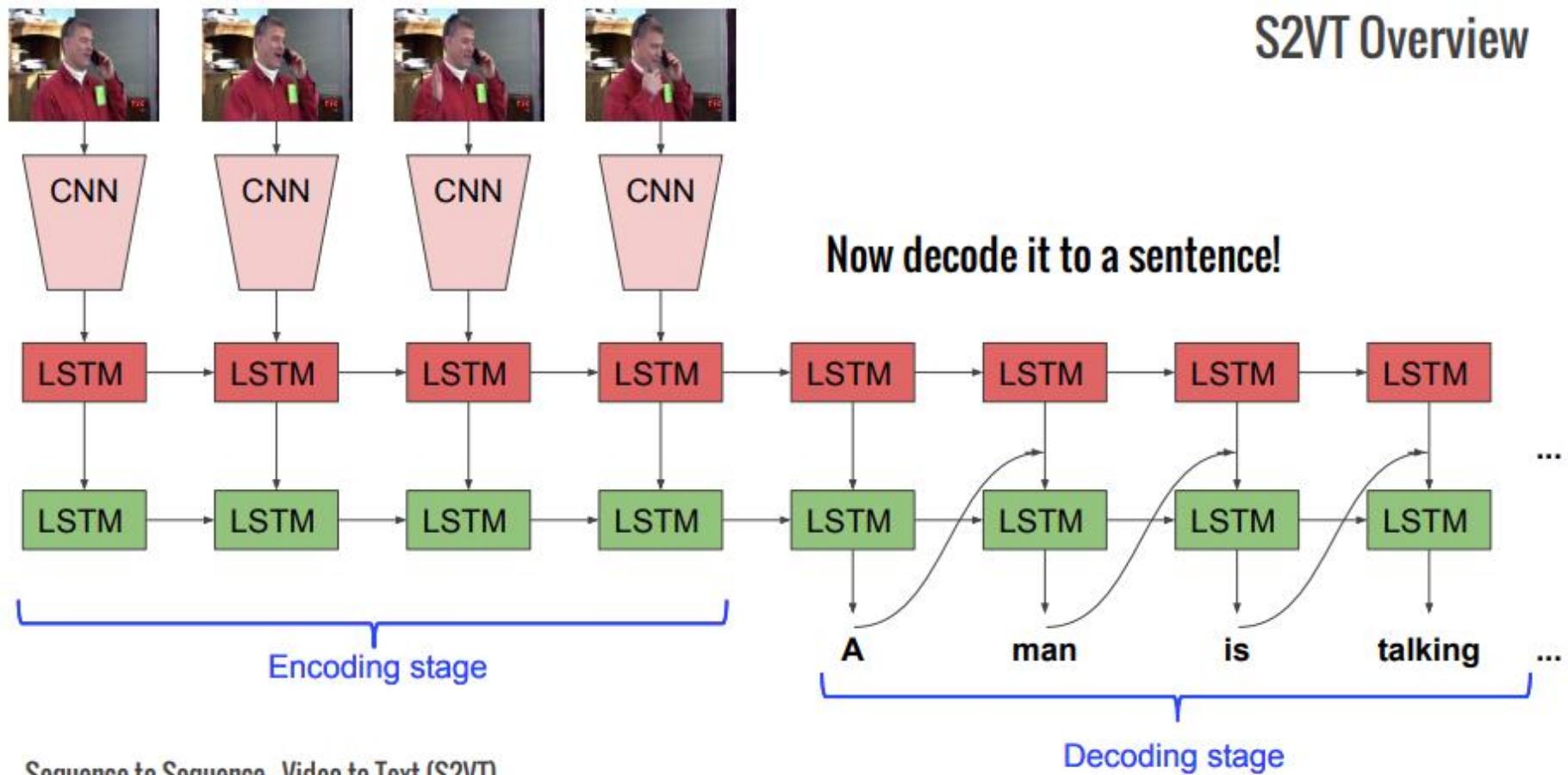
- Sequence to sequence: video to text

Objective



A monkey is pulling a dog's tail and is chased by the dog.

Video to Text

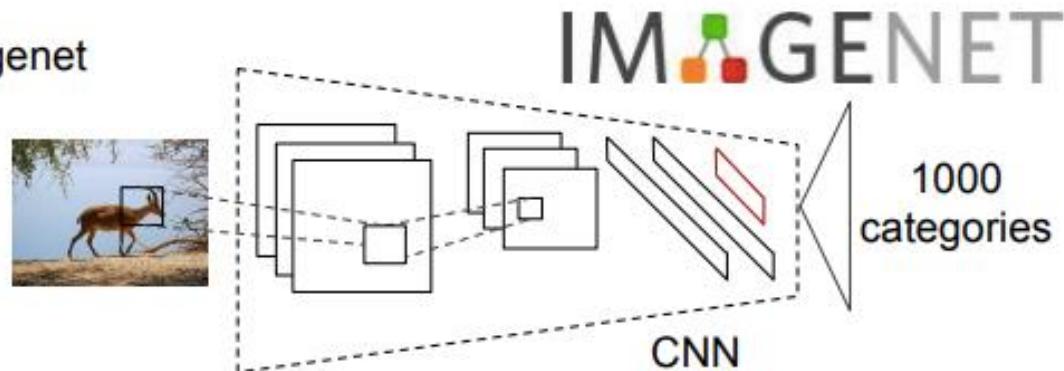


Sequence to Sequence - Video to Text (S2VT)

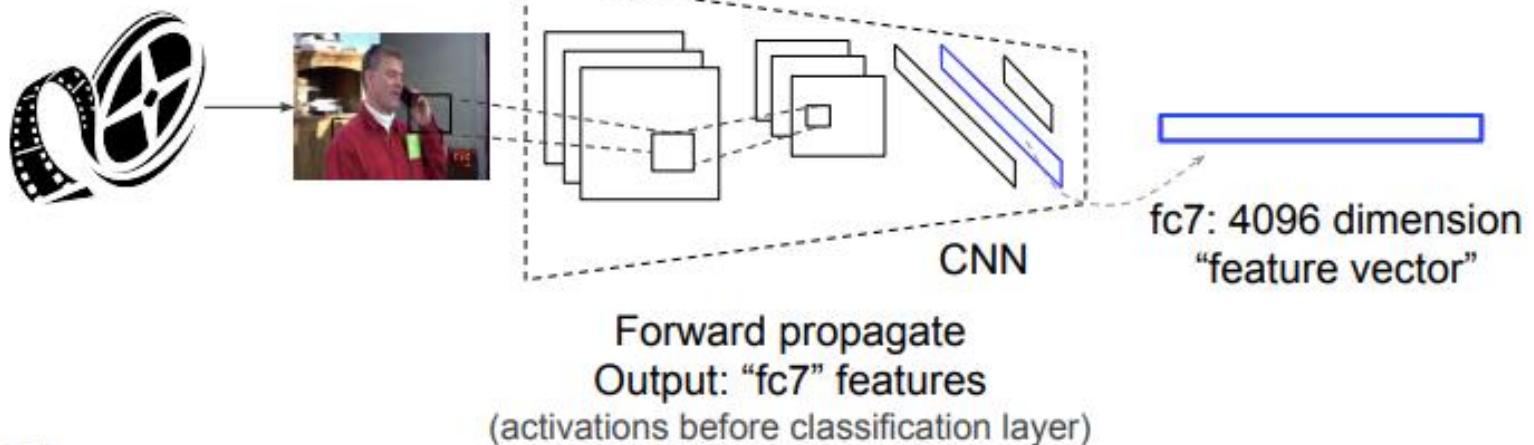
S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, K. Saenko

Video to Text

1. Train on Imagenet



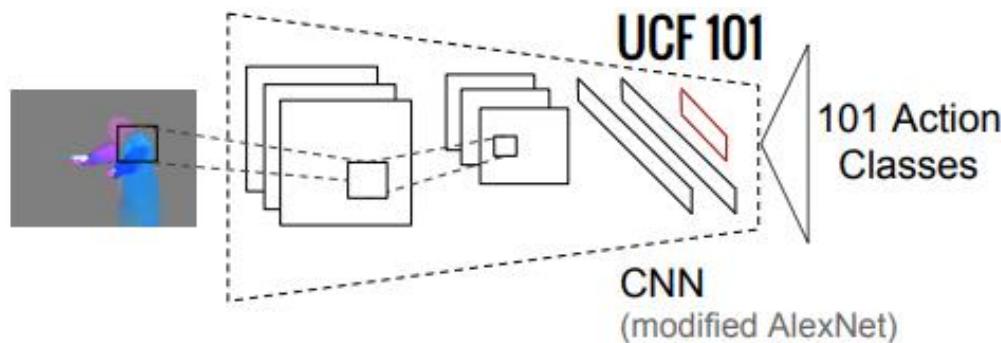
2. Take activations from layer before classification



Frames: RGB

Video to Text

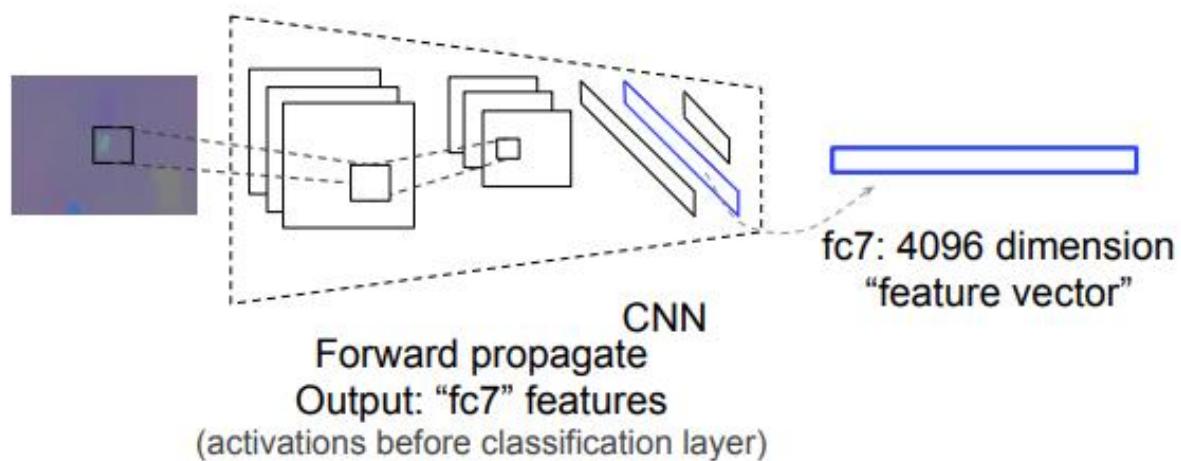
1. Train CNN on Activity classes



2. Use optical flow to extract flow images.



3. Take activations from layer before classification



Frames: Flow

Video to Text

Dataset: Youtube

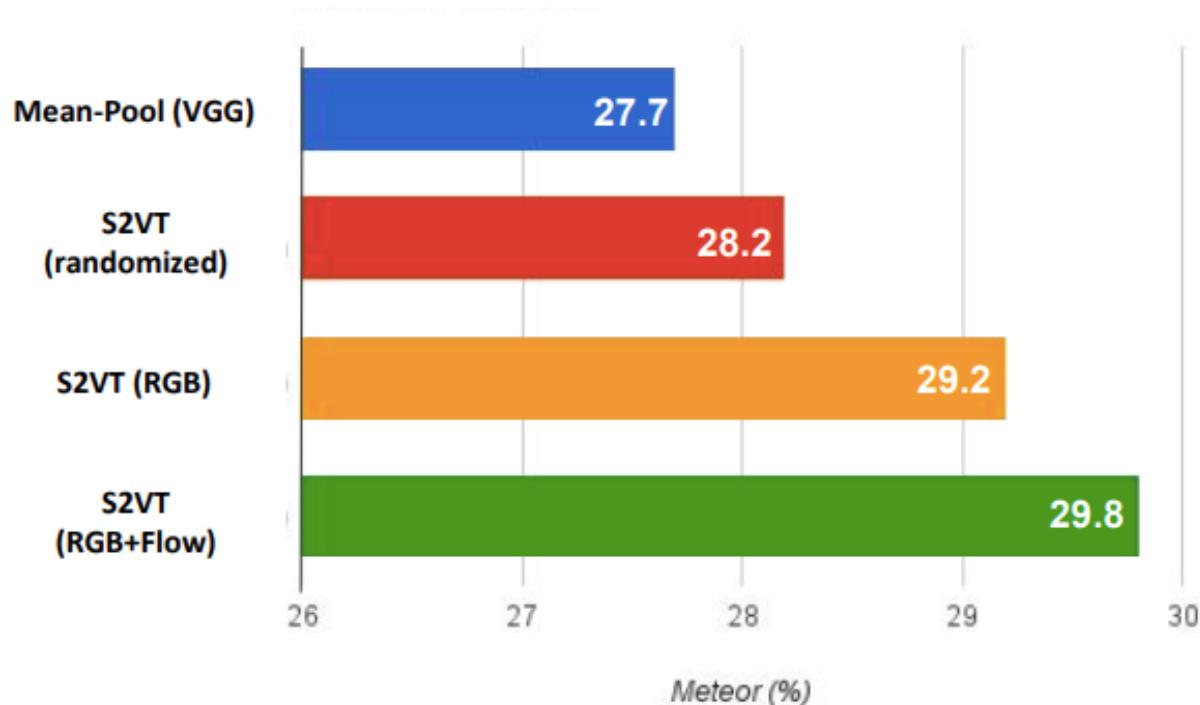
- ~2000 clips
- Avg. length: 11s per clip
- **~40 sentence per clip**
- ~81,000 sentences



- A man is **walking** on a **rope**.
- A man is **walking** across a **rope**.
- A man is **balancing** on a **rope**.
- A man is **balancing** on a **rope** at the beach.
- A man **walks** on a **tightrope** at the beach.
- A man is **balancing** on a **volleyball net**.
- A man is **walking** on a **rope** held by poles
- A man **balanced** on a **wire**.
- The man is **balancing** on the **wire**.
- A man is **walking** on a **rope**.
- A man is **standing** in the sea shore.

Video to Text

Results (Youtube)



METEOR: MT metric. Considers alignment, para-phrases and similarity.

Video to Text

Correct descriptions.



S2VT: A man is doing stunts on his bike.



S2VT: A herd of zebras are walking in a field.



S2VT: A young woman is doing her hair.



S2VT: A man is shooting a gun at a target.

Relevant but incorrect descriptions.



S2VT: A small bus is running into a building.



S2VT: A man is cutting a piece of a pair of a paper.



S2VT: A cat is trying to get a small board.



S2VT: A man is spreading butter on a tortilla.

Irrelevant descriptions.



S2VT: A man is pouring liquid in a pan.



S2VT: A polar bear is walking on a hill.



S2VT: A man is doing a pencil.



S2VT: A black clip to walking through a path.

Video to Text

Evaluation on movie corpus

M-VAD

- Univ. of Montreal
- DVS alignment: automated speech extraction
- 92 movies
- 46,009 clips
- Avg. length: 6.2s per clip
- **1-2 sentences per clip**
- 56,634 sentences

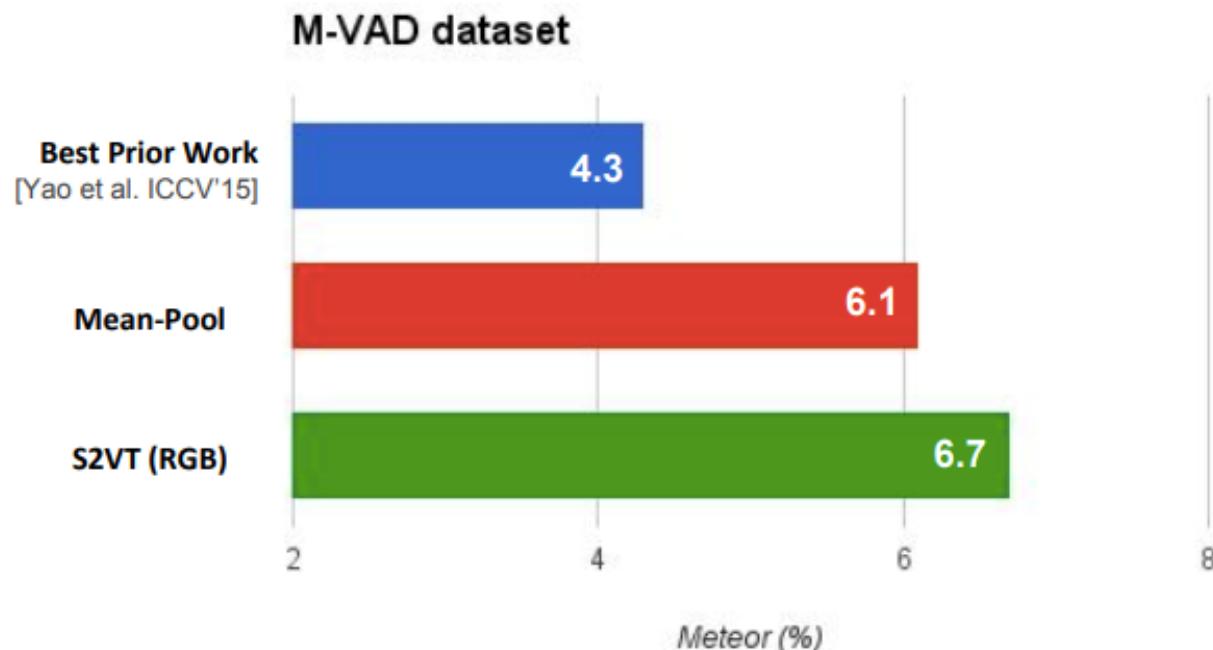


The Land Rover pulls away.

Three bodyguards quickly jump into a nearby car and follow her.

