

# Lecture 1: Introduction and Overview

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



## This lecture

- Introduction and Warm-up
- Housekeeping COMP90049
- Machine Learning



## **Intros & Warm-up**

---

# Introductions

## About Lida

- Lecturer in CIS since 2019
- Research in graph mining and information retrieval
- PhD from the University of Melbourne
- 4 years of research in academia

## About QiuHong

- Lecturer in CIS since 2020
- Research in machine learning and computer vision
- PhD from the University of Western Australia
- 1.5 years of research in Max Planck Institute for Informatics, Germany



# Introductions

## About Lida

- Lecturer in CIS since 2019
- Research in graph mining and information retrieval
- PhD from the University of Melbourne
- 4 years of research in academia

## About QiuHong

- Lecturer in CIS since 2020
- Research in machine learning and computer vision
- PhD from the University of Western Australia
- 1.5 years of research in Max Planck Institute for Informatics, Germany

## About you

Please go to: [pollev.com/comp90049](https://pollev.com/comp90049)



## Brainstorm / Discuss

**What is Learning?**



## Brainstorm / Discuss

**What is Machine Learning?**



# Definitions of Machine Learning

## Some proposed definitions...

“The computer automatically learns something”

“Statistics, plus marketing”

“ ... how to construct computer programs that automatically improve with experience .... A computer program is said to learn from experience ... if its performance ... improves with experience... ”

Mitchell [1997, pp. xv-17]



# Definitions of Machine Learning

“We are drowning in information, but we are starved for knowledge”

John Naisbitt, Megatrends

## Our definition of Machine Learning

automatic extraction of **valid**, **novel**, **useful** and **comprehensible knowledge** (rules, regularities, patterns, constraints, models, ...) from arbitrary sets of data



# Definitions of Machine Learning

## Learning what?

- **Task** to accomplish a goal, e.g.,
  - Assign continuous values to inputs (essay → grade)
  - Group inputs into known classes (email → {spam, no-spam})
  - Understand regularities in the data

## Learning from what?

- **Data**
- Where do the data come from? Is it reliable? Representative?

## How do we learn?

- define a **model** that explains how to get from input to output
- derive a **learning algorithm** to find the best model parameters

## How do we know learning is happening?

- The algorithm improves at its task with exposure to more data
- We need to be able to **evaluate** performance objectively



## About COMP90049

---

## COMP90049 – Teaching Staff

---

Coordinator & Lecturer1	Lida Rashidi	rashidil@unimelb.edu.au
Lecturer 2	QiuHong Ke	quihong.ke@unimelb.edu.au
Tutors	Tahrima Hashem	tahrima@unimelb.edu.au
	Pei-Yun Sun	pssun@unimelb.edu.au
	Ella Alipourchavary	ella.alipourchavary@unimelb.edu.au
	Kazi Adnan	kazi.adnan@unimelb.edu.au
	Hasti Samadi	hasti.samadi@unimelb.edu.au
	Zenan Zhai	zenan.zhai@unimelb.edu.au

---



THE UNIVERSITY OF  
MELBOURNE

- The lectures will be delivered **fully online**
- I'll aim for as much interaction as possible (and desired)
- All live lectures will be recorded. All recordings and other materials will be made available online through Canvas
- **Live lectures** via Zoom for the first couple of weeks
- Afterwards possibly **pre-recorded with live Q&A sessions**
- **Live and in-person** workshops throughout the semester **dual delivery mode**



## Lectures

---

Lecture 1                    Wed 16:15-17:15

Online; Zoom

---

Lecture 2                    Thu 14:15-15:15

Online; Zoom

---

## Lecture content

- Theory
- Derivation of ML algorithms from scratch
- Motivation and context
- Some coding demos in Python



## Workshops

- **start from week 2**
- 1 hour per week
- ~ 14 slots, please sign up and stick to one
- Online workshops are live via zoom and In-person workshops will be on campus
- At the moment, due to restrictions the on campus workshops are converted to online workshops
- Return to campus will be announced on LMS

## Workshop Content

- Practical exercises
- Working through numerical examples
- Revising theoretical concepts from the lectures



## Coding drop-in sessions

---

Session 1    Tuesday 12:00–13:00 (link via Canvas Zoom)

Session 2    Friday 15:00–16:00 (link via Canvas Zoom)

---

- **start from week 2** and run until week 5
- you can ask questions around Python / the weekly code snippets
- **Not** an assignment consultation



## Materials and announcements

- All materials will be made available through LMS (Canvas)
- Important news will be shared via Canvas Announcements (expect about 1 per week)

## General inquiries: Piazza forum on LMS

- We encourage all students to join in discussions answering other students questions is one of the best ways to improve your own understanding
- Please do not post sections of your code or reports publicly on Piazza! If you must include these, private-message the instructors

## Personal/private concerns: Email your tutor or lecturer

- If you email us about a general inquiry, we may ask you to re-post your question in the forum
- Please include COMP90049 in email subject



## I am looking for 2-3 Student Representatives

- Communication channel between class and teaching team
- Collect and pass on (anonymous) feedback or complaints
- Attend a student-staff meeting during the semester (TBD)
- Represent the **diversity** of the class

**Interested? Send me an email with a short paragraph on why you want this role.**



## Interaction and Engagement

- We'll experiment with breakout rooms, polls, shared whiteboards... please engage!
- Feel free to ask questions / use the chat / raise your hands (I'll do my best to monitor)
- Feedback surveys
- You are encouraged to switch on your camera in lectures and (particularly) workshops to maximize engagement. Please see the recent announcement / post on the subject Home page for acknowledgment of and details on privacy concerns.



- **Topics** include: classification, clustering, optimization, unsupervised learning, semi-supervised learning, neural networks
- All from a theoretical and practical perspective
- Refreshers on maths and programming basics
- Theory in the lectures (some live-coding and demo-ing of libraries and toolkits)
- Hands-on experience in workshops and projects
- **Guest lecture 1:** academic writing skills
- **Guest lecture 2:** Industry talk with focus on bias and fairness in machine learning



THE UNIVERSITY OF  
MELBOURNE

# Expected Background

## Programming concepts

- We will be using **Python** and **Jupyter Notebooks**
- Basic familiarity with libraries (numpy, scikit-learn, scipy)
- You need to be able to write code to process your data, apply different algorithms, and evaluate the output
- Optional practice / demo Jupyter notebooks (most weeks)
- Optional **coding consultation sessions** in the first weeks of semester



# Expected Background

## Programming concepts

- We will be using **Python** and **Jupyter Notebooks**
- Basic familiarity with libraries (numpy, scikit-learn, scipy)
- You need to be able to write code to process your data, apply different algorithms, and evaluate the output
- Optional practice / demo Jupyter notebooks (most weeks)
- Optional **coding consultation sessions** in the first weeks of semester

## Mathematical concepts

- formal maths notation
- basic probability, statistics, calculus, geometry, linear algebra
- (why?)



# What Level of Maths are we Talking?

$$\ln \frac{P(y = \text{true}|x)}{1 - P(y = \text{true}|x)} = w \cdot f$$

$$\frac{P(y = \text{true}|x)}{1 - P(y = \text{true}|x)} = e^{w \cdot f}$$

$$P(y = \text{true}|x) = e^{w \cdot f} / (1 + e^{w \cdot f})$$

$$P(y = \text{true}|x) + e^{w \cdot f} P(y = \text{true}|x) = e^{w \cdot f}$$

$$P(y = \text{true}|x) = h(x) = \frac{e^{w \cdot f}}{1 + e^{w \cdot f}} = \frac{1}{1 + e^{-w \cdot f}}$$

$$P(y = \text{false}|x) = \frac{1}{1 + e^{w \cdot f}} = \frac{e^{-w \cdot f}}{1 + e^{-w \cdot f}}$$



# What Level of Maths are we Talking?

$$P(y = 1|x; \beta) = h_\beta(x)$$

$$P(y = 0|x; \beta) = 1 - h_\beta(x)$$

$$\rightarrow P(y|x; \beta) = (h_\beta(x))^y * (1 - h_\beta(x))^{1-y}$$

$$\begin{aligned} & \underset{\beta}{\operatorname{argmax}} \prod_{i=1}^n P(y_i|x_i; \beta) \\ &= \underset{\beta}{\operatorname{argmax}} \prod_{i=1}^n (h_\beta(x_i))^{y_i} * (1 - h_\beta(x_i))^{1-y_i} \\ &= \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^n y_i \log h_\beta(x_i) + (1 - y_i) \log(1 - h_\beta(x_i)) \end{aligned}$$



# Assessment

## Two small coding projects (30%)

- Project 1: release week 2, due week 3
- Project 2: release week 5, due week 6
- Read in data, apply ML algorithm(s), evaluate.

## Open-ended research project (30%)

- Release week 7, due week 10
- You will be given a data set and will formulate a research question and write a short research paper on your findings. You will be graded based on the quality of your report.

## Final exam (40%)

- during exam period
- 2 hours; closed-book
- **Hurdle requirement:** you have to pass the exam ( $\geq 50\%$ ).



# Academic Honesty

- Videos & Quiz
- Linked from Canvas 'Home' page (or in Modules)
- CIS-specific scenarios

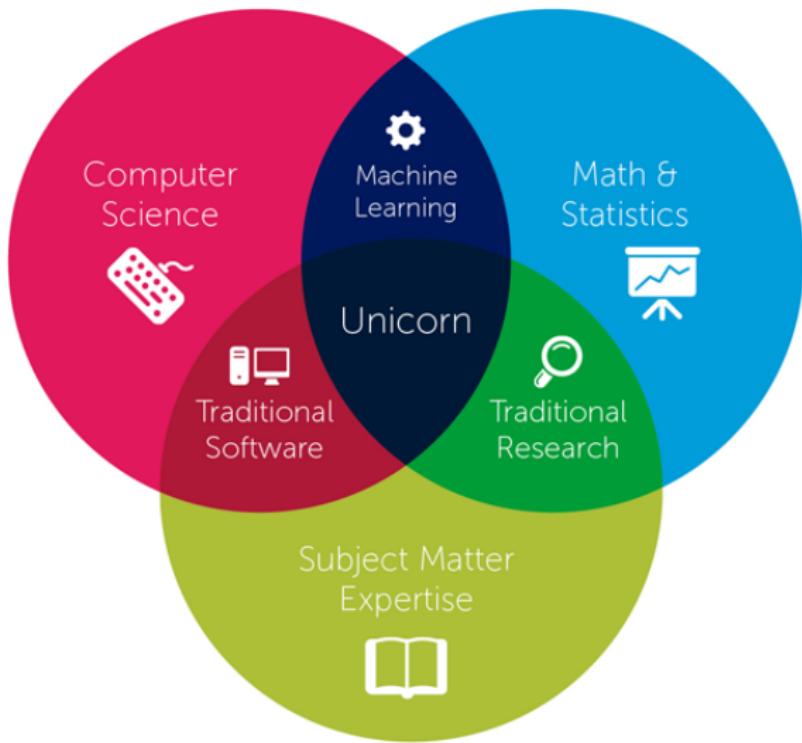
CIS Academic Honesty Training		Complete All Items
Videos		
<input checked="" type="checkbox"/>	Getting help from non-student friends	<input type="radio"/> Mark as done
<input checked="" type="checkbox"/>	Copying the answer from a fellow student	<input type="radio"/> Mark as done
<input checked="" type="checkbox"/>	Getting help from fellow students	<input type="radio"/> Mark as done
<input checked="" type="checkbox"/>	Copying the answer from online sources	<input type="radio"/>
<input checked="" type="checkbox"/>	Do not outsource assignments	<input type="radio"/> Mark as done
<input checked="" type="checkbox"/>	Lock screen when leaving your monitor	<input type="radio"/> Mark as done
<input checked="" type="checkbox"/>	Protect your code (Do not share on Github)	<input type="radio"/> Mark as done
<input checked="" type="checkbox"/>	Working and discussing with friends in the right way	<input type="radio"/> Mark as done
Quiz		
Further information		
<input checked="" type="checkbox"/>	Academic Integrity Principles at Unimelb	<input type="radio"/>
<input checked="" type="checkbox"/>	Further Resources	<input type="radio"/>



## **What and Why of Machine Learning?**

---

# What is Machine Learning?



Copyright © 2014 by Steven Geringer Raleigh, NC.  
Permission is granted to use, distribute, or modify this image,  
provided that this copyright notice remains intact.

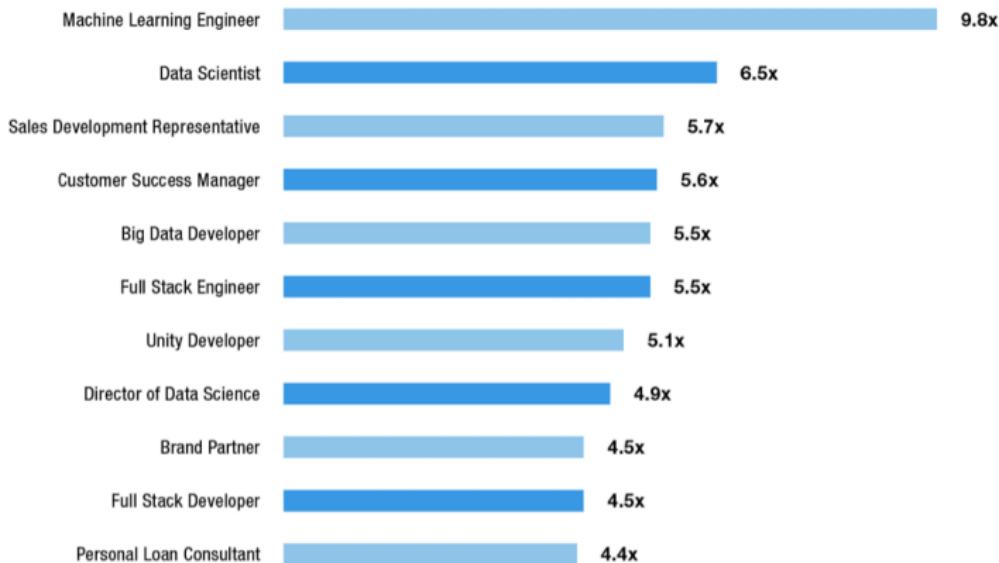
# What is Machine Learning?



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

(you're sitting in the right class!)

## Top 20 Emerging Jobs



Source: <https://www.springboard.com/blog/machine-learning-engineer-salary-guide/>



# Three ingredients for machine learning

... and related questions



# Three ingredients for machine learning

... and related questions

## 1. Data

- Discrete vs continuous vs ...
- Big data vs small data
- Labeled data vs unlabeled data
- Public vs sensitive data



# Three ingredients for machine learning

... and related questions

## Models

- function mapping from inputs to outputs
- motivated by a data *generating* hypothesis
- probabilistic machine learning models
- geometric machine learning models
- parameters of the function are unknown



# Three ingredients for machine learning

... and related questions

## Learning

- Improving (on a task) after data is taken into account
- Finding the best model parameters (for a given task)
- Supervised vs. unsupervised learning



## ML Example Problem

---

## ML Example Problem

- Scenario 1

You are an archaeologist in charge of classifying a mountain of fossilized bones, and want to quickly identify any “finds of the century” before sending the bones off to a museum

- Solution:

Identify bones which are of different size/dimensions/characteristics to others in the sample and/or pre-identified bones



## ML Example Problem

- Scenario 1

You are an archaeologist in charge of classifying a mountain of fossilized bones, and want to quickly identify any “finds of the century” before sending the bones off to a museum

- Solution:

Identify bones which are of different size/dimensions/characteristics to others in the sample and/or pre-identified bones

### CLUSTERING/OUTLIER DETECTION



THE UNIVERSITY OF  
MELBOURNE

## ML Example Problem

- Scenario 2:

You are an archaeologist in charge of classifying a mountain of fossilized bones, and want to come up with a consistent way of determining the species and type of each bone which doesn't require specialist skills

- Solution:

Identify some easily measurable properties of bones (size, shape, number of "lumps", ...) and compare any new bones to a pre-classified database of bones



## ML Example Problem

- Scenario 2:

You are an archaeologist in charge of classifying a mountain of fossilized bones, and want to come up with a consistent way of determining the species and type of each bone which doesn't require specialist skills

- Solution:

Identify some easily measurable properties of bones (size, shape, number of "lumps", ...) and compare any new bones to a pre-classified database of bones

**SUPERVISED CLASSIFICATION ;**



## ML Example Problem

- Scenario 3:

You are in charge of developing the next “release” of Coca Cola, and want to be able to estimate how well received a given recipe will be

- Solution:

Carry out taste tests over various “recipes” with varying proportions of sugar, caramel, caffeine, phosphoric acid, coca leaf extract, ... (and any number of “secret” new ingredients), and estimate the function which predicts customer satisfaction from these numbers



## ML Example Problem

- Scenario 3:

You are in charge of developing the next “release” of Coca Cola, and want to be able to estimate how well received a given recipe will be

- Solution:

Carry out taste tests over various “recipes” with varying proportions of sugar, caramel, caffeine, phosphoric acid, coca leaf extract, ... (and any number of “secret” new ingredients), and estimate the function which predicts customer satisfaction from these numbers

**REGRESSION**

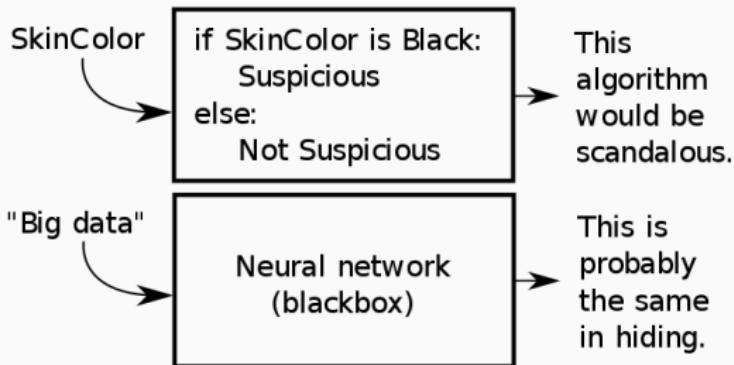


# More Applications

- natural language processing
- image classification
- stock market prediction
- movie recommendation
- web search
- medical diagnoses
- spam / malware detection
- ...



# Machine Learning, Ethics, and Transparency



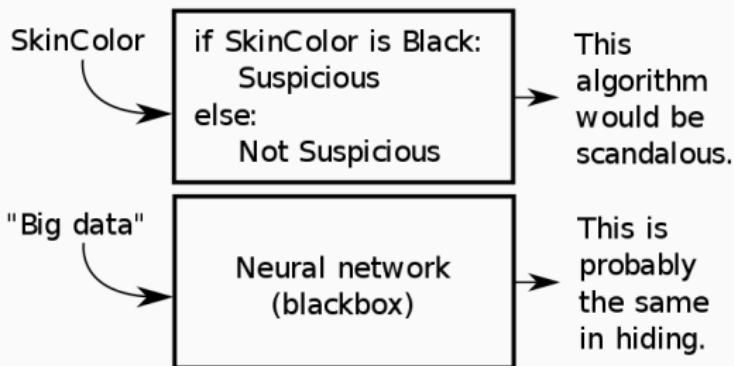
[commons.wikimedia.org/wiki/File:Pseudo-algorithm\\_comparison\\_for\\_my\\_slides\\_on\\_machine\\_learning.ethics.svg](https://commons.wikimedia.org/wiki/File:Pseudo-algorithm_comparison_for_my_slides_on_machine_learning.ethics.svg)

**Def 1. Discrimination= To make distinctions.**

For example, in supervised ML, for a given instance, we might try to discriminate between the various possible classes.



# Machine Learning, Ethics, and Transparency



[commons.wikimedia.org/wiki/File:Pseudo-algorithm\\_comparison\\_for\\_my\\_slides\\_on\\_machine\\_learning\\_ethics.svg](https://commons.wikimedia.org/wiki/File:Pseudo-algorithm_comparison_for_my_slides_on_machine_learning_ethics.svg)

**Def 2. Discrimination=** To make decisions based on prejudice.

Digital computers have no volition, and consequently cannot be prejudiced.

**However,** the data may contain information which leads to an application where the ensuing behavior is prejudicial, intentionally or otherwise.



RETAIL OCTOBER 11, 2018 / 10:04 AM / UPDATED 2 YEARS AGO

## Amazon scraps secret AI recruiting tool that showed bias against women

By Jeffrey Dastin

8 MIN READ



SAN FRANCISCO (Reuters) - Amazon.com Inc's [AMZN.O](#) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.



Take a new look  
at **Amazon jobs**

- Health care

[benefits day 1](#)



# Machine Learning gone wrong...



World Business Markets Breakingviews Video More

MEDIA AND TELECOMS MARCH 25, 2016 / 9:55 AM / UPDATED 5 YEARS AGO

## Microsoft's AI Twitter bot goes dark after racist, sexist tweets

By Amy Tennery, Gina Cherelus

3 MIN READ



(Reuters) - Tay, Microsoft Corp's so-called chatbot that uses artificial intelligence to engage with millennials on Twitter, lasted less than a day before it was hobbled by a barrage of racist and sexist comments by Twitter users that it parroted back to them.



# Machine Learning gone wrong...

BBC | [Sign in](#)

Home | News | Sport | Reel | Worklife | Travel

## NEWS

Home | Coronavirus | Video | World | Asia | UK | Business | Tech | Science | Stories | Entertainment & Arts

Tech

### Facial recognition to 'predict criminals' sparks row over AI bias

24 June 2020

<



GETTY IMAGES

A US university's claim it can use facial recognition to "predict criminality" has renewed debate over racial bias in technology.

Harrisburg University researchers said their software "can predict if someone is a criminal, based solely on a picture of their face".



## Not everything that *can* be done, *should* be done

- Attributes in the data can encode information in an indirect way
- For example, home address and occupation can be used (perhaps with other seemingly-banal data) to infer age and social standing of an individual
- Potential legal exposure due to implicit “knowledge” used by a classifier
- Just because you didn’t realize doesn’t mean that you shouldn’t have realized, or at least, made reasonable efforts to check

## Questions to Ask

- Who is permitted to access the data?
- For what purpose was the data collected?
- What kinds of conclusions are legitimate?
- If our conclusions defy common sense, are there confounding factors?
- Could my research / application be abused (*dual use*)?



# Summary

## Today

- COMP90049 Overview
- What is machine learning?
- Why is it important? Some use cases.
- What can go wrong?

**Next lecture:** Concepts in machine learning



## References i

- Jacob Eisenstein. Natural Language Processing. MIT Press (2019)
- Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. Mathematics for Machine Learning. Cambridge University Press (forthcoming)
- Chris Bishop. Pattern Recognition and Machine Learning. Springer (2009)
- Tom Mitchell. Machine Learning. McGraw-Hill, New York, USA (1997).



## References ii

Microsoft's AI robot goes dark.

<https://www.reuters.com/article/us-microsoft-twitter-bot-idUSKCN0WQ2LA>

Amazon scraps secret recruiting tool.

<https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G>

Predictive policing algorithms are biased.

<https://www.bbc.com/news/technology-53165286>



# Lecture 2: Machine Learning Concepts

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



# Roadmap

## Last lecture

- Warm-up
- Housekeeping COMP90049
- Machine Learning

## This lecture

- Establishing terminology
- Basic concepts of ML: instances, attributes, learning paradigms, ...
- Python demo



## **Basics of ML: Instances, Attributes and Learning Paradigms**

---

# Typical Workflow



# Terminology

- The input to a machine learning system consists of:
  - **Instances:** the individual, independent examples of a concept  
also known as **exemplars**
  - **Attributes:** measuring aspects of an instance  
also known as **features**
  - **Concepts:** things that we aim to learn  
generally in the form of **labels** or **classes**



THE UNIVERSITY OF  
MELBOURNE

## Example: weather.nominal Dataset

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
⋮	⋮	⋮	⋮	⋮



## Example: weather.nominal Dataset

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
⋮	⋮	⋮	⋮	⋮



## Example: weather.nominal Dataset

ATTRIBUTE 1	Outlook	Temperature	Humidity	Windy	Play
	sunny	hot	high	FALSE	no
	sunny	hot	high	TRUE	no
	overcast	hot	high	FALSE	yes
	rainy	mild	high	FALSE	yes
	rainy	cool	normal	FALSE	yes
	rainy	cool	normal	TRUE	no
ATTRIBUTE 2			:	:	:



## Quiz I

8	4	5	2	3	8	4	8
0	3	0	6	2	9	9	4
8	9	0	3	8	3	2	1

The MNIST digit classification data set

- How many **instances** do you see in the dataset above?



## Quiz I

8	4	5	2	3	8	4	8
0	3	0	6	2	9	9	4
8	9	0	3	8	3	2	1

### The MNIST digit classification data set

- How many **instances** do you see in the dataset above?
- What are these instances?



## Quiz I

8	4	5	2	3	8	4	8
0	3	0	6	2	9	9	4
8	9	0	3	8	3	2	1

### The MNIST digit classification data set

- How many **instances** do you see in the dataset above?
- What are these instances?
- What **features** or **attributes** do we have for the instances?



# What's a Concept?

**“Concepts” that we aim to learn:**

- Predicting a discrete class (Classification)
- Grouping similar instances into clusters (Clustering)
- Predicting a numeric quantity (Regression)
- Detecting associations between attribute values (Association Learning)



## A Word on Supervision

- **Supervised** methods have prior knowledge of a closed set of classes and set out to discover and categorise new instances according to those classes
- **Unsupervised** do not have access to an inventory of classes, and instead discover groups of ‘similar’ examples in a given dataset. Two flavors :



## A Word on Supervision

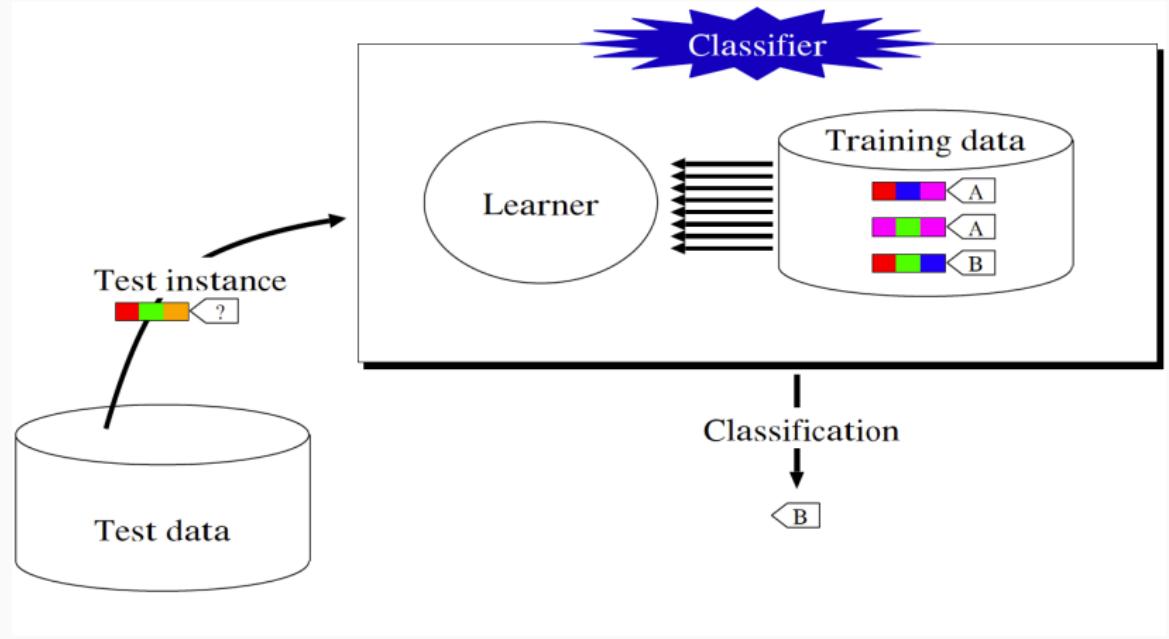
- **Supervised** methods have prior knowledge of a closed set of classes and set out to discover and categorise new instances according to those classes
- **Unsupervised** do not have access to an inventory of classes, and instead discover groups of ‘similar’ examples in a given dataset. Two flavors :
  - dynamically discover the “classes” (implicitly derived from grouping of instances) in the process of categorising the instances **[STRONG]**
  - ... *OR* ...
  - categorise instances as certain labels without the aid of pre-classified data **[WEAK]**



# Classification

- Assigning an instance a discrete class label
- Classification learning is **supervised**
- Scheme is provided with actual outcome or **class**
- The learning algorithm is provided with a set of classified **training data**
- Measure success on “held-out” data for which class labels are known  
**(test data)**





## Example Predictions for weather.nominal

Outlook	Temperature	Humidity	Windy	True Label	Classified
sunny	hot	high	FALSE	no	
sunny	hot	high	TRUE	no	
overcast	hot	high	FALSE	yes	
rainy	mild	high	FALSE	yes	
rainy	cool	normal	FALSE	yes	
rainy	cool	normal	TRUE	no	
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	
sunny	cool	normal	FALSE	yes	
rainy	mild	normal	FALSE	yes	
sunny	mild	normal	TRUE	yes	no
overcast	mild	high	TRUE	yes	yes
overcast	hot	normal	FALSE	yes	yes
rainy	mild	high	TRUE	no	yes



- Finding groups of items that are similar
- Clustering is **unsupervised** — the learner operates without a set of labelled training data
- The class of an example is not known ... or at least, not given to the learning algorithm
- Success often measured subjectively; evaluation is problematic



## Clustering over weather.nominal

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
:	:	:	:	:



# Regression

- Classification learning, but class is continuous (**numeric prediction**)
- Learning is **supervised**
- Why is this distinct from Classification?
  - In Classification, we can exhaustively enumerate all possible labels for a given instance; a correct prediction entails mapping an instance to the label which is truly correct
  - In Regression, infinitely many labels are possible, we cannot conceivably enumerate them; a “correct” prediction is when the numeric value is acceptably close to the true value



## Example Predictions for weather

Outlook	Humidity	Windy	Play	Actual Temp	Classified Temp
sunny	85	FALSE	no	85	
sunny	90	TRUE	no	80	
overcast	86	FALSE	yes	83	
rainy	96	FALSE	yes	70	
rainy	80	FALSE	yes	68	
rainy	70	TRUE	no	65	
overcast	65	TRUE	yes	64	
sunny	95	FALSE	no	72	
sunny	70	FALSE	yes	69	
rainy	80	FALSE	yes	75	
sunny	70	TRUE	yes	75	68.8
overcast	90	TRUE	yes	72	71.2
overcast	75	FALSE	yes	81	70.6
rainy	91	TRUE	no	71	76.5



## Quiz II

8	4	5	2	3	8	4	8
0	3	0	6	2	9	9	4
8	9	0	3	8	3	2	1

### The MNIST digit classification data set

- Design a **classification** task given this data set. What ‘concept(s)’ could be learnt?
- Could we perform **clustering** instead? What would change?
- Can you think of a meaningful **regression** task?



# Instance Topology

- Instances characterised as “feature vectors”, defined by a predetermined set of attributes
- Input to learning scheme: set of instances/dataset
  - Flat file representation
  - No relationships between objects
  - No explicit relationship between attributes
- Attribute Data types
  1. discrete: nominal (also: categorical) or ordinal
  2. continuous: numeric

**What about class label data types?**



## Nominal Quantities

- Values are distinct symbols (e.g. {sunny,overcast,rainy})
  - values themselves serve only as labels or names
- Also called **categorical**, or **discrete**
- Special case: dichotomy (“Boolean” attribute)
- No relation is implied among nominal values (no ordering or distance measure), and only equality tests can be performed



## Ordinal Quantities

- An explicit order is imposed on the values (e.g. {hot,mild,cool} where hot > mild > cool)
- No distance between values defined and addition and subtraction don't make sense
- Example rule: temperature < hot → play = yes
- Distinction between nominal and ordinal not always clear (e.g. outlook)



## Numeric Quantities

- Numeric quantities are real-valued attributes
- Scalar (a single number): attribute *distance*
- Vector-valued (a vector of numbers each pertaining to a feature or feature value): attribute *position* (x,y coordinate)
- All mathematical operations are allowed (of which addition, subtraction, scalar multiplication are most salient, but other operations are relevant in some contexts)



# Attribute Types and Machine Learning Models

**Most machine learning algorithms assume a certain type of attribute**

- Naive Bayes: nominal or numeric
- Logistic/Linear Regression: numeric
- Perceptron/Neural networks: numeric

**If we have the wrong attribute type for our algorithm (or attributes with different types for each instance), we can**

- Select only attributes with the correct type
- Change the model assumptions to match the data
- **Change the attributes to match the model**



# Converting Nominal to Numeric Attributes

## Option 1: Map category names to numbers

- “red”, “blue”, “green”, “yellow”
- 0, 1, 2, 3

Graphical representation:

- Problem: creates an artificial ordering. Some categories will appear more similar to each other than others
- Especially problematic with a large number of categories



# Converting Nominal to Numeric Attributes

## Option 2: One-hot encoding

“red”	=	[1, 0, 0, 0]
“blue”	=	[0, 1, 0, 0]
“green”	=	[0, 0, 1, 0]
“yellow”	=	[0, 0, 0, 1]

Graphical representation:

- Better way of encoding categorical attributes in a numeric way
- Problem: increases the dimensionality of the feature space



# Numeric Feature Normalization

## Features of vastly different scale can be problematic

- Some machine learning models assume features to follow a normal distribution
- Some learning algorithms are overpowered by large feature values (and ignore smaller ones)
- Feature **standardization** rescales features to be distributed around a 0 mean with a unit standard deviation.

$$x' = \frac{x - \mu}{\sigma}$$

- Feature **scaling** rescales features to a given range. For example, **Min-max scaling** rescales values between 0 and 1 using the minimum and maximum feature value observed in the data

$$x' = \frac{x - X_{min}}{X_{max} - X_{min}}$$



## Converting Numeric to Nominal Attributes

**Discretization:** Group numeric values into a pre-defined set of distinct categories. E.g., map housing prices to {"high", "medium", "low"}

To do this, we

- First, decide on the number of categories
- Secondly, decide on the category boundaries



# Converting Numeric to Nominal Attributes

**Discretization:** Group numeric values into a pre-defined set of distinct categories. E.g., map housing prices to {"high", "medium", "low"}

To do this, we

- First, decide on the number of categories
- Secondly, decide on the category boundaries

**Option 1:** Equal widths discretisation

- Find the minimum and maximum of the data
- Partition the values into  $n$  bins of width  $(\text{max-min})/n$  bins
- **Problem 1:** outliers
- **Problem 2:** bins may end up with vastly different number of items
- **Problem 3:** how to select  $n$ ?



# Converting Numeric to Nominal Attributes

**Discretization:** Group numeric values into a pre-defined set of distinct categories. E.g., map housing prices to {"high", "medium", "low"}

To do this, we

- First, decide on the number of categories
- Secondly, decide on the category boundaries

**Option 2:** Equal frequency discretisation

- Sort the values
- Partition them into  $n$  bins such that each bin has an identical number of items
- **Problem 1:** boundaries could be hard to interpret
- **Problem 2:** how to select  $n$ ?



# Converting Numeric to Nominal Attributes

**Discretization:** Group numeric values into a pre-defined set of distinct categories. E.g., map housing prices to {"high", "medium", "low"}

To do this, we

- First, decide on the number of categories
- Secondly, decide on the category boundaries

**Option 3:** Clustering

- Use unsupervised machine learning to group the value into  $n$  clusters
- For example: K-means clustering (more on that later)
- **Problem 1:** how to evaluate the result?
- **Problem 2:** how to select  $K$ ?



## **ML in the Wild**

---

## Preparing the Input

- Problem: different data sources (e.g. sales department, customer billing department, ...)
  - Differences: styles of record keeping, conventions, time periods, data aggregation, primary keys, errors
  - Data must be assembled, integrated, cleaned up
  - Data warehouse: consistent point of access
- External data/storage may be required
- Critical: type and level of data aggregation



# Missing Values

- The number of attributes may vary in practice
  - missing values
  - inter-dependent attributes
- Typical cases of out-of-range values:
  - Types: unknown, unrecorded, irrelevant
  - Reasons:
    - malfunctioning equipment
    - changes in experimental design
    - collation of different datasets
    - measurement not possible
- Most models assume that values are **missing at random**
- Missing value may have significance in itself (e.g. missing test in a medical examination) → **Missing not at random.**  
Missing may need to be coded discretely.



## Inaccurate Values

- Cause: a given data mining application is often not known at the time logging is set up
- Result: errors and omissions that don't affect original purpose of data (e.g. age of customer)
- Typographical errors in nominal attributes → values need to be checked for consistency
- Typographical and measurement errors in numeric attributes → outliers need to be identified
- Errors may be deliberate (e.g. wrong post codes)



## Getting to Know the Data

- Simple visualization tools are very useful
  - Nominal attributes: histograms (distribution consistent with background knowledge?)
  - Numeric attributes: scatter plots (any obvious outliers?)
- 2-D and 3-D plots show dependencies
- Need to consult domain experts
- Too much data to inspect? Take a sample!
- You can never know your data too well (**or can you?**)



# Coding Demo!

## Intended take-aways

- Starting Jupyter Notebook
- Reading in a dataset (using basic Python)
- Reading in a dataset (using the pandas library)
- Formatting a dataset into lists (of instances)
- Separating features from class labels (for each instance)



# Summary

## Today: establishing common vocabulary

- What are instances, attributes and concepts?
- Learning paradigms: supervised and unsupervised
- Concepts: Regression, Classification, Clustering
- Attributes: types and encodings
- Python and Jupyter

## Next: K-Nearest Neighbors



# Lecture 3: K-Nearest Neighbors

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



## Last time... Machine Learning concepts

- data, features, classes
- models, training
- practical considerations

## Today... Our first machine learning algorithm

- K-nearest neighbors
- Application to classification
- Application to regression

Also: the topic of your **first assignment!**

- Released on Canvas on Wed 4th August at 7pm!
- Questions: assignment1 discussion board (don't share solutions!)



## **Introduction**

---

## K-Nearest Neighbors: Example

Your 'photographic memory' of all handwritten digits you've ever seen:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9



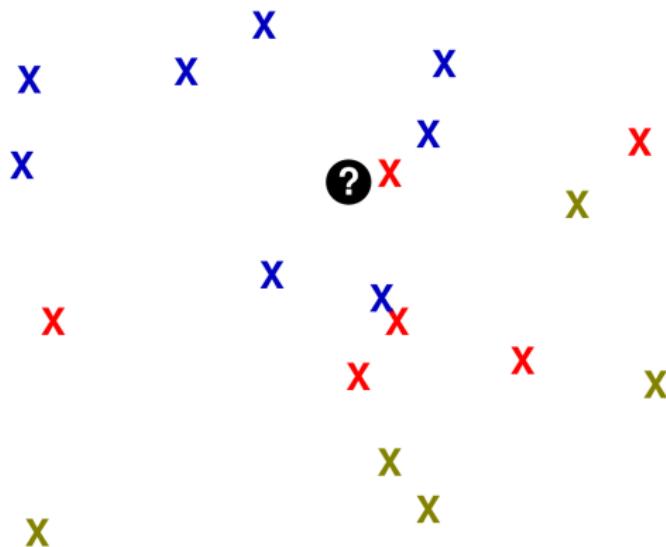
## K-Nearest Neighbors: Example

Your 'photographic memory' of all handwritten digits you've ever seen:

Given a new drawing, determine the digit by comparing it to all digits in your 'memory'.



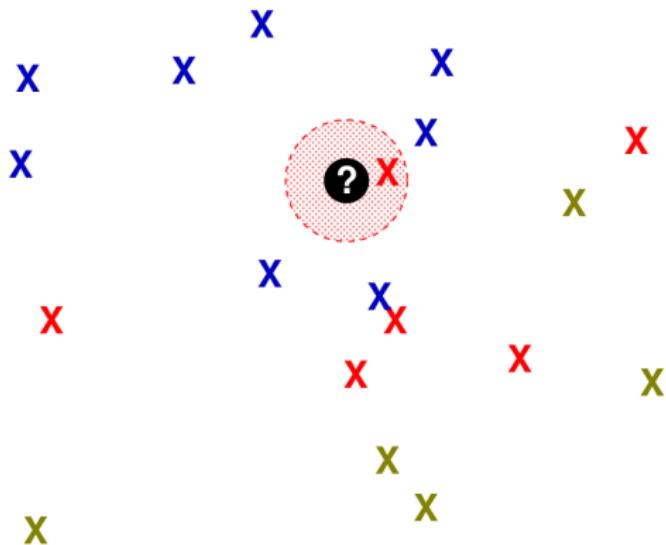
# K-Nearest Neighbors: Visualization



$K$  nearest neighbors =  $K$  closest stored data points



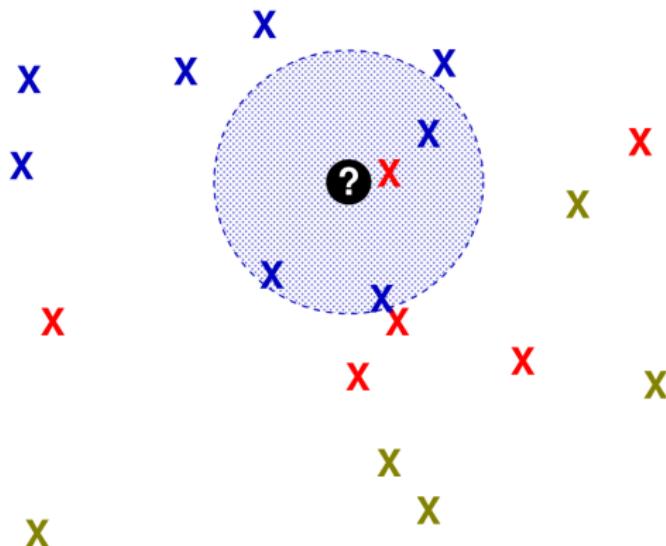
# K-Nearest Neighbors: Visualization



1 nearest neighbor = single closest stored data point



# K-Nearest Neighbors: Visualization



4 nearest neighbors = 4 closest stored data points



# K-Nearest Neighbors: Algorithm

## Training

- Store all training examples

## Testing

- Compute **distance** of test instance to all training data points
- Find the K closest training data points (*nearest neighbors*)
- Compute **target concept** of the test instance based on labels of the training instances



# K-Nearest Neighbors: Target Concepts

## KNN Classification

- Return the most common class label among neighbors
- Example: cat vs dog images; text classification; ...

## KNN Regression

- Return the average value of among  $K$  nearest neighbors
- Example: housing price prediction;



# Outline

## Four problems

1. How to represent each data point?
2. How to measure the distance between data points?
3. What if the neighbors disagree?
4. How to select  $K$ ?



## Feature Vectors

A data set of 6 instances (a...f) with 4 features and a label

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no



## Feature Vectors

A data set of 6 instances (a...f) with 4 features and a label

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no

We can represent each instance as a feature vector

$$\text{feature vector} = \begin{bmatrix} \text{Outlook} \\ \text{Temperature} \\ \text{Humidity} \\ \text{Windy} \end{bmatrix}$$



# Feature (or attribute) Types

Recall, from last lecture?

## 1. Nominal

- set of values with no intrinsic ordering
- possibly *boolean*

## 2. Ordinal

- explicitly ordered

## 3. Numerical

- real-valued, often no upper bound, easily mathematical manipulatable
- vector valued



# Outline

1. How to represent each data point?
2. **How to measure the distance between data points?**
3. What if the neighbors disagree?
4. How to select  $K$ ?



## Comparing Nominal Feature Vectors

First, we convert the nominal features into numeric features.

instance	features			
	color	shape	taste	size
apple	red	round	sweet	-
banana	yellow	curved	sweet	-
cherry	red	round	sweet	small

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1



## Comparing Nominal Features: Hamming Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of differing elements in two ‘strings’ of equal length.



## Comparing Nominal Features: Hamming Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of differing elements in two ‘strings’ of equal length.

$$d(\text{apple}, \text{banana}) = 4$$



## Comparing Nominal Features: Simple Matching Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of matching features divided by the number of all features in the sample

$$d = 1 - \frac{k}{m}$$

- $d$ : distance
- $k$ : number of matching features
- $m$ : total number of features



## Comparing Nominal Features: Simple Matching Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

The number of matching features divided by the number of all features in the sample

$$d = 1 - \frac{k}{m}$$

- $d$ : distance
- $k$ : number of matching features
- $m$ : total number of features

$$d(\text{apple}, \text{banana}) = 1 - \frac{2}{6} = \frac{4}{6}$$



## Comparing Nominal Feature Vectors: Jaccard Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

Jaccard *similarity*  $J$ : intersection of two **sets** divided by their union.

$$d = 1 - J$$

$$= 1 - \frac{|A \cap B|}{|A \cup B|}$$

$$= 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



## Comparing Nominal Feature Vectors: Jaccard Distance

instance	features					
	red	yellow	round	sweet	curved	small
apple	1	0	1	1	0	?
banana	0	1	0	1	1	?
cherry	1	0	1	1	0	1

Jaccard *similarity*  $J$ : intersection of two **sets** divided by their union.

$$d = 1 - J$$

$$= 1 - \frac{|A \cap B|}{|A \cup B|}$$

$$= 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$d(\text{apple}, \text{banana}) = 1 - \frac{1}{5} = \frac{4}{5}$$



# Comparing Numerical Feature Vectors: Manhattan Distance

## Manhattan Distance (or: L1 distance)

- Given two instances  $a$  and  $b$ , each with a set of numerical features, e.g.,  
 $a = [2.0, 1.4, 4.6, 5.5]$   
 $b = [1.0, 2.4, 6.6, 2.5]$
- Their distance  $d$  is the sum of absolute differences of each feature

$$d(a, b) = \sum_{i=1}^m |a_i - b_i| \quad (1)$$

## Example

$$\begin{aligned} d(a, b) &= |2.0 - 1.0| + |1.4 - 2.4| + |4.6 - 6.6| + |5.5 - 2.5| \\ &= 1 + 1 + 2 + 3 \\ &= 7 \end{aligned}$$



# Comparing Numerical Feature Vectors: Euclidean Distance

## Euclidean Distance (or: L2 distance)

- Given two instances  $a$  and  $b$ , each with a set of numerical features, e.g.,  
 $a = [2.0, 1.4, 4.6, 5.5]$   
 $b = [1.0, 2.4, 6.6, 2.5]$
- Their distance  $d$  is the distance in Euclidean space (2-dimensional space). Defined as the squared root of the sum of squared differences of each feature

$$d(a, b) = \sqrt{\sum_{i=1}^m (a_i - b_i)^2} \quad (2)$$

### Example

$$\begin{aligned} d(a, b) &= \sqrt{(2.0 - 1.0)^2 + (1.4 - 2.4)^2 + (4.6 - 6.6)^2 + (5.5 - 2.5)^2} \\ &= \sqrt{1 + 1 + 4 + 9} = \sqrt{15} \\ &= 3.87 \end{aligned}$$



# Comparing Numerical Feature Vectors: Cosine distance

## Cosine Distance

- Cosine similarity = cosine of angle between two vectors (= inner product of the *normalized* vectors)
- Cosine distance  $d$ : one minus cosine similarity

$$\cos(a, b) = \frac{a \cdot b}{|a||b|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}}$$

$$d(a, b) = 1 - \cos(a, b)$$

- Cosine distance is **normalized by the magnitude** of both feature vectors, i.e., we can compare instances of different magnitude
  - word counts: compare long vs short documents
  - pixels: compare high vs low resolution images



# Comparing Numerical Feature Vectors: Cosine distance

## Example

$$\cos(a, b) = \frac{a \cdot b}{|a||b|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}}$$
$$d(a, b) = 1 - \cos(a, b)$$

feature	doc1	doc2	doc3
word1	200	300	50
word2	300	200	40
word3	200	100	25

$$\cos(\text{doc1}, \text{doc2}) = \frac{200 \times 300 + 300 \times 200 + 200 \times 100}{\sqrt{200^2 + 300^2 + 200^2} \sqrt{300^2 + 200^2 + 100^2}} = 0.93$$

$$d(\text{doc1}, \text{doc2}) = 0.07$$

$$\cos(\text{doc2}, \text{doc3}) = \frac{300 \times 50 + 200 \times 40 + 100 \times 25}{\sqrt{300^2 + 200^2 + 100^2} \sqrt{50^2 + 40^2 + 25^2}} = 0.99$$

$$d(\text{doc2}, \text{doc3}) = 0.01$$



# Comparing Ordinal Feature Vectors

## Normalized Ranks

- sort values, and return a rank  $r \in \{0 \dots m\}$
- map ranks to evenly spaced values between 0 and 1

$$z = \frac{r}{m}$$

- compute a distance function for numeric features (e.g., Euclidean distance)

### Example: Customer ratings

1. Sorted ratings: { -2: 😢, -1: 😞, 0: 😐, 1: 😃, 2: 😊 } }
2. Ranks: { 0, 1, 2, 3, 4 } }

feature	A	B
safety	0	2
comfortable	-2	1
convenient	-1	2



# Comparing Ordinal Feature Vectors

## Normalized Ranks

- sort values, and return a rank  $r \in \{0 \dots m\}$
- map ranks to evenly spaced values between 0 and 1

$$z = \frac{r}{m}$$

- compute a distance function for numeric features (e.g., Euclidean distance)

### Example: Customer ratings

1. Sorted ratings: { -2: 😢, -1: 😞, 0: 😐, 1: 😃, 2: 😊 } }
2. Ranks: { 0, 1, 2, 3, 4 } }

feature	A	B
safety	0	2
comfortable	-2	1
convenient	-1	2



feature	A	B
safety	2/4	4/4
comfortable	0	3/4
convenient	1/4	4/4



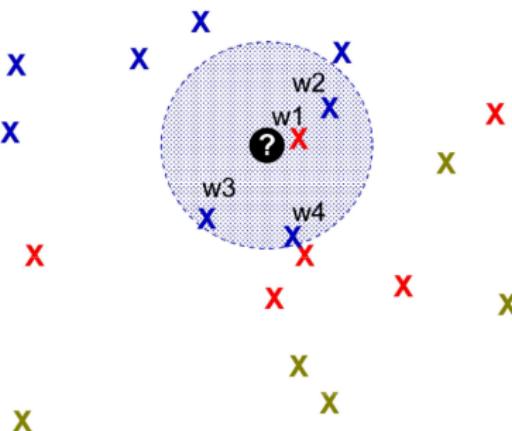
## Four problems

1. How to represent each data point?
2. How to measure the distance between data points?
3. **What if the neighbors disagree?**
4. How to select  $K$ ?



# Majority Voting

Equal weights (=majority vote)



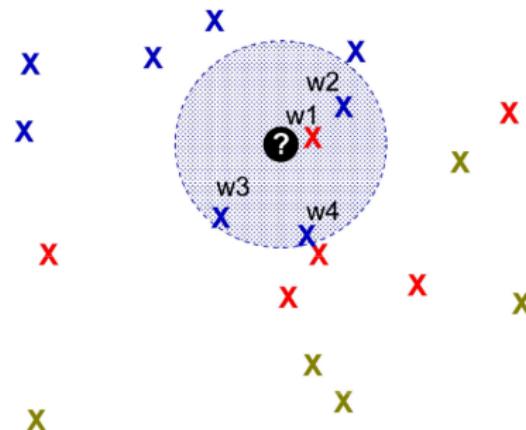
Voting Example (k=4)

- $w_1 = w_2 = w_3 = w_4 = 1$



# Majority Voting

Equal weights (=majority vote)



## Voting Example (k=4)

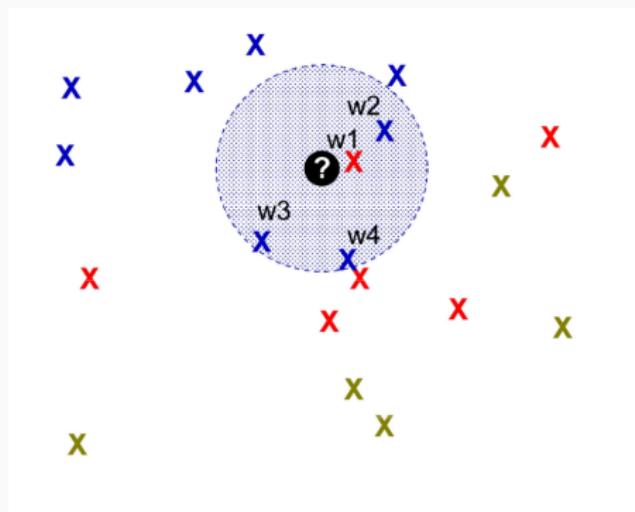
- $w_1 = w_2 = w_3 = w_4 = 1$
- red: 1
- blue:  $1+1+1=3$

# Weighted KNN: Inverse Distance

Inverse Distance

$$w_j = \frac{1}{d_j + \epsilon}$$

with  $\epsilon \approx 0$ , e.g.,  $1e - 10$



Voting Example (k=4)

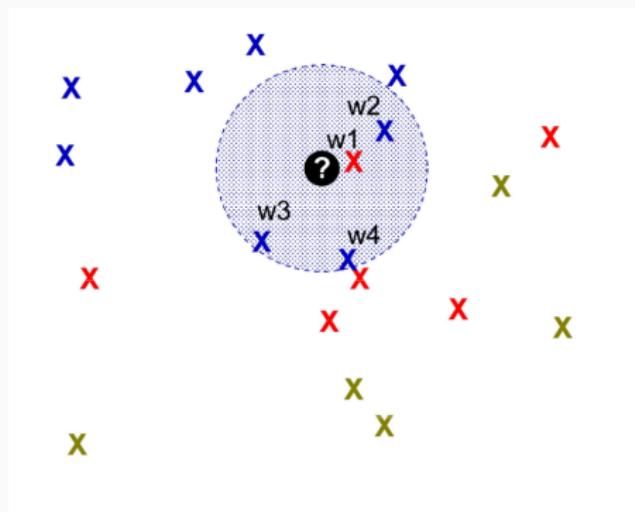
- $d_1=0$ ;  $d_2=1$ ;  $d_3=d_4=1.5$
- $\epsilon = 1e - 5$

# Weighted KNN: Inverse Distance

Inverse Distance

$$w_j = \frac{1}{d_j + \epsilon}$$

with  $\epsilon \approx 0$ , e.g.,  $1e-10$



Voting Example (k=4)

- $d_1=0$ ;  $d_2=1$ ;  $d_3=d_4=1.5$
- $\epsilon = 1e-5$

red:  $\frac{1}{0+\epsilon} = 100000$

blue:  $\frac{1}{1+\epsilon} + \frac{1}{1.5+\epsilon} + \frac{1}{1.5+\epsilon} = 1.0 + 0.67 + 0.67 = 2.34$

## Weighted K-NN: Inverse Linear Distance

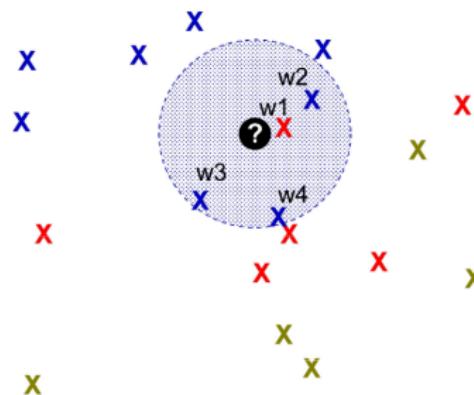
Inverse Linear distance

$$w_j = \frac{d_k - d_j}{d_k - d_1}$$

$d_1 = \min d$  among neighbors

$d_k = \max d$  among neighbors

$d_j$  = distance of  $j$ th neighbor



## Voting Example (k=4)

- $d_1=0$ ;  $d_2=1$ ;  $d_3=d_4=1.5$



# Weighted K-NN: Inverse Linear Distance

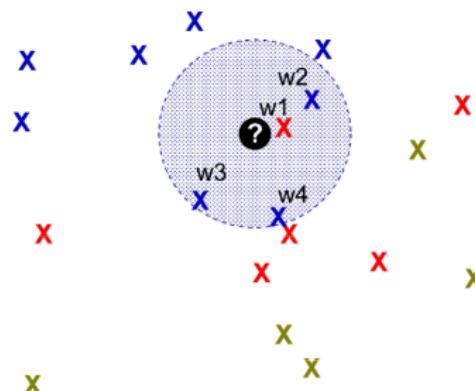
Inverse Linear distance

$$w_j = \frac{d_k - d_j}{d_k - d_1}$$

$d_1$  = min  $d$  among neighbors

$d_k$  = max  $d$  among neighbors

$d_j$  = distance of  $j$ th neighbor



Voting Example (k=4)

- $d_1=0$ ;  $d_2=1$ ;  $d_3=d_4=1.5$

red:  $\frac{1.5-0}{1.5-0} = 1$

blue:  $\frac{1.5-1}{1.5-0} + \frac{1.5-1.5}{1.5-0} + \frac{1.5-1.5}{1.5-0} = 0.3 + 0 + 0 = 0.3$

# Outline

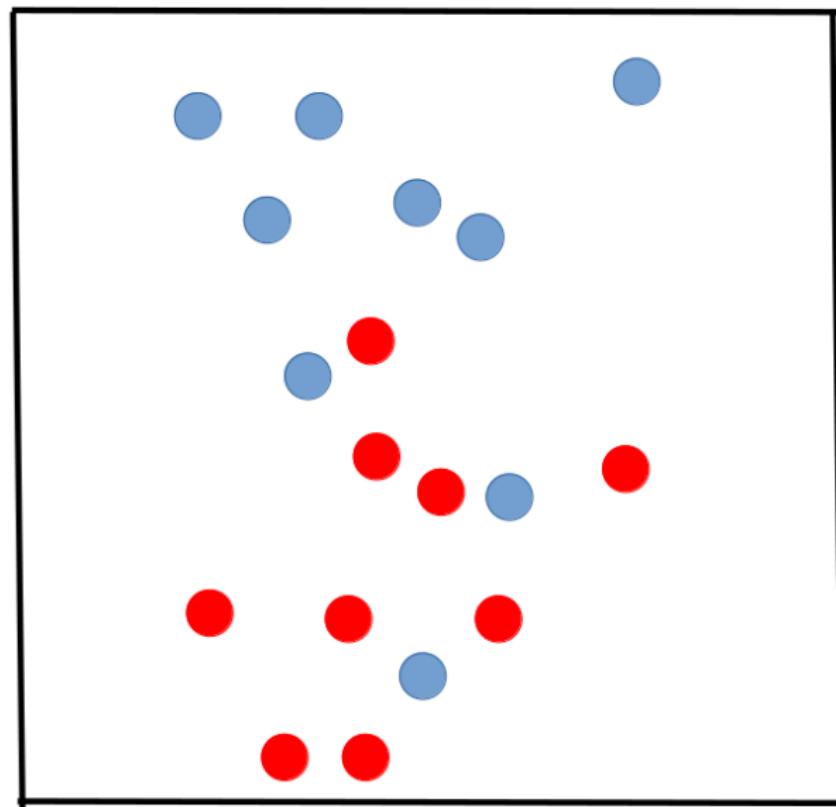
## Four problems

1. How to represent each data point?
2. How to measure the distance between data points?
3. What if the neighbors disagree?
4. **How to select  $K$ ?**



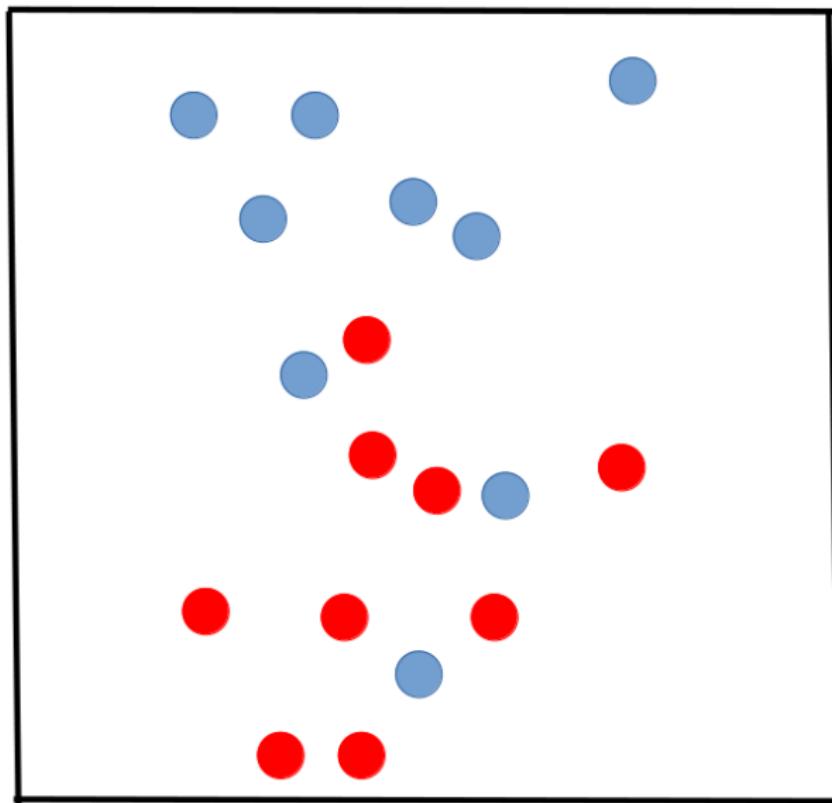
## Selecting the value of $K$

$K=1$



## Selecting the value of $K$

$K=3$



## Selecting the value of $K$

### Small $K$

- jagged decision boundary
- we capture noise
- lower classifier performance

Draw validation error:

### Large $K$

- smooth decision boundary
- danger of grouping together unrelated classes
- also: lower classifier performance!
- **what if  $K == N$ ? ( $N$ =number of training instances)**



## Breaking Ties

**What if more than K neighbors have the same (smallest) distance?**

- select at random
- change the distance metric

**What if two classes are equally likely given the current neighborhood?**

- avoid an even  $K$
- random tie breaking
- include  $K + 1$ th neighbor
- use class with highest prior probability



# Quiz!

[pollev.com/comp90049](http://pollev.com/comp90049)



# Why K-NN?

## Pros

- Intuitive and simple
- No assumptions
- Supports classification and regression
- **No training:** new data → evolve and adapt immediately

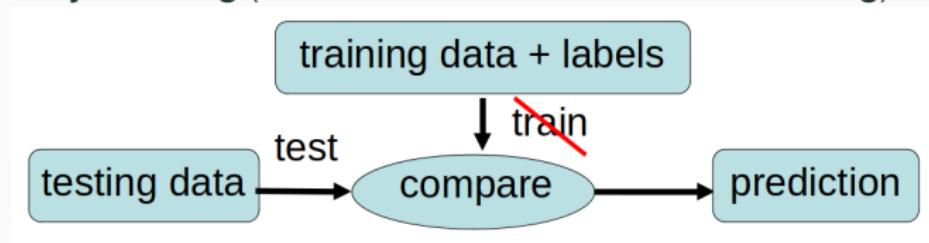
## Cons

- How to decide on best distance functions?
- How to combine multiple neighbors?
- How to select  $K$ ?
- Expensive with large (or growing) data sets



# Lazy vs Eager Learning

## Lazy Learning (also known as Instance-based Learning)

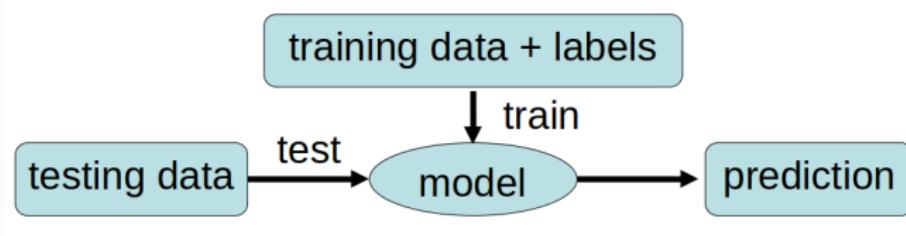


- **store** the training data
- **fixed** distance function
- **fixed** prediction rule (majority, weighting, ...)
- **compare** test instances with stored instances
- **no learning**



# Lazy vs Eager Learning

## Eager Learning



- **train a model** using labelled training instances
- the model will **generalize** from seen data to unseen data
- use the model to **predict** labels for test instances
- we will look at a variety of **eager** models and their learning algorithms over the next couple of weeks



## Today... Our first machine learning algorithm

- K-nearest neighbors
- Application to classification
- Application to regression

Also: the topic of your **first assignment!**

**Next: Probabilities (recap) and probabilistic modeling**



## Further Reading

- *Data Mining: Concepts and Techniques*, 2nd ed., Jiawei Han and Micheline Kamber, Morgan Kaufmann, 2006. Chapter 2, Chapter 9.5.
- *The elements of statistical learning*, 2nd ed., Trevor Hastie, Jerome Friedman and Robert Tibshirani. New York: Springer series in statistics, 2001. Chapter 2.3.2



# Lecture 4: Probability Theory and Probabilistic Modeling

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



## Last time... Concepts and KNN classification

- data, features, classes
- K Nearest Neighbors algorithm
- Application to classification

## Today... Probability

- basics / refresher
- distributions and parameterizations
- why probability in ML?

Estimating confidence in different possible outcomes



# Probability Theory

“The calculus of probability theory provides us with a **formal framework** for considering multiple possible **outcomes** and their **likelihood**. It defines a set of **mutually exclusive** and **exhaustive** possibilities, and associates each of them with a probability — **a number between 0 and 1**, so that the **total probability of all possibilities is 1**. This framework allows us to consider options that are **unlikely, yet not impossible**, without reducing our conclusions to content-free lists of every possibility.”

From Probabilistic Graphical Models: Principles and Techniques (2009; Koller and Friedman) <http://pgm.stanford.edu/intro.pdf>



# (Very) Basics of Probability Theory

**P(A): the probability of A** the fraction of times the event A is true in independent trials

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

$$P(\text{True}) = 1$$

$$P(\text{False}) = 0$$



# (Very) Basics of Probability Theory

**P(A): the probability of A** the fraction of times the event A is true in independent trials

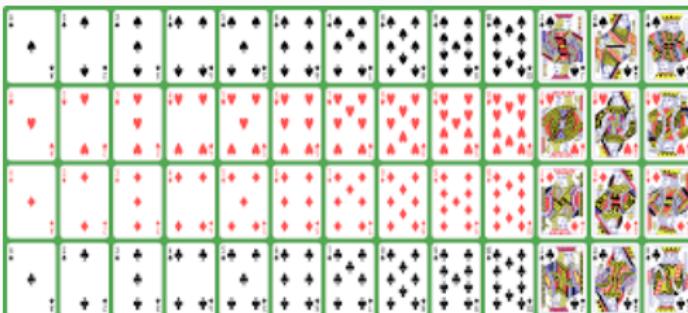
$$0 \leq P(A) \leq 1$$

$$P(\text{True}) = 1$$

$$P(\text{False}) = 0$$

**Given a deck of 52 cards**

- 13 ranks (ace, 2-10, jack, queen, king)
- of each of four suits (clubs, spades = black; hearts, diamonds = red)
- A is a random variable denoting the value of a randomly selected card.  
We denote the probability of A taking on a specific value  $a$  as  $P(A=a)$ .