Video to Text

Results (M-VAD Movie Corpus)

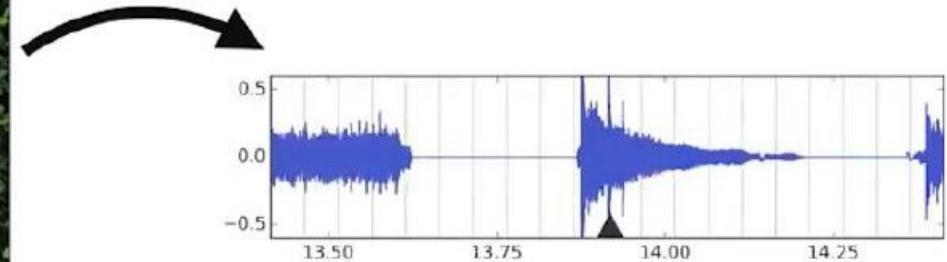


Adding Audio to Silent Film

<https://www.youtube.com/watch?v=0FW99AQmMc8>

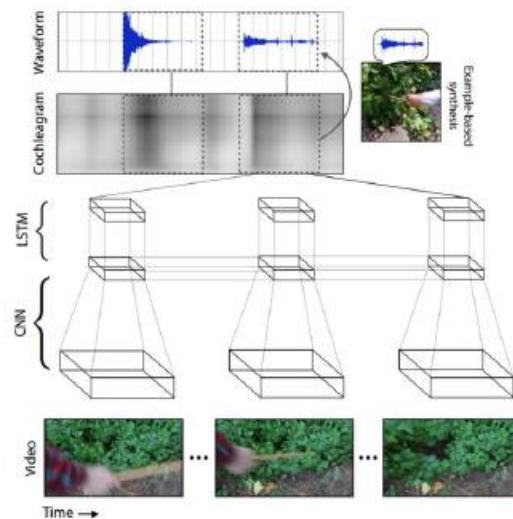


Silent video

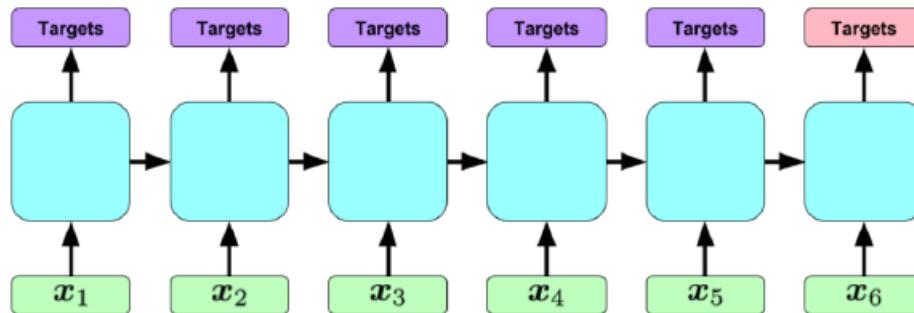


Predicted soundtrack

Owens, Andrew, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman. "**Visually Indicated Sounds.**" (2015).



Medical Diagnosis



- **Input:** patients electronic health record (EHR) data over multiple visits (meaning, variable length sequences)
- **Output:** 128 diagnoses

Top 6 diagnoses measured by F1 score

| Label | F1 | AUC | Precision | Recall |
|---|--------|--------|-----------|--------|
| Diabetes mellitus with ketoacidosis | 0.8571 | 0.9966 | 1.0000 | 0.7500 |
| Scoliosis, idiopathic | 0.6809 | 0.8543 | 0.6957 | 0.6667 |
| Asthma, unspecified with status asthmaticus | 0.5641 | 0.9232 | 0.7857 | 0.4400 |
| Neoplasm, brain, unspecified | 0.5430 | 0.8522 | 0.4317 | 0.7315 |
| Delayed milestones | 0.4751 | 0.8178 | 0.4057 | 0.5733 |
| Acute Respiratory Distress Syndrome (ARDS) | 0.4688 | 0.9595 | 0.3409 | 0.7500 |

Stock Market Prediction

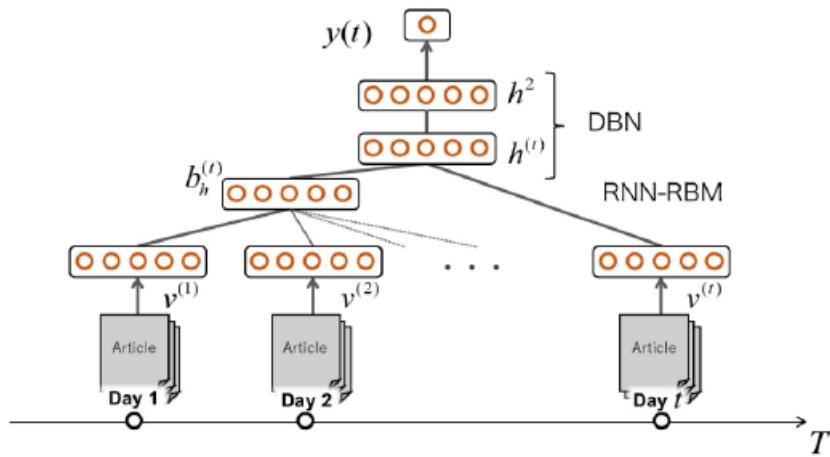


Table 3. Test error rates for stock price prediction

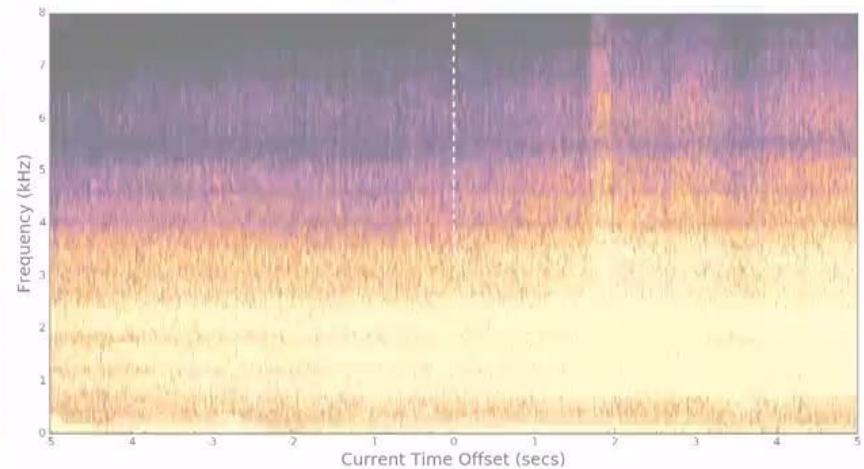
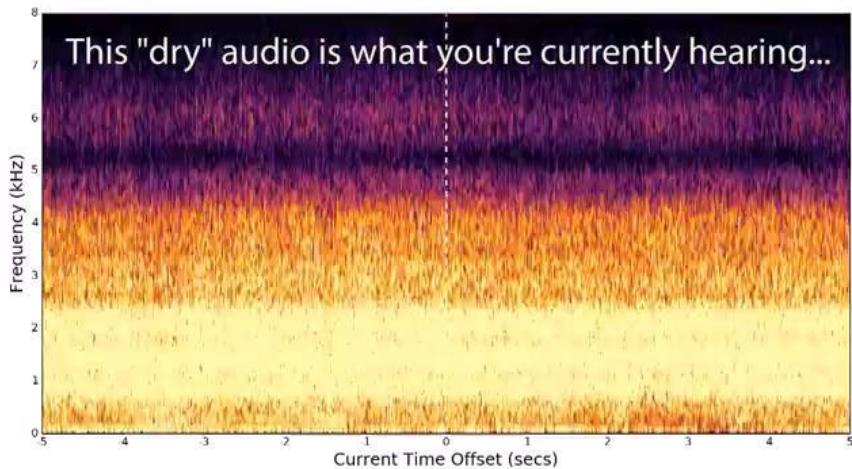
| Brands | Baseline | SVM | DBN | RNN-RBM
M + DBN |
|--------------|----------|-------|--------------|--------------------|
| Nikkei | 49.57 | 48.73 | 45.50 | 43.62 |
| Average | | | | |
| Hitachi | 35.71 | 37.29 | 32.00 | 32.00 |
| Toshiba | 39.52 | 41.95 | 38.50 | 38.50 |
| Fujitsu | 40.00 | 40.25 | 32.00 | 34.00 |
| Sharp | 42.00 | 47.88 | 40.00 | 40.00 |
| Sony | 43.00 | 47.46 | 41.43 | 40.95 |
| Nissan Motor | 40.00 | 45.34 | 39.50 | 37.00 |
| Toyota Motor | 44.29 | 53.39 | 43.81 | 42.38 |
| Canon | 43.81 | 53.39 | 43.00 | 39.11 |
| Mitsui | 46.96 | 47.88 | 41.43 | 41.43 |
| Mitsubishi | 43.81 | 49.15 | 43.33 | 40.43 |
| Average | 42.61 | 46.61 | 40.05 | 39.04 |

Table 5. Comparison of test error rates after a significant financial crisis

| Brands | SVM | RNN-RBM + DBN |
|----------------|--------------|---------------|
| Nikkei Average | 51.61 | 38.70 |
| Hitachi | 61.29 | 32.25 |
| Toshiba | 54.83 | 38.70 |
| Fujitsu | 45.16 | 32.25 |
| Sharp | 58.06 | 45.16 |
| Sony | 41.93 | 41.93 |
| Nissan Motor | 29.03 | 35.48 |
| Toyota Motor | 48.38 | 45.16 |
| Canon | 54.83 | 54.83 |
| Mitsui | 41.93 | 38.70 |
| Mitsubishi | 29.03 | 25.80 |
| Average | 46.92 | 39.00 |

Yoshihara et al. "Leveraging temporal properties of news events for stock market prediction." 2015.

Audio Classification





COMP3055

Machine Learning

Topic 17 – Deep Reinforcement Learning Intro

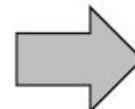
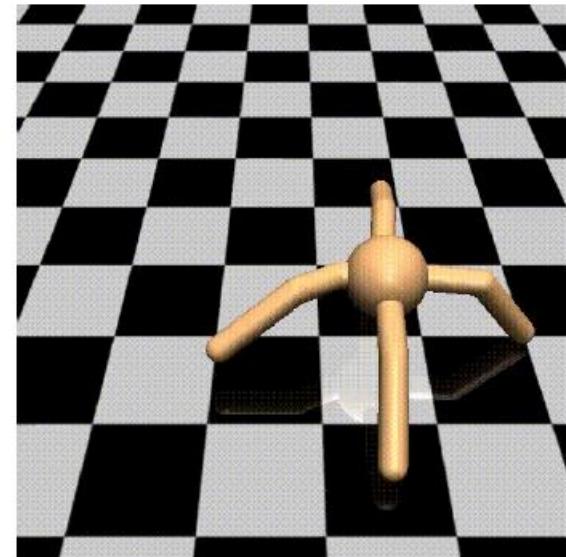
Zheng Lu
2024 Autumn

Deep Reinforcement Learning

Deep Learning

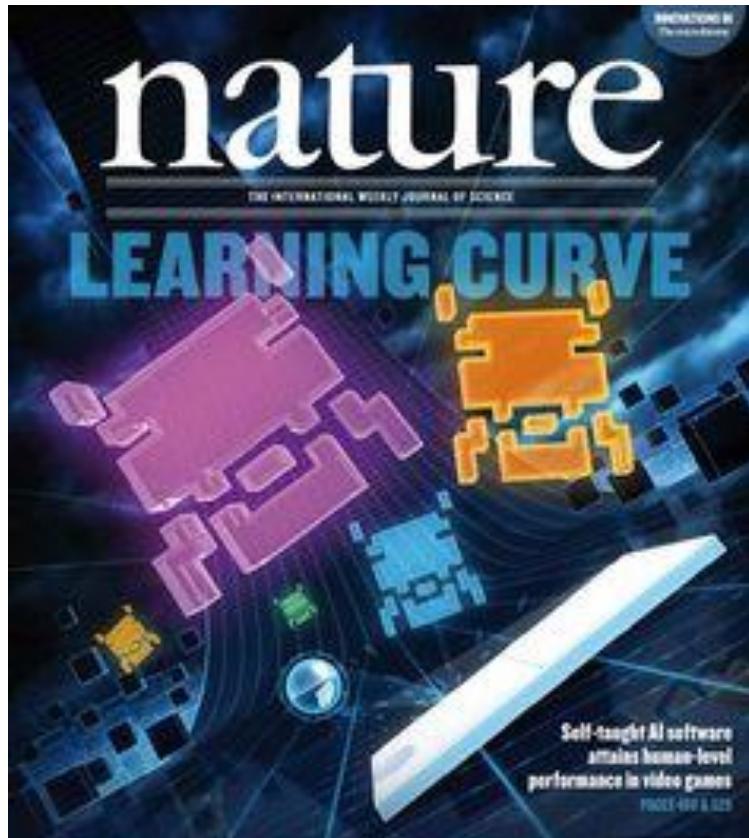


Deep RL



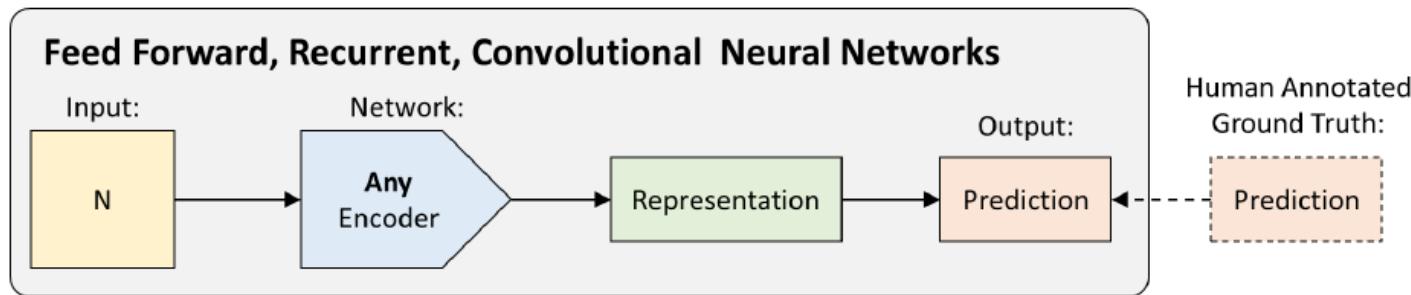
- **What is it?** Framework for learning to solve sequential decision making problems.
- **How?** Trial and error in a world that provides occasional rewards

Deep Reinforcement Learning



Deep Reinforcement Learning: $\text{AI} = \text{RL} + \text{DL}$

Deep Reinforcement Learning



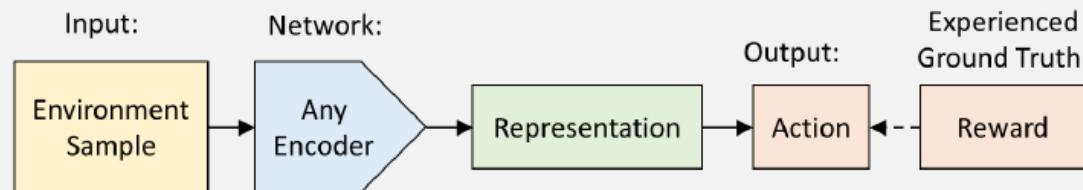
Supervised learning is “teach by **example**”:

Here's some examples, now learn patterns in these example.

Reinforcement learning is “teach by **experience**”:

Here's a world, now learn patterns by exploring it.

Networks for Learning Actions, Values, Policies, and/or Models



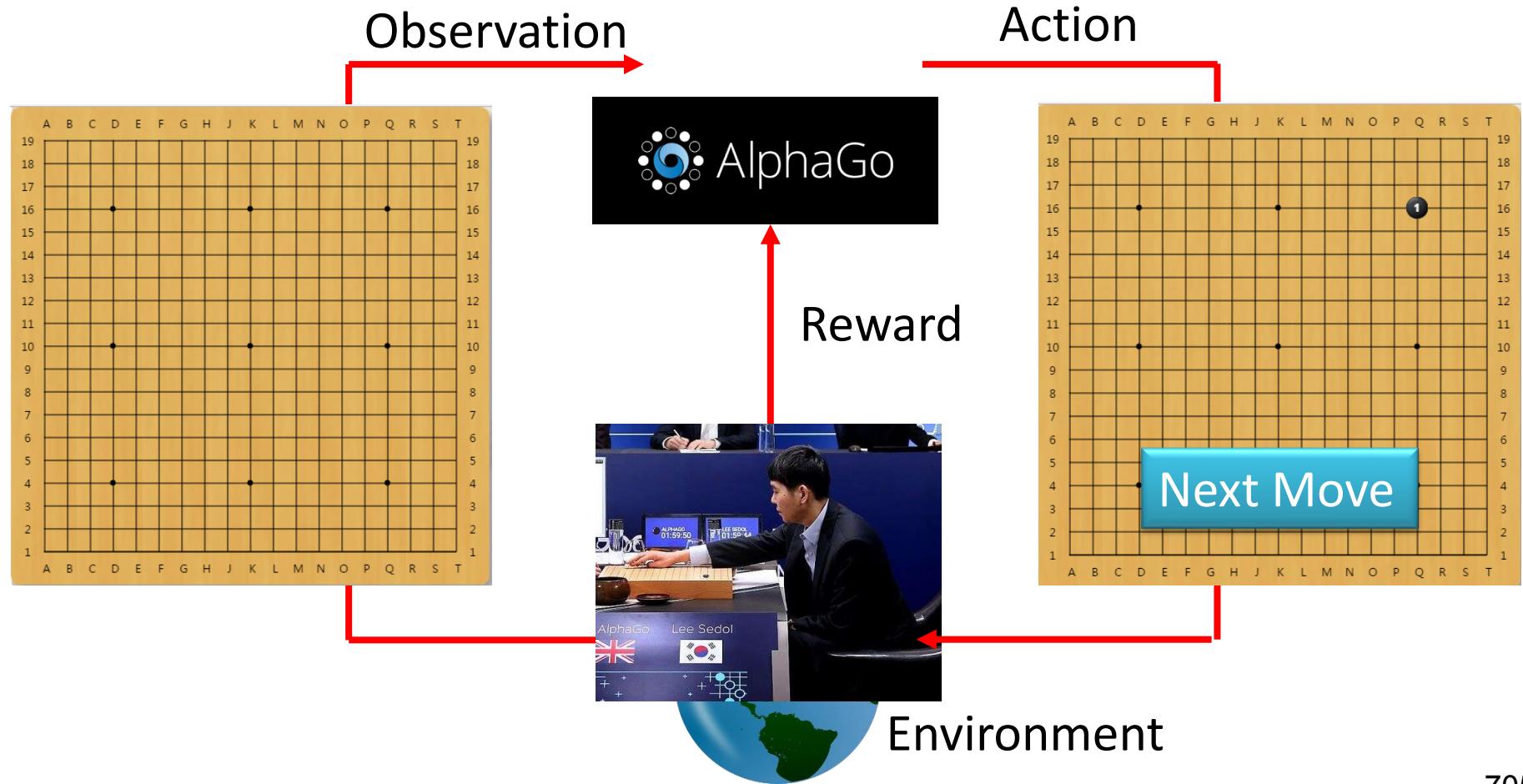
Scenario of Reinforcement Learning



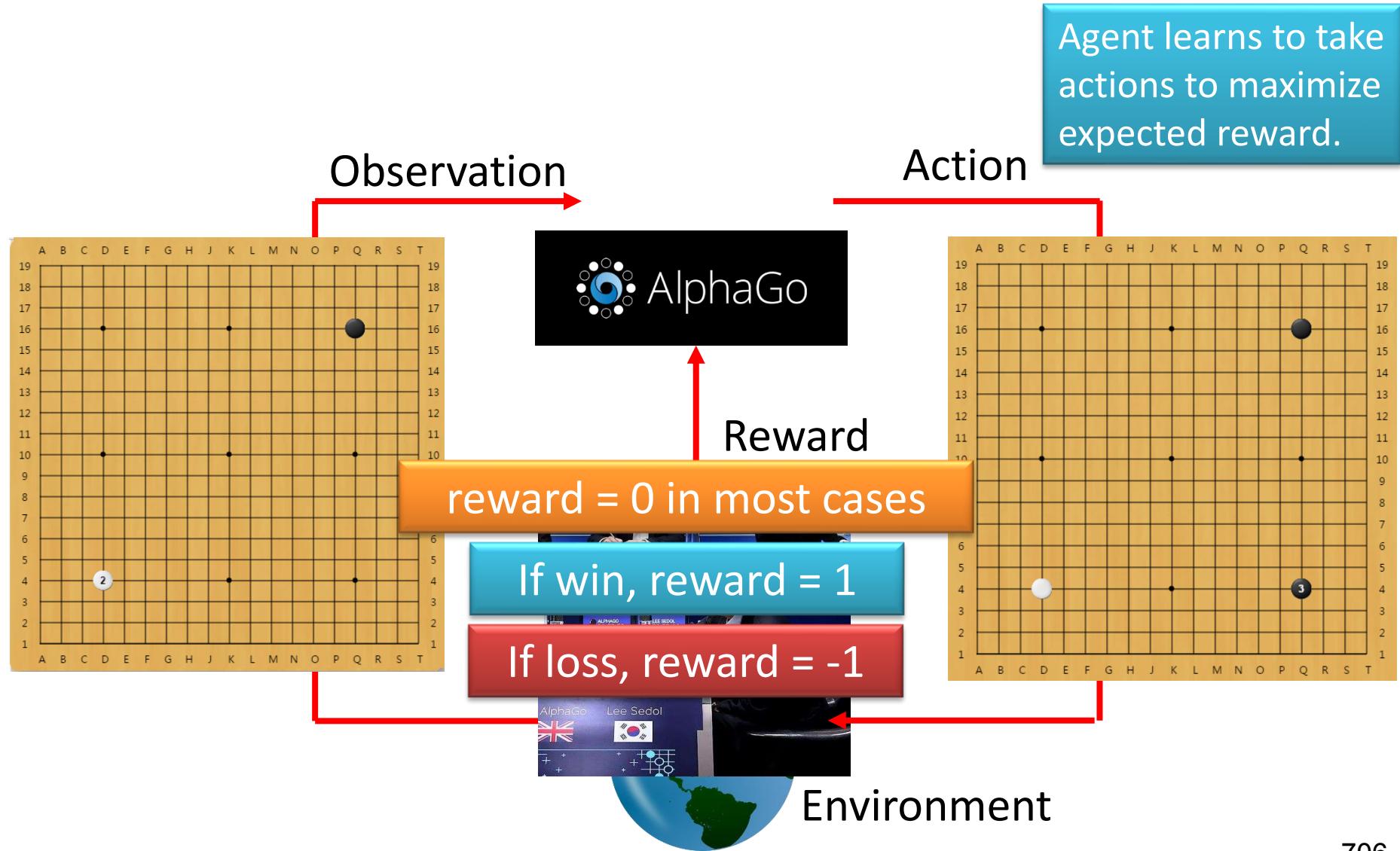
Scenario of Reinforcement Learning



Learning to play Go



Learning to play Go



Learning to play Go

- Supervised v.s. Reinforcement

- Supervised:

Learning from teacher



Next move:
“5-5”



Next move:
“3-3”

Learning from experience

- Reinforcement Learning

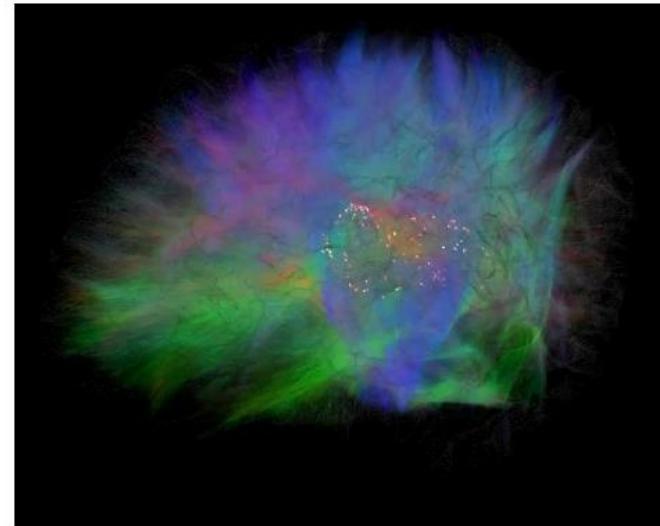
First move → many moves → Win!

(Two agents play with each other.)

Alpha Go is supervised learning + reinforcement learning.

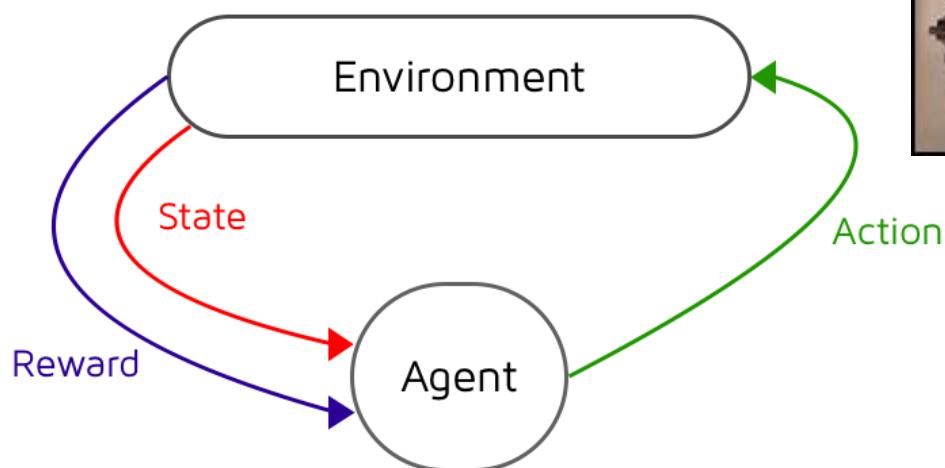
RL in Humans

- Human appear to learn to walk through “very few examples” of trial and error. How is an open question...
- Possible answers:
 - **Hardware**: 3 million years of evolutionary data
 - **Imitation Learning**: observation of other humans walking
 - **Algorithms**: better than backpropagation and stochastic gradient descent



RL Framework

- At each step, the agent should:
 - Executes **action**
 - Observes new **state**
 - Receives **reward**



Environment and Actions

- Fully Observable (Chess) vs Partially Observable (Poker, SC2)
- Single Agent (Atari) vs Multi Agent (SC2)
- Deterministic (Cart Pole) vs Stochastic (SC2)
- Static (Chess) vs Dynamic (SC2)
- Discrete (Chess) vs Continuous (Cart Pole)
- **Note:** *Real-world environment might not technically be stochastic or partially-observable but might as well be treated as such due to their complexity*

The Challenge for RL in Real-World Applications

Reminder:

Supervised learning:
teach by example

Reinforcement learning:
teach by experience

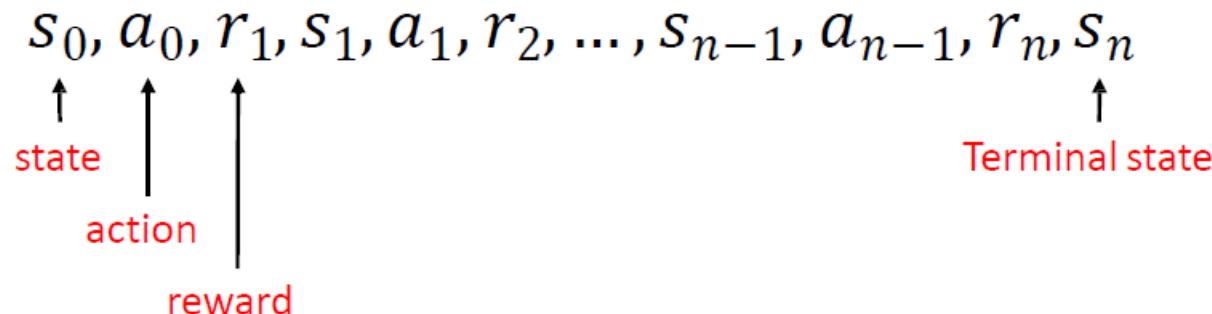
Open Challenges. Two Options:

1. Real world observation + one-shot trial & error
2. Realistic simulation + transfer learning



Major Components of an RL Agent

- An RL agent may be directly or indirectly trying to learn a:
 - **Policy**: agent's behavior function
 - **Value function**: how good is each state and/or action
 - **Model**: agent's representation of the environment



RL Agent: Maximize Reward

- Future reward: $R_t = r_t + r_{t+1} + r_{t+2} + \dots + r_n$
- Discounted future reward:

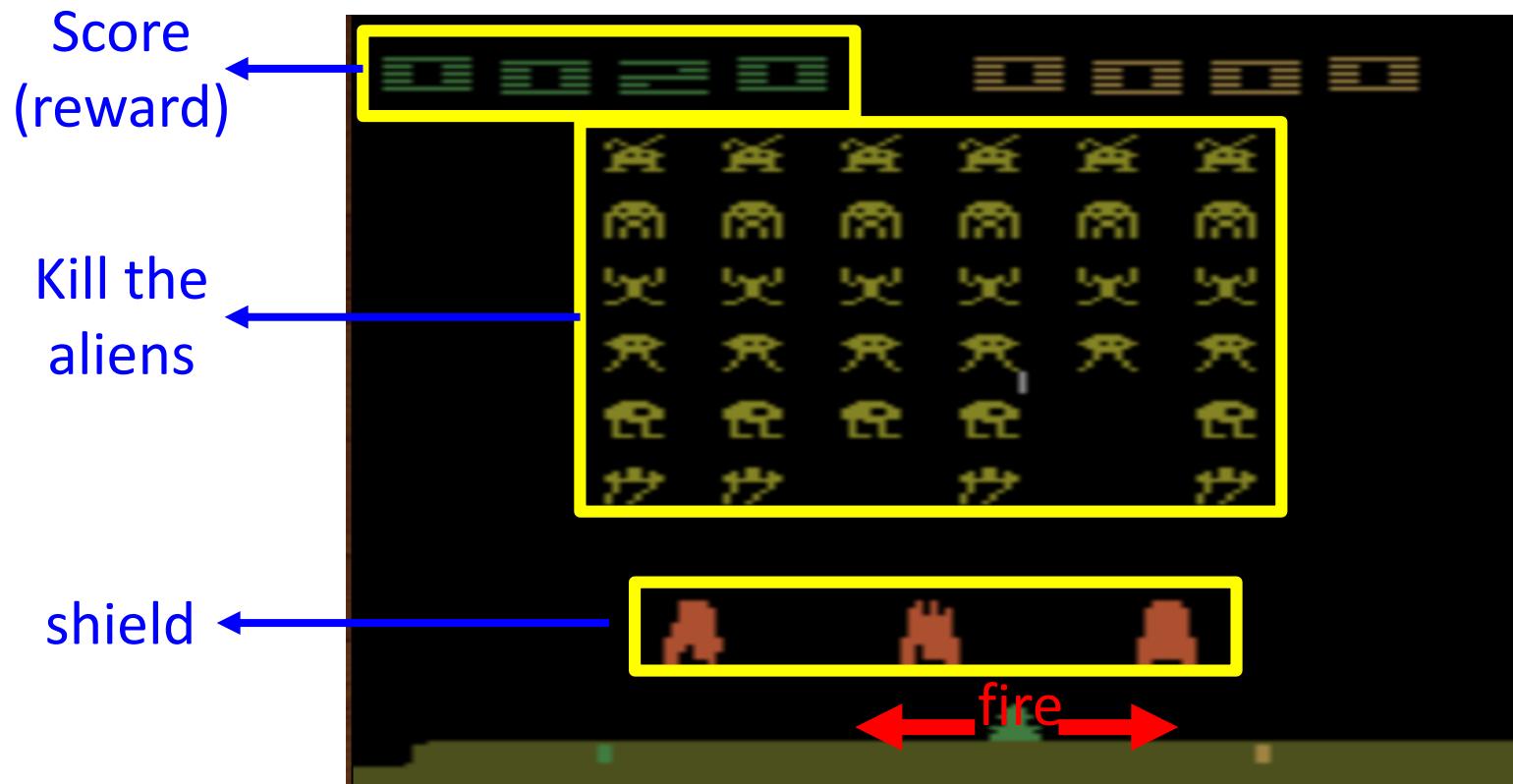
$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{n-t} r_n$$

- A good strategy for an agent would be to always choose an action that **maximizes the (discounted) future reward**
- Why “discounted”?
 - Math trick to help analyze convergence
 - Uncertainty due to environment stochasticity, partial observability, or that life can end at any moment:

Example: Playing Video Game

- Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.



Example: Playing Video Game

Start with
observation s_1

Observation s_2

Observation s_3



Obtain reward
 $r_1 = 0$

?

Action a_1 : “right”

Obtain reward
 $r_2 = 5$

?

Action a_2 : “fire”
(kill an alien)

Example: Playing Video Game

Start with
observation s_1



Observation s_2



Observation s_3



After many turns



Obtain reward r_T

Action a_T

This is an *episode*.

Learn to maximize the
expected cumulative
reward per episode

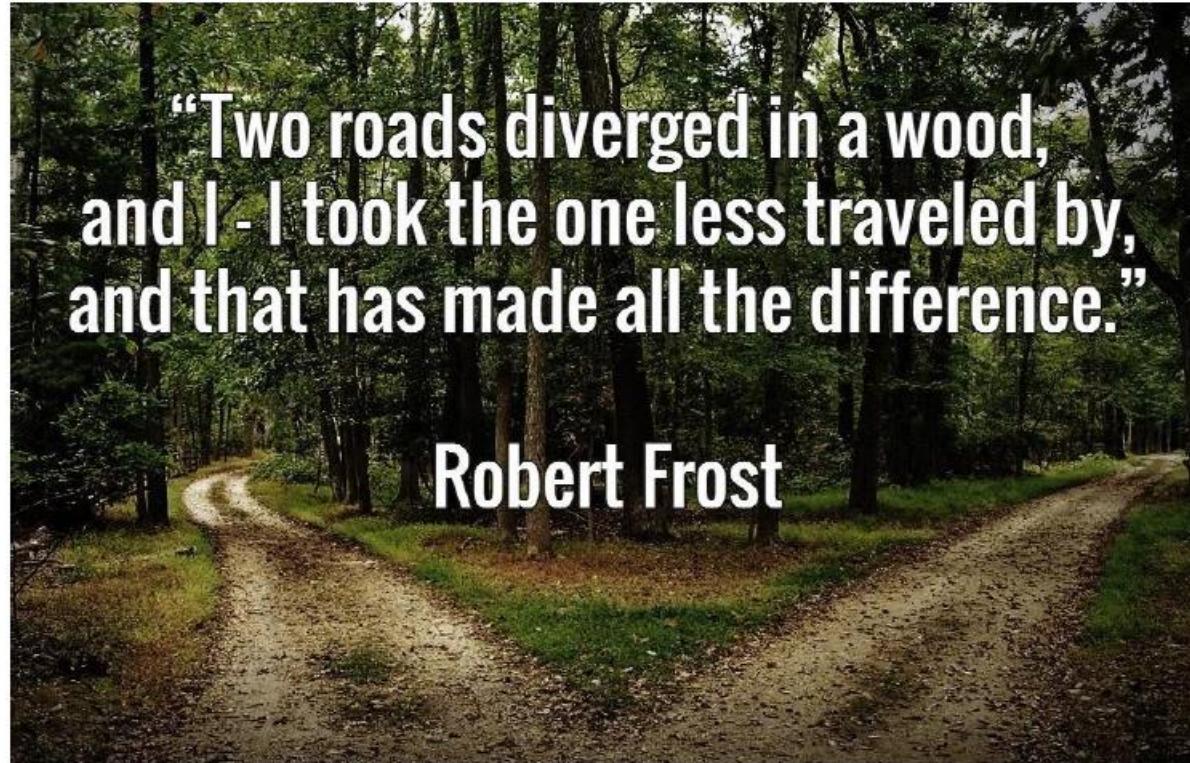
Difficulties of Reinforcement Learning

- Reward delay
 - In space invader, only “fire” obtains reward
 - Although the moving before “fire” is important
 - In Go playing, it may be better to sacrifice immediate reward to gain more long-term reward
- Agent’s actions affect the subsequent data it receives
 - E.g. Exploration

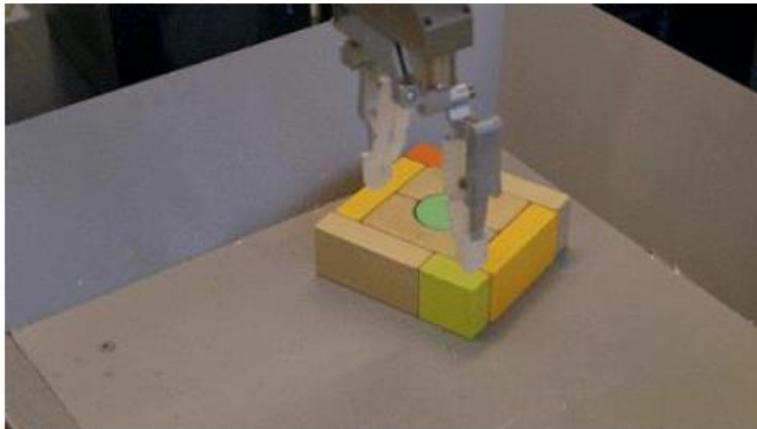


Lessons Take Away

- Environment model has big impact on optimal policy
- Reward structure has big impact on optimal policy



Real World Example



Grasping Objects with Robotic Arm

- **Goal** - Pick an object of different shapes
- **State** - Raw pixels from camera
- **Actions** – Move arm. Grasp.
- **Reward** - Positive when pickup is successful



RL in Humans

- Human – Life is a big training set
- **Goal:** Survival?
- **State:** Sight. Hearing. Taste. Smell. Touch.
- **Actions:** Think. Move.
- **Reward:** Happiness?