# (Very) Basics of Probability Theory

**P(A): the probability of A** the fraction of times the event A is true in independent trials

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

$$P(\text{True}) = 1$$

$$P(\text{False}) = 0$$

## Given a deck of 52 cards

- 13 ranks (ace, 2-10, jack, queen, king)
- of each of four suits (clubs, spades = black; hearts, diamonds = red)
- A is a random variable denoting the value of a randomly selected card.  
We denote the probability of A taking on a specific value  $a$  as  $P(A=a)$ .

$$P(A = \text{queen}) = ?$$

$$P(A = \text{red}) = ?$$

$$P(A = \text{heart}) = ?$$



# (Very) Basics of Probability Theory

**P(A): the probability of A** the fraction of times the event A is true in independent trials

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

$$P(\text{True}) = 1$$

$$P(\text{False}) = 0$$

Given a deck of 52 cards

- 13 ranks (ace, 2-10, jack, queen, king)
- of each of four suits (clubs, spades = black; hearts, diamonds = red)
- A is a random variable denoting the value of a randomly selected card.  
We denote the probability of A taking on a specific value  $a$  as  $P(A=a)$ .

$$P(A = \text{queen}) = \frac{1}{13}$$

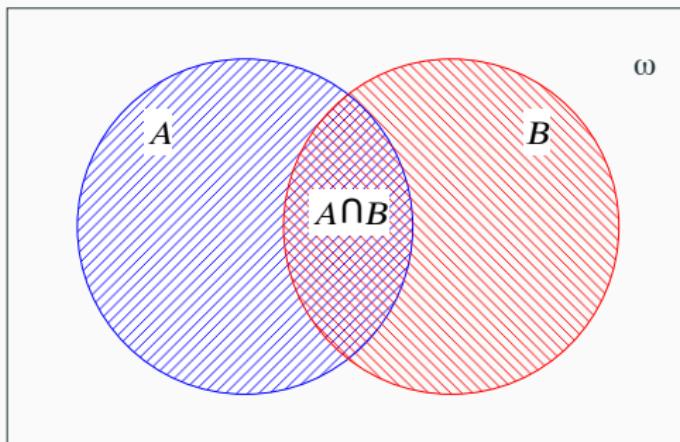
$$P(A = \text{red}) = \frac{1}{2}$$

$$P(A = \text{heart}) = \frac{1}{4}$$



# Basics of Probability Theory

**P(A, B): joint probability of the probability of both A and B occurring =  $P(A \cap B)$**

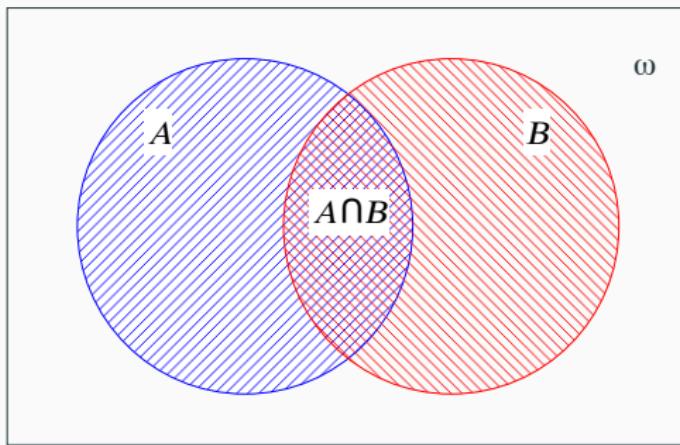


$$P(A = \text{ace}, B = \text{heart}) = ?$$

$$P(A = \text{heart}, B = \text{red}) = ?$$

# Basics of Probability Theory

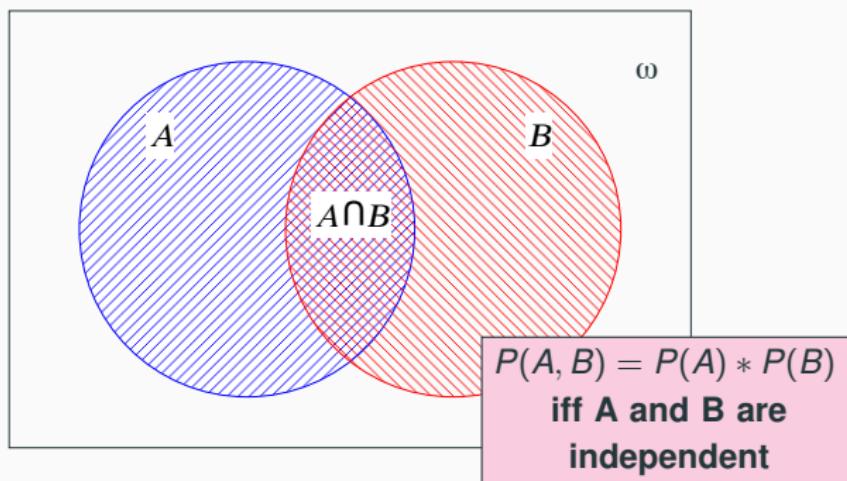
**P(A, B): joint probability of the probability of both A and B occurring =  $P(A \cap B)$**



$$P(A = \text{ace}, B = \text{heart}) = \frac{1}{52}$$
$$P(A = \text{heart}, B = \text{red}) = \frac{1}{4}$$

# Basics of Probability Theory

**P(A, B): joint probability of the probability of both A and two events A and B occurring =  $P(A \cap B)$**



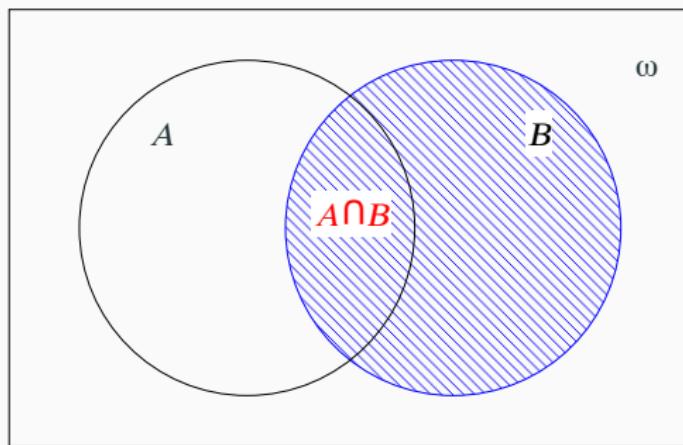
$$P(A = \text{ace}, B = \text{heart}) = \frac{1}{52}$$

$$P(A = \text{heart}, B = \text{red}) = \frac{1}{4}$$

# Conditional Probability

$P(A|B)$ : **conditional probability**

the probability of  $A=a$  given the observation  $B=b = \frac{P(A \cap B)}{P(B)}$



$$P(A = \text{ace} | B = \text{heart}) = ?$$

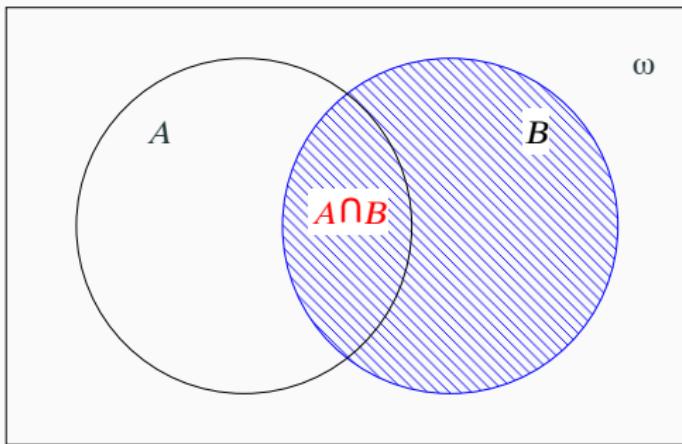
$$P(A = \text{heart} | B = \text{red}) = ?$$



# Conditional Probability

$P(A|B)$ : **conditional probability**

the probability of  $A=a$  given the observation  $B=b = \frac{P(A \cap B)}{P(B)}$



$$P(A = \text{ace} | B = \text{heart}) = \frac{1}{52} / \frac{1}{4} = \frac{1}{13}$$

$$P(A = \text{heart} | B = \text{red}) = \frac{1}{4} / \frac{1}{2} = \frac{1}{2}$$

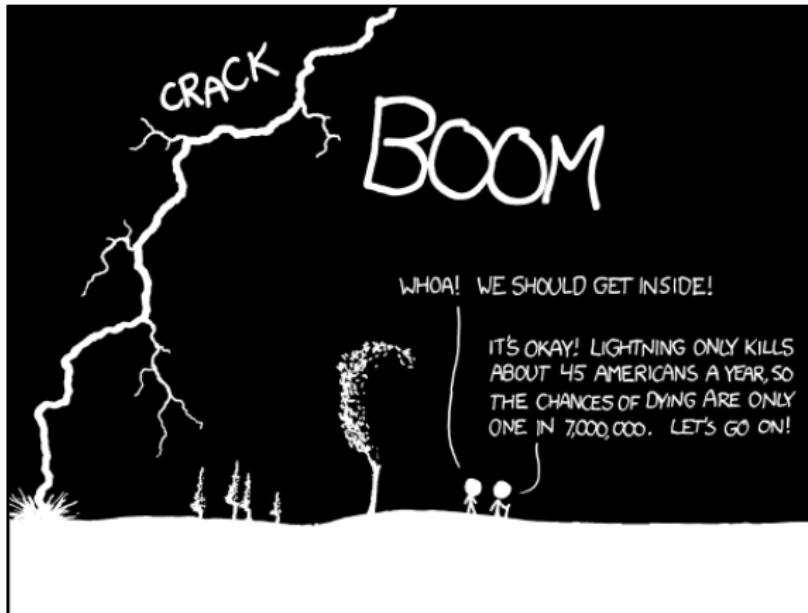


# Notation

1.  $P(A = x)$  probability that random variable A takes on value x
2.  $P(A)$  probability distribution over random variable A
3.  $P(x)$  I'll often use this as a short-hand for 1. if clear from the context



# What type of probability?



THE ANNUAL DEATH RATE AMONG PEOPLE  
WHO KNOW THAT STATISTIC IS ONE IN SIX.

[https://imgs.xkcd.com/comics/conditional\\_risk.png](https://imgs.xkcd.com/comics/conditional_risk.png)



THE UNIVERSITY OF  
MELBOURNE

# Rules of Probability I

- **Independence:**  $A$  and  $B$  are independent iff  $P(A \cap B) = P(A)P(B)$
- **Disjoint events:** The probability of two disjoint events, such that  $A \cap B = \emptyset$ , is  $P(A \text{ or } B) = P(A) + P(B)$
- **Product rule:**  $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$
- **Chain rule:**

$$P(A_1 \cap \dots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2 \cap A_1) \dots P(A_n | \cap_{i=1}^{n-1} A_i)$$



# Rules of Probability I

- **Independence:**  $A$  and  $B$  are independent iff  $P(A \cap B) = P(A)P(B)$
- **Disjoint events:** The probability of two disjoint events such that  $A \cap B = \emptyset$ , is  $P(A \text{ or } B) = P(A) + P(B)$   
e.g., draw an ace or a king:  $A = \text{draw an ace}$ ;  $B = \text{draw a king}$ .
- **Product rule:**  $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$
- **Chain rule:**

$$P(A_1 \cap \dots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2 \cap A_1) \dots P(A_n | \cap_{i=1}^{n-1} A_i)$$



# Rules of Probability I

- **Independence:**  $A$  and  $B$  are independent iff  $P(A \cap B) = P(A)P(B)$
- **Disjoint events:** The probability of two disjoint events such that  $A \cap B = \emptyset$ , is  $P(A \text{ or } B) = P(A) + P(B)$   
e.g., draw an ace or a king:  $A = \text{draw an ace}$ ;  $B = \text{draw a king}$ .
- **Product rule:**  $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$
- **Chain rule:**

$$P(A_1 \cap \dots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2 \cap A_1) \dots P(A_n|\cap_{i=1}^{n-1} A_i)$$

again, we can choose the factorization, e.g., :

$$P(\text{July}, 5^\circ C, \text{sick}) = P(\text{July}) \times P(5^\circ C|\text{July}) \times P(\text{sick}|5^\circ C, \text{July})$$

makes sense

$$= P(5^\circ C) \times P(\text{sick}|5^\circ C) \times P(\text{July}|5^\circ C, \text{sick})$$

???



# Rules of Probability II

## Bayes Rule

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

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

## Basic rule of probability

- Bayes' Rule allows us to compute  $P(A|B)$  given knowledge of the 'inverse' probability  $P(B|A)$ .

## More philosophically,

- Bayes' Rule allows us to update prior belief with empirical evidence



# Rules of Probability II

## Bayes Rule

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

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

### Posterior Probability $P(A|B)$

- the degree of belief having accounted for  $B$ .

### Prior Probability $P(A)$

- the initial degree of belief in  $A$ .
- the probability of  $A$  occurring, given no additional knowledge about  $A$

### Likelihood $P(B|A)$

- the support  $B$  provides for  $A$

**Normalizing constant ('Evidence')**  $P(B) = \sum_A P(B|A)P(A)$



## Rules of Probability II

### Bayes Rule

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \quad (\text{cf., } P(A|B) = \frac{P(A \cap B)}{P(B)})$$

### Example

Estimate the probability of a student **being smart** given that (s)he **achieved H1 score**,  $P(\text{smart}|H1)$  from the following information:

$$P(\text{Smart}) = 0.3 \quad \text{prior rate of smart students}$$

$$P(H1|\text{Smart}) = 0.6 \quad \text{empirically measured } H1|\text{smart}$$

$$P(H1) = 0.2 \quad \text{emprirically measured}$$

(What if  $P(H1) = 0.4?$ )



# Binomial Distributions

- A **binomial distribution** results from a series of independent trials with only two outcomes (aka **Bernoulli trials**)  
*e.g. multiple coin tosses ( $H, T, H, H, \dots, T$ )*



# Binomial Distributions

- A **binomial distribution** results from a series of independent trials with only two outcomes (aka **Bernoulli trials**)  
*e.g. multiple coin tosses ( $\langle H, T, H, H, \dots, T \rangle$ )*
- The probability  $P$  of an event with probability  $p$  occurring exactly  $m$  out of  $n$  times is given by

$$P(m, n, p) = \binom{n}{m} p^m (1 - p)^{n-m}$$

$$P(m, n, p) = \underbrace{\frac{n!}{m!(n-m)!}}_{\text{possible distributions of } m \text{ successes over } n \text{ trials}} \underbrace{p^m}_{m \text{ successes}} \underbrace{(1-p)^{n-m}}_{n-m \text{ failures}}$$



# Binomial Distributions

- A **binomial distribution** results from a series of independent trials with only two outcomes (aka **Bernoulli trials**)  
*e.g. multiple coin tosses ( $\langle H, T, H, H, \dots, T \rangle$ )*
- The probability  $P$  of an event with probability  $p$  occurring exactly  $m$  out of  $n$  times is given by

$$P(m, n, p) = \binom{n}{m} p^m (1 - p)^{n-m}$$

$$P(m, n, p) = \underbrace{\frac{n!}{m!(n-m)!}}_{\text{possible distributions of } m \text{ successes over } n \text{ trials}} \underbrace{p^m}_{m \text{ successes}} \underbrace{(1-p)^{n-m}}_{n-m \text{ failures}}$$

What is the probability of getting 2 heads out of 3 tosses of a fair coin?



## Binomial Example: Coin Toss

Go through solution:



## Binomial Example: Coin Toss

What is the probability of getting times 2 heads out of 3 tosses of a fair coin?



## Binomial Example: Coin Toss

What is the probability of getting times 2 heads out of 3 tosses of a fair coin?

1.  $m = 2$  successes (heads) when flipping coin  $n = 3$  times;  $P(X = 2)$



## Binomial Example: Coin Toss

What is the probability of getting times 2 heads out of 3 tosses of a fair coin?

1.  $m = 2$  successes (heads) when flipping coin  $n = 3$  times;  $P(X = 2)$
2. number of possible outcomes  $e$  from 3 coin flips:

$$2 * 2 * 2 = 2^3 = 8 \quad \text{each with } P(e) = \frac{1}{8}$$



## Binomial Example: Coin Toss

What is the probability of getting times 2 heads out of 3 tosses of a fair coin?

1.  $m = 2$  successes (heads) when flipping coin  $n = 3$  times;  $P(X = 2)$
2. number of possible outcomes  $e$  from 3 coin flips:

$$2 * 2 * 2 = 2^3 = 8 \quad \text{each with } P(e) = \frac{1}{8}$$

3. Choose 2 out of 3:  $C(3, 2) = \frac{3!}{2!1!} = 3$



## Binomial Example: Coin Toss

What is the probability of getting times 2 heads out of 3 tosses of a fair coin?

1.  $m = 2$  successes (heads) when flipping coin  $n = 3$  times;  $P(X = 2)$
2. number of possible outcomes  $e$  from 3 coin flips:

$$2 * 2 * 2 = 2^3 = 8 \quad \text{each with } P(e) = \frac{1}{8}$$

3. Choose 2 out of 3:  $C(3, 2) = \frac{3!}{2!1!} = 3$

4. 3 possible outcomes,  $\frac{1}{8}$  for each:  $P(X = 2) = \frac{3}{8}$



## Binomial Example: Coin Toss

What is the probability of getting times 2 heads out of 3 tosses of a fair coin?

1.  $m = 2$  successes (heads) when flipping coin  $n = 3$  times;  $P(X = 2)$
2. number of possible outcomes  $e$  from 3 coin flips:

$$2 * 2 * 2 = 2^3 = 8 \quad \text{each with } P(e) = \frac{1}{8}$$

3. Choose 2

$$P(m, n, p) = \frac{n!}{m!(n-m)!} p^m (1-p)^{n-m}$$

4. 3 possible outcomes,  $\frac{1}{8}$  for each:  $P(X = 2) = \frac{3}{8}$

$$P\left(2, 3, \frac{1}{2}\right) = \frac{3!}{2!(3-2)!} \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^{3-2} = 3 \left(\frac{1}{4}\right) \left(\frac{1}{2}\right)$$



## Multinomial Distributions

- A **multinomial distribution** models the probability of **counts** of different events from a series of independent trials with **more than two possible outcomes**, e.g.,
  - a fair 6-sided dice is rolled 5 times
  - what is the probability of observing exactly 3 'ones' and 2 'fives'?
  - what is the probability of observing 5 'threes'?



## Multinomial Distributions

- A **multinomial distribution** models the probability of **counts** of different events from a series of independent trials with **more than two possible outcomes**, e.g.,
  - a fair 6-sided dice is rolled 5 times
  - what is the probability of observing exactly 3 'ones' and 2 'fives'?
  - what is the probability of observing 5 'threes'?
- The probability of events  $X_1, X_2, \dots, X_n$  with probabilities  $\mathbf{p} = p_1, p_2, \dots, p_n$  occurring exactly  $x_1, x_2, \dots, x_n$  times, respectively, is given by

$$\begin{aligned}P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n; \mathbf{p}) &= \frac{(\sum_i x_i)!}{x_1! \dots x_n!} p_1^{x_1} \times p_2^{x_2} \times \dots \times p_n^{x_n} \\&= \frac{(\sum_i x_i)!}{x_1! \dots x_n!} \prod_i p_i^{x_i}\end{aligned}$$



## Categorical Distributions

- The **categorical distribution** models the probability of **events** resulting from a single trial with **more than two possible outcomes**, e.g.,
  - we roll a fair-sided dice once
  - what is the probability of observing a 'five'?
- The probability of events  $X_1, X_2, \dots, X_n$  with probabilities  $\mathbf{p} = p_1, p_2, \dots, p_n$  occurring exactly  $x_1, x_2, \dots, x_n$  times, respectively, is given by

$$\begin{aligned}P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n; \mathbf{p}) &= p_1^{x_1} \times p_2^{x_2} \times \cdots \times p_n^{x_n} \\&= \prod_i p_i^{x_i}\end{aligned}$$



# Marginalization

## Intuition

We want to know the probability of an event  $A$  *irrespective of* the outcome of another event  $B$ . We can obtain it, by summing over all possible outcomes  $\mathcal{B}$  of  $B$ .

- Take an event  $B$ . The set of *all* possible *individual* outcomes of  $B$ ,  $\mathcal{B}$  is the **partition** of the outcome space
- E.g.,  $\mathcal{B} = \{\text{head, tail}\}$  for a coin flip;  $\mathcal{B} = \{\text{king, heart, diamond, spades}\}$  for card suits
- We can **marginalize** over the set of outcomes of  $B$  as follows

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

or equivalently (remember the product rule?)

$$P(A) = \sum_{b \in \mathcal{B}} P(A|B = b)P(B = b)$$

and even for conditional probabilities

$$P(A|C) = \sum_{b \in \mathcal{B}} P(A|C, B = b)P(B = b|C)$$



# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)
romance	EU	0.05
romance	NA	0.1
romance	AUS	0.3
thriller	EU	0.1
thriller	NA	0.2
thriller	AUS	0.1
comedy	EU	0.1
comedy	NA	0.025
comedy	AUS	0.025

# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)	
romance	EU	0.05	
romance	NA	0.1	
romance	AUS	0.3	$\Sigma$
thriller	EU	0.1	thriller
thriller	NA	0.2	
thriller	AUS	0.1	
comedy	EU	0.1	comedy
comedy	NA	0.025	
comedy	AUS	0.025	

# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)		A	P(A)
romance	EU	0.05		romance	0.45
romance	NA	0.1	$\Sigma$	thriller	
romance	AUS	0.3		comedy	
thriller	EU	0.1			
thriller	NA	0.2			
thriller	AUS	0.1			
comedy	EU	0.1			
comedy	NA	0.025			
comedy	AUS	0.025			

# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)		A	P(A)
romance	EU	0.05		romance	0.45
romance	NA	0.1	$\Sigma$	thriller	
romance	AUS	0.3		comedy	
thriller	EU	0.1			
thriller	NA	0.2			
thriller	AUS	0.1	$\Sigma$		
comedy	EU	0.1			
comedy	NA	0.025			
comedy	AUS	0.025			

# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)	A	P(A)
romance	EU	0.05	romance	0.45
romance	NA	0.1		
romance	AUS	0.3		
thriller	EU	0.1	thriller	0.4
thriller	NA	0.2		
thriller	AUS	0.1		
comedy	EU	0.1	comedy	
comedy	NA	0.025		
comedy	AUS	0.025		

1.0



# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)	A	P(A)
romance	EU	0.05	romance	0.45
romance	NA	0.1		
romance	AUS	0.3		
thriller	EU	0.1	thriller	0.4
thriller	NA	0.2		
thriller	AUS	0.1		
comedy	EU	0.1	comedy	
comedy	NA	0.025		
comedy	AUS	0.025		

# Marginalization

## Example

We want to know the probability of success of movies of a specific genre ( $\mathcal{A} = \{\text{comedy}, \text{thriller}, \text{romance}\}$ ). But we only have data on movie success probabilities in a specific market, namely ( $\mathcal{B} = \{\text{EU}, \text{NA}, \text{AUS}\}$ ).

$$P(A) = \sum_{b \in \mathcal{B}} P(A, B = b)$$

A	B	P(A, B)	A	P(A)
romance	EU	0.05	romance	0.45
romance	NA	0.1		
romance	AUS	0.3		
thriller	EU	0.1	thriller	0.4
thriller	NA	0.2		
thriller	AUS	0.1		
comedy	EU	0.1	comedy	0.15
comedy	NA	0.025		
comedy	AUS	0.025		



# Quiz!

Please go to

<https://pollev.com/comp90049>

for a quick quiz on probabilities!



# Probability and Machine Learning

We probably all agree that probabilities are useful for thinking about card games or coin flips

... but why should we care in machine learning?

Consider typical classification problems

- document → {spam, no spam}
- hand-written digit → {0,1,2,3,4,5,6,7,8,9}
- purchase history → recommend {book a, book b, book c, ...}



# Probability and Machine Learning

We probably all agree that probabilities are useful for thinking about card games or coin flips

... but why should we care in machine learning?

Consider typical classification problems

- document → {spam, no spam}
- hand-written digit → {0,1,2,3,4,5,6,7,8,9}
- purchase history → recommend {book a, book b, book c, ...}
- **uncertainty**, due to few observations, noisy data, ...
- model features as following certain **probability distributions**
- **soft predictions** (“we are 60% confident that Bob will like *Harry Potter* given his purchase history”)
- ...



“All models are wrong, but some are useful.”

(George Box, Statistician)

## Probabilistic Models

- allow to reason about random events in a **principled** way.
- allow to formalize hypotheses as different types of probability distributions, and use the laws of probability to derive predictions

### Example: Spam classification

- An email is a random event with two possible outcomes: `spam`, `not spam`
- The probability of observing a spam email  $P(\text{spam}) = \theta$ , and trivially  $P(\text{not spam}) = 1 - \theta$ .
- We might care about a random variable  $X$  as the number of spam emails in an inbox of 100 emails.  $X$  is distributed according to the **binomial distribution**, and depends on the **parameters**  $\theta$  and  $N = 100$

$$X \sim \text{Binomial}(\theta, N = 100)$$



# Learning Probabilistic Models I

$X$  is distributed according to the **binomial distribution**, and depends on the **parameters**  $\theta$  and  $N = 100$

$$X \sim \text{Binomial}(\theta, N = 100)$$

- In order to make predictions of  $X$  we need to know the parameters  $\theta$ .  
**How do we learn them?**
- Typically,  $\theta$  is unknown, but if we have **data** available we can **estimate**  $\theta$
- One common choice is to pick  $\theta$  that maximizes the probability of the observed data

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(X; \theta, N)$$

That is the **maximum likelihood estimate (MLE)** of  $\theta$ .

- Once we have estimated  $\theta$  we can use it to **predict** values for unseen data



## The maximum likelihood principle

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(X; \theta, N) \quad (1)$$

- Consider a **data set** consisting of 100 emails, 20 of which are spam.
- Following from the binomial distribution

$$\mathcal{L}(\theta) = P(X; \theta, N) = \binom{n}{m} \theta^x (1 - \theta)^{N-x}$$

the **likelihood of the data**<sup>1</sup> is  $\propto \theta^{20} (1 - \theta)^{100-20}$

- What do you think would be a good value for  $\theta = p(\text{spam} = 1)$ ? Why?
- Next lecture, we will see how to derive this value in a principled way

---

<sup>1</sup> $\propto$  means 'proportional to'.  $\binom{n}{m}$  can be ignored because it is independent of  $\theta$ .



## Maximum likelihood is only one choice of estimator among many

- Consider a data set of one inbox with no spam email. MLE:  $\theta = 1$ , and hence  $P(\text{not spam}) = \theta = 1$  and  $P(\text{spam}) = 1 - \theta = 0$ .  
→ “spam emails don’t exist”
- We could modify this estimate with our **prior belief**. E.g., we might believe that about 80 of 100 emails are not spam. We ‘nudge’  $\theta$  from  $\theta = 1$  towards  $\theta = 0.80$
- We can combine our prior belief with the estimate from the data to arrive at a **posterior probability distribution** over  $\theta$ :  $P(\theta)$ .

$$P(\theta|x) = \frac{P(\theta)P(x|\theta)}{P(x)} \propto P(\theta)P(x|\theta) \quad (\text{looks familiar})?$$

- The **maximum a posteriori estimate** is then

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(\theta)P(x|\theta)$$



## Probability underlies many modern knowledge technologies

- estimate the (conditional, joint) probability of observations
- Bayes rule
- Expectations and marginalization
- Probabilistic models
- Maximum likelihood estimation (taster)
- Maximum a posteriori estimation (taster)

### Next Lecture(s):

- Optimization
- Naive Bayes Classification



## References

Chris Bishop. Pattern Recognition and Machine Learning. Chapters: 1.2 (intro), 1.2.3, 2 (intro), 2.1 (up to 2.1.1), 2.2 (up to 2.2.1)



## Optional / If time allows: Expectations

The **expectation** of a function (like a probability distribution) is the **weighted average** of all possible outcomes, weighted by their respective probability.

- For functions with discrete outputs

$$E[f(x)] = \sum_{x \in \mathcal{X}} f(x)P(x)$$

- For functions with continuous outputs

$$E[f(x)] = \int_{\mathcal{X}} f(x)P(x)dx$$



## Optional / If time allows: Expectations

The **expectation** of a function (like a probability distribution) is the **weighted average** of all possible outcomes, weighted by their respective probability.

- On sunny days Bob watches 1 movie
- On rainy days Bob watches 3 movies
- Bob lives in Melbourne, it rains on 70% of all days
- What is the expected number of movies Bob watches per day?



## Optional / If time allows: Expectations

The **expectation** of a function (like a probability distribution) is the **weighted average** of all possible outcomes, weighted by their respective probability.

- On sunny days Bob watches 1 movie
- On rainy days Bob watches 3 movies
- Bob lives in Melbourne, it rains on 70% of all days
- What is the expected number of movies Bob watches per day?

$$1 * 0.3 + 3 * 0.7 = 2.4$$



# Lecture 5: Introduction to Optimization

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



## Last time... Probability

- estimate the (conditional, joint) probability of observations
- Bayes rule
- Marginalization
- Probabilistic models
- Maximum likelihood estimation (taster)
- Maximum a posteriori estimation (taster)



## Last time... Probability

- estimate the (conditional, joint) probability of observations
- Bayes rule
- Marginalization
- Probabilistic models
- Maximum likelihood estimation (taster)
- Maximum a posteriori estimation (taster)

## Today... Optimization

- Curves, minima
- Gradients, derivatives
- Recipe for numerical optimization
- Maximum likelihood of the Binomial (from scratch!)



## Optimization

---

# Introduction I

We are all here to **learn** about Machine **Learning**.

- What is learning?



# Introduction I

We are all here to **learn** about Machine **Learning**.

- What is learning?
- It probably has something to do with **change** or **mastering** or **optimizing** performance on a specific task



THE UNIVERSITY OF  
MELBOURNE

# Introduction I

We are all here to **learn** about Machine **Learning**.

- What is learning?
- It probably has something to do with **change** or **mastering** or **optimizing** performance on a specific task
- Machine learning typically involves to build models (like seen last time), and learning boils down to **finding model parameters that optimize some measure of performance**



# Introduction I

We are all here to **learn** about Machine **Learning**.

- What is learning?
- It probably has something to do with **change** or **mastering** or **optimizing** performance on a specific task
- Machine learning typically involves to build models (like seen last time), and learning boils down to **finding model parameters that optimize some measure of performance**

**But, how do we know what is optimal?**



# Finding Optimal Points I

Finding the **parameters** that optimize a **target**

Ex1: Estimate the **study time** which leads to the **best grade** in COMP90049.

Ex2: Find the **shoe price** which leads to **maximum profit** of our shoe shop.



# Finding Optimal Points I

Finding the **parameters** that optimize a **target**

Ex1: Estimate the **study time** which leads to the **best grade** in COMP90049.

Ex2: Find the **shoe price** which leads to **maximum profit** of our shoe shop.



## Finding Optimal Points I

Finding the **parameters** that optimize a **target**

Ex1: Estimate the **study time** which leads to the **best grade** in COMP90049.

Ex2: Find the **shoe price** which leads to **maximum profit** of our shoe shop.

Ex3: Predicting **housing prices** from a **weighted** combination of house age and house location

Ex4: Find the **parameters  $\theta$**  of a spam classifier which lead to the **lowest error**

Ex5: Find the **parameters  $\theta$**  of a spam classifier which lead to the **highest data log likelihood**

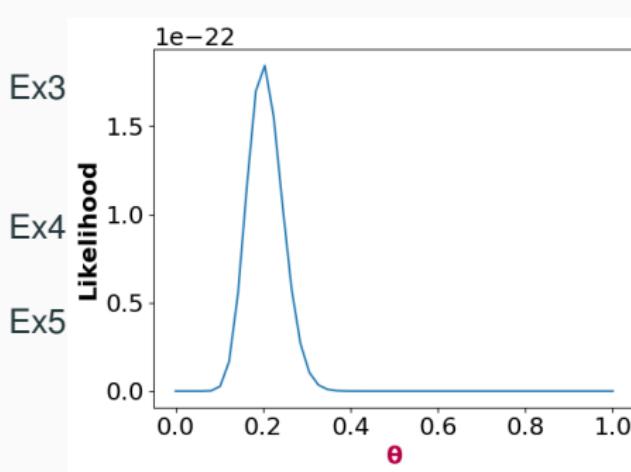


# Finding Optimal Points I

Finding the **parameters** that optimize a **target**

Ex1: Estimate the **study time** which leads to the **best grade** in COMP90049.

Ex2: Find the **shoe price** which leads to **maximum profit** of our shoe shop.



eighted combination of house age

ssifier which lead to the **lowest error**

ssifier which lead to the **highest**



## Objective functions

Find parameter values  $\theta$  that maximize (or minimize) the value of a function  $f(\theta)$

- we want to find the **extreme** points of the **objective function**.