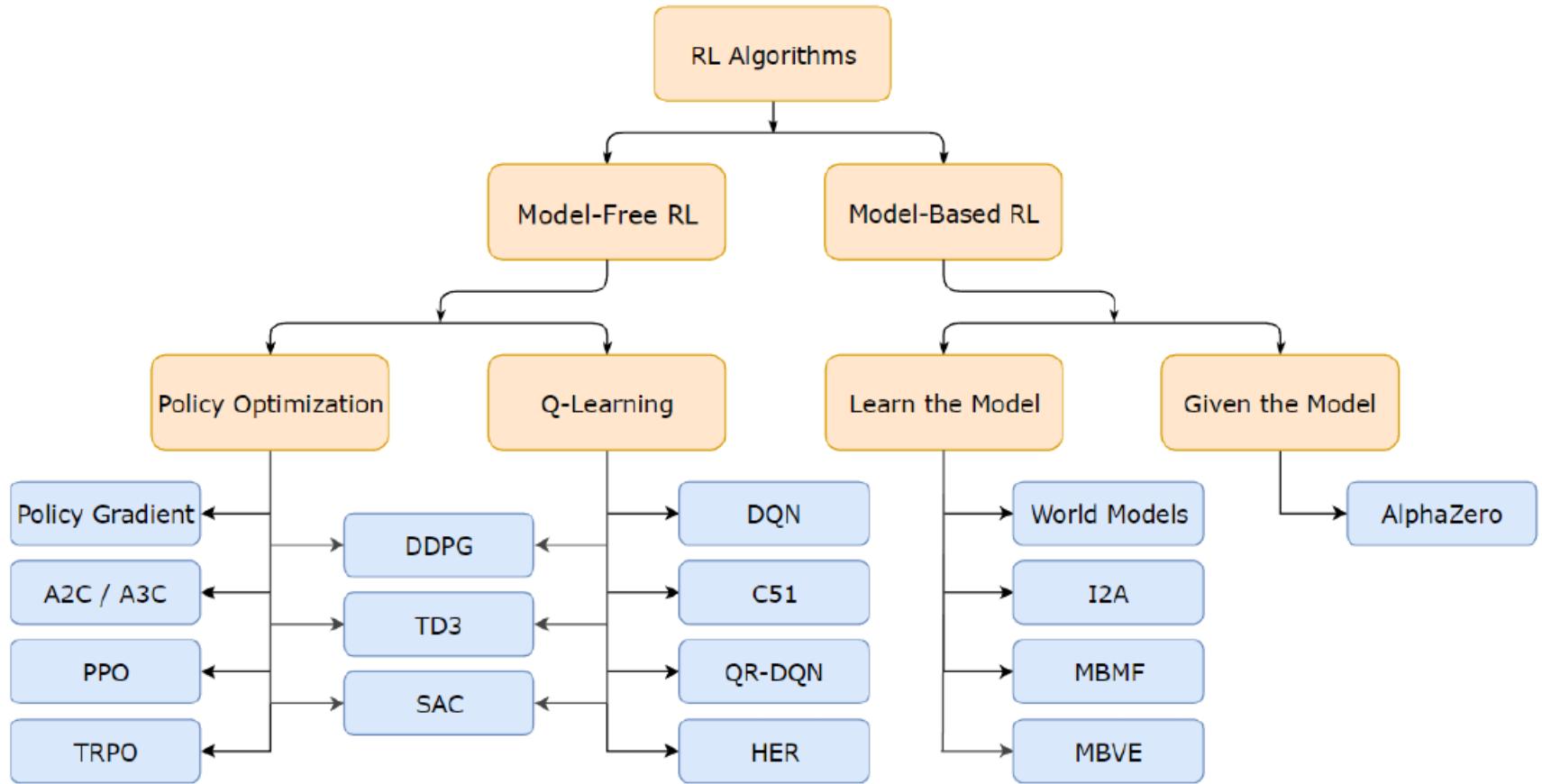
Key Takeaways

- **Deep Learning:**
 - Fun part: Good algorithms that learn from data
 - Hard part: Good questions, huge amounts of representative data
- **Deep Reinforcement Learning:**
 - Fun part: Good algorithms that learn from data
 - Hard part: Defining a useful state space, action space, and reward
 - Hardest part: Getting meaningful data for the above formalization

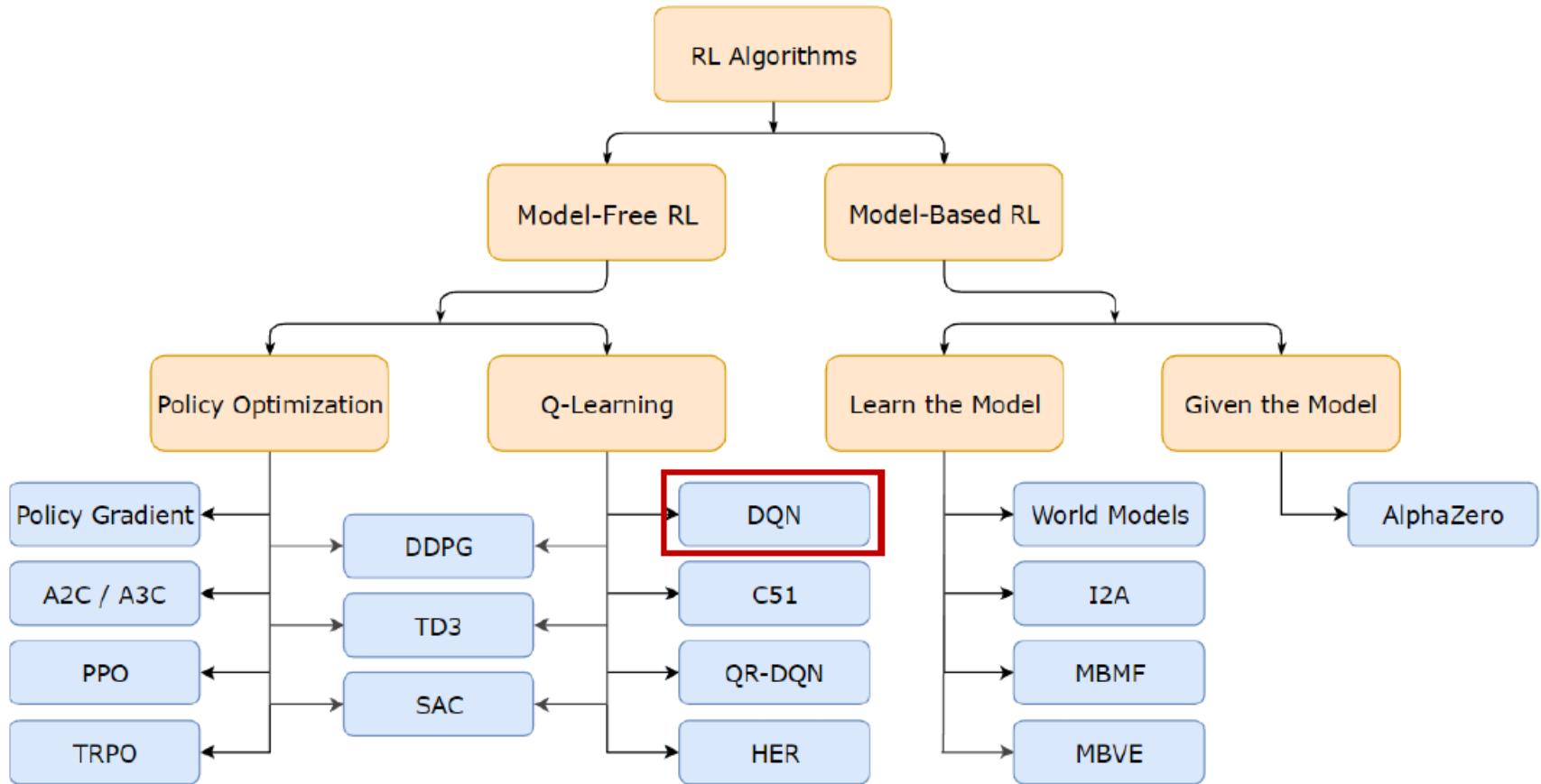
RL Methods

- **Model-based**
 - Learn the model of the world, then plan using the model
 - Update model often
 - Re plan often
- **Value-based**
 - Learn the state or state action value
 - Act by choosing best action in state
 - Exploration is a necessary add on
- **Policy-based**
 - Learn the stochastic policy function that maps state to action
 - Act by sampling policy
 - Exploration is baked in

RL Methods

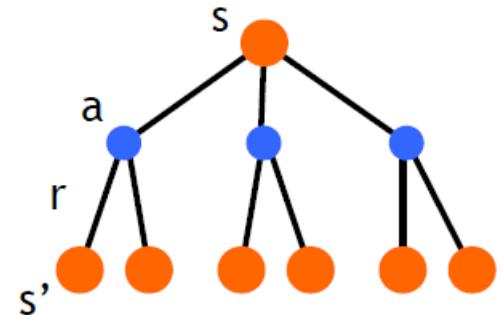


RL Methods



Q-Learning

- State-action value function: $Q^\pi(s, a)$
 - Expected return when starting in s , performing a , and following π
- Q-Learning: Use **any policy** to estimate Q that maximizes future reward:
 - Q directly approximates Q^* (Bellman optimality equation)
 - Independent of the policy being followed
 - Only requirement: keep updating each (s, a) pair



$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$$

Learning Rate Discount Factor

New State Old State Reward

Exploration vs Exploitation

- Deterministic/greedy policy won't explore all actions
 - Don't know anything about the environment at the beginning
 - Need to try all actions to find the optimal one
- ϵ -greedy policy
 - With probability $1-\epsilon$ perform the optimal/greedy action, otherwise random action
 - Slowly move it towards greedy policy: $\epsilon \rightarrow 0$



Q-Learning: Value Iteration

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$$

| | A1 | A2 | A3 | A4 |
|----|----|----|----|----|
| S1 | +1 | +2 | -1 | 0 |
| S2 | +2 | 0 | +1 | -2 |
| S3 | -1 | +1 | 0 | -2 |
| S4 | -2 | 0 | +1 | +1 |

```

initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
    select and carry out an action a
    observe reward r and new state s'
    Q[s,a] = Q[s,a] + α(r + γ maxa' Q[s',a'] - Q[s,a])
    s = s'
until terminated

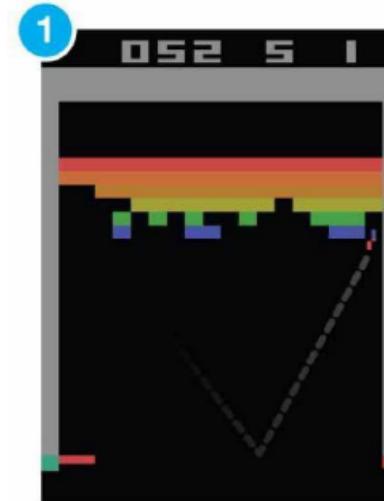
```

Q-Learning: Representation

- In practice, Value Iteration is impractical
 - Very limited states/actions
 - Cannot generalize to unobserved states

- Think about the **Breakout** game

- State: screen pixels
 - Image size: **84 × 84** (resized)
 - Consecutive **4** images
 - Grayscale with **256** gray levels

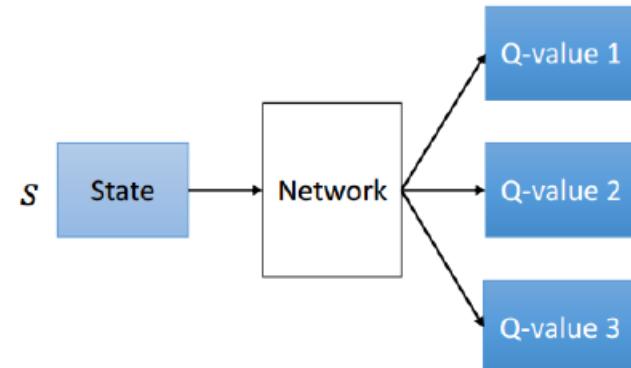
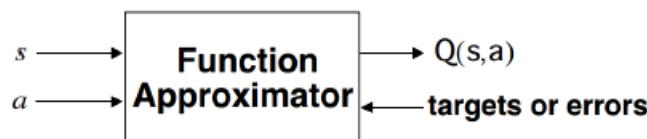
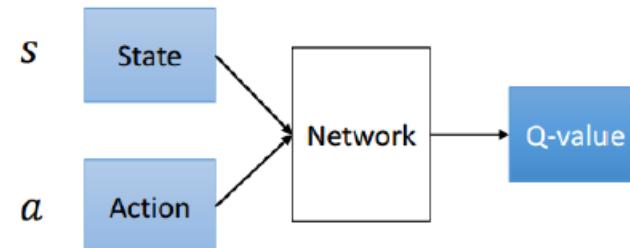


256^{84×84×4} rows in the Q-table!
 $= 10^{69,970} >> 10^{82}$ atoms in the universe

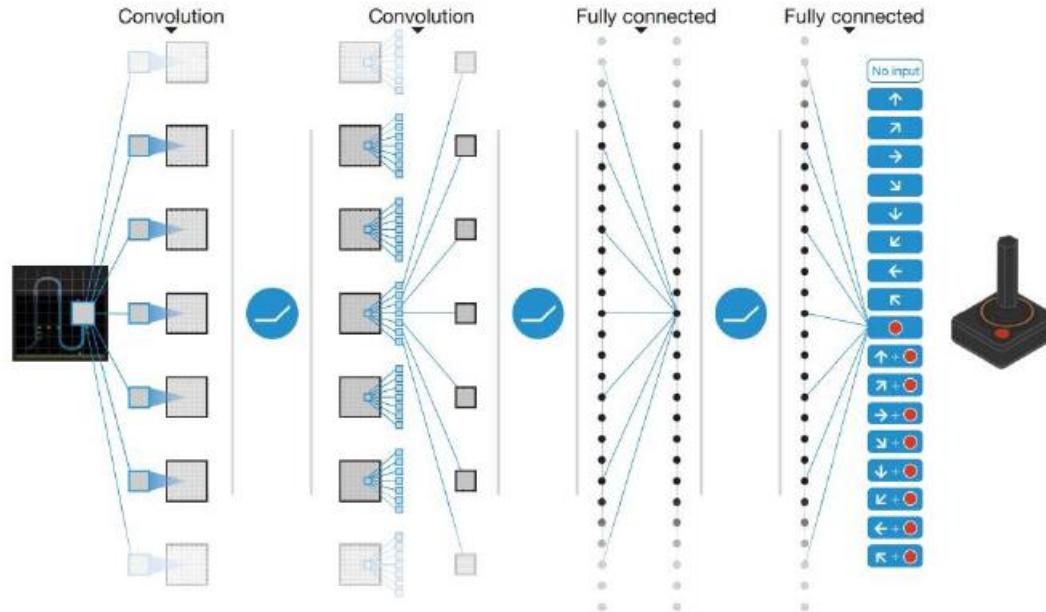
Deep Q-Learning

Use a neural network to approximate the Q-function:

$$Q(s, a; \theta) \approx Q^*(s, a)$$



DQN: Atari



| Layer | Input | Filter size | Stride | Num filters | Activation | Output |
|-------|----------|-------------|--------|-------------|------------|----------|
| conv1 | 84x84x4 | 8x8 | 4 | 32 | ReLU | 20x20x32 |
| conv2 | 20x20x32 | 4x4 | 2 | 64 | ReLU | 9x9x64 |
| conv3 | 9x9x64 | 3x3 | 1 | 64 | ReLU | 7x7x64 |
| fc4 | 7x7x64 | | | 512 | ReLU | 512 |
| fc5 | 512 | | | 18 | Linear | 18 |

Mnih et al. "Playing atari with deep reinforcement learning." 2013.

Double DQN

- Loss function (squared error):

$$L = \mathbb{E}[(\underbrace{r + \gamma \max_{a'} Q(s', a')}_\text{target} - \underbrace{Q(s, a)}_\text{prediction})^2]$$

- DQN: same network for both Q
- Double DQN: separate network for each Q
 - Helps reduce bias introduced by the inaccuracies of Q network at the beginning of training

DQN Tricks

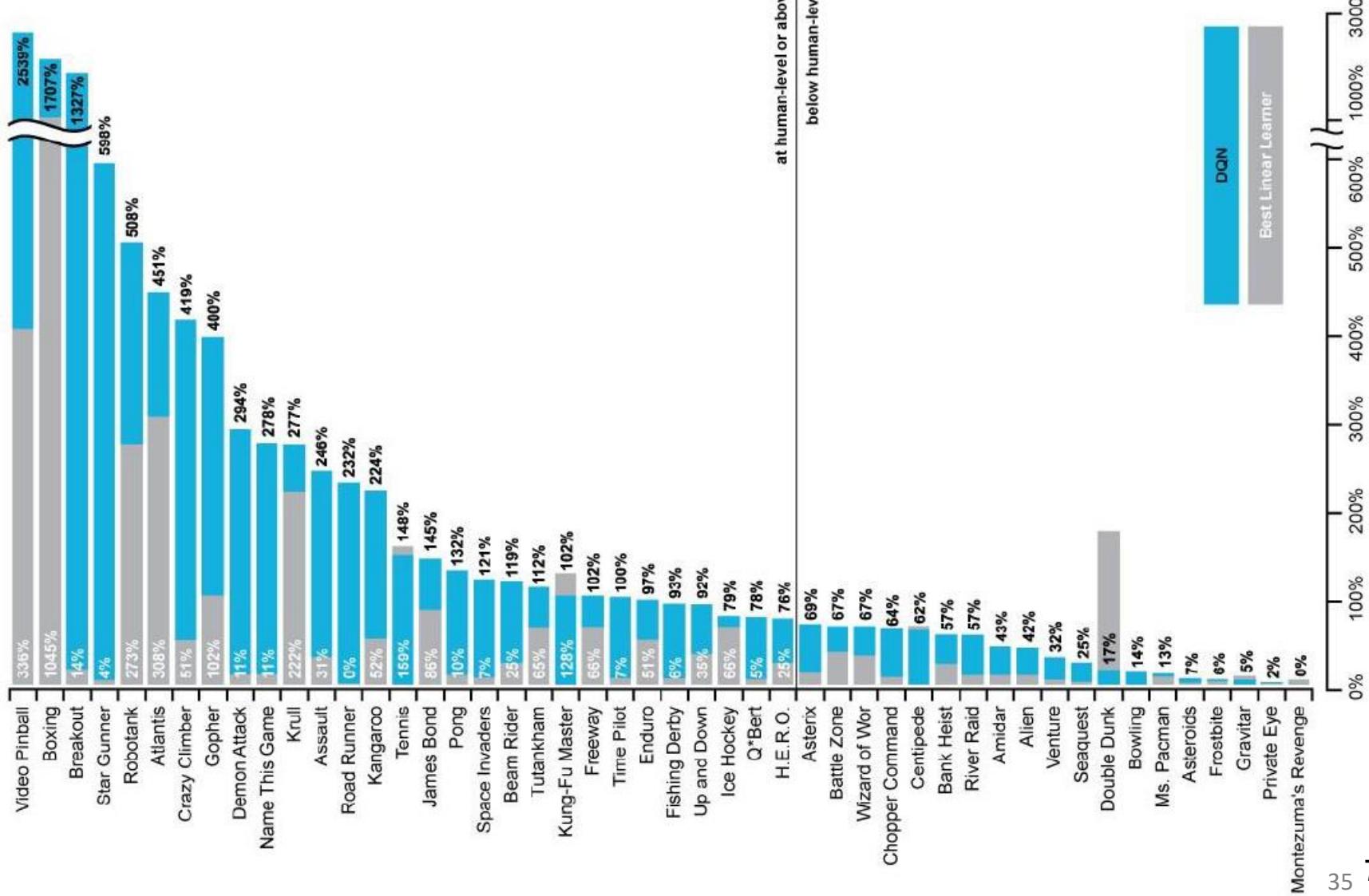
- Experience Replay
 - Stores experiences (actions, state transitions, and rewards) and creates mini-batches from them for the training process
- Fixed Target Network
 - Error calculation includes the target function depends on network parameters and thus changes quickly. Updating it only every 1,000 steps increases stability of training process.