Depending on our **target**, this could be

- ...the **maximum**

E.g., the **maximum** profit of our shoe shop

E.g., the **largest** possible (log) likelihood of the data

$$\hat{\theta} = \operatorname{argmax}_{\theta} f(\theta)$$

- ...or the **minimum** (in which case we often call  $f$  a **loss function**)

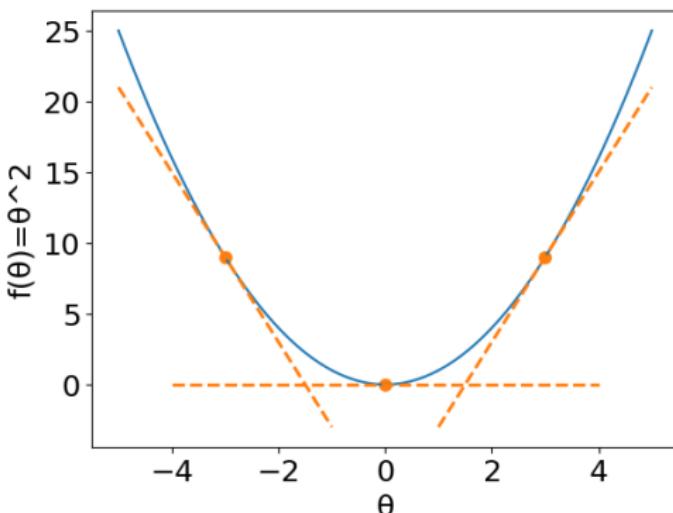
E.g., the **smallest** possible classification error

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



## Finding extreme points of a function

- At its **extreme point**,  $f(\theta)$  is ‘flat’: its **slope** is equal to **zero**.
- We can measure the **slope** of a function at any point through its first **derivative** at that point
- The derivative measures the change of the output  $f(\theta)$  given a change in the input  $\theta$
- We write the derivative of  $f$  with respect to  $\theta$  as  $\frac{\partial f}{\partial \theta}$

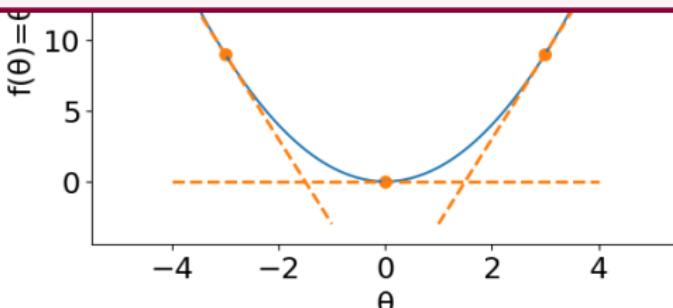


## Finding extreme points of a function

- At its **extreme point**,  $f(\theta)$  is ‘flat’: its **slope** is equal to **zero**.
- We can measure the **slope** of a function at any point through its first **derivative** at that point
- The derivative measures the change of the output  $f(\theta)$  given a change in the input  $\theta$
- We write the derivative of  $f$  with respect to  $\theta$  as  $\frac{\partial f}{\partial \theta}$

In order to find the parameters that maximize / minimize an objective function, we find those inputs at which the derivative of the function evaluates to zero.

That's it!



# Finding a Minimum / Maximum

## Example

- For our function, with a single 1-dimensional parameter  $\theta$

$$f(\theta) = \theta^2$$

Take the derivative

$$\frac{\partial f}{\partial \theta} = 2\theta$$

We want to find the point where this derivative is zero, so

$$2\theta = 0$$

and solve for  $\theta$

$$\theta = 0$$



# Finding a Minimum / Maximum

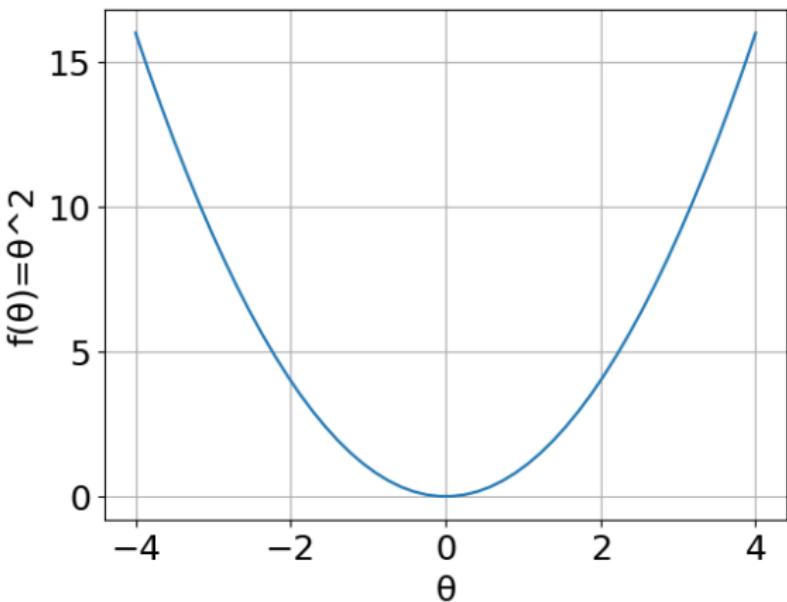
## Example

- For our function, we

Take the derivative

We want to find the

and solve for  $\theta$



The global minimum of  $f(\theta) = \theta^2$  occurs at the point where  $\theta=0$ .



## Recipe for finding Minima / Maxima

1. Define your function of interest  $f(\theta)$  (e.g., data log likelihood)
2. Compute its first derivative with respect to its input  $\theta$
3. Set the derivative equal to zero
4. Solve for  $\theta$



## Recipe for finding Minima / Maxima

1. Define your function of interest  $f(\theta)$  (e.g., data log likelihood)
2. Compute its first derivative with respect to its input  $\theta$
3. Set the derivative equal to zero
4. Solve for  $\theta$

Let's do this for a more interesting problem. Recall our binomial spam model from the last lecture?



# Maximum Likelihood Optimization of the Binomial Spam Model

## 1. Problem setup / identifying the function of interest

- Consider a data set of emails, where each email is an observation  $x$  which is labeled either as spam or not spam
- We have  $N$  observations, each with 2 possible outcomes. The data consequently follows a **binomial distribution** and the data likelihood is

$$\mathcal{L}(\theta) = p(X; N, \theta) = \frac{N!}{x!(N-x)!} \theta^x (1-\theta)^{N-x}$$

- So the parameter  $\theta = P(\text{spam})$



# Maximum Likelihood Optimization of the Binomial Spam Model

## 1. Problem setup / identifying the function of interest

- Consider a data set of emails, where each email is an observation  $x$  which is labeled either as spam or not spam
- We have  $N$  observations, each with 2 possible outcomes. The data consequently follows a **binomial distribution** and the data likelihood is

$$\mathcal{L}(\theta) = p(X; N, \theta) = \frac{N!}{x!(N-x)!} \theta^x (1-\theta)^{N-x}$$

- So the parameter  $\theta = P(\text{spam})$
- Imagine we have a data set of 100 emails: 20 are spam (and consequently 80 emails are not spam).
- In the last lecture, we agreed intuitively that  $P(\text{spam}) = \theta = 20/100 = \frac{x}{N}$ .
- We will now derive the same result mathematically, and show that  $\theta = \frac{x}{N}$  is the  $\hat{\theta}$  that maximizes the likelihood of the observed data



# Maximum Likelihood Optimization of the Binomial Spam Model

$$\mathcal{L}(\theta) = p(X; N, \theta) = \frac{N!}{x!(N-x)!} \theta^x (1-\theta)^{N-x} \approx \theta^x (1-\theta)^{N-x}$$



# Maximum Likelihood Optimization of the Binomial Spam Model



# Maximum Likelihood Optimization of the Binomial Spam Model



# Maximum Likelihood Optimization of the Binomial Spam Model

## 2. Computing its first derivative

$$\begin{aligned}\mathcal{L}(\theta) &= p(X; N, \theta) = \frac{N!}{x!(N-x)!} \theta^x (1-\theta)^{N-x} \\ &\approx \theta^x (1-\theta)^{N-x}\end{aligned}$$

Move to log space (makes our life easier)

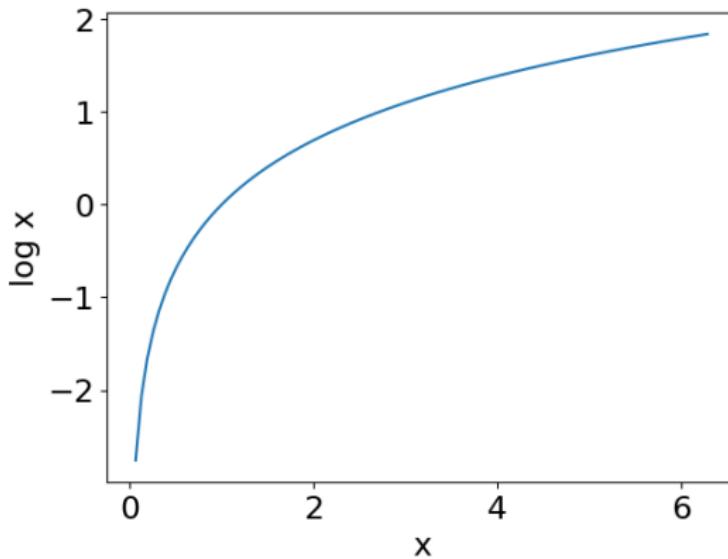
$$\log \mathcal{L}(\theta) = x \log \theta + (N - x) \log(1 - \theta)$$



# Maximum Likelihood Optimization of the Binomial Spam Model

(Log transformation aside)

- Log is a monotonic transformation: The same  $\theta$  will maximize both  $p(x, y)$  and  $\log p(x, y)$
- Log values are less extreme (cf. x scale vs y scale)
- Products become sums (avoid under/overflow)



# Maximum Likelihood Optimization of the Binomial Spam Model

## 2. Computing its first derivative

$$\mathcal{L}(\theta) = p(X; N, \theta) = \frac{N!}{x!(N-x)!} \theta^x (1-\theta)^{N-x} \approx \theta^x (1-\theta)^{N-x}$$

Move to log space (makes or life easier)

$$\log \mathcal{L}(\theta) = x \log \theta + (N - x) \log(1 - \theta)$$



# Maximum Likelihood Optimization of the Binomial Spam Model

## 2. Computing its first derivative

$$\mathcal{L}(\theta) = p(X; N, \theta) = \frac{N!}{x!(N-x)!} \theta^x (1-\theta)^{N-x} \approx \theta^x (1-\theta)^{N-x}$$

Move to log space (makes our life easier)

$$\log \mathcal{L}(\theta) = x \log \theta + (N-x) \log(1-\theta)$$

Take the derivative of  $\mathcal{L}$  wrt the parameters  $\theta$

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{x}{\theta} - \frac{N-x}{1-\theta}$$



# Maximum Likelihood Optimization of the Binomial Spam Model

## 3. Set the derivative to zero

$$0 = \frac{x}{\theta} - \frac{N-x}{1-\theta}$$

## 4. Solve for $\theta$

$$\frac{x}{\theta} = \frac{N-x}{1-\theta} \quad [\times(1-\theta)]$$

$$\frac{x \times (1-\theta)}{\theta} = N-x \quad [\times \frac{1}{x}]$$

$$\frac{1-\theta}{\theta} = \frac{N-x}{x} \quad [\text{rearrange}]$$

$$\frac{1}{\theta} - 1 = \frac{N}{x} - 1 \quad [+1]$$

$$\frac{1}{\theta} = \frac{N}{x} \quad [\text{flip}]$$

$$\hat{\theta} = \frac{x}{N}$$

Which corresponds to our estimate of  $\frac{x}{N} = \frac{20}{100} = 0.2$  for our spam classification problem.



## Possible Complications

Can you think of scenarios where this approach breaks down?



## Possible Complications

**Can you think of scenarios where this approach breaks down?**

- Our loss function is not differentiable
- It is mathematically impossible to set the derivative to 0 and solve for the parameters  $\theta$ . “No closed-form solution”.
- Our function has multiple ‘extreme points’ where the slope equals zero. Which one is the correct one?

**to be continued...**



# Summary

- What is optimization?
- Objective function / loss function
- Gradients, derivatives, and slopes

**Next: Naive Bayes**



## **Solution subject to Constraints**

---

## Constrained Optimization

Finding the **parameters** that optimize a **target** subject to one or more constraints.

- Buy **3 pieces of fruit** which lead to the best **nutritional value**. But we only have a budget of **3\$**.
- I want to estimate the **parameters of a Categorical distribution** to maximize the **data log likelihood** and I know that **the parameters must sum to 1**.



# Constrained Optimization

It often happens that the parameters we want to learn have to obey constraints

$$\operatorname{argmin}_{\theta} f(\theta)$$

$$\text{subject to } g(\theta) = 0,$$

- ideally, we would like to incorporate such constraints and still be able to follow the general recipe for optimization discussed before
- **Lagrangians** allow us to do exactly that in the case of **equality constraints** (there are also boundary constraints, which we won't cover)
- we combine our target functions with (sets of) constraints multiplied through **Lagrange multipliers**  $\lambda$

$$\mathcal{L}(\theta, \lambda) = f(\theta) - \lambda g(\theta)$$

- proceed as before: derivative, set to zero, solve for  $\theta$



# Constrained Optimization

## Example

- Find an optimal parameter vector  $\theta$  such that each all  $\theta_i$  sum up to a certain constant  $b$ .
- Formalize the constraint:

$$\sum_i \theta_i = b$$

- Set the constraint to zero

$$0 = \sum_i \theta_i - b = -b + \sum_i \theta_i$$

- set the constraint and write the Lagrangian

$$g_c(\theta) = -b + \sum_i \theta_i$$

$$\begin{aligned}\mathcal{L}(\theta, \lambda) &= f(\theta) - \lambda g_c(\theta) \\ &= f(\theta) - \lambda(-b + \sum_i \theta_i)\end{aligned}$$

- proceed as before: derivative, set to zero, solve for  $\theta$



Jacob Eisenstein. Introduction to Natural Language Processing, Appendix B  
(up to B.1)

Dan Klein. Lagrange Multipliers without Permanent Scarring. <https://people.eecs.berkeley.edu/~klein/papers/lagrange-multipliers.pdf> .  
Sections 1, 2 (up to 2.4), 3.1, 3.5



# Lecture 6: Classification with Naive Bayes

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



# Roadmap

## Last time...

- Machine learning concepts and approaches
- Review of probability
- Review of (basic) optimization

## Today

- Back to Machine learning: Naive Bayes Classification
- Deriving the classifier (drawing on foundations in lecture 4)
- Finding the optimal parameters (drawing on foundations in lecture 5)
- Example and implementation



## **Naive Bayes Theory**

---

## A little thought experiment...

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
overcast	cool	high	true	no

What (do you think) is the class of sunny, cool, normal, false?



## A little thought experiment...

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
sunny	cool	normal	false	yes
sunny	mild	high	false	no
overcast	cool	high	true	no

What (do you think) is the class of rainy, hot, normal, true?



## A little thought experiment...

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
overcast	mild	normal	true	yes
sunny	mild	normal	false	yes
overcast	hot	high	true	yes
sunny	cool	high	false	yes
rainy	cool	normal	true	no
overcast	hot	normal	true	no
sunny	hot	normal	false	no
sunny	mild	normal	true	no
rainy	cool	high	true	no

What (do you think) is the class of overcast, mild, high, false?



# Notation

- $y$  label (e.g., spam, play, ...)
- $x$  observation (e.g., email, day, ...)
- $x_m, m \in \{1, 2, \dots, M\}$  features of  $x$  (e.g., temperature, word, ...)
- $(x^i, y^i)$  observation-label pair; the  $i$ th data point
- $\theta, \phi, \psi, \dots$  parameters
- $f(x, y; \theta)$  function of  $x$  and  $y$  with parameters  $\theta$ , equivalently:  $f_\theta(x, y)$



# A Probabilistic Learner I

- Let's come up with a **supervised machine learning** method
- We build a probabilistic model of the training data  $D^{train}$ ,

$$P_{\theta}(x, y) = \prod_{i \in D^{train}} P_{\theta}(x^i, y^i)$$

- We learn our model parameters  $\theta$  such that they maximize the data log likelihood
- We subsequently use that trained model to predict the class labels of the test data
- So, *given* a test instance  $x \in D^{test}$ , which class  $y$  is most likely?

$$\hat{y} = \operatorname{argmax}_{y \in Y} P(y|x)$$



## A Probabilistic Learner II

The obvious way of doing this:

- For each class  $y$ :
  - Find the instances in the training data labelled as  $y$
  - Count the number of times  $x$  has been observed
- Choose  $\hat{y}$  with the greatest frequency of observed  $x$



## A Probabilistic Learner III

The obvious way of doing this:

- Would require an *enormous* amount of data
- A test instance  $x$  is a bundle of attribute values: to classify an (as-yet) unseen instance would require that *every possible* combination of attribute values has been attested in the training data a non-trivial number of times
- For  $m$  attributes, each taking  $k$  different values, and  $|Y|$  classes, this means  $\mathcal{O}(|Y| \cdot k^m)$  instances
  - Weather example: perhaps 100s of instances
  - 2-class problem, 20 binary attributes: at least 2M instances
  - 4 classes, 60 ternary attributes: at least  $10^{28}$  instances
- Would only be meaningful for the instances that we've actually seen



## Bayes' Rule

Reformulate the probability of class under features as probability of features under class

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

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



## Bayes' Rule

Reformulate the probability of class under features as probability of features under class

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

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

Recall our objective

$$\hat{y} = \operatorname{argmax}_{y \in Y} P(y|x)$$



## Bayes' Rule

Reformulate the probability of class under features as probability of features under class

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

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

Recall our objective

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y \in Y} P(y|x) \\ &= \operatorname{argmax}_{y \in Y} \frac{P(x|y)P(y)}{P(x)} \\ &= \operatorname{argmax}_{y \in Y} P(x|y)P(y)\end{aligned}$$



## Bayes' Rule

Reformulate the probability of class under features as probability of features under class

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

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

Recall our objective

$$\hat{y} = \operatorname{argmax}_{y \in Y} P(y|x)$$

$$= \operatorname{argmax}_{y \in Y} \frac{P(x|y)P(y)}{P(x)}$$

$$= \operatorname{argmax}_{y \in Y} P(x|y)P(y)$$

Recall that each observation consists of many features  $x = x_1, x_2, \dots, x_M$

$$\hat{y} = \operatorname{argmax}_{y \in Y} P(x_1, x_2, \dots, x_M|y)P(y)$$

That is still infeasible!



# Putting ‘Naive’ in Naive Bayes

To arrive at a more feasible solution, we make a naive assumption

$$\begin{aligned} P(x_1, x_2, \dots, x_M | y)P(y) &\approx P(x_1 | y)P(x_2 | y)\dots P(x_M | y)P(y) \\ &= P(y) \prod_{m=1}^M P(x_m | y) \end{aligned}$$

- The **conditional independence assumption**: Conditioned on the class  $y$ , the features are assumed to be independent
- Intuitively: if I know that the class of the email is spam, none of the words depend on their surrounding words
- Clearly, this is nonsense. But the model works surprisingly well, anyway!



# The Naive Bayes Model: Generative Story

## The complete Naive Bayes Classifier

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y \in Y} P(y)P(x_1, x_2, x_3, x_4, \dots x_n | y) \\ &= \operatorname{argmax}_{y \in Y} P(y) \prod_{m=1}^M P(x_m | y)\end{aligned}$$

## The Underlying Probabilistic Model

$$P(x, y) = \prod_{i=1}^N P(y^i) \prod_{m=1}^M P(x_m^i | y^i)$$

## Intuition

---

### Algorithm 1 Generative Story of Naive Bayes

---

- 1: **for** Observation  $i \in \{1, 2, \dots N\}$  **do**
  - 2:   Generate the label  $y^i$  from  $P(y)$
  - 3:   **for** Feature  $m \in \{1, 2, \dots M\}$  **do**
  - 4:     Generate feature value  $x_m^i$  given that label= $y^i$  from  $P(x_m^i | y^i)$
- 



## Naive Bayes Assumptions

$$P(x, y) = \prod_{i=1}^N P(y^i) \prod_{m=1}^M P(x_m^i | y^i)$$

- Features of an instance are conditionally independent given the class
- Instances are independent of each other
- The distribution of data in the training instances is the same as the distribution of data in the test instances



# Gaussian Naive Bayes with 2 classes

---

**Observations** real-valued feature vectors of length M  
labelled with binary class (0,1)

---

## Example

---

**Model**  $y$  drawn from Bernoulli distribution  
 $x_m$  drawn from Gaussian distribution

$$\begin{aligned} p(x, y) &= p_{\phi, \psi}(x_1, x_2, \dots, x_m, y) = p_\phi(y) \prod_m^M p_\psi(x_k|y) \\ &= BN(y|\phi) \prod_m^M N(x_k|\psi = \{\mu_{m,y}, \sigma_{m,y}\}) \\ &= \phi^y(1-\phi)^{(1-y)} \prod_{m=1}^M \frac{1}{\sqrt{2\pi\sigma_{m,y}^2}} \exp\left(-\frac{1}{2} \frac{(x_m - \mu_{m,y})^2}{\sigma_{m,y}^2}\right) \end{aligned}$$

---

**Prediction**  $\hat{y} = \text{argmax}_y p(y|x)$



# Bernoulli Naive Bayes with 2 classes

---

**Observations** binary feature vectors of length M  
labelled with binary class (0,1)

---

## Example

---

**Model**  $y$  drawn from Bernoulli distribution  
 $x_m$  drawn from Bernoulli distribution

$$\begin{aligned} p(x, y) &= p_{\phi, \psi}(x_1, x_2, \dots, x_m, y) = p_\phi(y) \prod_m p_\psi(x_k | y) \\ &= BN(y|\phi) \prod_m BN(x_k | \psi_{m,y}) \\ &= \phi^y (1 - \phi)^{1-y} \prod_{m=1}^M (\psi_{y,m})^{x_m} (1 - \psi_{y,m})^{(1-x_m)} \end{aligned}$$

---

**Prediction**  $\hat{y} = \text{argmax}_y p(y|x)$



# Categorical Naive Bayes with $C$ classes

---

**Observations** categorical feature vectors of length M  
labelled with one of  $C$  classes ( $C > 2$ )

---

## Example

---

**Model**  $y$  drawn from Categorical distribution with  $C$  classes  
 $x_m$  drawn from Categorical distribution over  $K$  values

$$\begin{aligned} p(x, y) &= p_{\phi, \psi}(x_1, x_2, \dots, x_m, y) = p_\phi(y) \prod_m^M p_\psi(x_k | y) \\ &= \text{Cat}(y|\phi) \prod_m^M \text{Cat}(x_k | \psi_{m,y}) \\ &= \phi_y \prod_{m=1}^M \prod_{k=1}^K (\psi_{y,m,k}) \end{aligned}$$

---

**Prediction**  $\hat{y} = \text{argmax}_y p(y|x)$



But where do the parameters come from?

- Parameters  $\phi$  of the **Categorical distribution over class labels** are the relative frequencies of classes observed in the training data

$$\phi_y = \frac{\text{count}(y)}{N}$$

- Parameters  $\psi$  of the **Categorical distributions over features given a class label** are the observed relative frequencies of (class, label) among all instances with that class

$$\psi_{y,m} = \frac{\text{count}(y, m)}{\text{count}(y)}$$



## But where do the parameters come from?

- Parameters  $\phi$  of the **Categorical distribution over class labels** are the relative frequencies of classes observed in the training data

$$\phi_y = \frac{\text{count}(y)}{N}$$

- Parameters  $\psi$  of the **Categorical distributions over features given a class label** are the observed relative frequencies of (class, label) among all instances with that class

$$\psi_{y,m} = \frac{\text{count}(y, m)}{\text{count}(y)}$$

- These parameters maximize the probability of the observed dataset  $P(\{(x^i, y^i)\}_{i=1}^N; \phi, \psi)$ . They are the **maximum likelihood estimate** of  $\phi$  and  $\psi$ .
- You are invited to derive this result using the optimization techniques we learnt in the last lecture!



But where do the parameters come from?

- Parameters  $\phi$  of the **Categorical distribution over class labels** are the relative frequencies of classes observed in the training data

$$\phi_y = \frac{\text{count}(y)}{N}$$

- Parameters  $\psi$  of the **Categorical distributions over features given a class label** are the observed relative frequencies of (class, label) among all instances with that class

$$\psi_{y,m} = \frac{\text{count}(y, m)}{\text{count}(y)}$$

**Activity:** What are the four steps we would follow in finding the

- The optimal parameters?

$$P(\{$$

$$\text{and } \psi.$$

- You are invited to derive this result using the optimization techniques we learnt in the last lecture!



# Maximum Likelihood Estimation for Gaussian Naive Bayes

For each class  $y$  and each feature  $x_m$ , we learn an individual Gaussian distribution parameterized by a mean  $\mu_{y,m}$  and a standard deviation  $\sigma_{y,m}$

**Mean:** the average of all observed feature value for  $x_m$  under class  $y$

$$\mu_{y,m} = \frac{1}{\text{count}(y)} \sum_{i:y_i=y} x_m^i$$

**Standard deviation:** Sum of squared differences of observed values from the mean. Normalized, and square rooted.

$$\sigma_{y,m} = \sqrt{\frac{\sum_{i:y_i=y} (x_m^i - \mu_{y,m})^2}{\text{count}(y)}}$$



## Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu



## Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

We need  $P(y = k)$ ,  $P(x = f|y = k)$ , for every possible value  $k$  for  $y$  and every possible value  $f$  for  $x$



# Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

We need  $P(y = k)$ ,  $P(x = f|y = k)$ , for every possible value  $k$  for  $y$  and every possible value  $f$  for  $x$

$$P(\text{Flu}) = 3/5$$

$$P(\text{Headache} = \text{severe}|\text{Flu}) = 2/3$$

$$P(\text{Headache} = \text{mild}|\text{Flu}) = 1/3$$

$$P(\text{Headache} = \text{no}|\text{Flu}) = 0/3$$

$$P(\text{Sore} = \text{severe}|\text{Flu}) = 1/3$$

$$P(\text{Sore} = \text{mild}|\text{Flu}) = 2/3$$

$$P(\text{Sore} = \text{no}|\text{Flu}) = 0/3$$

$$P(\text{Temp} = \text{high}|\text{Flu}) = 1/3$$

$$P(\text{Temp} = \text{normal}|\text{Flu}) = 2/3$$

$$P(\text{Cough} = \text{yes}|\text{Flu}) = 3/3$$

$$P(\text{Cough} = \text{no}|\text{Flu}) = 0/3$$

$$P(\text{Cold}) = 2/5$$

$$P(\text{Headache} = \text{severe}|\text{Cold}) = 0/2$$

$$P(\text{Headache} = \text{mild}|\text{Cold}) = 1/2$$

$$P(\text{Headache} = \text{no}|\text{Cold}) = 1/2$$

$$P(\text{Sore} = \text{severe}|\text{Cold}) = 1/2$$

$$P(\text{Sore} = \text{mild}|\text{Cold}) = 0/2$$

$$P(\text{Sore} = \text{no}|\text{Cold}) = 1/2$$

$$P(\text{Temp} = \text{high}|\text{Cold}) = 0/2$$

$$P(\text{Temp} = \text{normal}|\text{Cold}) = 2/2$$

$$P(\text{Cough} = \text{yes}|\text{Cold}) = 1/2$$

$$P(\text{Cough} = \text{no}|\text{Cold}) = 1/2$$



## Naive Bayes Example II

Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a *Cold*, or the *Flu*?



## Naive Bayes Example II

Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a *Cold*, or the *Flu*?

Cold:

$$\begin{aligned} P(Co) &\times P(H = m|Co)P(S = s|Co)P(T = n|Co)P(C = n|Co) \\ \frac{2}{5} &\times \left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\left(\frac{2}{2}\right)\left(\frac{1}{2}\right) = 0.05 \end{aligned}$$

Flu:

$$\begin{aligned} P(Fl) &\times P(H = m|Fl)P(S = s|Fl)P(T = n|Fl)P(C = n|Fl) \\ \frac{3}{5} &\times \left(\frac{1}{3}\right)\left(\frac{1}{3}\right)\left(\frac{2}{3}\right)\left(\frac{0}{3}\right) = 0 \end{aligned}$$



## Naive Bayes Example III

Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{aligned} P(Co) &\times P(H = s|Co)P(S = m|Co)P(T = h|Co)P(C = n|Co) \\ \frac{2}{5} &\times \left(\frac{0}{2}\right)\left(\frac{0}{2}\right)\left(\frac{0}{2}\right)\left(\frac{1}{2}\right) = 0 \end{aligned}$$

Flu:

$$\begin{aligned} P(Fl) &\times P(H = s|Fl)P(S = m|Fl)P(T = h|Fl)P(C = n|Fl) \\ \frac{3}{5} &\times \left(\frac{2}{3}\right)\left(\frac{2}{3}\right)\left(\frac{1}{3}\right)\left(\frac{0}{3}\right) = 0 \end{aligned}$$



## The problem with unseen features

- If any term  $P(x_m|y) = 0$  then the class probability  $P(y|x) = 0$
- But, we already established that in any realistic scenario we won't see every class-feature combination during training
- A single zero renders many additional meaningful observations irrelevant
- **Solution:** no event is impossible:  $P(x_m|y) > 0 \forall x_m \forall y$
- We need to readjust the remaining model parameters to maintain valid probability distributions ( $\sum_i \psi_i = 1$ )



## Simplest approach

- if we calculate  $P(x_m|y) = 0$ , we replace 0 with a very (!) small constant typically called  $\epsilon$
- $\epsilon$  needs to be smaller (preferably much smaller) than  $\frac{1}{N}$  ( $N$ =the number of training instances). **Why?**
- Effectively it reduces most comparisons to the cardinality of  $\epsilon$  (fewest  $\epsilon$ s wins)
- We assume that  $\epsilon$  is so small that  $1 + \epsilon \approx 1$ , so we do not need to renormalize or adjust the other probabilities in the model



THE UNIVERSITY OF  
MELBOURNE

## Epsilon Smoothing

Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{aligned} P(Co) &\times P(H = s|Co)P(S = m|Co)P(T = h|Co)P(C = n|Co) \\ \frac{2}{5} &\times (\epsilon)(\epsilon)(\epsilon)\left(\frac{1}{2}\right) = \frac{\epsilon^3}{5} \end{aligned}$$

Flu:

$$\begin{aligned} P(Fl) &\times P(H = s|Fl)P(S = m|Fl)P(T = h|Fl)P(C = n|Fl) \\ \frac{3}{5} &\times \left(\frac{2}{3}\right)\left(\frac{2}{3}\right)\left(\frac{1}{3}\right)(\epsilon) = \frac{12\epsilon}{135} = \frac{4\epsilon}{45} \end{aligned}$$



## Laplace Smoothing

Add a “pseudocount”  $\alpha$  to each feature count observed during training

$$P(x_m = j|y = k) = \frac{\alpha + \text{count}(y = k, x_m = j)}{M\alpha + \text{count}(y = k)}$$

- the value of  $\alpha$  is a parameter; very often  $\alpha = 1$
- all **counts** are incremented to ensure to maintain monotonicity (for  $\alpha = 1$ : 0 becomes 1, 1 becomes 2, 2 becomes 3, ...)
- $M$  is the number of values  $x_m$  can take on



# Laplace Smoothing

Add a “pseudocount”  $\alpha$  to each feature count observed during training

$$P(x_m = j|y = k) = \frac{\alpha + \text{count}(y = k, x_m = j)}{M\alpha + \text{count}(y = k)}$$

## Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

	original estimate	smoothed estimate ( $\alpha = 1$ )
$P(\text{Headache} = \text{severe} \text{Flu})$	2/3	(2 + 1)/(3 + 3) = 3/6
$P(\text{Headache} = \text{mild} \text{Flu})$	1/3	(1 + 1)/(3 + 3) = 2/6
$P(\text{Headache} = \text{no} \text{Flu})$	0/3	(0 + 1)/(3 + 3) = 1/6
$P(\text{Cough} = \text{yes} \text{Flu})$	3/3	(3 + 1)/(3 + 2) = 4/5
$P(\text{Cough} = \text{no} \text{Flu})$	0/3	(0 + 1)/(3 + 2) = 1/5
...		



## Laplace Smoothing

Add a “pseudocount”  $\alpha$  to each feature count observed during training

$$P(x_m = j|y = k) = \frac{\alpha + \text{count}(y = k, x_m = j)}{M\alpha + \text{count}(y = k)}$$

- Probabilities are changed drastically when there are few instances; with a large number of instances, the changes are small
- Laplace smoothing (and smoothing in general) **reduces variance** of the NB classifier because it reduces sensitivity to individual (non-)observations in the training data
- Laplace smoothing (and smoothing in general) **adds bias** to the NB classifier. We no longer have a true maximum likelihood estimator.
- How to choose  $\alpha$ ?
- There are other smoothing methods, including Good-Turing, Kneser-Ney, Regression, ... (outside the scope of this class)



## **Implementation of Categorical Naive Bayes**

---

# Implementing a Naive Bayes Classifier

Naive Bayes is a supervised machine learning method:

- We need to build a model (“training phase”)
- We need to make predictions using that model and evaluate the predictions against the ground truth (“testing phase”)



## Training a NB Classifier I

Our model consists of two kinds of probabilities:

- *priors*  $P(Y = k)$  (one per class)
- *likelihoods*  $P(X = j|Y = k)$  (one per attribute value, per class)



# Calculating priors by counting I

There is one prior  $P(Y = k)$  per class

$X_1$ (Headache)	$X_2$ (Sore)	$X_3$ (Temperature)	$X_4$ (Cough)	$Y$ (Diagnosis)
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
2	3



# Calculating priors by counting I

There is one prior  $P(Y = k)$  per class

$X_1$ (Headache)	$X_2$ (Sore)	$X_3$ (Temperature)	$X_4$ (Cough)	$Y$ (Diagnosis)
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
2	3

We need to normalize these counts by the total number of training instances  
 $N$ . Options:

- divide each entry by the sum of the entries in the list
- keep a separate counter for the total number of instances  $N$ , which is often useful



## Calculating likelihoods by counting I

There is one likelihood  $P(x = j|y = k)$  per attribute value, per class: 2D array?



## Calculating likelihoods by counting I

There is one likelihood  $P(x = j|y = k)$  per attribute value, per class, **for each attribute  $X$ : 2D array? 3D array?**

- But each attribute might have a different number of possible attribute values,
- And we might not know all of the various attribute values before we start counting.
- So...
  - 2D array of dictionaries?
  - 1D array of dictionaries of dictionaries?
  - Dictionary of dictionaries of dictionaries?



## Calculating likelihoods by counting II

Headache	Sore	Temperature	Cough	Diagnosis
<b>severe</b>	mild	high	yes	<b>Flu</b>
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache

Temperature

Cold: {}		Cold: {}
Flu: {severe:1}		Flu: {}

Sore

Cough

Cold: {}		Cold: {}
Flu: {}		Flu: {}



## Calculating likelihoods by counting II

Headache	Sore	Temperature	Cough	Diagnosis
severe	<b>mild</b>	high	yes	<b>Flu</b>
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache	Temperature
Cold: {}	Cold: {}
Flu: {severe:1}	Flu: {}

Sore	Cough
Cold: {}	Cold: {}
Flu: {mild:1}	Flu: {}



## Calculating likelihoods by counting II

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache

Temperature

Cold: {no:1, mild:1}		Cold: {normal:2}
Flu: {severe:2, mild:1}		Flu: {high:1, normal:2}

Sore

Cough

Cold: {severe:1, no:1}		Cold: {yes:1, no:1}
Flu: {mild:2, severe:1}		Flu: {yes:3}



## Calculating likelihoods by counting II

We need to know the number of instances of class  $c_j$  to turn these counts into probabilities:

- The slow way: sum the entries in the corresponding dictionary
- The fast way: read off the class array

Smoothing can be done:

- When accessing values, e.g. if value is 0, replace with  $\epsilon$
- Using a `defaultdict`, e.g. default for Laplace is 1



THE UNIVERSITY OF  
MELBOURNE

## Making predictions using a NB Classifier i

$$\hat{y} = \operatorname{argmax}_{k \in Y} P(y = k) \prod_m P(x_m = j | y = k)$$

- These values can be read off the data structures from the training phase.
- We only care about the class corresponding to the maximal value, so as we progress through the classes, we can keep track of the greatest value so far.



## Making predictions using a NB Classifier ii

We're multiplying a bunch of numbers  $(0, 1]$  together — because of our floating-point number representation, we tend to get **underflow**.

One common solution is a **log-transformation**:

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{k \in Y} P(y = k) \prod_m P(x_m = j | y = k) \\ &= \operatorname{argmax}_{k \in Y} \left[ \log P(y = k) + \sum_m \log P(x_m = j | y = k) \right]\end{aligned}$$



## Evaluating a NB classifier

Evaluation in a supervised ML context (for NB and other methods):

- fundamentally based around comparing predicted labels with the actual labels

We'll talk about this in much more detail in the upcoming lectures.



# Naive Bayes: Final thoughts

**Why does it work given that it's a blatantly wrong model of the data?**

- we don't need the true distribution over  $P(y|x)$ , we just need to be able to identify the most likely outcome

**Advantages of Naive Bayes**

- easy to build and estimate
- easy to scale to many feature dimensions (e.g., words in the vocabulary) and data sizes
- reasonably easy to explain why a specific class was predicted
- good starting point for a classification project



## Naive Bayes

- What is the Naive Bayes algorithm?
- What is Bayes' Rule and how does it relate to the Naive Bayes algorithm
- What are the simplifying assumptions we make?
- How and why do we use smoothing in Naive Bayes?
- How can we implement a Naive Bayes classifier?

**Next Lecture:** Evaluation



## References

Jacob Eisenstein. *Natural Language Processing*. MIT Press (2019).  
Chapter 2.2



## MLE of Categorical Naive Bayes – for the Math hungry

- The **likelihood** is the probability of the data as a function of only the parameters:

$$\mathcal{L}(\phi, \psi) = \sum_i^N \log \text{Cat}(x^i; \psi_{y^i}) + \log \text{Cat}(y^i | \phi)$$

- Focussing only on terms involving  $\psi$  (the same steps can be applied to  $\phi$ , separately)

$$\begin{aligned}\mathcal{L}(\psi) &= \sum_i^N \log \text{Cat}(x^i; \psi_{y^i}) \\ &= \sum_i^N \sum_{f=1}^V x_f \times \log \psi_{y^i, f}\end{aligned}$$

- now choose  $\psi$  to maximize  $\mathcal{L}$  under the constraint that

$$\sum_{f=1}^V \psi_{y, f} = 1 \quad \forall y,$$

(i.e., all possible outcomes add up to 1)



## MLE of Categorical Naive Bayes – for the Math hungry

- integrate the constraint by adding a set of Lagrange multipliers

$$\ell(\psi_y) = \sum_{i:y^i=y} \sum_{f=1}^V x_f \times \log \psi_{y,f} - \lambda \left( \sum_{f=1}^V \psi_{y,f} - 1 \right)$$

- we differentiate the likelihood wrt.  $\psi_{y,f}$  i.e., the probability of feature value  $f$  under class  $y$

$$\frac{\partial \ell(\psi_y)}{\partial \psi_{y,f}} = \sum_{i:y^i=y} x_f / \psi_{y,f} - \lambda$$

- set the derivatives to zero, and rearrange

$$\lambda \psi_{y,f} = \sum_{i:y^i=y} x_f$$

$$\psi_{y,f} \propto \sum_{i:y^i=y} x_f = \text{count}(y, f)$$

- and the only way to find an exact solution that obeys our sum-to-one constraint is

$$\psi_{y,f} = \frac{\text{count}(y, f)}{\sum_{f' \in V} \text{count}(y, f')} = \frac{\text{count}(y, f)}{\text{count}(y)}$$

...which is the exact quantity we defined on slide 14. Hurray!



## MLE of Categorical Naive Bayes – for the Math hungry

Following an almost identical procedure we can derive that

$$\phi_y = \frac{\text{count}(y)}{N}$$



# Lecture 7: Model Evaluation I

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



## So far

- Instance-based classification with KNN
- Probabilities and probabilistic modeling
- Optimization and MLE
- Probabilistic classification with Naive Bayes

## Today... Evaluation

- How do we know that we succeeded in learning?
- Evaluation paradigms
- Evaluation methods

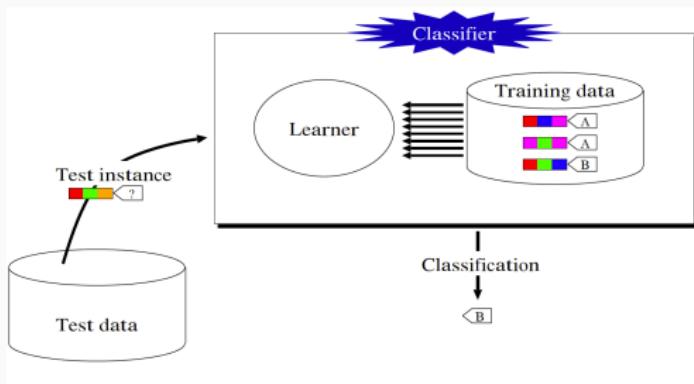


## **Classification Evaluation**

---

# The Nature of “Classification”

- Input: set of labelled training instances; set of unlabelled test instances
- Model: an estimate of the underlying target function
- Output: prediction of the classes of the test instances



## Goal 1: Model Evaluation

**Our goal (in a supervised Machine Learning framework):**

- Have a perfect model?
  - Not necessarily clear what that means
  - More difficult than necessary
- Make predictions that are correct!
- Train and test machine learning models based on the limited data available
- Estimate the *true* performance on any data based on the errors made on the limited labelled data available



## Goal 2: Model Selection

**Choosing the best model from a number of possibilities**

- Different machine learning algorithms
- Different parameterizations for the same algorithm
- Different training parameters
- Other considerations: Efficiency, explainability, fairness, ...



## Evaluation Strategies

---

# Evaluating Classification I

Main idea:

- Train (build model) using **training data**
- Test (evaluate model) on **test data**

But often, we just have **data** — a collection of instances



## Evaluating Classification II

An obvious strategy, which is highly **not recommended**:

- Use all of the instances as training data
  - Build the model using all of the instances
- Use all of the (same) instances as test data
  - Evaluate the model using all of the instances

“Testing on the training data” tends to grossly over-estimate classifier performance.

Effectively, we are telling the classifier what the correct answers are, and then asking whether it can come up with the correct answers.



## Holdout

One solution: **Holdout** evaluation strategy

- Each instance is randomly assigned as either a training instance **or** a testing instance
- Effectively, the data is **partitioned** — no overlap between datasets
- Evaluation strategy:
  - Build the model using (only) the training instances
  - Evaluate the model using (only) the (different) test instances

Very commonly used strategy; typical split sizes are approximately 50–50, 80–20, 90–10 (train, test)

Source(s): Tan et al. [2006, pp 186–7]



## Advantages

- simple to work with and implement
- fairly high reproducability

## Disadvantages

- size of the split affects estimate of the model's behaviour:
  - lots of test instances, few training instances: learner doesn't have enough information to build an accurate model
  - lots of training instances, few test instances: learner builds an accurate model, but test data might not be representative (so estimates of performance can be too high/too low)



## Repeated Random Subsampling

Slower, but somewhat better solution: **Repeated Random Subsampling**

- Like Holdout, but iterated multiple times:
  - A new training set and test set are randomly chosen each time
  - Relative size of training–test is fixed across iterations
  - New model is built each iteration
- Evaluate by averaging (chosen metric) across the iterations



# Repeated Random Subsampling

## Advantages:

- averaging Holdout method tends to produce more reliable results

## Disadvantages:

- more difficult to reproduce
- slower than Holdout (by a constant factor)
- wrong choice of training set–test set size can still lead to highly misleading results (that are now very difficult to sanity–check)



## Cross–Validation

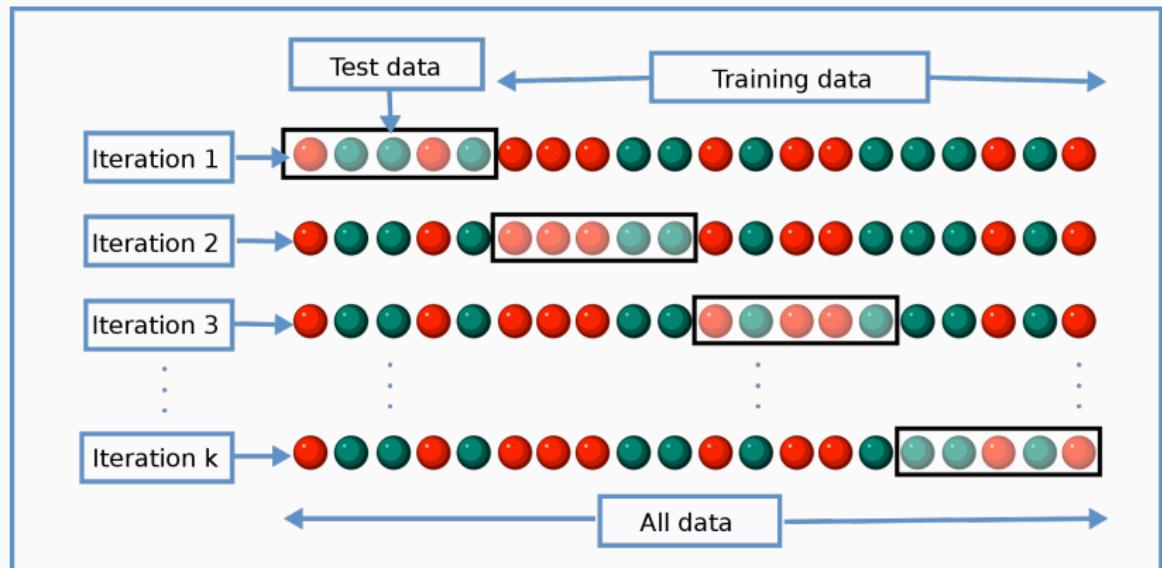
Usually preferred alternative: **Cross–Validation**

- Data is progressively split into a number of partitions  $m$  ( $\geq 2$ )
- Iteratively:
  - One partition is used as test data
  - The other  $m - 1$  partitions are used as training data
- Evaluation metric is aggregated across  $m$  test partitions
  - This could mean averaging, but more often, counts are added together across iterations



# Cross-Validation

Usually preferred alternative: **Cross-Validation**



[https://en.wikipedia.org/wiki/Cross-validation\\_\(statistics\)#/media/File:K-fold\\_cross\\_validation\\_EN.svg](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.svg)



## Cross-Validation

Why is this better than Holdout/Repeated Random Subsampling?

- Every instance is a test instance, for some partition
  - Similar to testing on the training data, but without dataset overlap
  - Evaluation metrics are calculated with respect to a dataset that looks like the entire dataset (i.e. the entire dataset)
- Takes roughly the same amount of time as Repeated Random Subsampling (but see below)
- Very reproducible
- Can be shown to minimise **bias** and **variance** of our estimates of the classifier's performance (more on this in Evaluation II)



# Cross-Validation

How big is  $m$ ?

- Number of folds directly impacts runtime *and* size of datasets:
  - Fewer folds: more instances per partition, more variance in performance estimates
  - More folds: fewer instances per partition, less variance but slower
- Most common choice of  $m$ : 10 (occasionally, 5)
  - Mimics 90–10 Holdout, but far more reliable
- Best choice:  $m=N$ , the number of instances (known as **Leave-One-Out Cross-Validation**):
  - Maximises training data for the model
  - Mimics actual testing behaviour (every test instance is treated as an individual test “set”)
  - Way too slow to use in practice



# Inductive Biases and No Free Lunch

*A learner that makes no **a priori assumptions** regarding the identity of the target concept has no **rational basis** for classifying any unseen instances”*

[Mitchell, 1997, Ch. 2.7.3]

## The No Free Lunch Theorem (Wolpert and Macready, 1997)

- Intuition: You don't get *something* for *nothing*. You won't get a *good classifier* (or any ML algorithm) without *making assumptions*.
- Averaged across all possible problems, the performance of any two algorithms is identical.
  - If algorithm *A* does well on *p1* it necessarily performs worse on some *p2*
- Averaged across all classification problems, the generalization error on a held-out test set of any two classifiers is identical. There is no universally superior classifier.



# Inductive Biases and No Free Lunch

*A learner that makes no **a priori assumptions** regarding the identity of the target concept has no **rational basis** for classifying any unseen instances”*

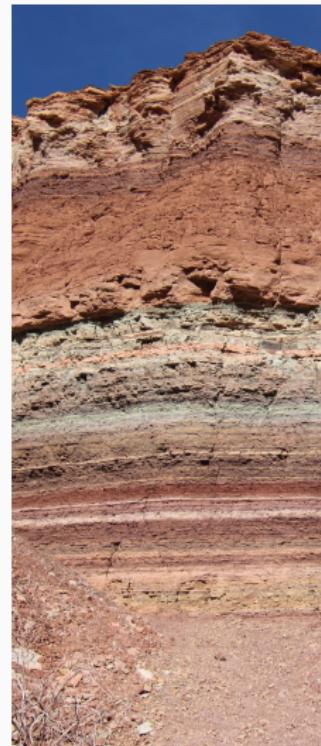
[Mitchell, 1997, Ch. 2.7.3]

- **Inductive Learning Hypothesis:** Any hypothesis found to approximate the target function well over (a sufficiently large) training data set will also approximate the target function well over **unseen** test examples.
- Assumptions must be made about the data to build a model and make predictions (**inductive biases**)
  - Different assumptions will lead to different predictions



# Stratification

- Typical inductive bias in our evaluation framework (n.b. independent of model): **Stratification**
  - Assume that **class distribution** of unseen instances will be the same as distribution of seen instances
- When constructing Holdout/Cross-Validation partitions, ensure that training data and test data **both** have same class distribution as dataset as a whole



## **Evaluation Measures**

---

## Error $E$

The fraction of incorrect predictions.

$$E = \frac{1}{N} (1 - I(y_i, \hat{y}_i)), \text{ where } I(a, b) \begin{cases} 1, & \text{if } a == b \\ 0, & \text{otherwise} \end{cases}$$

Error rate reduction:  $\text{ERR} = \frac{E_0 - E}{E_0}$



## Error $E$

The fraction of incorrect predictions.

$$E = \frac{1}{N} (1 - I(y_i, \hat{y}_i)), \text{ where } I(a, b) \begin{cases} 1, & \text{if } a == b \\ 0, & \text{otherwise} \end{cases}$$

Error rate reduction:  $\text{ERR} = \frac{E_0 - E}{E_0}$

Outlook	Temperature	Humidity	Windy	Actual	Classified
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	
sunny	cool	normal	FALSE	yes	
rainy	mild	normal	FALSE	yes	
sunny	mild	normal	TRUE	yes	no
overcast	mild	high	TRUE	yes	no
overcast	hot	normal	FALSE	yes	yes
rainy	mild	high	TRUE	no	yes



## Two Types of Errors

### Contingency Tables

$\downarrow y, \hat{y} \rightarrow$	1	0
1	true positive (TP)	false negative (FN)
0	false positive (FP)	true negative (TN)

### Not all Errors are equal

- Some problems want to strongly **penalize false negative errors** e.g.,
- Other problems want to strongly **penalize false positive errors**,
- Attach a “cost” to each type of error



# Accuracy

$\downarrow y, \hat{y} \rightarrow$	1	0
1	true positive (TP)	false negative (FN)
0	false positive (FP)	true negative (TN)

**Accuracy:** The basic evaluation metric

$$\text{Accuracy} = \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}} = \frac{TP + TN}{TP + FP + TN + FN}$$

- Same as  $1 - E$
- Quantifies how frequently the classifier is correct



# Accuracy

Outlook	Temperature	Humidity	Windy	Actual	Classified
sunny	hot	high	FALSE	no	
sunny	hot	high	TRUE	no	
overcast	hot	high	FALSE	yes	
rainy	mild	high	FALSE	yes	
rainy	cool	normal	FALSE	yes	
rainy	cool	normal	TRUE	no	
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	
sunny	cool	normal	FALSE	yes	
rainy	mild	normal	FALSE	yes	
sunny	mild	normal	TRUE	yes	no
overcast	mild	high	TRUE	yes	yes
overcast	hot	normal	FALSE	yes	yes
rainy	mild	high	TRUE	no	yes



# Accuracy

4 test instances; 2 correct predictions, 2 incorrect predictions

$$\begin{aligned}\text{Accuracy} &= \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}} \\ &= \frac{2}{4} = 50\%\end{aligned}$$



## Precision and Recall

$\downarrow y, \hat{y} \rightarrow$	1 (interesting)	0 (uninteresting)
1 (interesting)	true positive (TP)	false negative (FN)
0 (uninteresting)	false positive (FP)	true negative (TN)

With respect to **just the interesting class**:

- **Precision:** How often are we correct, when we predict that an instance is interesting?

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** What proportion of the truly interesting instances have we correctly identified as interesting?

$$\text{Recall} = \frac{TP}{TP + FN}$$



## Precision and Recall

Precision/Recall are typically in an **inverse relationship**. We can generally set up our classifier, so that:

- The classifier has high Precision, but low Recall
- The classifier has high Recall, but low Precision

But, we want **both** Precision and Recall to be high. A popular metric that evaluates this is **F-score**:

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

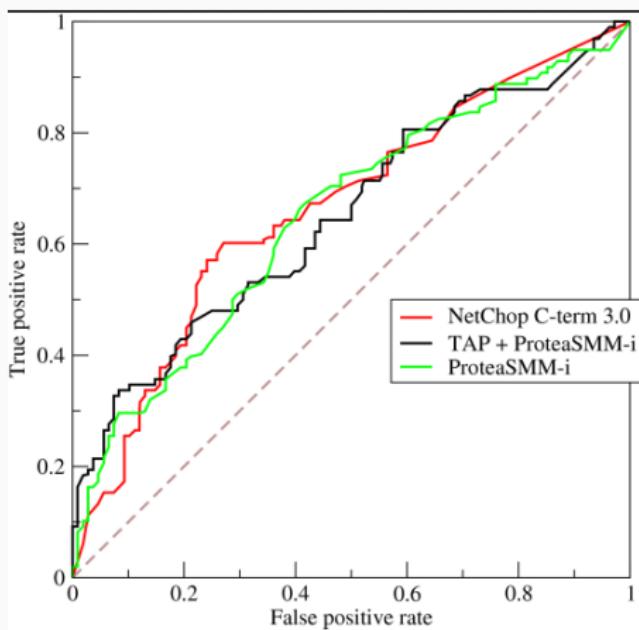
$$F_1 = \frac{2PR}{P + R}$$



# Capturing trade-offs between different objectives

## Receiver Operating characteristics (ROC-curve)

- Goal: Maximize the area under the ROC curve: (a) minimize the **false alarms** and maximize the **true alarms**
- History: trade off between **false alarms** and **true alarms** in signal processing



- X-axis: percentage of false positives  
Y-axis: percentage of true positives

# Many other Metrics!

		Predicted condition			
Total population		Predicted Condition positive	Predicted Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	
True condition	condition positive	True positive	False Negative (Type II error)	True positive rate (TPR), Sensitivity, Recall = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False negative rate (FNR), Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$
	condition negative	False Positive (Type I error)	True negative	False positive rate (FPR), Fall-out $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$
$\text{Accuracy (ACC)} = \frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$		Positive predictive value (PPV), Precision $= \frac{\sum \text{True positive}}{\sum \text{Test outcome positive}}$	False omission rate (FOR) $= \frac{\sum \text{False negative}}{\sum \text{Test outcome negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR}^+}{\text{LR}^-}$
		False discovery rate (FDR) $= \frac{\sum \text{False positive}}{\sum \text{Test outcome positive}}$	Negative predictive value (NPV) $= \frac{\sum \text{True negative}}{\sum \text{Test outcome negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$	

From Wikipedia:

[https://en.wikipedia.org/wiki/Sensitivity\\_and\\_specificity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity)



## Multi-class Evaluation

For a multi-class problem, assume an **Interesting** class ( $I$ ) and several **Uninteresting** classes ( $U_1, U_2, \dots$ ).

		<i>Predicted</i>			
		<i>I</i>	<i>U<sub>1</sub></i>	<i>U<sub>2</sub></i>	...
<i>Actual</i>	<i>I</i>	TP	FN	FN	...
	<i>U<sub>1</sub></i>	FP	TN	TN	...
	<i>U<sub>2</sub></i>	FP	TN	TN	...
	...	...	...	...	...

A multi-class **Confusion Matrix**.

Typically, all classes are “interesting” in a multi-class context.



## Multi-class Evaluation

- The natural definition of Accuracy still makes sense in a multi-class context,
- The technical definition behaves strangely: re-interprets the multi-class problem as a special case of a two-class problem, namely One-vs-Rest
- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
- **macro-averaging**: calculate P, R per class and then average

$$\text{Precision}_M = \frac{\sum_{i=1}^c \text{Precision}_i}{c}$$
$$\text{Recall}_M = \frac{\sum_{i=1}^c \text{Recall}_i}{c}$$



## Multi-class Evaluation

- The natural definition of Accuracy still makes sense in a multi-class context,
- The technical definition behaves strangely: re-interprets the multi-class problem as a special case of a two-class problem, namely One-vs-Rest
- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
- **micro-averaging**: combine all test instances into a single pool

$$\text{Precision}_{\mu} = \frac{\sum_{i=1}^c TP_i}{\sum_{i=1}^c TP_i + FP_i}$$
$$\text{Recall}_{\mu} = \frac{\sum_{i=1}^c TP_i}{\sum_{i=1}^c TP_i + FN_i}$$



## Multi-class Evaluation

- The natural definition of Accuracy still makes sense in a multi-class context,
- The technical definition behaves strangely: re-interprets the multi-class problem as a special case of a two-class problem, namely One-vs-Rest
- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
- **weighted averaging**: calculate P, R per class and then average, based on the proportion of instances in that class

$$\text{Precision}_W = \sum_{i=1}^c \text{Precision}_i \times \left(\frac{n_i}{N}\right)$$

$$\text{Recall}_W = \sum_{i=1}^c \text{Recall}_i \times \left(\frac{n_i}{N}\right)$$



## How to average your predictions?

- Macro average treats all classes equal: emphasizes small classes.  
Micro average is dominated by large classes.
- **Macro-averaged F-score:** is it the F-score of macro-averaged P (over classes) and macro-averaged R (over classes)? Or the macro-average (over classes) of the F-score for each class?
- If we are doing **Repeated Random Subsampling**, and want **weighted-averaged Precision**, do we average the weighted Precision (over classes) for each iteration of Random Subsampling? or do we take the weighted average (over classes) of the Precision averaged over the iterations of Subsampling? Or the weighted average over the instances aggregated over the iterations?



## **Model comparison**

---

## Baselines vs. Benchmarks

- **Baseline** = naive method which we would expect any reasonably well-developed method to better
  - e.g. for a novice marathon runner, the time to walk 42km*
- **Benchmark** = established rival technique which we are pitching our method against
  - e.g. for a marathon runner, the time of our last marathon run/the world record time/3 hours/...*
- “Baseline” often used as umbrella term for both meanings



# The Importance of Baselines

- Baselines are important in establishing whether any proposed method is doing better than “dumb and simple”  
*“dumb” methods often work surprisingly well*
- Baselines are valuable in getting a sense for the intrinsic difficulty of a given task (cf. accuracy = 5% vs. 99%)
- In formulating a baseline, we need to be sensitive to the importance of positives and negatives in the classification task  
*limited utility of a baseline of unsuitable for a classification task aimed at detecting potential sites for new diamond mines (as nearly all sites are unsuitable)*



**Method 1:** randomly assign a class to each test instance

- Often the only option in unsupervised/semi-supervised contexts

**Method 2:** randomly assign a class to each test instance, weighting the class assignment according to  $P(C_k)$

- Assumes we know the prior probabilities
- Alleviate effects of variance by:
  - running method  $N$  times and calculating the mean accuracy  
*OR*
  - arriving at a deterministic estimate of the accuracy of random assignment =  $\sum_i P(C_i)^2$



## Zero-R (Zero rules)

Also known as **majority class** baseline

- **Method:** classify all instances according to the most common class in the training data
- The most commonly used baseline in machine learning
- Inappropriate if the majority class is FALSE and the learning task is to identify needles in the haystack

Outlook	Temperature	Humidity	Windy	$y$ (Play)	$\hat{y}$
sunny	hot	high	FALSE	no	
sunny	hot	high	TRUE	no	
overcast	hot	high	FALSE	yes	
rainy	mild	high	FALSE	yes	
rainy	cool	normal	FALSE	yes	
rainy	cool	normal	TRUE	no	
overcast	cool	normal	TRUE	yes	
sunny	mild	high	FALSE	no	?



# One-R (One Rule)

## Introduction

- Select **one** attribute and use it to predict an instance's class
- Test each attribute, and select the one with the smallest error rate
- Each attribute-specific test is often called "Decision stump" (more on that in the Trees lecture)

## Intuition

- If there is a single, simple feature with which we can classify most of our features correctly – do we really need a sophisticated machine learner?



THE UNIVERSITY OF  
MELBOURNE

## One-R pseudo-code

For each attribute

For each value of the attribute, make a rule:

- (i) count how often each class appears
- (ii) find the most frequent class
- (iii) make the rule assign that class to this value

Calculate the error rate of the rule

Choose the rules with the smallest error rate

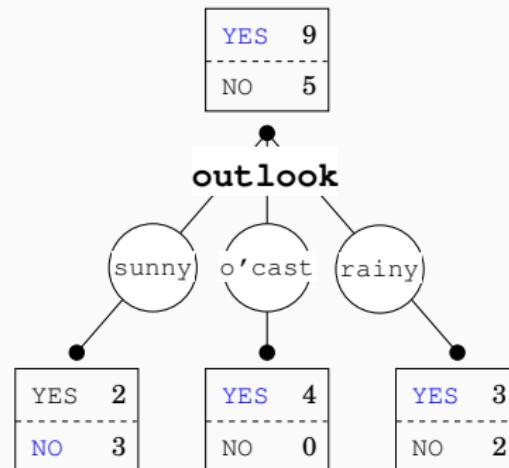


## Weather dataset

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no



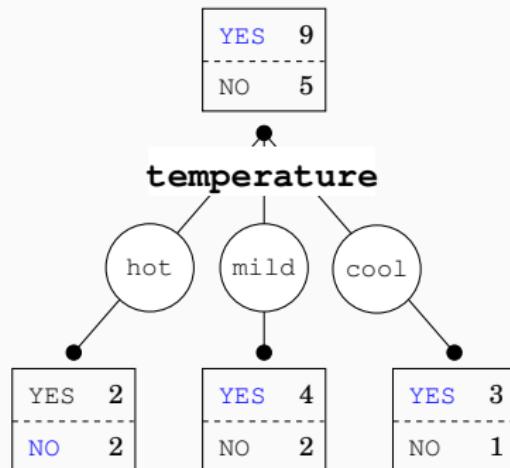
## Attribute Option 1 (outlook)



Total errors =  $\frac{4}{14}$



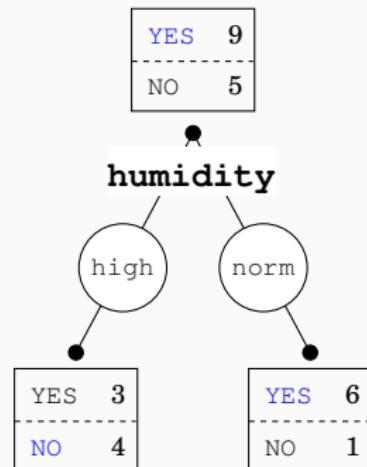
## Attribute Option 2 (temperature)



$$\text{Total errors} = \frac{5}{14}$$

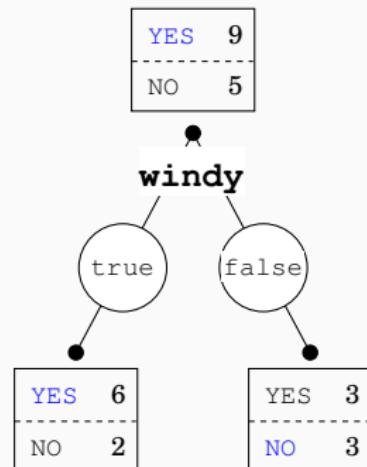


## Attribute Option 3 (humidity)



Total errors =  $\frac{4}{14}$

## Attribute Option 4 (windy)



$$\text{Total errors} = \frac{5}{14}$$

## One-R: Reflections

### Advantages:

- simple to understand and implement
- simple to comprehend the results
- surprisingly good results

### Disadvantages:

- unable to capture attribute interactions
- bias towards high-arity attributes (attributes with many possible values)



# Summary

## Today

- How do we set up an evaluation of a classification system?
- What are the measures we use to assess the performance of the classification system?
- What is a baseline? What are some examples of reasonable baselines to compare with?

## Next lecture

- Feature Selection



## References

Data Mining: Concepts and Techniques, 3rd ed., Jiawei Han and Micheline Kamber, Morgan Kaufmann, 2006. Chapter 8.5

Chris Bishop. Pattern Recognition and Machine Learning. Chapters: 1.3

Addison Wesley, 2006. David Wolpert and William Macready. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1:6782, 1997.