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[r_{t+1} + \gamma \max_p Q(s_{t+1}, p) - Q(s_t, a) \right]$$

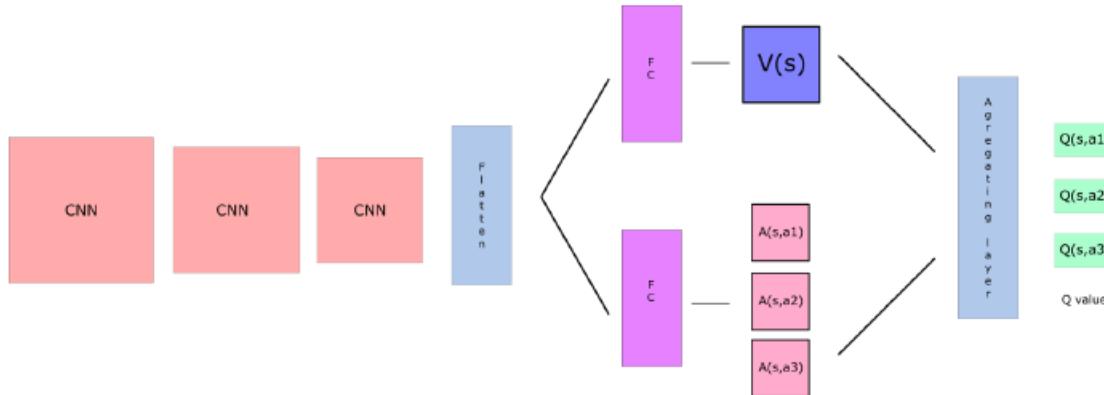
target Q function in the red rectangular is fixed

| Replay | ○ | ○ | ✗ | ✗ |
|----------------|---------------|--------|--------|--------|
| Target | ○ | ✗ | ○ | ✗ |
| Breakout | 316.8 | 240.7 | 10.2 | 3.2 |
| River Raid | 7446.6 | 4102.8 | 2867.7 | 1453.0 |
| Seaquest | 2894.4 | 822.6 | 1003.0 | 275.8 |
| Space Invaders | 1088.9 | 826.3 | 373.2 | 302.0 |

DQN in Atari



Dueling DQN (DDQN)



- Decompose $Q(s,a)$
$$Q(s, a) = A(s, a) + V(s)$$
 - $V(s)$: the value of being at that state
 - $A(s,a)$: the **advantage** of taking action a in state s versus all other possible actions at that state
- Use two streams:
 - one that estimates the **state value** $V(s)$
 - one that estimates the **advantage for each action** $A(s,a)$
- Useful for states where action choice does not affect $Q(s,a)$

Prioritized Experience Replay

$$p_t = \underline{|\delta_t|} + \underline{e}$$

Magnitude of our TD error

Constant assures that no experience has 0 probability to be taken.

$$P(i) = \frac{p_i^a}{\sum_k p_k^a}$$

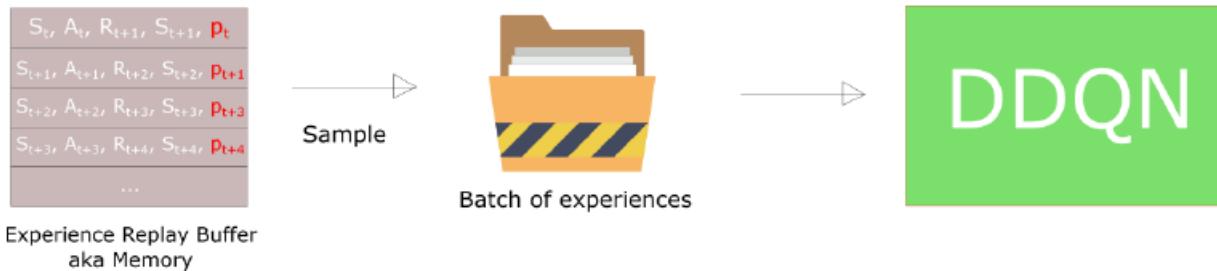
Priority value

Hyperparameter used to reintroduce some randomness in the experience selection for the replay buffer

If $a = 0$ pure uniform randomness

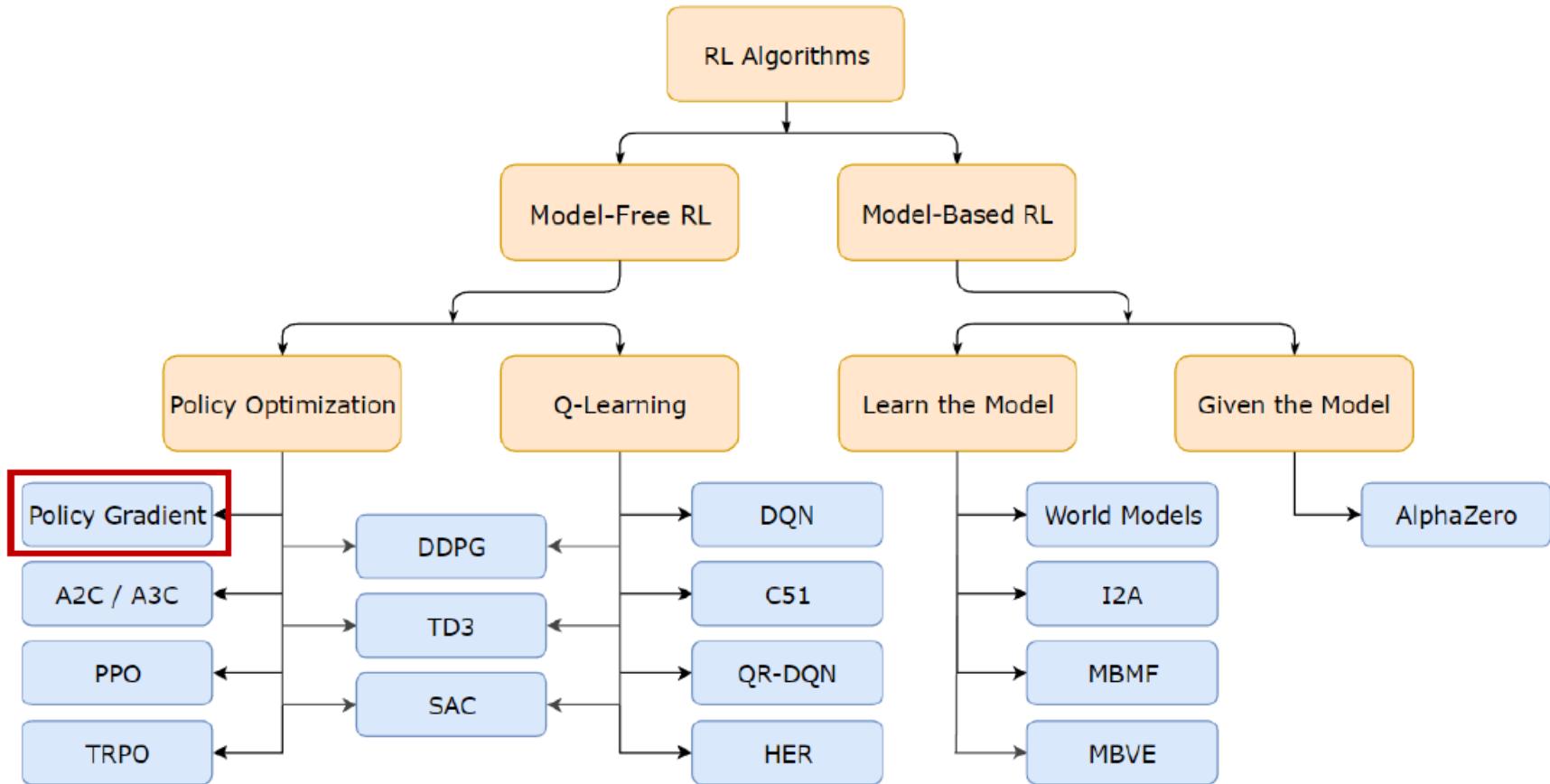
If $a = 1$ only select the experiences with the highest priorities

Normalized by all priority values in Replay Buffer



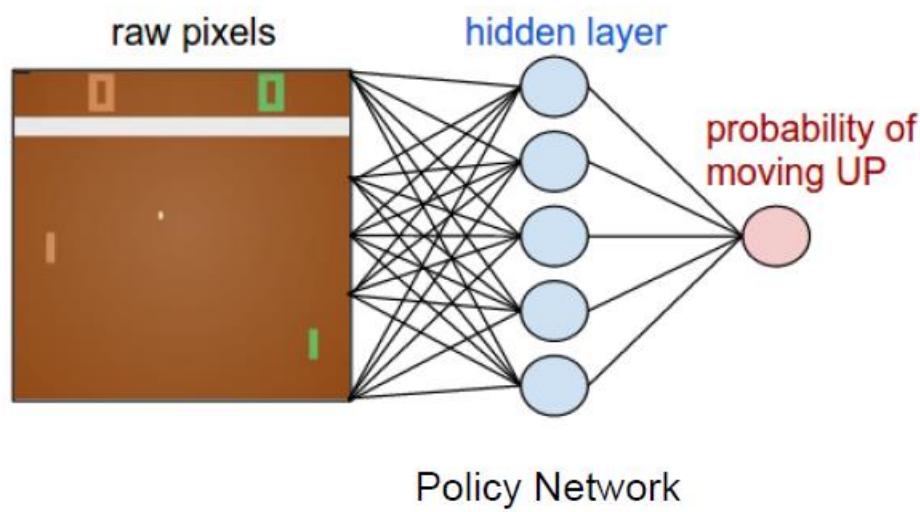
- Sample experiences based on impact not frequency of occurrence

RL Methods



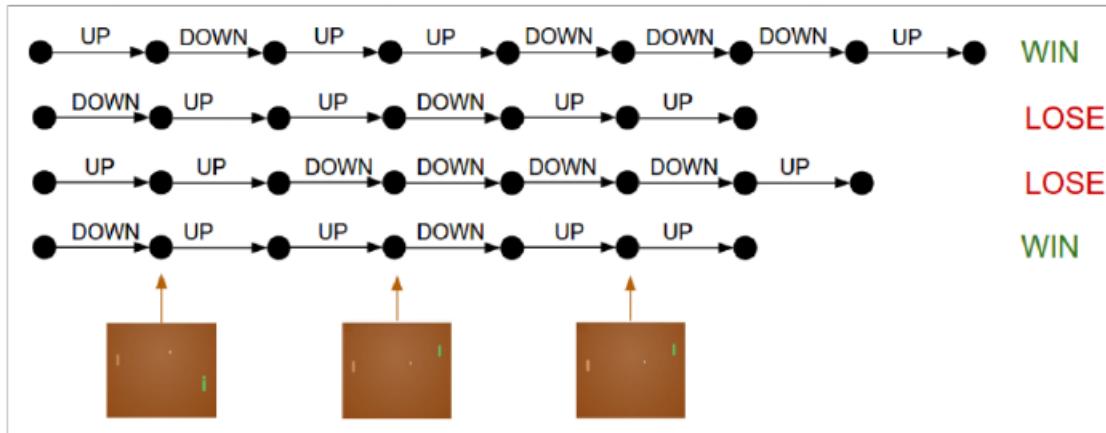
Policy Gradient (PG)

- **DQN (off-policy):** Approximate Q and infer optimal policy
- **PG (on-policy):** Directly optimize policy space



Policy Gradient: REINFORCE

Policy Gradients: Run a policy for a while. See what actions led to high rewards. Increase their probability.



- **REINFORCE:** Policy gradient that increases probability of good actions and decreases probability of bad action:

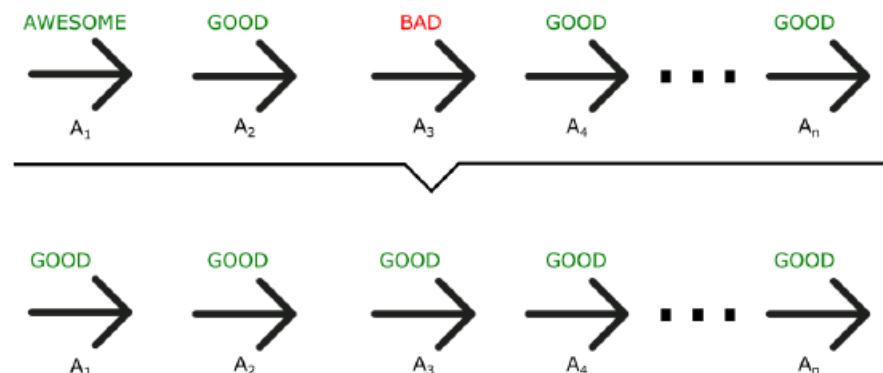
$$\nabla_{\theta} E[R_t] = E[\nabla_{\theta} \log P(a) R_t]$$

Policy Gradient

- Pros vs DQN:
 - **Messy World:** If Q function is too complex to be learned, DQN may fail miserably, while PG will still learn a good policy.
 - **Speed:** Faster convergence
 - **Stochastic Policies:** Capable of learning stochastic policies - DQN can't
 - **Continuous actions:** Much easier to model continuous action space
- Cons vs DQN:
 - **Data:** Sample inefficient (needs more data)
 - **Stability:** Less stable during training process.
 - Poor **credit assignment** to (state, action) pairs for delayed rewards

Problem with REINFORCE:

Calculating the reward at the end, means all the actions will be averaged as good because the total reward was high.



Actor-Critic

- Combine DQN (value-based) and REINFORCE (policy-based)
- Two neural networks (Actor and Critic):
 - **Actor** is policy-based: Samples the action from a policy
 - **Critic** is value-based: Measures how good the chosen action is

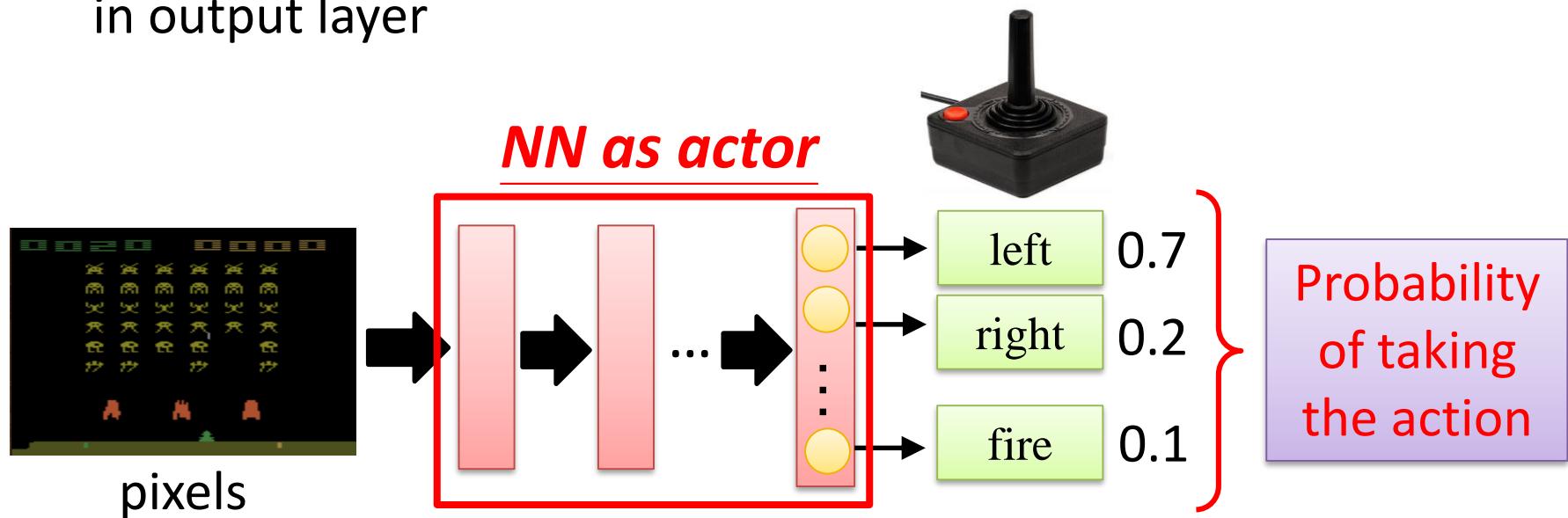
Policy Update: $\Delta\theta = \alpha * \nabla_\theta * (\log \pi(S_t, A_t, \theta)) * \cancel{R(t)}$

New update: $\Delta\theta = \alpha * \nabla_\theta * (\log \pi(S_t, A_t, \theta)) * \boxed{Q(S_t, A_t)}$

- Update at each time step - temporal difference (TD) learning

Neural network as Actor

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network : each action corresponds to a neuron in output layer