Jacob Eisenstein. Natural Language Processing. Chapter 4.4



# Lecture 8: Feature Selection and Analysis

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Jeremy Nicholson, Tim Baldwin, Karin Verspoor & Lea Frermann

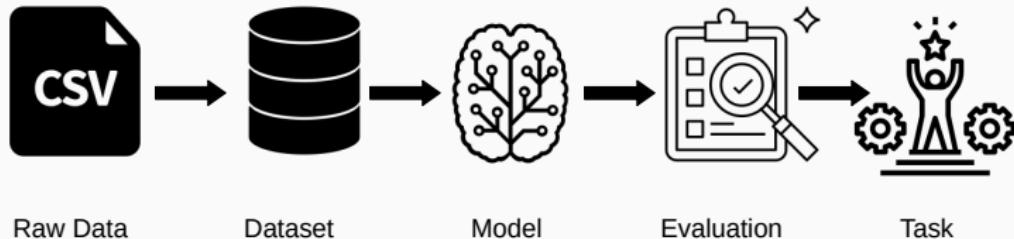


## **Features in Machine Learning**

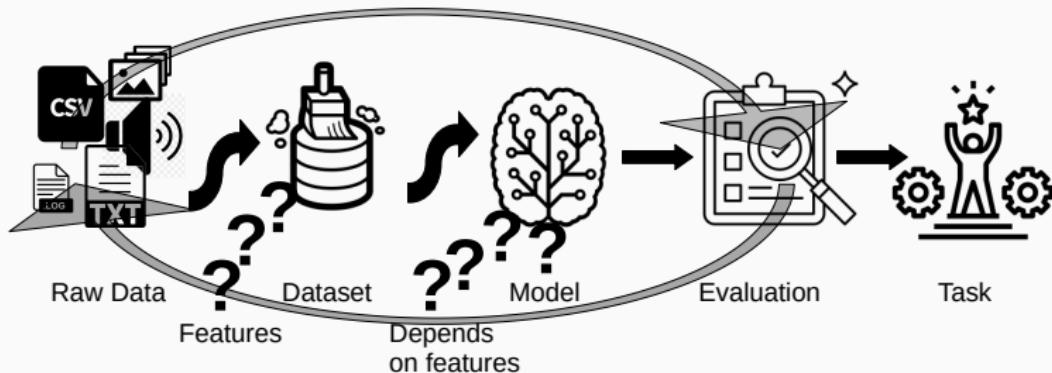
---

# Machine Learning Workflow

## The Dream



## Reality



# Data Preparation vs Feature Selection

**GIGO:** Garbage In, Garbage Out



**Data Preparation and Cleansing** (discussed before)

- Data Cleaning
- Data Aggregation
- Dealing with missing values
- Transformation (e.g., log transform)
- Binarization
- Binning
- Scaling or Normalization

**Feature Selection** (this lecture)

- Wrapper methods (aka recursive elimination)
- Filtering (aka univariate filtering)
- Glance into some other common approaches



# Data Preparation vs Feature Selection

## Our job as Machine Learning experts:

- Inspect / clean the data
- Choose a model suitable for classifying the data according to the attributes
- Choose attributes suitable for classifying the data according to the model
  - Inspection
  - Intuition



# Data Preparation vs Feature Selection

## Our job as Machine Learning experts:

- Inspect / clean the data
- Choose a model suitable for classifying the data according to the attributes
- Choose attributes suitable for classifying the data according to the model
  - Inspection
  - Intuition
  - Neither possible in practice



## **Feature Selection**

---

# What makes features good?

## Lead to better models

- Better performance according to some evaluation metric

## Side-goal 1

- Seeing important features can suggest other important features
- Tell us interesting things about the problem

## Side-goal 2

- Fewer features → smaller models → faster answer
  - More accurate answer >> faster answer



## **Iterative feature selection: Wrappers**

---

# Choosing a good feature set

## “Wrapper” methods

- Choose subset of attributes that give best performance on the development data
- For example: for the Weather data set:
  - Train model on {Outlook}
  - Train model on {Temperature}
  - ...
  - Train model on {Outlook, Temperature}
  - ...
  - Train model on {Outlook, Temperature, Humidity}
  - ...
  - Train model on {Outlook, Temperature, Humidity, Windy}



# Choosing a good feature set

## “Wrapper” methods

- Choose subset of attributes that give best performance on the development data
- For example: for the Weather data set:
  - Evaluate model on {Outlook}
  - Evaluate model on {Temperature}
  - ...
  - Evaluate model on {Outlook, Temperature}
  - ...
  - Evaluate model on {Outlook, Temperature, Humidity}
  - ...
  - Evaluate model on {Outlook, Temperature, Humidity, Windy}
- Best performance on data set → best feature set



THE UNIVERSITY OF  
MELBOURNE

# Choosing a good feature set

## “Wrapper” methods

- Choose subset of attributes that give best performance on the development data
- Advantages:
  - Feature set with **optimal** performance on development data
- Disadvantages:
  - Takes a **long** time



THE UNIVERSITY OF  
MELBOURNE

## Aside: how long does the full wrapper method take?

Assume we have a fast method (e.g. Naive Bayes) over a data set of non-trivial size ( $\sim 10K$  instances):

- Assume: train–evaluate cycle takes 10 sec to complete

How many cycles? For  $m$  features:

- $2^m$  subsets =  $\frac{2^m}{6}$  minutes
- $m = 10 \rightarrow 3$  hours
- $m = 60 \rightarrow$  heat death of universe

Only practical for very small data sets.



## More practical wrapper methods: Greedy search

### Greedy approach

- Train and evaluate model on each single attribute
- Choose best attribute
- Until convergence:
  - Train and evaluate model on best attribute(s), plus each remaining single attribute
  - Choose best attribute out of the remaining set
- Iterate until performance (e.g. accuracy) stops increasing



# More practical wrapper methods: Greedy search

## Greedy approach

- Bad news:
  - Takes  $\frac{1}{2}m^2$  cycles, for  $m$  attributes
  - In theory, 386 attributes → days
- Good news:
  - In practice, converges much more quickly than this
- Bad news again:
  - Converges to a sub-optimal (and often very bad) solution



# More practical wrapper methods: Ablation

## “Ablation” approach

- Start with all attributes
- Remove one attribute, train and evaluate model
- Until divergence:
  - From remaining attributes, remove each attribute, train and evaluate model
  - Remove attribute that causes least performance degradation
- Termination condition usually: performance (e.g. accuracy) starts to degrade by more than  $\epsilon$



# More practical wrapper methods: Ablation

## “Ablation” approach

for example:

- Start with all features
  - Train, evaluate model on {Outlook, Temperature, Humidity, Windy}
- Consider feature subsets of size 3:
  - Train, evaluate model on {Outlook, Temperature, Humidity}
  - Train, evaluate model on {Outlook, Temperature, Windy}
  - Train, evaluate model on {Outlook, Humidity, Windy}
  - Train, evaluate model on {Temperature, Humidity, Windy}
- Choose best of previous five (let's say THW):
- Consider feature subsets of size 2:
  - Train, evaluate model on {Temperature, Humidity}
  - Train, evaluate model on {Temperature, Windy}
  - Train, evaluate model on {Humidity, Windy}
- etc...



# More practical wrapper methods: Ablation

## “Ablation” approach

- Good news:
  - Mostly removes irrelevant attributes (at the start)
- Bad news:
  - Assumes independence of attributes  
(Actually, both approaches do this)
  - Takes  $O(m^2)$  time; cycles are slower with more attributes
  - Not feasible on non-trivial data sets.



## **Feature Filtering**

---

**Intuition:** Evaluate the “goodness” of each feature, separate from other features

- Consider each feature separately: linear time in number of attributes
- Possible (but difficult) to control for inter-dependence of features
- Typically most popular strategy



## Feature “goodness”

What makes a feature set single feature good?



## Toy example

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N

Which of  $a_1$ ,  $a_2$  is good?



## Toy example

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N



## Toy example

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N



## Pointwise Mutual Information

Discrepancy between the **observed joint probability** of two random variables  $A$  and  $C$  and the expected joint probability **if  $A$  and  $C$  were independent.**

Recall independence:  $P(C|A) = P(C)$



## Pointwise Mutual Information

Discrepancy between the **observed joint probability** of two random variables  $A$  and  $C$  and the expected joint probability **if  $A$  and  $C$  were independent.**

Recall independence:  $P(C|A) = P(C)$

**PMI** is defined as

$$PMI(A, C) = \log_2 \frac{P(A, C)}{P(A)P(C)}$$

We want to find attributes that are **not** independent of the class.

- If  $PMI >> 0$ , attribute and class occur together much more often than randomly.
- If  $RHS \sim 0$ , attribute and class occur together as often as we would expect from random chance
- If  $RHS << 0$ , attribute and class are negatively correlated.  
(More on that later!)

**Attributes with greatest PMI: best attributes**



## Toy example, revisited

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N

Calculate PMI of  $a_1, a_2$  with respect to  $c$



## Toy example, revisited

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N

$$P(a_1) =$$

$$P(c) =$$

$$P(a_1, c) =$$

$$PMI(a_1, c) =$$



## Toy example, revisited

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N

$$P(a_1) =$$

$$P(c) =$$

$$P(a_1, c) =$$

$$PMI(a_1, c) =$$



## Toy example, revisited

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N

$$P(a_2) = \frac{2}{4}$$

$$P(c) = \frac{2}{4}$$

$$P(a_2, c) = \frac{1}{4}$$



## Toy example, revisited

$a_1$	$a_2$	$c$
Y	Y	Y
Y	N	Y
N	Y	N
N	N	N

$$P(a_2) = \frac{2}{4}$$

$$P(c) = \frac{2}{4}$$

$$P(a_2, c) = \frac{1}{4}$$

$$\begin{aligned} PMI(a_2, c) &= \log_2 \frac{\frac{1}{4}}{\frac{1}{2} \cdot \frac{1}{2}} \\ &= \log_2(1) = 0 \end{aligned}$$



# Feature “goodness”, revisited

## What makes a single feature good?

- Well correlated with class
  - Knowing  $a$  lets us predict  $c$  with more confidence
- Reverse correlated with class
  - Knowing  $\bar{a}$  lets us predict  $c$  with more confidence
- Well correlated (or reverse correlated) with not class
  - Knowing  $a$  lets us predict  $\bar{c}$  with more confidence
  - Usually not quite as good, but still useful



## Mutual Information

- Expected value of PMI over all possible events
- For our example: Combine PMI of all possible combinations:  $a, \bar{a}, c, \bar{c}$



## Aside: Contingency tables

**Contingency tables:** compact representation of these frequency counts

	$a$	$\bar{a}$	Total
$c$	$\sigma(a, c)$	$\sigma(\bar{a}, c)$	$\sigma(c)$
$\bar{c}$	$\sigma(a, \bar{c})$	$\sigma(\bar{a}, \bar{c})$	$\sigma(\bar{c})$
Total	$\sigma(a)$	$\sigma(\bar{a})$	$N$

$$P(a, c) = \frac{\sigma(a, c)}{N}, \text{ etc.}$$



## Aside: Contingency tables

Contingency tables for toy example:

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	2	0	2
$c = N$	0	2	2
Total	2	2	4

$a_2$	$a = Y$	$a = N$	Total
$c = Y$	1	1	2
$c = N$	1	1	2
Total	2	2	4



# Mutual Information

Combine PMI of all possible combinations:  $a, \bar{a}, c, \bar{c}$

$$MI(A, C) = P(a, c)PMI(a, c) + P(\bar{a}, c)PMI(\bar{a}, c) + \\ P(a, \bar{c})PMI(a, \bar{c}) + P(\bar{a}, \bar{c})PMI(\bar{a}, \bar{c})$$

$$MI(A, C) = P(a, c) \log_2 \frac{P(a, c)}{P(a)P(c)} + P(\bar{a}, c) \log_2 \frac{P(\bar{a}, c)}{P(\bar{a})P(c)} + \\ P(a, \bar{c}) \log_2 \frac{P(a, \bar{c})}{P(a)P(\bar{c})} + P(\bar{a}, \bar{c}) \log_2 \frac{P(\bar{a}, \bar{c})}{P(\bar{a})P(\bar{c})}$$



## Mutual Information

Combine PMI of all possible combinations:  $a, \bar{a}, c, \bar{c}$

$$MI(A, C) = P(a, c)PMI(a, c) + P(\bar{a}, c)PMI(\bar{a}, c) + \\ P(a, \bar{c})PMI(a, \bar{c}) + P(\bar{a}, \bar{c})PMI(\bar{a}, \bar{c})$$

$$MI(A, C) = P(a, c) \log_2 \frac{P(a, c)}{P(a)P(c)} + P(\bar{a}, c) \log_2 \frac{P(\bar{a}, c)}{P(\bar{a})P(c)} + \\ P(a, \bar{c}) \log_2 \frac{P(a, \bar{c})}{P(a)P(\bar{c})} + P(\bar{a}, \bar{c}) \log_2 \frac{P(\bar{a}, \bar{c})}{P(\bar{a})P(\bar{c})}$$

Often written more compactly as:

$$MI(A, C) = \sum_{i \in \{a, \bar{a}\}} \sum_{j \in \{c, \bar{c}\}} P(i, j) \log_2 \frac{P(i, j)}{P(i)P(j)}$$

We define that  $0 \log 0 \equiv 0$ .



# Mutual Information Example

## Contingency Table for attribute $a_1$

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	2	0	2
$c = N$	0	2	2
Total	2	2	4



# Mutual Information Example

## Contingency Table for attribute $a_1$

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	2	0	2
$c = N$	0	2	2
Total	2	2	4

$$P(a, c) = \frac{2}{4}; \quad P(a) = \frac{2}{4}; \quad P(c) = \frac{2}{4}; \quad P(a, \bar{c}) = 0$$
$$P(\bar{a}, \bar{c}) = \frac{2}{4}; \quad P(\bar{a}) = \frac{2}{4}; \quad P(\bar{c}) = \frac{2}{4}; \quad P(\bar{a}, c) = 0$$



# Mutual Information Example

## Contingency Table for attribute $a_1$

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	2	0	2
$c = N$	0	2	2
Total	2	2	4

$$P(a, c) = \frac{2}{4}; \quad P(a) = \frac{2}{4}; \quad P(c) = \frac{2}{4}; \quad P(a, \bar{c}) = 0$$

$$P(\bar{a}, \bar{c}) = \frac{2}{4}; \quad P(\bar{a}) = \frac{2}{4}; \quad P(\bar{c}) = \frac{2}{4}; \quad P(\bar{a}, c) = 0$$

$$\begin{aligned} MI(A_1, C) &= P(a_1, c) \log_2 \frac{P(a_1, c)}{P(a_1)P(c)} + P(\bar{a}_1, c) \log_2 \frac{P(\bar{a}_1, c)}{P(\bar{a}_1)P(c)} + \\ &\quad P(a_1, \bar{c}) \log_2 \frac{P(a_1, \bar{c})}{P(a_1)P(\bar{c})} + P(\bar{a}_1, \bar{c}) \log_2 \frac{P(\bar{a}_1, \bar{c})}{P(\bar{a}_1)P(\bar{c})} \end{aligned}$$



# Mutual Information Example

## Contingency Table for attribute $a_1$

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	2	0	2
$c = N$	0	2	2
Total	2	2	4

$$P(a, c) = \frac{2}{4}; \quad P(a) = \frac{2}{4}; \quad P(c) = \frac{2}{4}; \quad P(a, \bar{c}) = 0$$

$$P(\bar{a}, \bar{c}) = \frac{2}{4}; \quad P(\bar{a}) = \frac{2}{4}; \quad P(\bar{c}) = \frac{2}{4}; \quad P(\bar{a}, c) = 0$$

$$\begin{aligned} MI(A_1, C) &= P(a_1, c) \log_2 \frac{P(a_1, c)}{P(a_1)P(c)} + P(\bar{a}_1, c) \log_2 \frac{P(\bar{a}_1, c)}{P(\bar{a}_1)P(c)} + \\ &\quad P(a_1, \bar{c}) \log_2 \frac{P(a_1, \bar{c})}{P(a_1)P(\bar{c})} + P(\bar{a}_1, \bar{c}) \log_2 \frac{P(\bar{a}_1, \bar{c})}{P(\bar{a}_1)P(\bar{c})} \\ &= \frac{1}{2} \log_2 \frac{\frac{1}{2}}{\frac{1}{2} \frac{1}{2}} + 0 \log_2 \frac{0}{\frac{1}{2} \frac{1}{2}} + 0 \log_2 \frac{0}{\frac{1}{2} \frac{1}{2}} + \frac{1}{2} \log_2 \frac{\frac{1}{2}}{\frac{1}{2} \frac{1}{2}} \\ &= \frac{1}{2}(1) + 0 + 0 + \frac{1}{2}(1) = 1 \end{aligned}$$



## Mutual Information Example continued

Contingency Table for attribute  $a_2$

$a_2$	$a = Y$	$a = N$	Total
$c = Y$	1	1	2
$c = N$	1	1	2
Total	2	2	4



## Mutual Information Example continued

### Contingency Table for attribute $a_2$

$a_2$	$a = Y$	$a = N$	Total
$c = Y$	1	1	2
$c = N$	1	1	2
Total	2	2	4

$$P(a, c) = \frac{1}{4}; \quad P(a) = \frac{2}{4}; \quad P(c) = \frac{2}{4}; \quad P(\bar{a}, c) = \frac{1}{4}$$
$$P(\bar{a}, \bar{c}) = \frac{1}{4}; \quad P(\bar{a}) = \frac{2}{4}; \quad P(\bar{c}) = \frac{2}{4}; \quad P(a, \bar{c}) = \frac{1}{4}$$



## Mutual Information Example continued

### Contingency Table for attribute $a_2$

$a_2$	$a = Y$	$a = N$	Total
$c = Y$	1	1	2
$c = N$	1	1	2
Total	2	2	4

$$P(a, c) = \frac{1}{4}; \quad P(a) = \frac{2}{4}; \quad P(c) = \frac{2}{4}; \quad P(\bar{a}, c) = \frac{1}{4}$$

$$P(\bar{a}, \bar{c}) = \frac{1}{4}; \quad P(\bar{a}) = \frac{2}{4}; \quad P(\bar{c}) = \frac{2}{4}; \quad P(a, \bar{c}) = \frac{1}{4}$$

$$\begin{aligned} MI(A_2, C) &= P(a_2, c) \log_2 \frac{P(a_2, c)}{P(a_2)P(c)} + P(\bar{a}_2, c) \log_2 \frac{P(\bar{a}_2, c)}{P(\bar{a}_2)P(c)} + \\ &\quad P(a_2, \bar{c}) \log_2 \frac{P(a_2, \bar{c})}{P(a_2)P(\bar{c})} + P(\bar{a}_2, \bar{c}) \log_2 \frac{P(\bar{a}_2, \bar{c})}{P(\bar{a}_2)P(\bar{c})} \end{aligned}$$



## Mutual Information Example continued

### Contingency Table for attribute $a_2$

$a_2$	$a = Y$	$a = N$	Total
$c = Y$	1	1	2
$c = N$	1	1	2
Total	2	2	4

$$P(a, c) = \frac{1}{4}; \quad P(a) = \frac{2}{4}; \quad P(c) = \frac{2}{4}; \quad P(\bar{a}, c) = \frac{1}{4}$$

$$P(\bar{a}, \bar{c}) = \frac{1}{4}; \quad P(\bar{a}) = \frac{2}{4}; \quad P(\bar{c}) = \frac{2}{4}; \quad P(a, \bar{c}) = \frac{1}{4}$$

$$\begin{aligned} MI(A_2, C) &= P(a_2, c) \log_2 \frac{P(a_2, c)}{P(a_2)P(c)} + P(\bar{a}_2, c) \log_2 \frac{P(\bar{a}_2, c)}{P(\bar{a}_2)P(c)} + \\ &\quad P(a_2, \bar{c}) \log_2 \frac{P(a_2, \bar{c})}{P(a_2)P(\bar{c})} + P(\bar{a}_2, \bar{c}) \log_2 \frac{P(\bar{a}_2, \bar{c})}{P(\bar{a}_2)P(\bar{c})} \\ &= \frac{1}{4} \log_2 \frac{\frac{1}{4}}{\frac{1}{2} \frac{1}{2}} + \frac{1}{4} \log_2 \frac{\frac{1}{4}}{\frac{1}{2} \frac{1}{2}} + \frac{1}{4} \log_2 \frac{\frac{1}{4}}{\frac{1}{2} \frac{1}{2}} + \frac{1}{4} \log_2 \frac{\frac{1}{4}}{\frac{1}{2} \frac{1}{2}} \\ &= \frac{1}{4}(0) + \frac{1}{4}(0) + \frac{1}{4}(0) + \frac{1}{4}(0) = 0 \end{aligned}$$



# Chi-square

Similar idea, different solution:

	$a$	$\bar{a}$	Total
$c$	$\sigma(a, c)$	$\sigma(\bar{a}, c)$	$\sigma(c)$
$\bar{c}$	$\sigma(a, \bar{c})$	$\sigma(\bar{a}, \bar{c})$	$\sigma(\bar{c})$
Total	$\sigma(a)$	$\sigma(\bar{a})$	$N$

Contingency table (shorthand):

	$a$	$\bar{a}$	Total
$c$	$W$	$X$	$W + X$
$\bar{c}$	$Y$	$Z$	$Y + Z$
Total	$W + Y$	$X + Z$	$N = W + X + Y + Z$

If  $a, c$  were independent (uncorrelated), what value would you expect in  $W$ ?

Denote the expected value as  $E(W)$ .



## Chi-square

If  $a, c$  were independent, then  $P(a, c) = P(a)P(c)$

$$P(a, c) = P(a)P(c)$$

$$\frac{\sigma(a, c)}{N} = \frac{\sigma(a)}{N} \frac{\sigma(c)}{N}$$

$$\sigma(a, c) = \frac{\sigma(a)\sigma(c)}{N}$$

$$E(W) = \frac{(W + Y)(W + X)}{W + X + Y + Z}$$



Compare the value we actually observed  $O(W)$  with the expected value  $E(W)$ :

- If the **observed value is much greater than the expected value**,  $a$  occurs more often with  $c$  than we would expect at random — **predictive**
- If the observed value is **much smaller than the expected value**,  $a$  occurs less often with  $c$  than we would expect at random — **predictive**
- If the **observed value is close to the expected value**,  $a$  occurs as often with  $c$  as we would expect randomly — **not predictive**

Similarly with  $X$ ,  $Y$ ,  $Z$



## Chi-square

### Actual calculation (to fit to a chi-square distribution)

$$\begin{aligned}\chi^2 &= \frac{(O(W) - E(W))^2}{E(W)} + \frac{(O(X) - E(X))^2}{E(X)} + \\ &\quad \frac{(O(Y) - E(Y))^2}{E(Y)} + \frac{(O(Z) - E(Z))^2}{E(Z)} \\ &= \sum_{i=1}^r \sum_{j=1}^c \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}\end{aligned}$$

- $i$  sums over rows and  $j$  sums over columns.
- Because the values are squared,  $\chi^2$  becomes much greater when  $|O - E|$  is large, even if  $E$  is also large.



## Chi-square Example

Contingency table for toy example (observed values):

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	2	0	2
$c = N$	0	2	2
Total	2	2	4

Contingency table for toy example (expected values):

$a_1$	$a = Y$	$a = N$	Total
$c = Y$	1	1	2
$c = N$	1	1	2
Total	2	2	4



## Chi-square Example

$$\begin{aligned}\chi^2(A_1, C) &= \frac{(O_{a,c} - E_{a,c})^2}{E_{a,c}} + \frac{(O_{\bar{a},c} - E_{\bar{a},c})^2}{E_{\bar{a},c}} + \\&\quad \frac{(O_{a,\bar{c}} - E_{a,\bar{c}})^2}{E_{a,\bar{c}}} + \frac{(O_{\bar{a},\bar{c}} - E_{\bar{a},\bar{c}})^2}{E_{\bar{a},\bar{c}}} \\&= \frac{(2-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(2-1)^2}{1} \\&= 1 + 1 + 1 + 1 = 4\end{aligned}$$

$\chi^2(A_2, C)$  is obviously 0, because all observed values are equal to expected values.



## **Common Issues**

---

## Types of Attribute

So far, we've only looked at binary (Y/N) attributes:

- Nominal attributes
- Continuous attributes
- Ordinal attributes



# Types of Attributes: Nominal

## Two common strategies

### 1. Treat as multiple binary attributes:

- e.g. sunny=Y, overcast=N, rainy=N, etc.
- Can just use the formulae as given
- Results sometimes difficult to interpret
  - For example, Outlook=sunny is useful, but Outlook=overcast and Outlook=rainy are not useful... Should we use Outlook?

### 2. Modify contingency tables (and formulae)

o	s	o	r
$c = Y$	$U$	$V$	$W$
$c = N$	$X$	$Y$	$Z$



## Types of Attributes: Nominal

### Modified MI:

$$\begin{aligned} MI(O, C) &= \sum_{i \in \{s, o, r\}} \sum_{j \in \{c, \bar{c}\}} P(i, j) \log_2 \frac{P(i, j)}{P(i)P(j)} \\ &= P(s, c) \log_2 \frac{P(s, c)}{P(s)P(c)} + P(s, \bar{c}) \log_2 \frac{P(s, \bar{c})}{P(s)P(\bar{c})} + \\ &\quad P(o, c) \log_2 \frac{P(o, c)}{P(o)P(c)} + P(o, \bar{c}) \log_2 \frac{P(o, \bar{c})}{P(o)P(\bar{c})} + \\ &\quad P(r, c) \log_2 \frac{P(r, c)}{P(r)P(c)} + P(r, \bar{c}) \log_2 \frac{P(r, \bar{c})}{P(r)P(\bar{c})} \end{aligned}$$

- Biased towards attributes with many values.



## Types of Attributes: Nominal

**Chi-square can be used as normal, with 6 observed/expected values.**

- To control for score inflation, we need to consider “number of degrees of freedom”, and then use the significance test explicitly (beyond the scope of this subject)



# Types of Attributes: Continuous

## Continuous attributes

- Usually dealt with by estimating probability based on a Gaussian (normal) distribution
- With a large number of values, most random variables are normally distributed due to the **Central Limit Theorem**
- For small data sets or pathological features, we may need to use binomial/multinomial distributions

All of this is beyond the scope of this subject



## Types of Attributes: Ordinal

**Three possibilities**, roughly in order of popularity:

1. Treat as binary
  - Particularly appropriate for frequency counts where events are low-frequency (e.g. words in tweets)
2. Treat as continuous
  - The fact that we haven't *seen* any intermediate values is usually not important
  - Does have all of the technical downsides of continuous attributes, however
3. Treat as nominal (i.e. throw away ordering)



## Multi-class problems

So far, we've only looked at binary (Y/N) classification tasks.

Multiclass (e.g. LA, NY, C, At, SF) classification tasks are usually much more difficult.



## Multi-class problems

### What makes a single feature good?

- Highly correlated with class
- Highly reverse correlated with class
- Highly correlated (or reverse correlated) with not class

... What if there are many classes?



## Multi-class problems

### What makes a single feature good?

- Highly correlated with class
- Highly reverse correlated with class
- Highly correlated (or reverse correlated) with not class

... What if there are many classes?

### What makes a feature bad?

- Irrelevant
- Correlated with other features
- Good at only predicting one class (but is this truly bad?)



## Multi-class problems

Consider multi-class problem over LA, NY, C, At, SF:

- PMI, MI,  $\chi^2$  are all calculated *per-class*
- (Some other feature selection metrics, e.g. Information Gain, work for all classes at once)
- Need to make a point of selecting (hopefully uncorrelated) features for *each* class to give our classifier the best chance of predicting everything correctly.



# Multi-class problems

Actual example (MI):

LA	NY	C	At	SF
la	nyc	chicago	atlanta	sf
angeles	york	bears	atl	<a href="http://dealnay.com">httpdealnaycom</a>
los	ny	il	ga	francisco
chicago	chicago	<a href="http://bitlyczmk">httpbitlyczmk</a>	lol	san
hollywood	atlanta	cubs	u	u
atlanta	yankees	la	georgia	lol
lakers	sf	chi	chicago	save



# Multi-class problems

Intuitive features:

LA	NY	C	At	SF
la	nyc	chicago	atlanta	sf
angelaes	york	bears	atl	httpdealnaycom
los	ny	il	ga	francisco
chicago	chicago	httpbitlyczmk	lol	san
hollywood	atlanta	cubs	u	u
atlanta	yankees	la	georgia	lol
lakers	sf	chi	chicago	save



# Multi-class problems

Features for predicting not class (MI):

LA	NY	C	At	SF
la	nyc	chicago	atlanta	sf
angeles	york	bears	atl	<a href="http://dealnay.com">httpdealnaycom</a>
los	ny	il	ga	francisco
<b>chicago</b>	<b>chicago</b>	http://bitlyczmk	lol	san
hollywood	<b>atlanta</b>	cubs	u	u
<b>atlanta</b>	yankees	<b>la</b>	georgia	lol
lakers	<b>sf</b>	chi	<b>chicago</b>	save



# Multi-class problems

## Unintuitive features:

LA	NY	C	At	SF
la	nyc	chicago	atlanta	sf
angeles	york	bears	atl	<b>httpdealnaycom</b>
los	ny	il	ga	francisco
chicago	chicago	<b>httpbitlyczmk</b>	<b>lol</b>	san
hollywood	atlanta	cubs	<b>u</b>	<b>u</b>
atlanta	yankees	la	georgia	<b>lol</b>
lakers	sf	chi	chicago	<b>save</b>



## What's going on with MI?

### Mutual Information is biased toward rare, uninformative features

- All probabilities: no notion of the raw frequency of events
- If a feature is seen rarely, but always with a given class, it will be seen as "good"
- Best features in the Twitter dataset only had MI of about 0.01 bits; 100<sup>th</sup> best for a given class had MI of about 0.002 bits



**Glance into a few other common  
approaches to feature selection**

---

## A common (unsupervised) alternative

### Term Frequency Inverse Document Frequency (TFIDF)

- Detect important words / Natural Language Processing
- Find words that are relevant to a document in a given document collection
- To be relevant, a word should be
  - Frequent enough in the corpus (TF). A word that occurs only 5 times in a corpus of 5,000,000 words is probably not too interesting
  - Special enough (IDF). A word that is very general and occurs in (almost) every document is probably not too interesting



## A common (unsupervised) alternative

### Term Frequency Inverse Document Frequency (TFIDF)

- Detect important words / Natural Language Processing
- Find words that are relevant to a document in a given document collection
- To be relevant, a word should be
  - Frequent enough in the corpus (TF). A word that occurs only 5 times in a corpus of 5,000,000 words is probably not too interesting
  - Special enough (IDF). A word that is very general and occurs in (almost) every document is probably not too interesting

$$tfidf(d, t, D) = tf + idf$$

$$tf = \log(1 + freq(t, d))$$

$$idf = \log\left(\frac{|D|}{count(d \in D : t \in d)}\right)$$

$d$ =document,  $t$ =term,  $D$ =document collection;

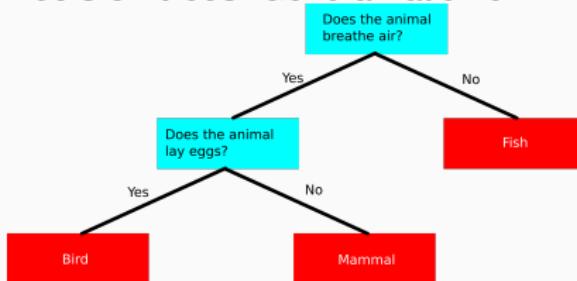
$|D|$ =number of documents in  $D$



# Embedded Methods:

Some ML models include feature selection inherently

## 1. Decision trees: Generalization of 1-R



## 2. Regression models with regularization

$$\text{house\_price} = \beta_0 + \beta_1 \times \text{size} + \beta_2 \times \text{location} + \beta_3 \times \text{age}$$

**Regularization** (or ‘penalty’) nudges the weight  $\beta$  of unimportant features towards zero

Image:

<https://towardsdatascience.com/a-beginners-guide-to-decision-tree-classification-6d3209353ea?gi=e0ee0b2b622e>



# And there are many more strategies

[https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature\\_selection](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection)

## sklearn.feature\_selection: Feature Selection

The `sklearn.feature_selection` module implements feature selection algorithms. It currently includes univariate filter selection methods and the recursive feature elimination algorithm.

**User guide:** See the [Feature selection](#) section for further details.

<code>feature_selection.GenericUnivariateSelect(...)</code>	Univariate feature selector with configurable strategy.
<code>feature_selection.SelectPercentile(...)</code>	Select features according to a percentile of the highest scores.
<code>feature_selection.SelectKBest([score_func, k])</code>	Select features according to the k highest scores.
<code>feature_selection.SelectFpr([score_func, alpha])</code>	Filter: Select the pvalues below alpha based on a FPR test.
<code>feature_selection.SelectFdr([score_func, alpha])</code>	Filter: Select the p-values for an estimated false discovery rate
<code>feature_selection.SelectFromModel(estimator, *)</code>	Meta-transformer for selecting features based on importance weights.
<code>feature_selection.SelectFwe([score_func, alpha])</code>	Filter: Select the p-values corresponding to Family-wise error rate
<code>feature_selection.SequentialFeatureSelector(...)</code>	Transformer that performs Sequential Feature Selection.
<code>feature_selection.RFE(estimator, *, ...)</code>	Feature ranking with recursive feature elimination.
<code>feature_selection.RFECV(estimator, *, ...)</code>	Feature ranking with recursive feature elimination and cross-validated selection of the best number of features.
<code>feature_selection.VarianceThreshold([threshold])</code>	Feature selector that removes all low-variance features.
<code>feature_selection.chi2(X, y)</code>	Compute chi-squared stats between each non-negative feature and class.
<code>feature_selection.f_classif(X, y)</code>	Compute the ANOVA F-value for the provided sample.
<code>feature_selection.f_regression(X, y, *, [center])</code>	Univariate linear regression tests.
<code>feature_selection.mutual_info_classif(X, y, *)</code>	Estimate mutual information for a discrete target variable.
<code>feature_selection.mutual_info_regression(X, y, *)</code>	Estimate mutual information for a continuous target variable.



# So ... is feature selection worth it?

**Absolutely!**

- Even marginally relevant features usually a vast improvement on an unfiltered data set
- Some models **need** feature selection
  - k-Nearest Neighbors, hugely
  - Naive Bayes, to a lesser extent
- Machine learning experts (us!) need to think about the data!