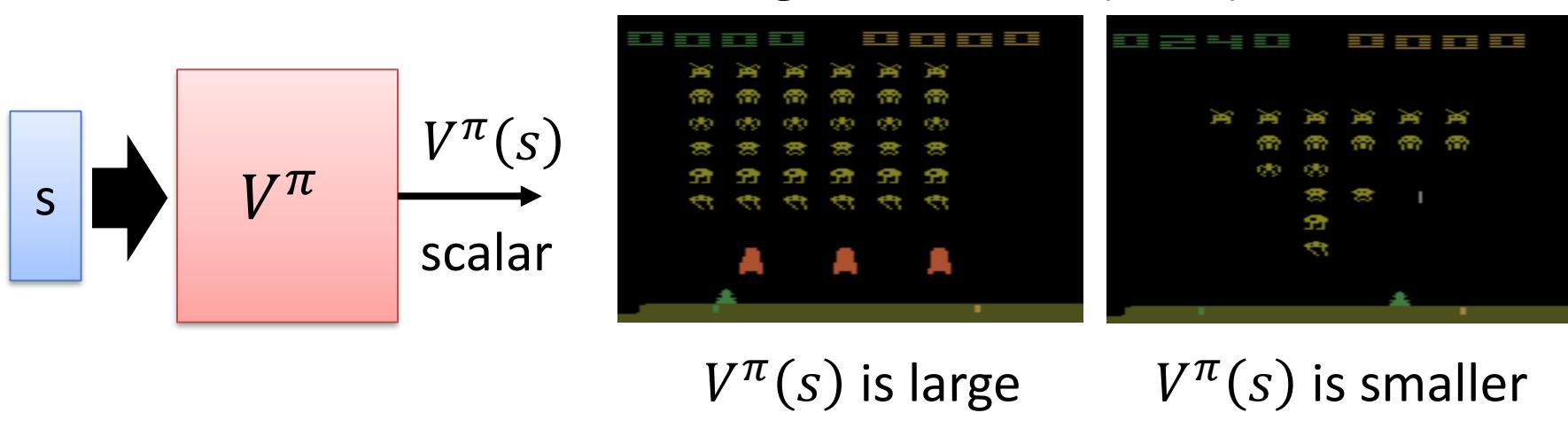
Critic

- A critic does not determine the action.
- Given an actor, it evaluates the how good the actor is



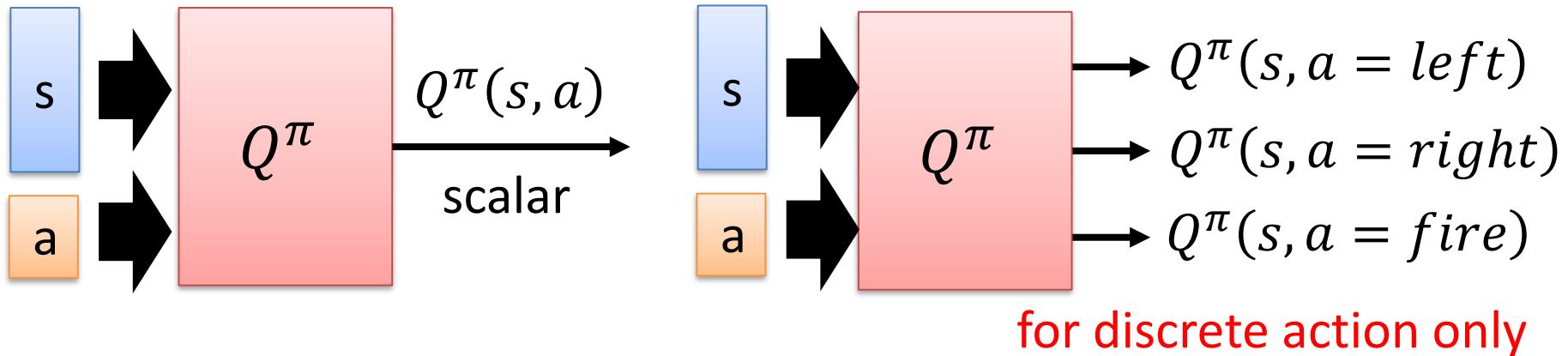
Critics

- A critic is a function depending on the actor π it is evaluated
 - The function is represented by a neural network
- State value function $V^\pi(s)$
 - When using actor π , the *cumulated* reward expects to be obtained after seeing observation (state) s



Critics

- State-action value function $Q^\pi(s, a)$
 - When using actor π , the *cumulated* reward expects to be obtained after seeing observation s and taking a

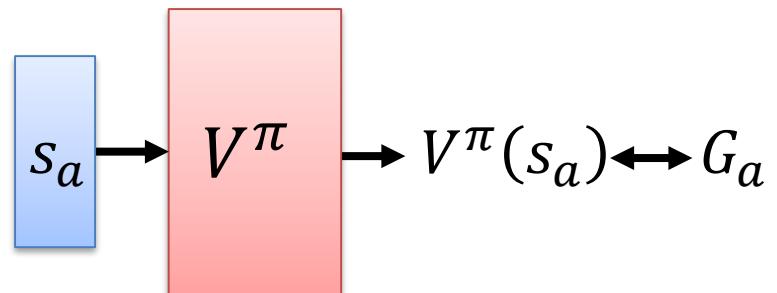


How to estimate $V^\pi(s)$

- Monte-Carlo based approach
 - The critic watches π playing the game

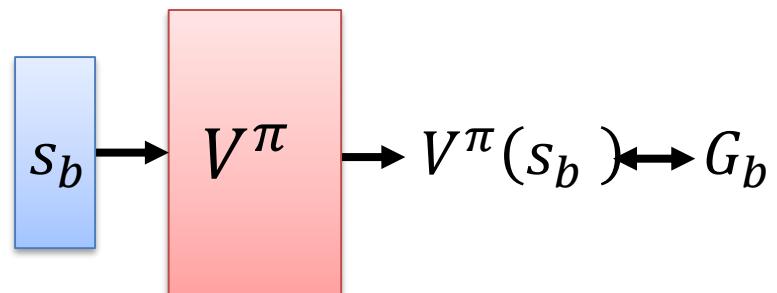
After seeing s_a ,

Until the end of the episode,
the cumulated reward is G_a



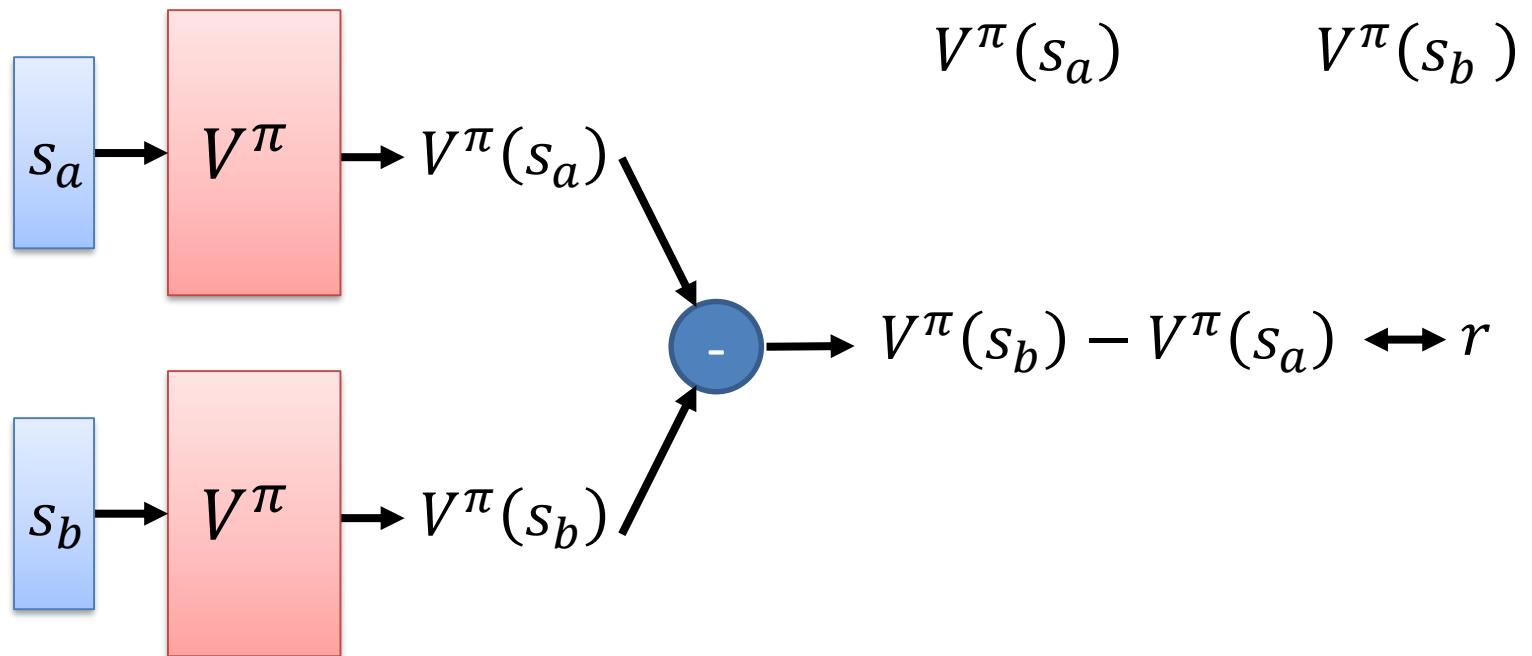
After seeing s_b ,

Until the end of the episode,
the cumulated reward is G_b



How to estimate $V^\pi(s)$

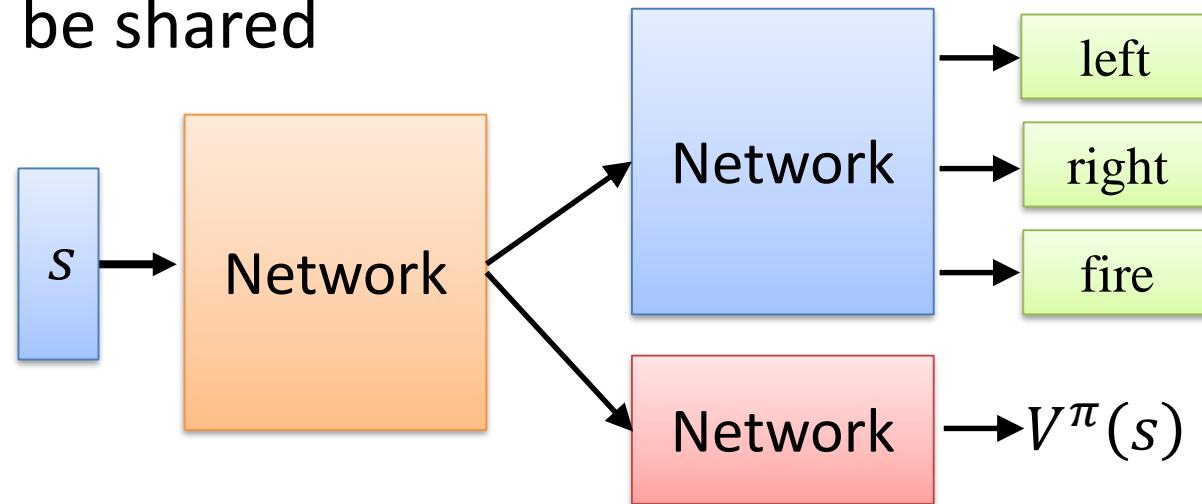
- Temporal-difference approach ... $s_a, a, r, s_b \dots$



Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

Actor-Critic

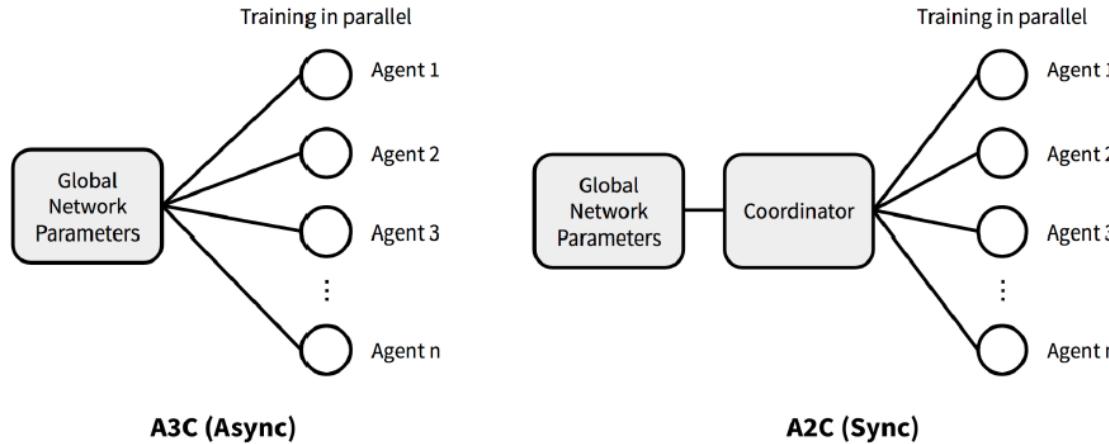
- Tips
 - The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared



- Use output entropy as regularization for $\pi(s)$
 - Larger entropy is preferred → exploration

A2C, A3C

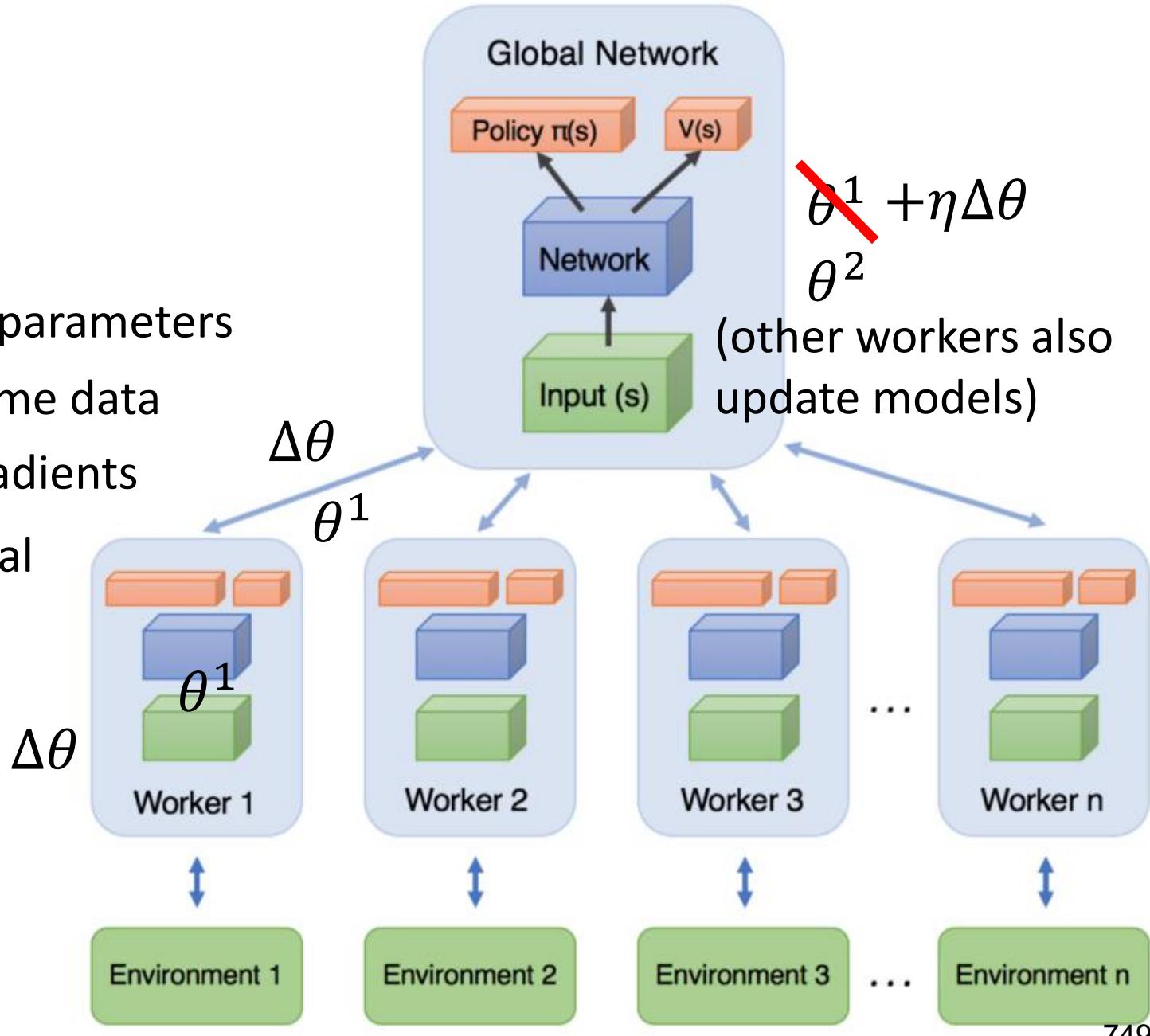
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)



- Both use parallelism in training
- A2C syncs up for global parameter update and then start each iteration with the same policy

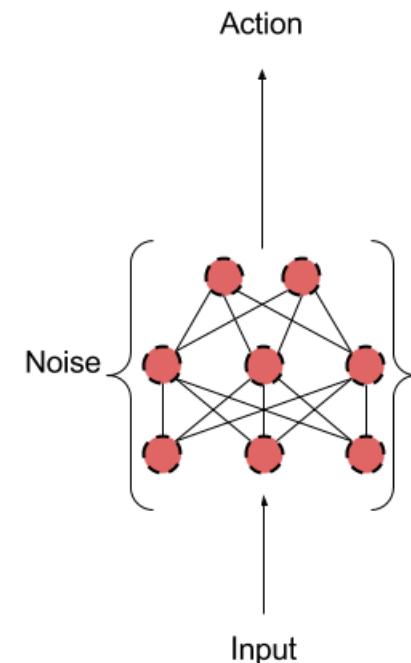
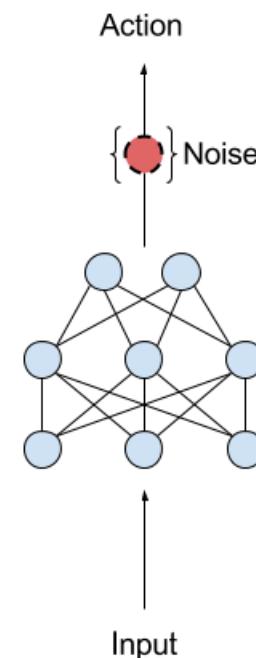
A3C

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models

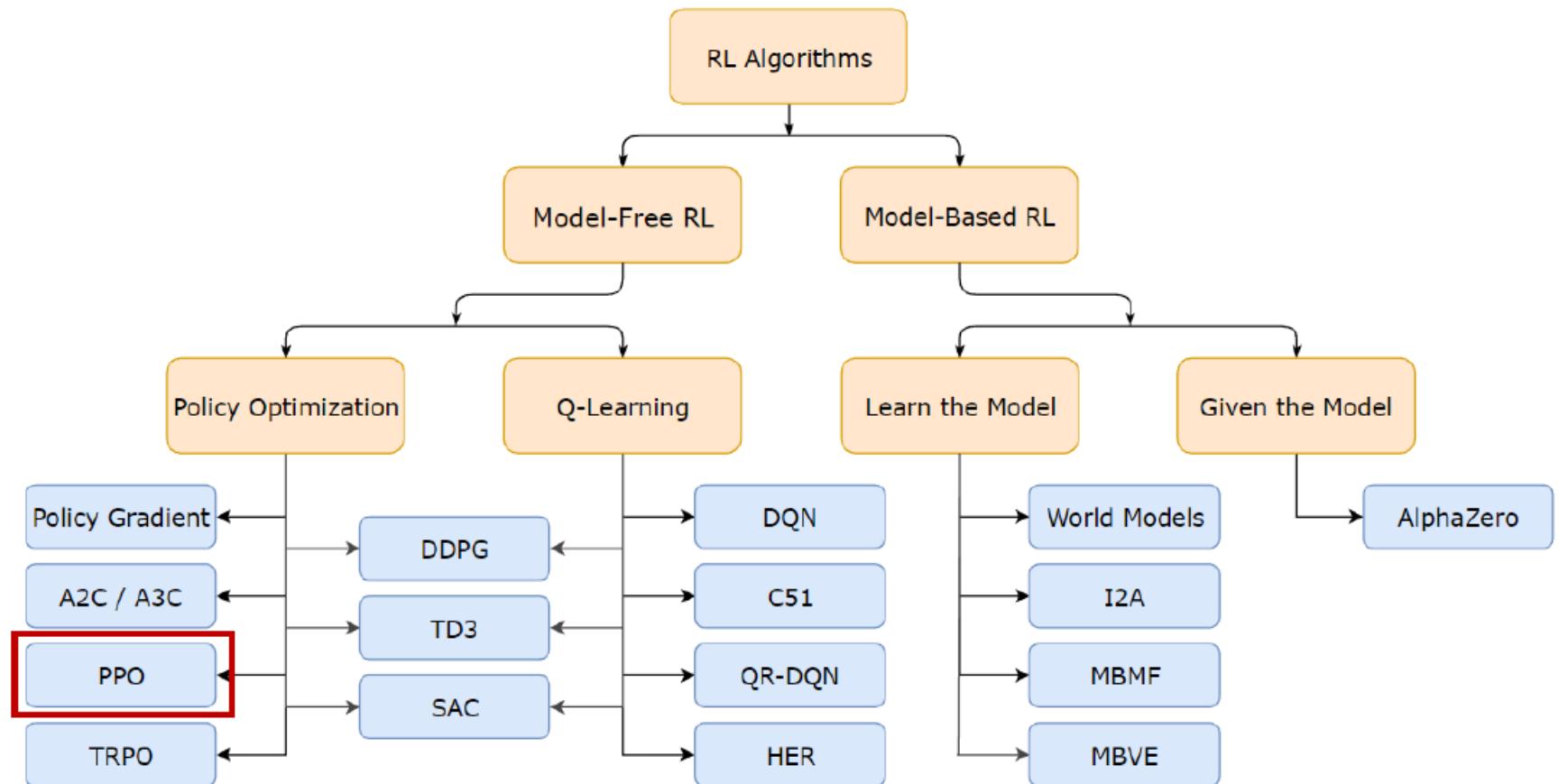


Deep Deterministic Policy Gradient (DDPG)

- Actor-Critic framework for learning a **deterministic** policy
- Can be thought of as: DQN for **continuous action** spaces
- As with all DQN, following tricks are required:
 - Experience Replay
 - Target network
- **Exploration:** add noise to actions, reducing scale of the noise as training progresses



RL Methods



Policy Optimization



- **Progress** beyond Vanilla Policy Gradient:
 - Natural Policy Gradient
 - TRPO
 - PPO
- Basic idea in **on-policy optimization**:
Avoid taking bad actions that collapse the training performance.



Line Search:

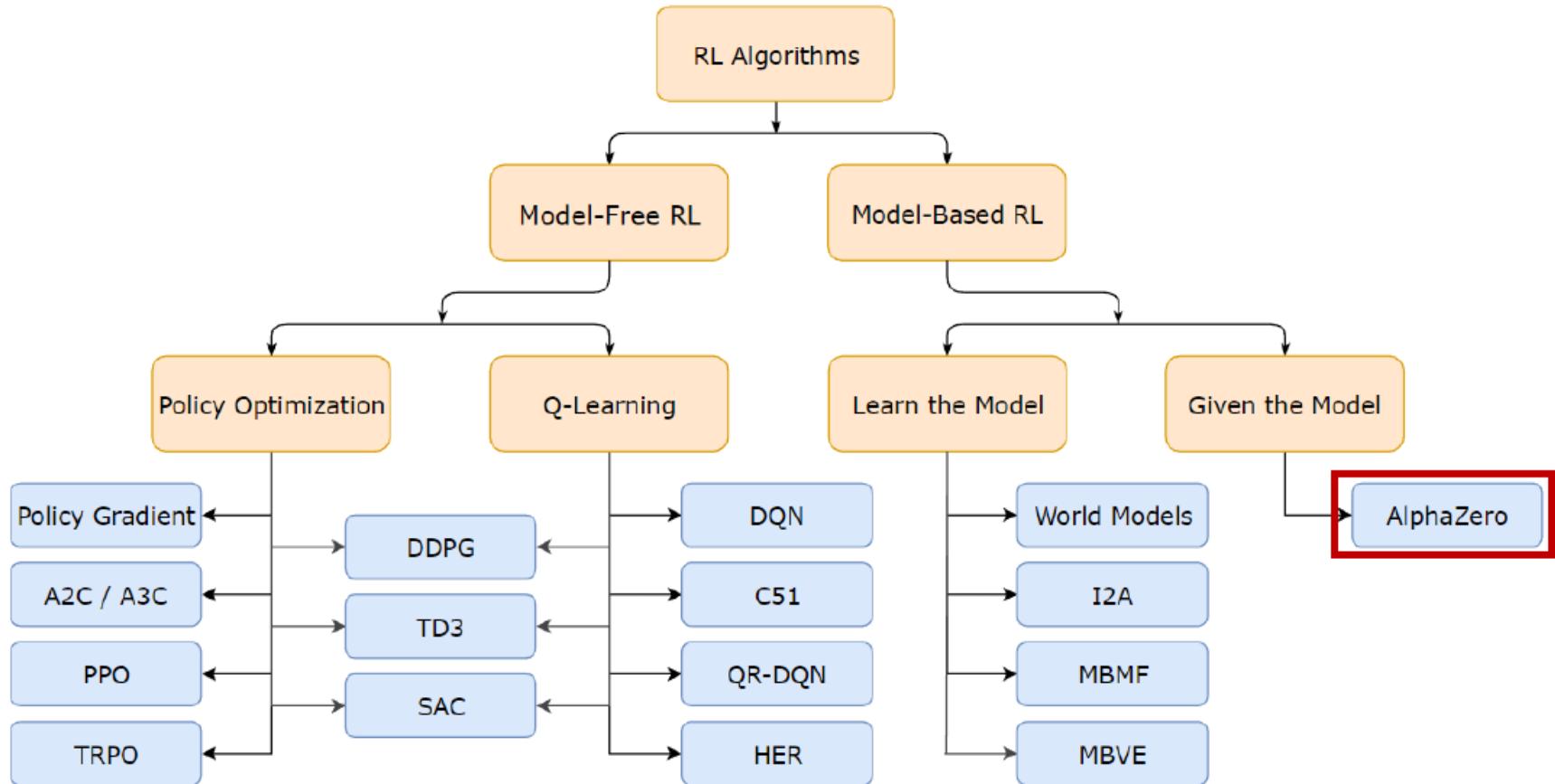
First pick direction, then step size



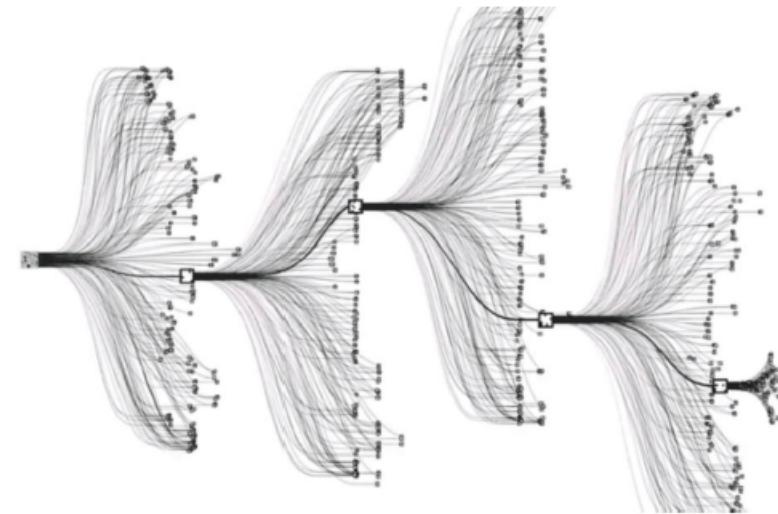
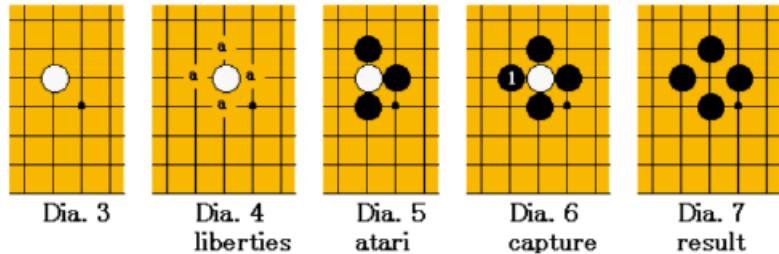
Trust Region:

First pick step size, then direction

RL Methods

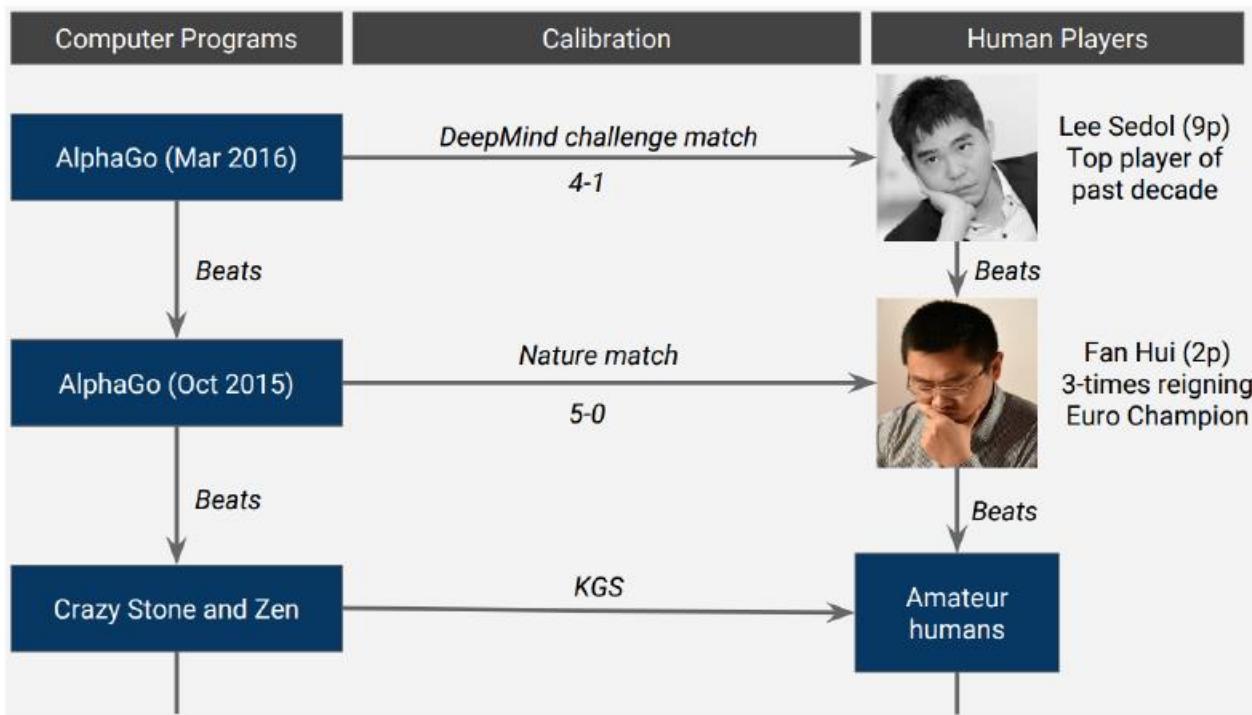
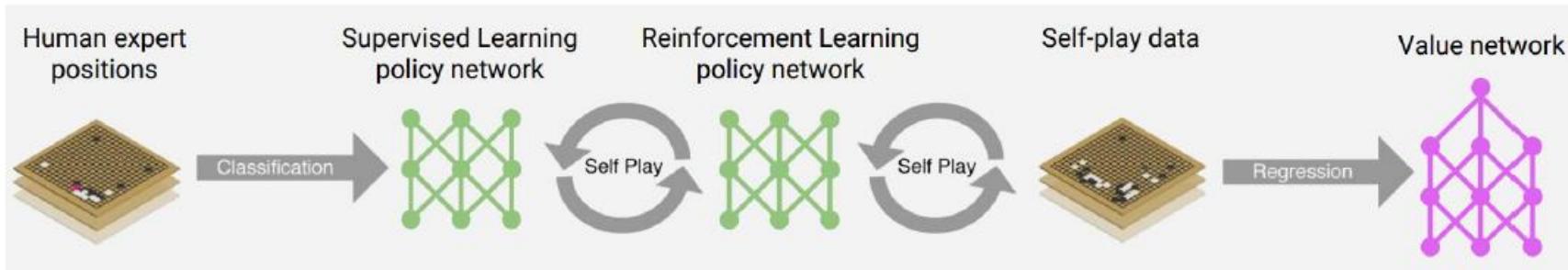


Game of Go



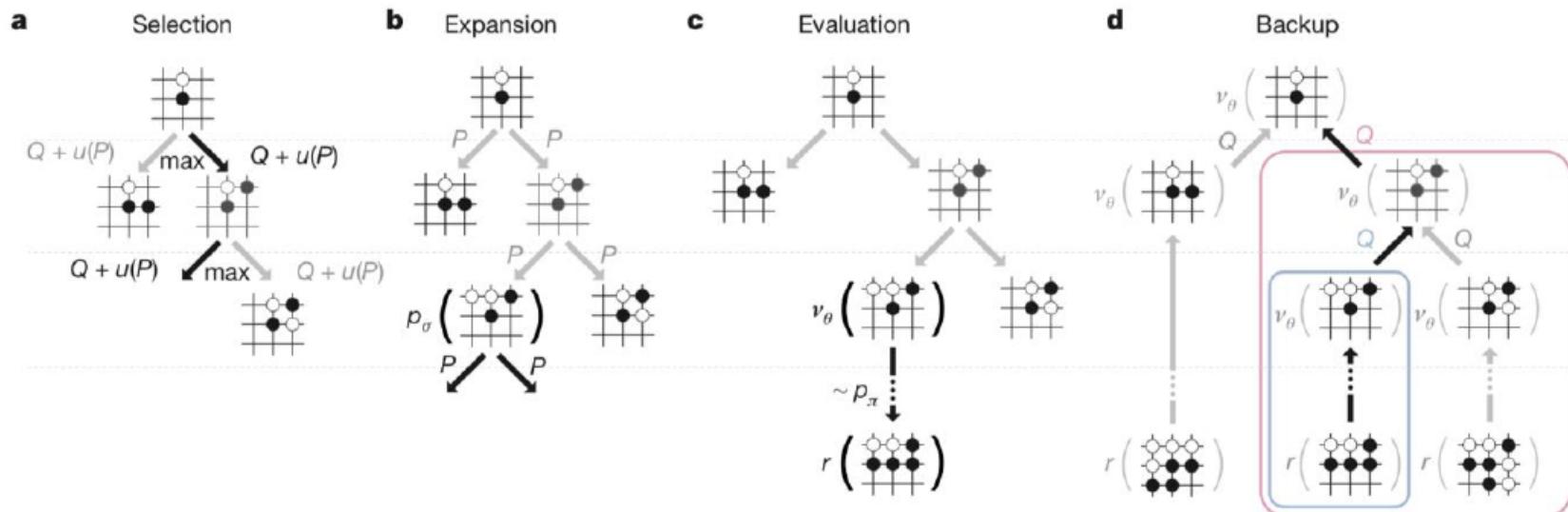
| Game size | Board size N | 3^N | Percent legal | legal game positions (A094777) ^[11] |
|--------------|--------------|------------------------|---------------|--|
| 1×1 | 1 | 3 | 33% | 1 |
| 2×2 | 4 | 81 | 70% | 57 |
| 3×3 | 9 | 19,683 | 64% | 12,675 |
| 4×4 | 16 | 43,046,721 | 56% | 24,318,165 |
| 5×5 | 25 | 8.47×10^{11} | 49% | 4.1×10^{11} |
| 9×9 | 81 | 4.4×10^{38} | 23.4% | 1.039×10^{38} |
| 13×13 | 169 | 4.3×10^{80} | 8.66% | $3.72497923 \times 10^{79}$ |
| 19×19 | 361 | 1.74×10^{172} | 1.196% | $2.08168199382 \times 10^{170}$ |

AlphaGo (2016)



AlphaZero

- Same as the best before: Monte Carlo Tree Search (MCTS)
 - Balance exploitation/exploration (going deep on promising positions or exploring new underplayed positions)
- Use a neural network as “intuition” for which positions to expand as part of MCTS (same as AlphaGo)



AlphaZero

- Same as the best before: Monte Carlo Tree Search (MCTS)
 - Balance exploitation/exploration (going deep on promising positions or exploring new underplayed positions)
- Use a neural network as “intuition” for which positions to expand as part of MCTS (same as AlphaGo)
- “Tricks”
 - Use MCTS intelligent look-ahead (instead of human games) to improve value estimates of play options
 - Multi-task learning: “two-headed” network that outputs (1) move probability and (2) probability of winning.
 - Updated architecture: use residual networks

AlphaZero vs Chess, Shogi, Go

Chess



AlphaZero vs. Stockfish

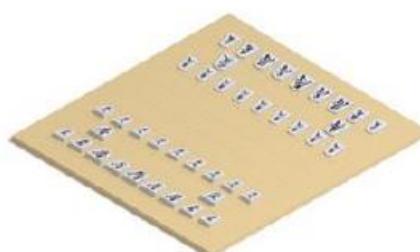
W:29.0% D:70.6% L:0.4%



W:2.0% D:97.2% L:0.8%



Shogi



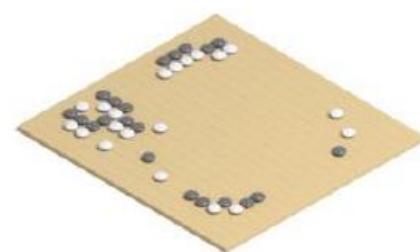
AlphaZero vs. Elmo

W:84.2% D:2.2% L:13.6%



W:98.2% D:0.0% L:1.8%

Go



AlphaZero vs. AGO

W:86.9% D:0.0% L:31.1%



W:53.7% D:0.0% L:46.3%

AZ wins



AZ draws



AZ loses



AZ white



AZ black





University of
Nottingham
UK | CHINA | MALAYSIA

COMP3055

Machine Learning

Topic 18 – Others

Zheng Lu
2024 Autumn

Pre-training and Fine-tuning

- Practical situation
 - You have some specific task that you want to apply machine learning techniques, for example, classifying medical dialogues into certain medical categories.
 - You only have a tiny labelled dataset to train with.
 - There is a general dataset with large number of labelled data.