# Summary

## Today

- Wrappers vs. Filters
- Popular filters: PMI, MI,  $\chi^2$ , how should we use them and what are the results going to look like
- Importance of feature selection for different methods (even though it sometimes isn't the solution we were hoping for)

## Next week(s):

- Logistic regression
- Perceptron and neural networks
- ...and their respective learning algorithms (iterative optimization)

The lectures will be pre-recorded, as sequences of shorter videos.  
We will have only one Q&A session on Thursdays.



## References

- Guyon, Isabelle, and Andre Elisseeff. 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research*. Vol 3, 1157–1182.
- Guyon, Isabelle, et al., eds. Feature extraction: foundations and applications. Vol. 207. Springer, 2008. Chapter 3.1, 3.2, 3.8, 4.1, 4.3, 4.7, 6.2–6.5
- Yang, Yiming and Jan Pedersen (1997). A Comparative Study on Feature Selection in Text Categorization.



# Lecture 9: Iterative Optimization with Gradient Descent

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



# Finding Optimal Points I

Finding the **parameters** that optimize a **target**

Ex1: Estimate the **study time** which leads to the **best grade** in COMP90049.

Ex2: Find the **shoe price** which leads to **maximum profit** of our shoe shop.

Ex3: Predicting **housing prices** from a **weighted** combination of house age and house location

Ex4: Find the **parameters  $\theta$**  of a spam classifier which lead to the **lowest error**

Ex5: Find the **parameters  $\theta$**  of a spam classifier which lead to the **highest data log likelihood**



## Recipe for finding Minima / Maxima

1. Define your function of interest  $f(x)$  (e.g., data log likelihood)
2. Compute its first derivative wrt its input  $x$
3. Set the derivative to zero
4. Solve for  $x$



# Closed-form vs Iterative Optimization

## Closed-form solutions

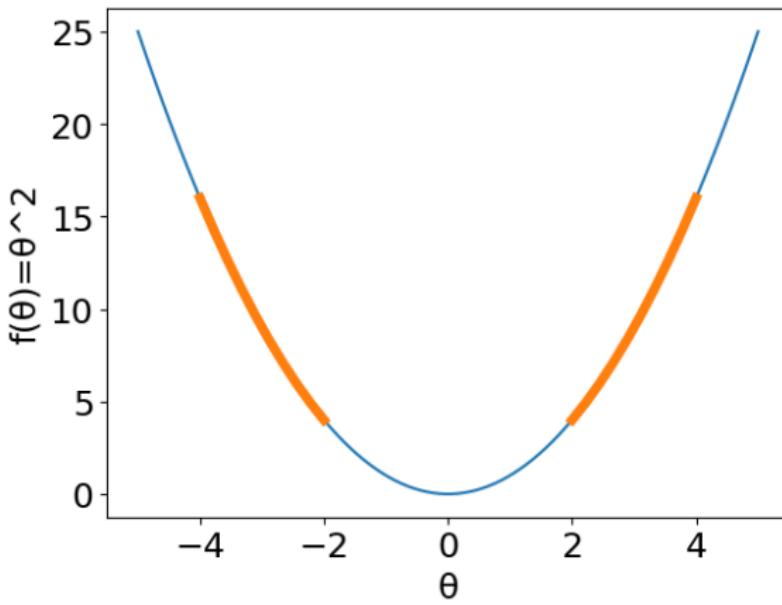
- Previously, we computed the **closed form** solution for the MLE of the binomial distribution
- We follow our recipe, and arrive at a single solution

## Unfortunately, life is not always as easy

- Often, no closed-form solution exists
- Instead, we have to **iteratively** improve our estimate of  $\hat{\theta}$  until we arrive at a satisfactory solution
- Gradient descent is one popular iterative optimization method



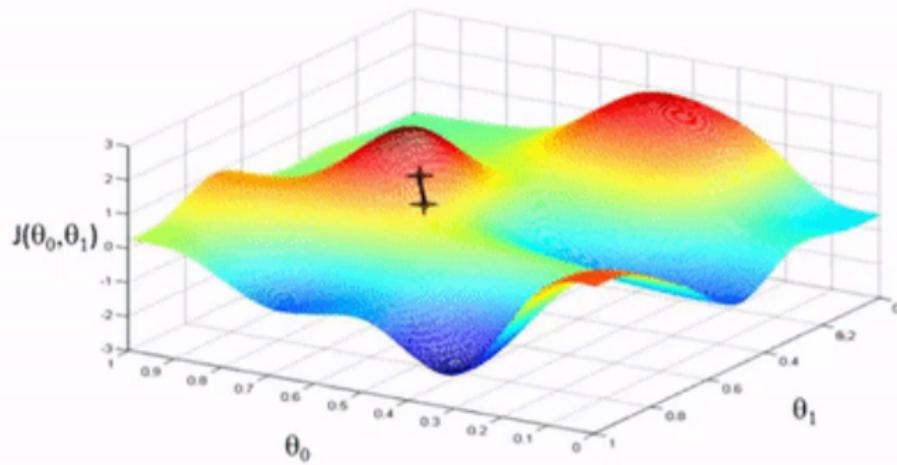
## 'Descending' the function to find the Optimum



- 1-dimensional case: find parameter  $\theta$  that minimizes the function
- follow the curvature of the line step by step



## 'Descending' the function to find the Optimum



- 2-dimensional case: find parameters  $\theta = [\theta_0, \theta_1]$  that minimize the function  $J$
- follow the curvature step by step along the steepest way

Andrew Ng



Source: <https://medium.com/binaryandmore/>

beginners-guide-to-deriving-and-implementing-backpropagation-e3c1a5a1e536

## Intuition

- Descending a mountain (aka. our function) as fast as possible: at every position take the next step that takes you most directly into the valley
- We compute a series of solutions  $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \dots$  by ‘walking’ along the function and taking steps in the direction with the steepest local slope (or gradient).
- each solution depends on the current location



## Gradient Descent: Details

### Learn the model parameters $\theta$

- such that we **minimize the error**
- traverse over the loss function step by step ('descending into a valley')
- we would like an algorithm that tells how to update our parameters

$$\theta \leftarrow \theta + \Delta\theta$$



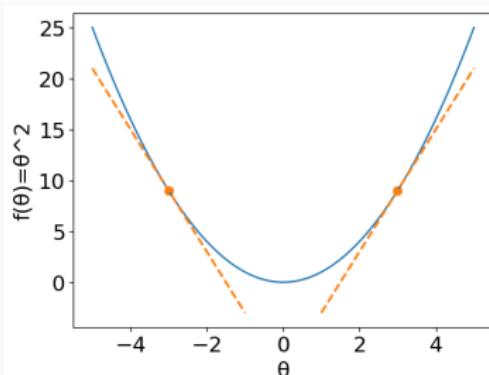
## Gradient Descent: Details

### Learn the model parameters $\theta$

- such that we **minimize the error**
- traverse over the loss function step by step ('descending into a valley')
- we would like an algorithm that tells how to update our parameters

$$\theta \leftarrow \theta + \Delta\theta$$

- $\Delta\theta$  is the **derivative**  $\frac{\partial f}{\partial \theta}$
- tells us how much  $f$  changes in response to a change in  $\theta$ .
- a measure of the **slope** or **gradient** of a function  $f$  at point  $\theta$
- the **gradient** points to the greatest **increase** of a function



# Gradient Descent: Details

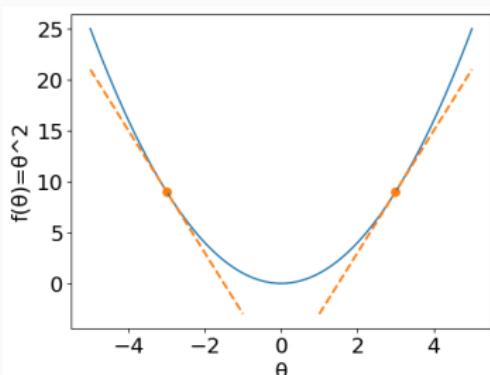
## Learn the model parameters $\theta$

- such that we **minimize the error**
- traverse over the loss function step by step ('descending into a valley')
- we would like an algorithm that tells how to update our parameters

$$\theta \leftarrow \theta + \Delta\theta$$

- if  $\frac{\partial f}{\partial \theta} > 0$ :  $f(\theta)$  :  $\nearrow$  as  $\theta$  :  $\nearrow$
- if  $\frac{\partial f}{\partial \theta} < 0$ :  $f(\theta)$  :  $\nearrow$  as  $\theta$  :  $\searrow$
- if  $\frac{\partial f}{\partial \theta} = 0$ : we are at a minimum
- so, to approach the minimum:

$$\theta \leftarrow \theta - \eta \frac{\partial f}{\partial \theta}$$



## Gradient Descent for multiple parameters

- Usually, our models have **several parameters** which need to be optimized to minimize the error
- We compute **partial derivatives** of  $f(\theta)$  wrt. individual  $\theta_i$
- Partial derivatives measure change in a function of multiple parameters given a change in a single parameter, with all others held constant
- For example for  $f(\theta_1, \theta_2)$  we can compute  $\frac{\partial f}{\partial \theta_1}$  and  $\frac{\partial f}{\partial \theta_2}$
- We then **update each parameter individually**

$$\theta_1 \leftarrow \theta_1 + \Delta \theta_1 \quad \text{with} \quad \Delta \theta_1 = -\eta \frac{\partial f}{\partial \theta_1}$$
$$\theta_2 \leftarrow \theta_2 + \Delta \theta_2 \quad \text{with} \quad \Delta \theta_2 = -\eta \frac{\partial f}{\partial \theta_2}$$



# Gradient Descent: Recipe

## Recipe for Gradient Descent (single parameter)

---

- 1: Define objective function  $f(\theta)$
- 2: Initialize parameter  $\theta^{(0)}$
- 3: **for** iteration  $t \in \{0, 1, 2, \dots T\}$  **do**
- 4:   Compute the first derivative of  $f$  at that point  $\theta^{(t)} : \frac{\partial f}{\partial \theta^{(t)}}$
- 5:   Update your parameter by subtracting the (scaled) derivative

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial f}{\partial \theta^{(t)}}$$

---

- $\eta$  is the **step size** or **learning rate**, a parameter
- When to stop? Fix number of iterations, or define other criteria



# Gradient Descent: Recipe

## Recipe for Gradient Descent (multiple parameters)

---

- 1: Define objective function  $f(\theta)$
- 2: Initialize parameters  $\{\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}, \dots\}$
- 3: **for** iteration  $t \in \{0, 1, 2, \dots T\}$  **do**
- 4:   Initialize vector of *gradients*  $\leftarrow []$
- 5:   **for** parameter  $f \in \{1, 2, 3, \dots F\}$  **do**
- 6:     Compute the first derivative of  $f$  at that point  $\theta_f^{(t)} : \frac{\partial f}{\partial \theta_f^{(t)}}$
- 7:     append  $\frac{\partial f}{\partial \theta_f^{(t)}}$  to *gradients*
- 8:   **Update all** parameters by subtracting the (scaled) gradient

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial f}{\partial \theta^{(t)}}$$

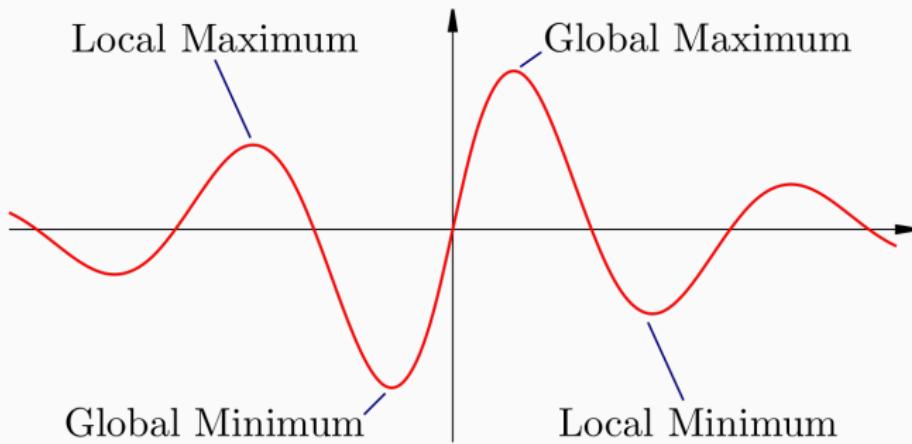
---



## Aside: Global and Local Minima and Maxima

Possible issue: local maxima and minima!

- A function is **convex** if a line between any two points of the function lies above the function
- A global **maximum** is the single highest value of the function
- A global **minimum** is the single lowest value of the function



# Gradient Descent Guarantees

- with an appropriate learning rate, GD will find the global minimum for differentiable convex functions
- with an appropriate learning rate, GD will find a local minimum for differentiable non-convex functions



# Summary

## Now you know:

- What optimization is, and why it's important
- How to do closed-form optimization (aka. "set the derivative of  $f(\theta)$  to zero and solve for  $\theta$ )
- That closed-form solutions are not always computable
- In that case, iterative optimization can help us
- Gradient descent is one instance of an iterative optimization method
- How gradient descent works!

## Next lecture(s)

- Logistic Regression
- The perceptron
- Neural networks



# Lecture 10: Logistic Regression

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



# Roadmap

## Sofar...

- Naive Bayes + KNN
- Optimization (closed-form and iterative)
- Evaluation + Feature Selection

**Today** : more classification!

- Logistic Regression



## **Logistic Regression**

---

# Quick Refresher

## Recall Naive Bayes

$$P(x, y) = P(y)P(x|y) = \prod_{i=1}^N P(y^i) \prod_{m=1}^M P(x_m^i | y^i)$$

- a **probabilistic generative model** of the joint probability  $P(x, y)$
- optimized to maximize the likelihood of the observed data
- **naive** due to unrealistic feature independence assumptions



## Recall Naive Bayes

$$P(x, y) = P(y)P(x|y) = \prod_{i=1}^N P(y^i) \prod_{m=1}^M P(x_m^i | y^i)$$

- a **probabilistic generative model** of the joint probability  $P(x, y)$
- optimized to maximize the likelihood of the observed data
- **naive** due to unrealistic feature independence assumptions

For **prediction**, we apply **Bayes Rule** to obtain the conditional distribution

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

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

$$\hat{y} = \operatorname{argmax}_y P(y|x) \approx P(y)P(x|y)$$

How about we model  $P(y|x)$  directly? → **Logistic Regression**



# Introduction to Logistic Regression

## Logistic Regression on a high level

- Is a **binary** classification model
- Is a **probabilistic discriminative model** because it optimizes  $P(y|x)$  directly
- Learns to optimally discriminate between inputs which belong to different classes
- No model of  $P(x|y) \rightarrow$  no conditional feature independence assumption



## Aside: Linear Regression

- Regression: predict a real-valued quantity  $y$  given features  $x$ , e.g.,

housing price            given    {location, size, age, ...}

success of movie (\$)    given    {cast, genre, budget, ...}

air quality              given    {temperature, timeOfDay, CO2, ...}



## Aside: Linear Regression

- Regression: predict a real-valued quantity  $y$  given features  $x$ , e.g.,

housing price            given    {location, size, age, ...}

success of movie (\$)    given    {cast, genre, budget, ...}

air quality              given    {temperature, timeOfDay, CO2, ...}

- **linear regression** is the simplest regression model
- a real-valued  $\hat{y}$  is predicted as a linear combination of weighted feature values

$$\begin{aligned}\hat{y} &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \\ &= \theta_0 + \sum_i \theta_i x_i\end{aligned}$$

- The weights  $\theta_0, \theta_1, \dots$  are model parameters, and need to be optimized during training
- Loss (error) is the sum of squared errors (SSE):  $L = \sum_{i=1}^N (\hat{y}^i - y^i)^2$



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach
- How about:  $p(x)$  as a linear function of  $x$ . Problem: probabilities are bounded in 0 and 1, linear functions are not.

$$p(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_F x_F$$



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach
- How about:  $p(x)$  as a linear function of  $x$ . Problem: probabilities are bounded in 0 and 1, linear functions are not.



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach
- How about:  $p(x)$  as a linear function of  $x$ . Problem: probabilities are bounded in 0 and 1, linear functions are not.
- How about:  $\log p(x)$  as a linear function of  $x$ . Problem:  $\log$  is bounded in one direction, linear functions are not.

$$\log p(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_F x_F$$



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach
- How about:  $p(x)$  as a linear function of  $x$ . Problem: probabilities are bounded in 0 and 1, linear functions are not.
- How about:  $\log p(x)$  as a linear function of  $x$ . Problem:  $\log$  is bounded in one direction, linear functions are not.



## Logistic Regression: Derivation I

- Let's assume a **binary** classification task,  $y$  is true (1) or false (0).
- We model **probabilities**  $P(y = 1|x; \theta) = p(x)$  as a function of observations  $x$  under parameters  $\theta$ . [ What about  $P(y = 0|x; \theta)$ ? ]
- We want to use a **regression** approach
- How about:  $p(x)$  as a linear function of  $x$ . Problem: probabilities are bounded in 0 and 1, linear functions are not.
- How about:  $\log p(x)$  as a linear function of  $x$ . Problem:  $\log$  is bounded in one direction, linear functions are not.
- How about: minimally modifying  $\log p(x)$  such that it is unbounded, by applying the **logistic** transformation

$$\log \frac{p(x)}{1 - p(x)} = \theta_0 + \theta_1 x_1 + \dots + \theta_F x_F$$



## Logistic Regression: Derivation II

$$\log \frac{p(x)}{1 - p(x)} = \theta_0 + \theta_1 x_1 + \dots + \theta_F x_F$$

- also called the **log odds**
- the **odds** are defined as the fraction of success over the fraction of failures

$$odds = \frac{P(\text{success})}{P(\text{failures})} = \frac{P(\text{success})}{1 - P(\text{success})}$$

- e.g., the odds of rolling a 6 with a fair dice are:

$$\frac{1/6}{1 - (1/6)} = \frac{0.17}{0.83} = 0.2$$



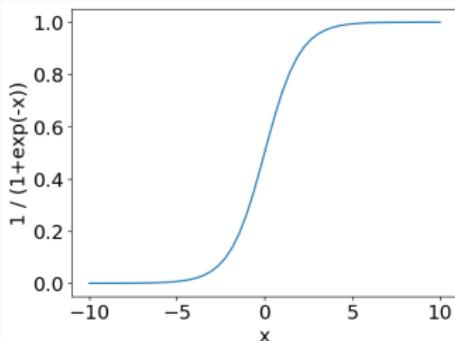
## Logistic Regression: Derivation III

$$\log \frac{P(x)}{1 - P(x)} = \theta_0 + \theta_1 x_1 + \dots + \theta_F x_F$$

If we rearrange and solve for  $P(x)$ , we get

$$\begin{aligned} P(x) &= \frac{\exp(\theta_0 + \theta_1 x_1 + \dots + \theta_F x_F)}{1 + \exp(\theta_0 + \theta_1 x_1 + \dots + \theta_F x_F)} = \frac{\exp(\theta_0 + \sum_{f=1}^F \theta_f x_f)}{1 + \exp(\theta_0 + \sum_{f=1}^F \theta_f x_f)} \\ &= \frac{1}{1 + \exp(-(\theta_0 + \theta_1 x_1 + \dots + \theta_F x_F))} = \frac{1}{1 + \exp(-(\theta_0 + \sum_{f=1}^F \theta_f x_f))} \end{aligned}$$

- where the RHS is the inverse logit (or **logistic function**)
- we pass a regression model through the logistic function to obtain a valid probability prediction

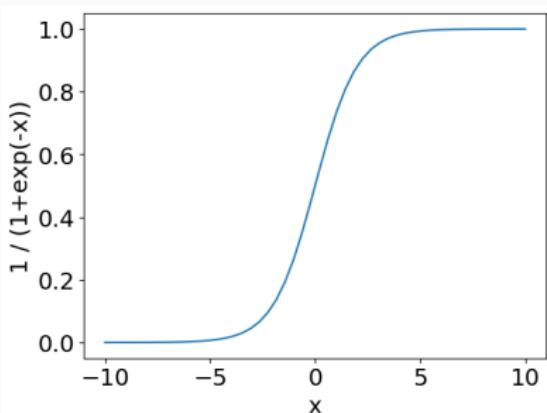


# Logistic Regression: Interpretation

$$P(y|x; \theta) = \frac{1}{1 + \exp(-(\theta_0 + \sum_{f=1}^F \theta_f x_f))}$$

## A closer look at the logistic function

Most inputs lead to  $P(y|x)=0$  or  $P(y|x)=1$ . That is intended, because all true labels are either 0 or 1.



- $(\theta_0 + \sum_{f=1}^F \theta_f x_f) > 0$  means  
 $y = 1$
- $(\theta_0 + \sum_{f=1}^F \theta_f x_f) \approx 0$  means  
most uncertainty
- $(\theta_0 + \sum_{f=1}^F \theta_f x_f) < 0$  means  
 $y = 0$



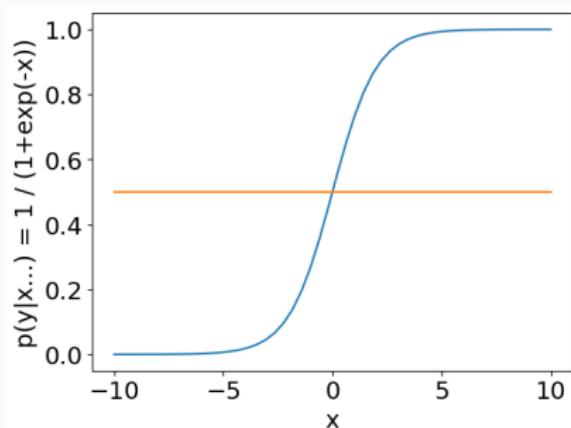
## Logistic Regression: Prediction

- The logistic function returns the probability of  $P(y = 1)$  given an input  $x$

$$P(y = 1 | x_1, x_2, \dots, x_F; \theta) = \frac{1}{1 + \exp(-(\theta_0 + \sum_{f=1}^F \theta_f x_f))} = \sigma(x; \theta)$$

- We define a **decision boundary**, e.g., predict  $y = 1$  if

$P(y = 1 | x_1, x_2, \dots, x_F; \theta) > 0.5$  and  $y = 0$  otherwise



# Example!

$$P(y = 1 | x_1, x_2, \dots, x_F; \theta) = \frac{1}{1 + \exp(-(\theta_0 + \sum_{f=1}^F \theta_f x_f))} = \frac{1}{1 + \exp(-(\theta^T x))} = \sigma(\theta^T x)$$

## Model parameters

$$\theta = [0.1, -3.5, 0.7, 2.1]$$

## (Small) Test Data set

Outlook	Temp	Humidity	Class
<i>rainy</i>	<i>cool</i>	<i>normal</i>	0
<i>sunny</i>	<i>hot</i>	<i>high</i>	1

## Feature Function

$$x_0 = 1 \text{ (bias term)}$$

$$x_1 = \begin{cases} 1 & \text{if outlook=sunny} \\ 2 & \text{if outlook=overcast} \\ 3 & \text{if outlook=rainy} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if temp=hot} \\ 2 & \text{if temp=mild} \\ 3 & \text{if temp=cool} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if humidity=normal} \\ 2 & \text{if humidity=high} \end{cases}$$



# Example!

$$P(y = 1 | x_1, x_2, \dots, x_F; \theta) = \frac{1}{1 + \exp(-(\theta_0 + \sum_{f=1}^F \theta_f x_f))} = \frac{1}{1 + \exp(-(\theta^T x))} = \sigma(\theta^T x)$$

## Model parameters

$$\theta = [0.1, -3.5, 0.7, 2.1]$$

## (Small) Test Data set

Outlook	Temp	Humidity	Class
<i>rainy</i>	<i>cool</i>	<i>normal</i>	0
<i>sunny</i>	<i>hot</i>	<i>high</i>	1

## Feature Function

$$x_0 = 1 \text{ (bias term)}$$

$$x_1 = \begin{cases} 1 & \text{if outlook=sunny} \\ 2 & \text{if outlook=overcast} \\ 3 & \text{if outlook=rainy} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if temp=hot} \\ 2 & \text{if temp=mild} \\ 3 & \text{if temp=cool} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if humidity=normal} \\ 2 & \text{if humidity=high} \end{cases}$$



# Parameter Estimation

What are the four steps we would follow in finding the optimal parameters?



# Objective Function

Mimimize the Negative conditional log likelihood

$$\mathcal{L}(\theta) = -P(Y|X; \theta) = -\prod_{i=1}^N P(y^i|x^i; \theta)$$

note that

$$P(y = 1|x; \theta) = \sigma(\theta^T x)$$

$$P(y = 0|x; \theta) = 1 - \sigma(\theta^T x)$$



# Objective Function

Mimimize the Negative conditional log likelihood

$$\mathcal{L}(\theta) = -P(Y|X; \theta) = -\prod_{i=1}^N P(y^i|x^i; \theta)$$

note that

$$P(y = 1|x; \theta) = \sigma(\theta^T x)$$

$$P(y = 0|x; \theta) = 1 - \sigma(\theta^T x)$$

so

$$\begin{aligned}\mathcal{L}(\theta) &= -P(Y|X; \theta) = -\prod_{i=1}^N P(y^i|x^i; \theta) \\ &= -\prod_{i=1}^N (\sigma(\theta^T x^i))^{y^i} * (1 - \sigma(\theta^T x^i))^{1-y^i}\end{aligned}$$



# Objective Function

Mimimize the Negative conditional log likelihood

$$\mathcal{L}(\theta) = -P(Y|X; \theta) = -\prod_{i=1}^N P(y^i|x^i; \theta)$$

note that

$$P(y = 1|x; \theta) = \sigma(\theta^T x)$$

$$P(y = 0|x; \theta) = 1 - \sigma(\theta^T x)$$

so

$$\begin{aligned}\mathcal{L}(\theta) &= -P(Y|X; \theta) = -\prod_{i=1}^N P(y^i|x^i; \theta) \\ &= -\prod_{i=1}^N (\sigma(\theta^T x^i))^{y^i} * (1 - \sigma(\theta^T x^i))^{1-y^i}\end{aligned}$$

take the log of this function

$$\log \mathcal{L}(\theta) = -\sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$



## Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$



# Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

## Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

## Also

- Derivative of sum = sum of derivatives  $\rightarrow$  focus on 1 training input
- Compute  $\frac{\partial \mathcal{L}}{\partial \theta_j}$  for each  $\theta_j$  individually, so focus on 1  $\theta_j$



# Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

## Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$



## Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

↓

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial p} = -\left(\frac{y}{p} - \frac{1-y}{1-p}\right)$$

( because  $\mathcal{L}(\theta) = -[y \log p + (1-y) \log(1-p)]$ )



## Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$\frac{\partial p}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$$



## Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$\frac{\partial z}{\partial \theta_j} = \frac{\partial \theta^T x}{\partial \theta_j} = x_j$$



# Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

## Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$= -\frac{y}{p} - \frac{1-y}{1-p} \times \sigma(z)[1 - \sigma(z)] \times x_j$$



# Take 1st Derivative of the Objective Function

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

## Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial D} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C} \times \frac{\partial C}{\partial D}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$\begin{aligned} &= -\frac{y}{p} - \frac{1-y}{1-p} \times \sigma(z)[1 - \sigma(z)] \times x_j \\ &= [\sigma(\theta^T x) - y] \times x_j \end{aligned}$$



## Logistic Regression: Parameter Estimation III

The derivative of the log likelihood wrt. a single parameter  $\theta_j$  for **all** training examples

$$\frac{\log \mathcal{L}(\theta)}{\partial \theta_j} = \sum_{i=1}^N (\sigma(\theta^T x^i) - y^i) x_j^i$$

- Now, we would set derivatives to zero (**Step 3**) and solve for  $\theta$  (**Step 4**)
- Unfortunately, that's not straightforward here (as for Naive Bayes)
- Instead, we will use an iterative method: **Gradient Descent**



## Logistic Regression: Parameter Estimation III

The derivative of the log likelihood wrt. a single parameter  $\theta_j$  for **all** training examples

$$\frac{\log \mathcal{L}(\theta)}{\partial \theta_j} = \sum_{i=1}^N \left( \sigma(\theta^T x^i) - y^i \right) x_j^i$$

- Now, we would set derivatives to zero (**Step 3**) and solve for  $\theta$  (**Step 4**)
- Unfortunately, that's not straightforward here (as for Naive Bayes)
- Instead, we will use an iterative method: **Gradient Descent**

$$\theta_j^{(new)} \leftarrow \theta_j^{(old)} - \eta \frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j}$$

$$\theta_j^{(new)} \leftarrow \theta_j^{(old)} - \eta \sum_{i=1}^N \left( \sigma(\theta^T x^i) - y^i \right) x_j^i$$



## Multinomial Logistic Regression

- So far we looked at problems where either  $y = 0$  or  $y = 1$  (e.g., spam classification:  $y \in \{\text{play}, \text{not\_play}\}$ )

$$\begin{aligned} P(y = 1|x; \theta) &= \sigma(\theta^T x) &= \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)} \\ P(y = 0|x; \theta) &= 1 - \sigma(\theta^T x) &= 1 - \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)} \end{aligned}$$



## Multinomial Logistic Regression

- So far we looked at problems where either  $y = 0$  or  $y = 1$  (e.g., spam classification:  $y \in \{\text{play}, \text{not\_play}\}$ )

$$P(y = 1|x; \theta) = \sigma(\theta^T x) = \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)}$$
$$P(y = 0|x; \theta) = 1 - \sigma(\theta^T x) = 1 - \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)}$$

- But what if we have more than 2 classes, e.g.,  $y \in \{\text{positive}, \text{negative}, \text{neutral}\}$
- we predict the probability of each class  $c$  by passing the input representation through the **softmax** function, a generalization of the sigmoid

$$p(y = c|x; \theta) = \frac{\exp(\theta_c x)}{\sum_k \exp(\theta_k x)}$$

- we learn a parameter vector  $\theta_c$  for each class  $c$



## Example! Multi-class with 1-hot features

$$p(y = c|x; \theta) = \frac{\exp(\theta_c x)}{\sum_k \exp(\theta_k x)}$$

### Model parameters

$$\theta_{c0} = [0.1, -3.5, 0.7, 2.1]$$

$$\theta_{c1} = [0.6, 2.5, 2.7, -2.1]$$

$$\theta_{c2} = [3.1, 1.5, 0.07, 3.6]$$

### (Small) Test Data set

Outlook	Temp	Humidity	Class
<i>rainy</i>	<i>cool</i>	<i>normal</i>	0 (don't play)
<i>sunny</i>	<i>cool</i>	<i>normal</i>	1 (maybe play)
<i>sunny</i>	<i>hot</i>	<i>high</i>	2 (play)

### Feature Function

$$x_0 = 1 \text{ (bias term)}$$

$$x_1 = \begin{cases} 1 & \text{if outlook=sunny} \\ 2 & \text{if outlook=overcast} \\ 3 & \text{if outlook=rainy} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if temp=hot} \\ 2 & \text{if temp=mild} \\ 3 & \text{if temp=cool} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if humidity=normal} \\ 2 & \text{if humidity=high} \end{cases}$$

$$x_0 = 1 \text{ (bias term)}$$

$$x_1 = \begin{cases} [100] & \text{if outlook=sunny} \\ [010] & \text{if outlook=overcast} \\ [001] & \text{if outlook=rainy} \end{cases}$$

$$x_2 = \begin{cases} [100] & \text{if temp=hot} \\ [010] & \text{if temp=mild} \\ [001] & \text{if temp=cool} \end{cases}$$

$$x_3 = \begin{cases} [10] & \text{if humidity=normal} \\ [01] & \text{if humidity=high} \end{cases}$$



## Example! Multi-class with 1-hot features

$$p(y = c|x; \theta) = \frac{\exp(\theta_c x)}{\sum_k \exp(\theta_k x)}$$

### (Small) Test Data set

Outlook	Temp	Humidity	Class
0 0 1	0 0 1	1 0	0
1 0 0	0 0 1	1 0	1
1 0 0	1 0 0	0 1	2

### Feature Function

$$x_0 = 1 \text{ (bias term)}$$

$$x_1 = \begin{cases} [100] & \text{if outlook=sunny} \\ [010] & \text{if outlook=overcast} \\ [001] & \text{if outlook=rainy} \end{cases}$$

$$x_2 = \begin{cases} [100] & \text{if temp=hot} \\ [010] & \text{if temp=mild} \\ [001] & \text{if temp=cool} \end{cases}$$

$$x_3 = \begin{cases} [10] & \text{if humidity=normal} \\ [01] & \text{if humidity=high} \end{cases}$$



# Logistic Regression: Final Thoughts

## Pros

- Probabilistic interpretation
- No restrictive assumptions on features
- Often outperforms Naive Bayes
- Particularly suited to frequency-based features (so, popular in NLP)

## Cons

- Can only learn *linear* feature-data relationships
- Some feature scaling issues
- Often needs a lot of data to work well
- Regularisation a nuisance, but important since overfitting can be a big problem



# Summary

- Derivation of logistic regression
- Prediction
- Derivation of maximum likelihood



## References

Cosma Shalizi. *Advanced Data Analysis from an Elementary Point of View*.  
Chapters 11.1 and 11.2. Online Draft.  
<https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf>

Dan Jurafsky and James H. Martin. *Speech and Language Processing*.  
Chapter 5. Online Draft V3.0.  
<https://web.stanford.edu/~jurafsky/slp3/>



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

### Also

- Derivative of sum = sum of derivatives  $\rightarrow$  focus on 1 training input
- Compute  $\frac{\partial \mathcal{L}}{\partial \theta_j}$  for each  $\theta_j$  individually, so focus on 1  $\theta_j$