Pre-training and Fine-tuning

- Pre-training
 - Train the large network models on a large amount of general data
 - Save the network parameters
 - Learn general things

Pre-training and Fine-tuning

- Fine-tuning
 - Initialize the large network model using parameters trained during pre-training.
 - Continue training with additional data (usually small number and task-specific).
 - Adapt to the task.

Using Large Model for Your Own Task

- Large network models trained on a large general dataset.
- Usually target general tasks.
- Using large language models as encoder or feature extractor for your own problem.
 - Backbone for object detection, resnet, inception, etc.
 - Language model for natural language processing, BERT, GPT, etc.

Using Large Model for Your Own Task

- Fine-tune model for your own task.
 - Directly add a classifier after the large model.
 - Update the parameters during the fin
- Create a new network model
 - Add new networks after the large model.
 - During training
 - Freeze the large model and update the subsequent network only.
 - Update both the large model and the subsequent network.

Transformer

- Self-attention
 - Each word' representation as a query to access and incorporate information from a set of values.
 - How important a word is related to others.
- Position representations
 - Specify the sequence order, since self attention is an unordered function of its inputs.
- Nonlinearities
 - At the output of the self-attention block.
 - Frequently implemented as a simple feed-forward network.
- Masking
 - Keeps information about the future from “leaking” to the past.

Self-Attention

- Attention operates on **queries**, **keys**, and **values**.
 - We have some **queries** q_1, q_2, \dots, q_T . Each query is $q_i \in \mathbb{R}^d$
 - We have some **keys** k_1, k_2, \dots, k_T . Each query is $k_i \in \mathbb{R}^d$
 - We have some **values** v_1, v_2, \dots, v_T . Each query is $v_i \in \mathbb{R}^d$
- In **self-attention**, the queries, keys, and values are drawn from the same source.
 - For example, if the output of the previous layer is x_1, \dots, x_T , (one vec per word) we could let $v_i = k_i = q_i = x_i$ (that is, use the same vectors for all of them!)
- The (dot product) self-attention operation is as follows:

$$e_{ij} = q_i^\top k_j$$

Compute **key-query** affinities

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

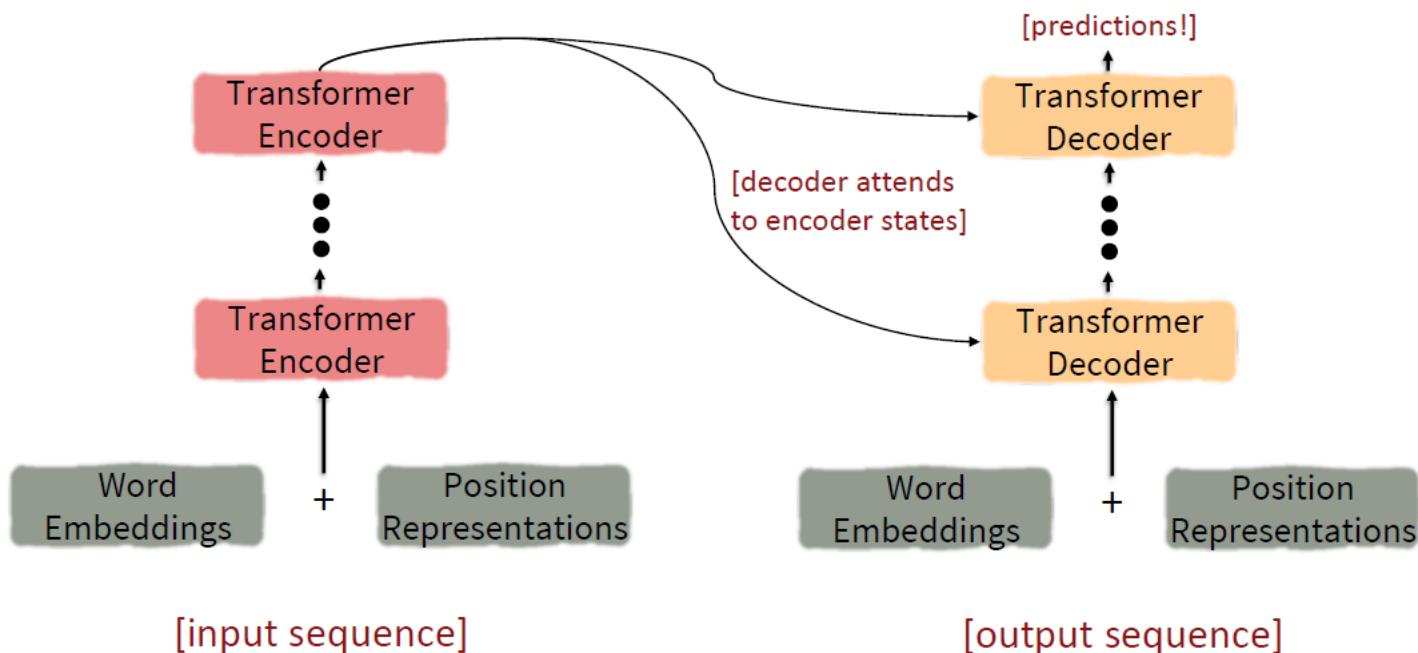
Compute attention weights from affinities
(softmax)

$$\text{output}_i = \sum_j \alpha_{ij} v_j$$

Compute outputs as weighted sum of **values**

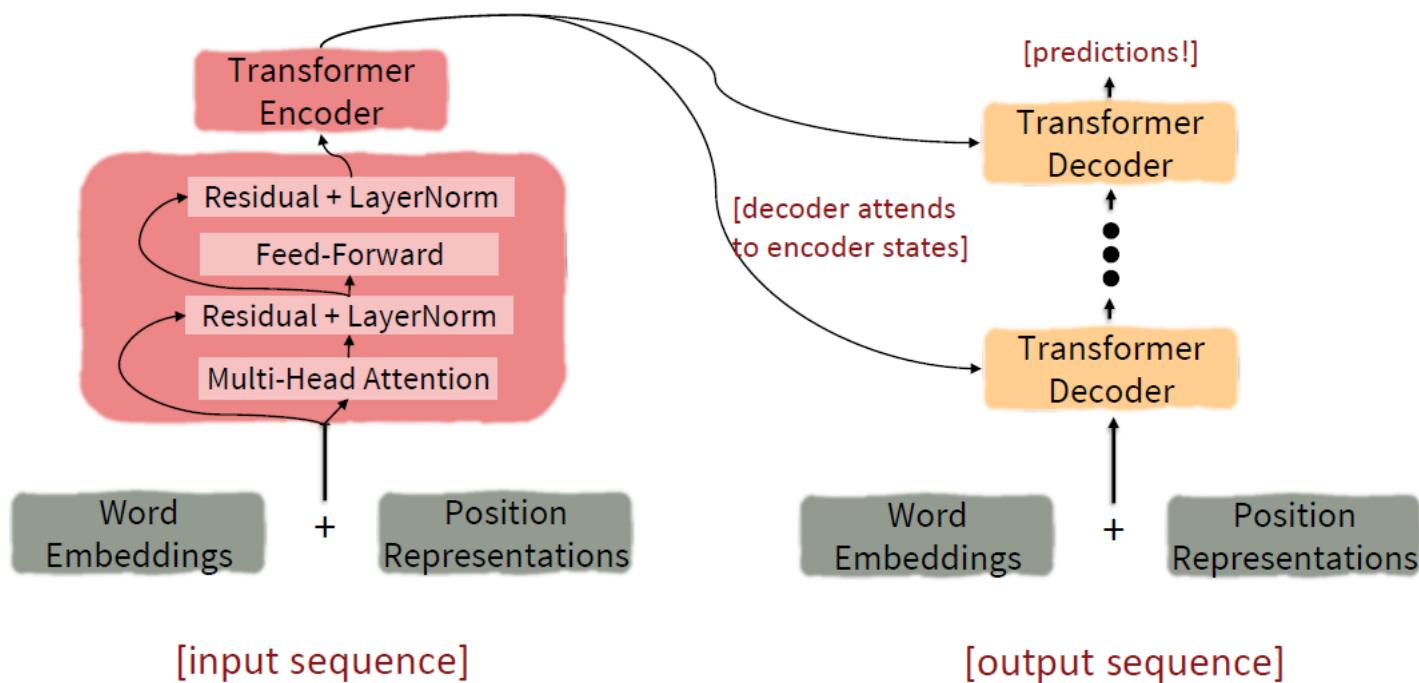
Transformer

- Encoder + decoder



Transformer

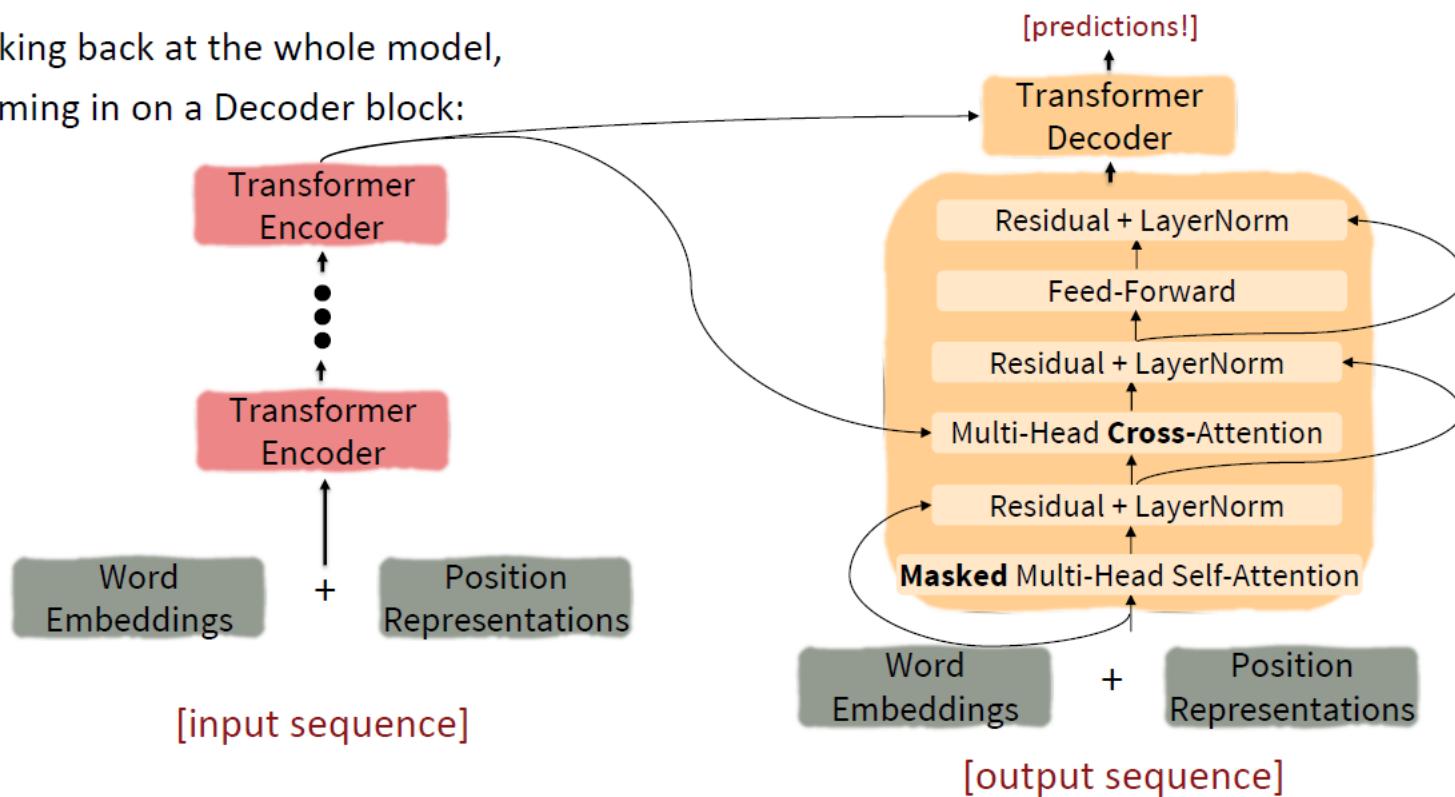
- Zooming into an encoder block



Transformer

- Zooming into an encoder block

Looking back at the whole model,
zooming in on a Decoder block:

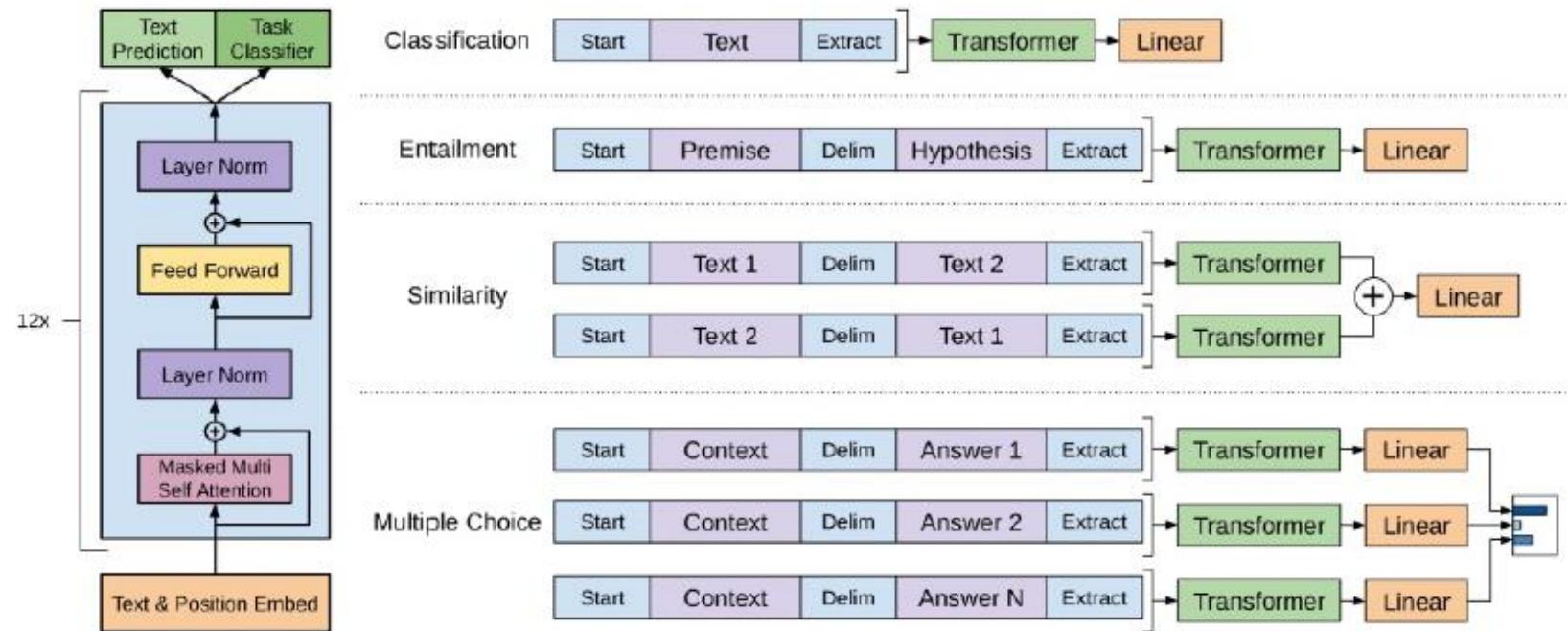


GPT

- Generative pre-trained transformers.
 - A type of large language model.
 - Based on the transformer architecture.
 - Pre-trained on large datasets of unlabeled text.

GPT

How do we format inputs to our decoder for finetuning tasks?



The linear classifier is applied to the representation of the [EXTRACT] token.

GPT-2

- A larger version of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. “By the time we reached the top of one peak, the water looked blue, with some crystals on top,” said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

GPT-2

- Trained on 40GB of text collected from upvoted links from reddit ->1.5B parameters.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. “By the time we reached the top of one peak, the water looked blue, with some crystals on top,” said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

GPT-3

- In-context learning, very large models
 - Very large language models seem to perform some kind of learning without gradient steps simply from examples you provide within their contexts.
- The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

“ thanks -> merce

hello -> bonjour

mint -> menthe

otter -> ”

Output (conditional generations):

loutre...”

GPT-n series

OpenAI's "GPT-n" series

| Model | Architecture | Parameter count | Training data | Release date | Training cost |
|---------|---|--|---|---|---|
| GPT-1 | 12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax. | 117 million | BookCorpus: ^[27] 4.5 GB of text, from 7000 unpublished books of various genres. | June 11, 2018 ^[8] | 30 days on 8 P600 GPUs, or 1 petaFLOP/s-day. ^[8] |
| GPT-2 | GPT-1, but with modified normalization | 1.5 billion | WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit. | February 14, 2019 (initial/limited version) and November 5, 2019 (full version) ^[28] | "tens of petaflop/s-day", ^[29] or 1.5e21 FLOP. ^[30] |
| GPT-3 | GPT-2, but with modification to allow larger scaling | 175 billion ^[31] | 499 billion tokens consisting of CommonCrawl (570 GB), WebText, English Wikipedia, and two books corpora (Books1 and Books2). | May 28, 2020 ^[29] | 3640 petaflop/s-day (Table D.1 ^[29]), or 3.1e23 FLOP. ^[30] |
| GPT-3.5 | Undisclosed | 175 billion ^[31] | Undisclosed | March 15, 2022 | Undisclosed |
| GPT-4 | Also trained with both text prediction and RLHF; accepts both text and images as input. Further details are not public. ^[26] | Undisclosed.
Estimated 1.7 trillion ^[32] | Undisclosed | March 14, 2023 | Undisclosed.
Estimated 2.1e25 FLOP. ^[30] |