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial p} = -\left(\frac{y}{p} - \frac{1-y}{1-p}\right)$$

( because  $\mathcal{L}(\theta) = -[y \log p + (1-y) \log(1-p)]$  )



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$\frac{\partial p}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$$



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$\frac{\partial z}{\partial \theta_j} = \frac{\partial \theta^T x}{\partial z} = x_j$$



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$= - \left[ \frac{y}{p} - \frac{1-y}{1-p} \right] \times \sigma(z)[1 - \sigma(z)] \times x_j \quad [[ \text{combine 3 derivatives} ]]$$

$$= - \left[ \frac{y}{p} - \frac{1-y}{1-p} \right] \times p[1-p] \times x_j \quad [[ \sigma(z) = p ]]$$

$$= - \left[ \frac{y(1-p)}{p(1-p)} - \frac{p(1-y)}{p(1-p)} \right] \times p[1-p] \times x_j \quad [[ \times \frac{1-p}{1-p} \text{ and } \frac{p}{p} ]]$$

$$= - [y(1-p) - p(1-y)] \times x_j \quad [[ \text{cancel terms} ]]$$



## Optional: Detailed Parameter Estimation

**Step 2** Differentiate the loglikelihood wrt. the parameters

$$\log \mathcal{L}(\theta) = - \sum_{i=1}^N y^i \log \sigma(\theta^T x^i) + (1 - y^i) \log(1 - \sigma(\theta^T x^i))$$

### Preliminaries

- The derivative of the logistic (sigmoid) function is  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)[1 - \sigma(z)]$
- The chain rule tells us that  $\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \times \frac{\partial B}{\partial C}$

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_j} = \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \times \frac{\partial p}{\partial z} \times \frac{\partial z}{\partial \theta_j} \quad \text{where } p = \sigma(\theta^T x) \text{ and } z = \theta^T x$$

$$= -[y(1 - p) - p(1 - y)] \times x_j \quad [[ \text{copy from last slide} ]]$$

$$= -[y - yp - p + yp] \times x_j \quad [[ \text{multiply out} ]]$$

$$= -[y - p] \times x_j \quad [[ -yp+yp=0 ]]$$

$$= [p - y] \times x_j \quad [[ -[y-p] = -y+p = p-y ]]$$

$$= [\sigma(\theta^T x) - y] \times x_j \quad [[p = \sigma(z), z = \theta^T x ]]$$



# Lecture 11: The Perceptron

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



## **Introduction**

---

## So far... Naive Bayes and Logistic Regression

- KNN
- Probabilistic models
- Maximum likelihood estimation
- Examples and code



## So far... Naive Bayes and Logistic Regression

- KNN
- Probabilistic models
- Maximum likelihood estimation
- Examples and code

## Today... The Perceptron

- Geometric motivation
- Error-based optimization
- ...towards neural networks



# Recap: Classification algorithms

## Naive Bayes

- Generative model of  $p(x, y)$
- Find optimal parameter that maximize the log data likelihood
- Unrealistic independence assumption  $p(x|y) = \prod_i p(x_i|y)$

## Logistic Regression

- Discriminative model of  $p(y|x)$
- Find optimal parameters that maximize the conditional log data likelihood
- Allows for more complex features (fewer assumptions)



## Recap: Classification algorithms

### Naive Bayes

- Generative model of  $p(x, y)$
- Find optimal parameter that maximize the log data likelihood
- Unrealistic independence assumption  $p(x|y) = \prod_i p(x_i|y)$

### Logistic Regression

- Discriminative model of  $p(y|x)$
- Find optimal parameters that maximize the conditional log data likelihood
- Allows for more complex features (fewer assumptions)

### Perceptron

- Biological motivation: imitating neurons in the brain
- No more probabilities
- Instead: minimize the classification error directly



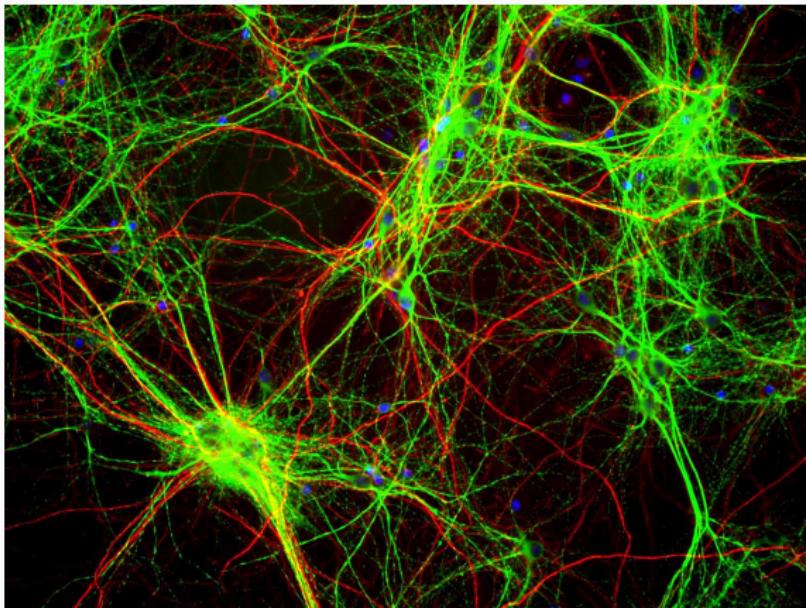
# Introduction: Neurons in the Brain

- Humans are the best learners we know
- Can we take inspiration from human learning
- → the brain!

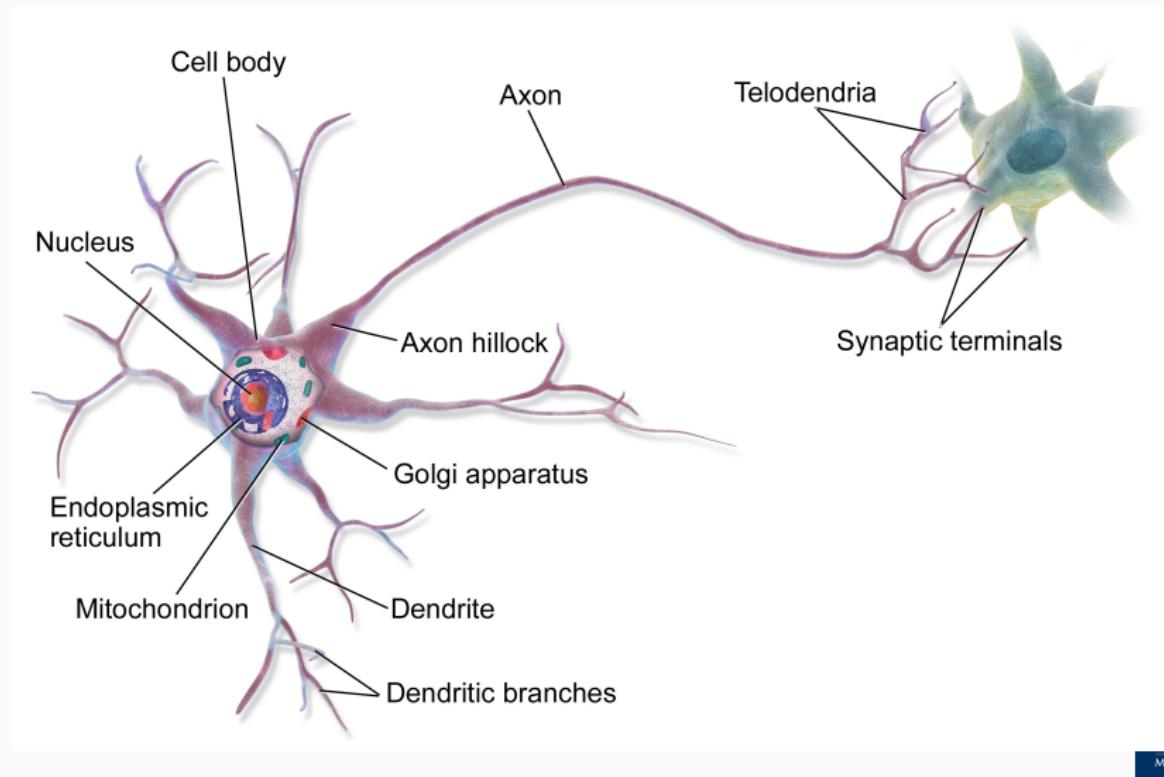


# Introduction: Neurons in the Brain

<https://vimeo.com/227026686>



# Introduction: Neurons in the Brain



Source: [https://upload.wikimedia.org/wikipedia/commons/1/10/Blausen\\_0657\\_MultipolarNeuron.png](https://upload.wikimedia.org/wikipedia/commons/1/10/Blausen_0657_MultipolarNeuron.png)

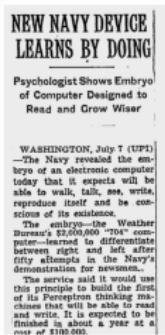


UNIVERSITY OF  
MELBOURNE

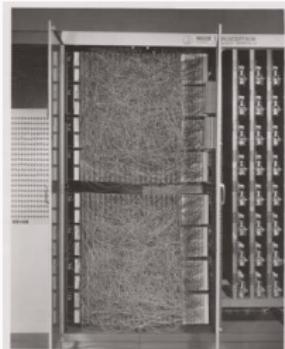
# Introduction: Neurons in the Brain

## The hype

- 1943 McCulloch and Pitts introduced the first ‘artificial neurons’
  - If the **weighted sum of inputs** is equal to or greater than a **threshold**, then the **output** is 1. Otherwise the output is 0.
  - the **weights** needed to be designed by hand
- 
- In 1958 Rosenblatt invented the **Perceptron**, which can learn the optimal parameters through the **perceptron learning rule**
  - The perceptron can be **trained** to learn the correct weights, even if randomly initialized [[ for a limited set of problems ]].



The New York Times, July 8 1958

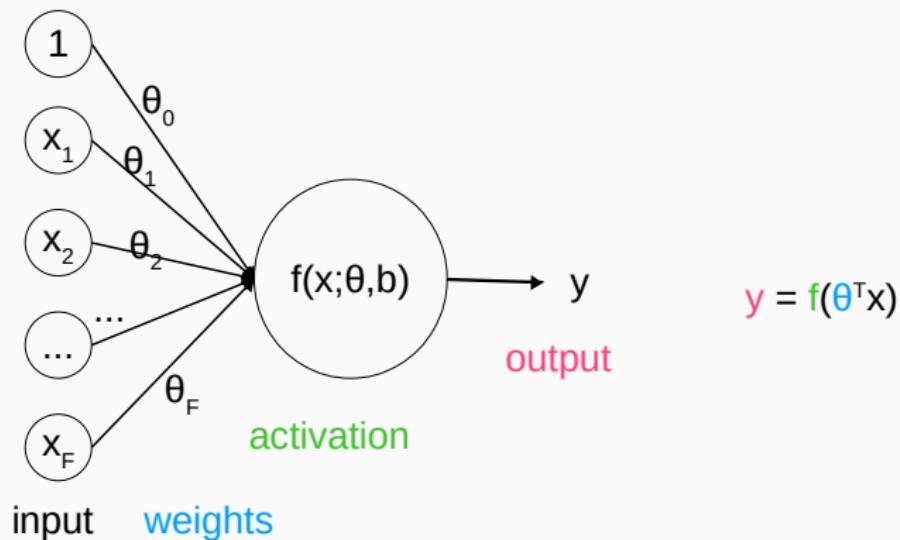


## The AI winter

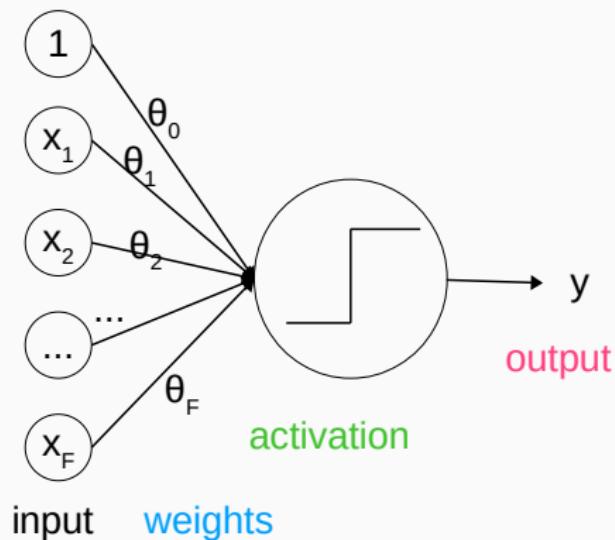
- A few years later Misky and Papert (too?) successfully pointed out the fundamental limitations of the perceptron.
- As a result, research on artificial neural networks stopped until the mid-1980s
- But the limitations can be overcome by combining multiple perceptrons into **Artificial Neural Networks**
- The perceptron is the basic component of today's deep learning success!



# Introduction: Artificial Neurons I



# Introduction: Artificial Neurons I



$$y = f(\theta x + b)$$

$$\begin{aligned} f: & y = 1 & \text{if} & f(\theta^T x) \geq 0 \\ & y = -1 & \text{if} & f(\theta^T x) < 0 \end{aligned}$$



## Perceptron: Definition I

- The Perceptron is a **minimal neural network**
- **neural networks** are composed of **neurons**
- A neuron is defined as follows:
  - input = a vector  $x$  of numeric inputs ( $\langle 1, x_1, x_2, \dots x_n \rangle$ )
  - output = a scalar  $y_i \in \mathbb{R}$
  - hyper-parameter: an **activation function**  $f$
  - parameters:  $\theta = \langle \theta_0, \theta_1, \theta_2, \dots \theta_n \rangle$
- Mathematically:

$$y^i = f \left( \left[ \sum_j \theta_j x_j^i \right] \right) = f(\theta^T x^i)$$

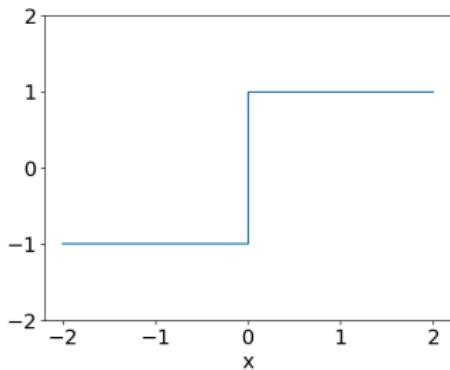


## Perceptron: Definition II

- Task: binary classification of instances into classes 1 and  $-1$
- Model: a single-neuron (aka a “perceptron”) :

$$f(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- $\theta^T x$  is the decision boundary
- Graphically,  $f$  is the **step function**



## Perceptron: Definition II

- Task: binary classification of instances into classes 1 and  $-1$
- Model: a single-neuron (aka a “perceptron”):

$$f(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- $\theta^T x$  is the decision boundary
- Example: 2-d case:



# Towards the Perceptron Algorithm I

- As usual, **learning** means to modify the **parameters** (i.e., weights) of the perceptron so that performance is **optimized**
- The perceptron is a **supervised** classification algorithm, so we learn from observations of input-label pairs

$$(x^1, y^1), (x^2, y^2), \dots (x^N, y^N)$$

- Simplest way to learn: compare predicted outputs  $\hat{y}$  against true outputs  $y$  and minimize the number of mis-classifications. Unfortunately, mathematically inconvenient.
- Second simplest idea: Find  $\theta$  such that gap between the predicted value  $\hat{y}^i \leftarrow f(\theta^T x^i)$  and the true class label  $y \in \{-1, 1\}$  is minimized



# Towards the Perceptron Algorithm I

**Intuition** Iterate over the **training data** and modify weights:

- if the true label  $y = 1$  and  $\hat{y} = 1$  then **do nothing**
- if the true label  $y = -1$  and  $\hat{y} = -1$  then **do nothing**
- if the true label  $y = 1$  but  $\hat{y} = -1$  then **increase** weights
- if the true label  $y = -1$  but  $\hat{y} = 1$  then **decrease** weights



# Towards the Perceptron Algorithm I

**Intuition** Iterate over the **training data** and modify weights:

- if the true label  $y = 1$  and  $\hat{y} = 1$  then **do nothing**
- if the true label  $y = -1$  and  $\hat{y} = -1$  then **do nothing**
- if the true label  $y = 1$  but  $\hat{y} = -1$  then **increase** weights
- if the true label  $y = -1$  but  $\hat{y} = 1$  then **decrease** weights

## More formally

---

```
Initialize parameters  $\theta \leftarrow 0$ 
for training sample  $(x, y)$  do
    Calculate the output  $\hat{y} = f(\theta^T x)$ 
    if  $y = 1$  and  $\hat{y} = -1$  then
         $\theta^{(new)} \leftarrow \theta^{(old)} + x$ 
    if  $y = -1$  and  $\hat{y} = 1$  then
         $\theta^{(new)} \leftarrow \theta^{(old)} - x$ 
until tired
```



## Towards the Perceptron Algorithm II

- We set a **learning rate** or **step size**  $\eta$
- and note that

$$(y^i - \hat{y}^i) = \begin{cases} 0 & \text{if } y^i == \hat{y}^i \\ 2 & \text{if } y^i = 1 \text{ and } \hat{y}^i = -1 \\ -2 & \text{if } y^i = -1 \text{ and } \hat{y}^i = 1 \end{cases} \quad (1)$$

- For each individual weight  $\theta_j$ , we compute an update such that

$$\theta_j \leftarrow \theta_j + \eta(y^i - \hat{y}^i)x_j^i$$



# The Perceptron Algorithm

---

$D = \{(\mathbf{x}^i, y^i) | i = 1, 2, \dots, N\}$  the set of training instances

Initialise the weight vector  $\theta \leftarrow 0$

$t \leftarrow 0$

**repeat**

$t \leftarrow t+1$

**for** each training instance  $(x^i, y^i) \in D$  **do**

    compute  $\hat{y}^{i,(t)} = f(\theta^T x^i)$

**if**  $\hat{y}^{i,(t)} \neq y^i$  **then**

**for** each weight  $\theta_j$  **do**

            update  $\theta_j^{(t)} \leftarrow \theta_j^{(t-1)} + \eta(y^i - \hat{y}^{i,(t)})x_j^i$

**else**

$\theta_j^{(t)} \leftarrow \theta_j^{(t-1)}$

**until** tired

Return  $\theta^{(t)}$



## An example

---

# Perceptron Example I

- Training instances:

$\langle x_{i1}, x_{i2} \rangle$	$y_i$
$\langle 1, 1 \rangle$	1
$\langle 1, 2 \rangle$	1
$\langle 0, 0 \rangle$	-1
$\langle -1, 0 \rangle$	-1

- Learning rate  $\eta = 1$



## Perceptron Example II

- $\theta = \langle 0, 0, 0 \rangle$
- learning rate:  $\eta = 1$
- Epoch 1:

$\langle x_1, x_2 \rangle$	$\theta_0 \cdot 1 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2$	$\hat{y}_i^{(1)}$	$y_i$
$\langle 1, 1 \rangle$	$0 + 1 \times 0 + 1 \times 0 = 0$	1	1
$\langle 1, 2 \rangle$	$0 + 1 \times 0 + 2 \times 0 = 0$	1	1
$\langle 0, 0 \rangle$	$0 + 0 \times 0 + 0 \times 0 = 0$	1	-1
Update to $\theta = \langle -2, 0, 0 \rangle$			
$\langle -1, 0 \rangle$	$-2 + -1 \times 0 + 0 \times 0 = -2$	-1	-1



## Perceptron Example III

- $\theta = \langle -2, 0, 0 \rangle$
- learning rate:  $\eta = 1$
- Epoch 2:

$\langle x_1, x_2 \rangle$	$\theta_0 \cdot 1 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2$	$\hat{y}_i^{(2)}$	$y_i$
$\langle 1, 1 \rangle$	$-2 + 1 \times 0 + 1 \times 0 = -2$	-1	1
Update to $\theta = \langle 0, 2, 2 \rangle$			
$\langle 1, 2 \rangle$	$0 + 1 \times 2 + 2 \times 2 = 6$	1	1
$\langle 0, 0 \rangle$	$0 + 0 \times 2 + 0 \times 2 = 0$	1	-1
Update to $\theta = \langle -2, 2, 2 \rangle$			
$\langle -1, 0 \rangle$	$-2 + -1 \times 2 + 0 \times 2 = -4$	-1	-1



## Perceptron Example IV

- $\theta = \langle -2, 2, 2 \rangle$
- learning rate:  $\eta = 1$
- Epoch 3:

$\langle x_1, x_2 \rangle$	$\theta_0 \cdot 1 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2$	$\hat{y}_i^{(3)}$	$y_i$
$\langle 1, 1 \rangle$	$-2 + 1 \times 2 + 1 \times 2 = 2$	1	1
$\langle 1, 2 \rangle$	$-2 + 1 \times 2 + 2 \times 2 = 4$	1	1
$\langle 0, 0 \rangle$	$-2 + 0 \times 2 + 0 \times 2 = -2$	-1	-1
$\langle -1, 0 \rangle$	$-2 + -1 \times 2 + 0 \times 2 = -4$	-1	-1

- Convergence, as no updates throughout epoch



# Perceptron Convergence

## Perceptron Rule:

$$\theta_j^{(t+1)} \leftarrow \theta_j^{(t)} + \eta(y_i - \hat{y}^i)x_j^i$$

- So, all we're doing is adding and subtracting constants every time we make a mistake.
- Does this really work!?



# Perceptron Convergence

## Perceptron Convergence

- The Perceptron algorithm is guaranteed to **converge** for linearly-separable data
  - the convergence point will depend on the initialisation
  - the convergence point will depend on the learning rate
  - (no guarantee of the margin being maximised)
- No guarantee of convergence over non-linearly separable data



# Back to Logistic Regression and Gradient Descent

## Perceptron Rule

$$\theta_j^{(t+1)} \leftarrow \theta_j^{(t)} + \eta(y_i - \hat{y}^i)x_j^i$$

## Gradient Descent

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial f}{\partial \theta^{(t)}}$$

## Activation Functions



# Back to Logistic Regression and Gradient Descent

## Perceptron Rule

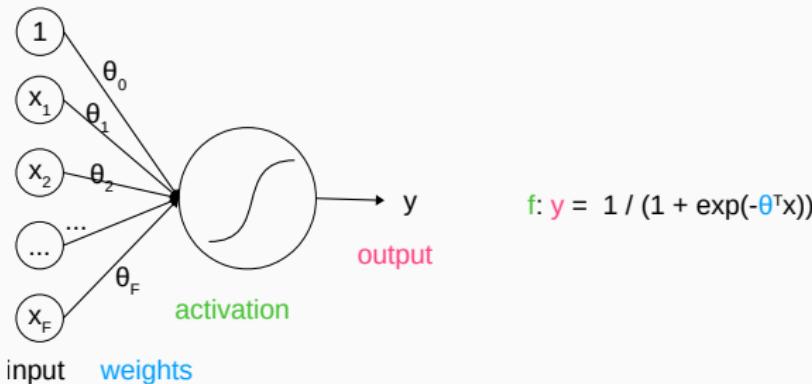
$$\theta_j^{(t+1)} \leftarrow \theta_j^{(t)} + \eta(y_i - \hat{y}^i)x_j^i$$

## Gradient Descent

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial f}{\partial \theta^{(t)}}$$

## Activation Functions

A single ‘neuron’ with a **sigmoid activation** which optimizes the **cross-entropy loss** (negative log likelihood) is equivalent to **logistic regression**



## Online learning vs. Batch learning

- It is an **online algorithm**: we update the weights after each training example
- In contrast, Naive Bayes and logistic regression (with Gradient Descent) are updated as a **batch** algorithm:
  - compute statistics of the *whole* training data set
  - update all parameters at once
- Online learning can be more efficient for large data sets
- Gradient Descent can be converted into an online version: **stochastic gradient descent**



# Multi-Class Perceptron

We can generalize the perceptron to more than 2 classes

- create a weight vector for each class  $k \in Y$ ,  $\theta^k$
- score input wrt each class:  $\theta_k^T x$  for all  $k$
- predict the class with maximum output  $\hat{y} = \operatorname{argmax}_{k \in Y} \theta_k^T x$
- learning works as before: if for some  $(x^i, y^i)$  we make a wrong prediction  $\hat{y}^i \neq y^i$  such that  $\theta_{y^i}^T x^i < \theta_{\hat{y}^i}^T x^i$ ,

$$\theta_{y^i} \leftarrow \theta_{y^i} + \eta x^i \quad \text{move towards predicting } y^i \text{ for } x^i$$

$$\theta_{\hat{y}^i} \leftarrow \theta_{\hat{y}^i} - \eta x^i \quad \text{move away from predicting } \hat{y}^i \text{ for } x^i$$



# Summary

## This lecture: The Perceptron

- Biological motivation
- Error-based classifier
- The Perceptron Rule
- Relation to Logistic Regression
- Multi-class perceptron

## Next

- More powerful machine learning through combining perceptrons
- More on linear separability
- More on activation functions
- Learning with backpropagation



## References

- Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.
- Minsky, Marvin, and Seymour Papert. "Perceptrons: An essay in computational geometry." MIT Press. (1969).
- Bishop, Christopher M. *Pattern recognition and machine learning*. Springer, 2006. Chapter 4.1.7



# Lecture 12: Multilayer Perceptron

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne



# About Me

## Experience:

- 2020.01 - now: Lecturer in The University of Melbourne
- 2018.05-2019.12: Post-doc in Max Planck Institute for Informatics, Germany
- 2015.02-2018.04: PhD in The University of Western Australia

Research: computer vision and machine learning

Contact: [qiuhsong.ke@unimelb.edu.au](mailto:qiuhsong.ke@unimelb.edu.au)



# Roadmap

## So far:

- Classifiers: KNN, Naive Bayes, Logistic Regression and Perceptron
- Feature Selection
- Evaluation

## Today: Multilayer Perceptron

- Introduction
- Architecture
- Reflections

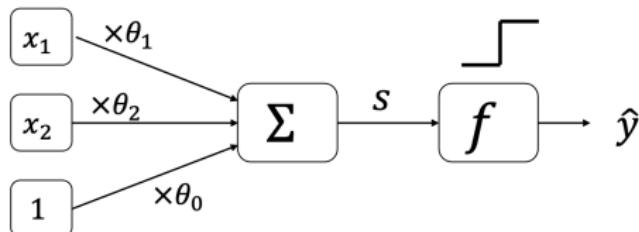


## **Introduction**

---

# Classifier Recap

## Perceptron



$$\hat{y} = f(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Single processing ‘unit’
- Inspired by neurons in the brain
- Activation: step-function (discrete, non-differentiable)
- Perceptron vs Logistic Regression?

# Linear classifiers

## Perceptron

$$\hat{y} = f(\theta^T x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Single processing ‘unit’
- Inspired by neurons in the brain
- Activation: step-function (discrete, non-differentiable)

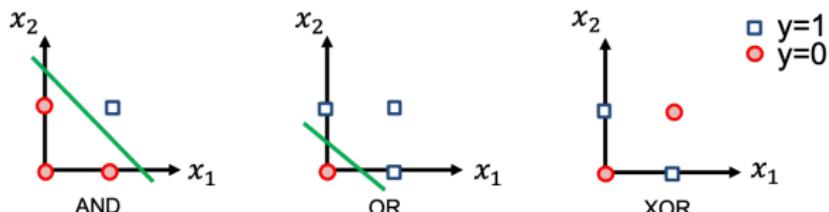
## Logistic Regression

$$P(y = 1|x; \theta) = f(\theta^T x) = \frac{1}{1 + \exp(-(\sum_{f=0}^F \theta_f x_f))}$$

- View 1: Model of  $P(y = 1|x)$ , maximizing the data log likelihood
- View 2: Single processing ‘unit’
- Activation: sigmoid (continuous, differentiable)



# Limitations of linear classifiers

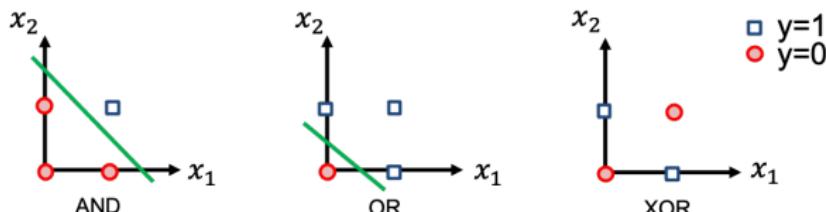


Linear classifier:

- Decision boundary is a linear combination of features  $\sum_i \theta_i x_i$
- Cannot learn ‘feature interactions’ naturally
- can solve only linearly separable problems



# Motivations of multilayer perceptron



Possible solution: composition

$$x_1 \text{ XOR } x_2 = (x_1 \text{ OR } x_2) \text{ AND } [\text{NOT}(x_1 \text{ AND } x_2)]$$

Non-linear separable data classification:

compose perceptrons → Multilayer perceptron

# Neural Networks and Deep Learning

## Neural Networks

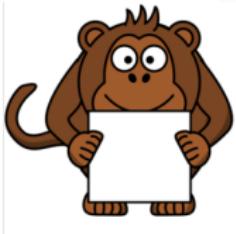
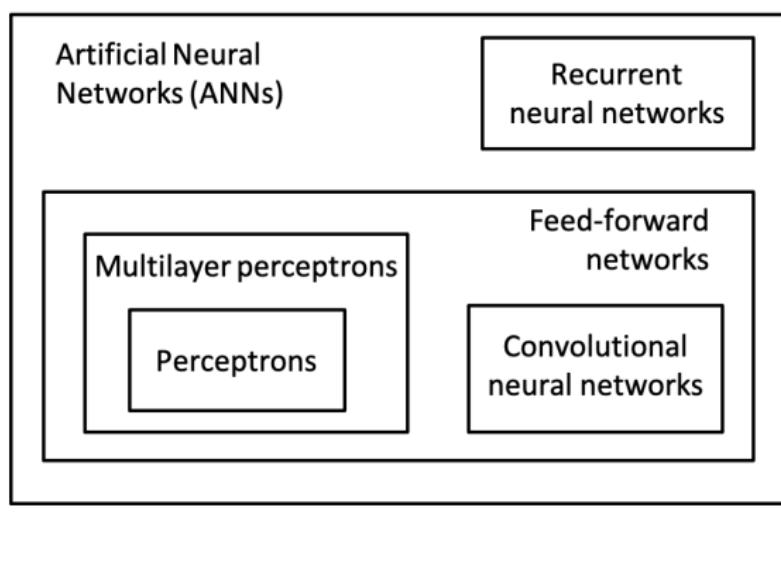
- Connected sets of many such units
- Connected into many layers → **Deep** neural networks

## Multilayer Perceptron

- This lecture!
- One specific type of neural network
- Feed-forward
- Fully connected
- Supervised learner



# Neural networks: “Animals” in the zoo

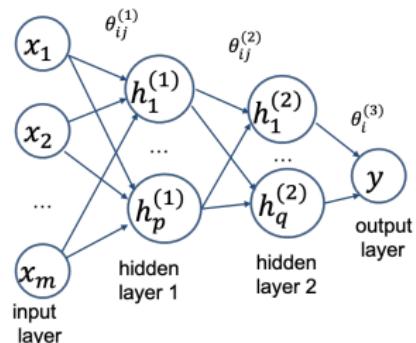


art: OpenClipartVectors  
at pixabay.com (CC0)

## **Architecture**

---

# Multilayer Perceptron



## Terminology

- depth: number of hidden layers and output layer.
- Each layer  $l$  has a number of units  $K_l$ .  $K_l$  is the **width** of layer  $l$ .
- Each layer  $l$  is **fully connected** to its neighboring layers  $l - 1$  and  $l + 1$ .

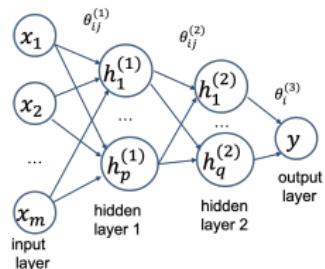
# Prediction

$$h_j^{(1)} = \phi^{(1)} \left( \sum_{i=1}^m \theta_{ij}^{(1)} x_i + \theta_{0j}^{(1)} \right) \quad \text{or } h^{(1)} = \phi^{(1)} \left( \theta^{(1)T} x \right), \text{ assume } x_0 = 1$$

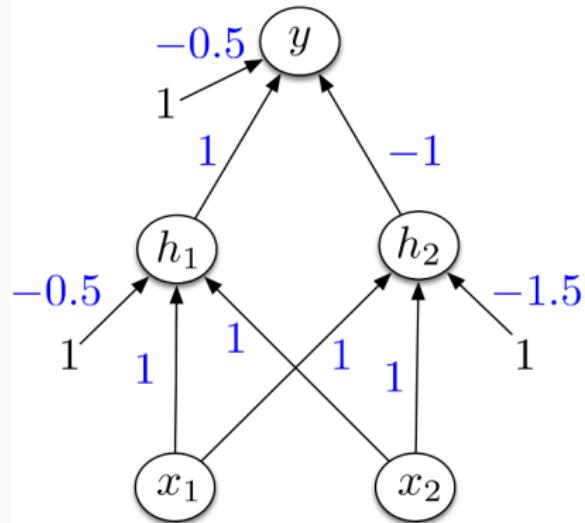
$$h_j^{(2)} = \phi^{(2)} \left( \sum_{i=1}^p \theta_{ij}^{(2)} h_i^{(1)} + \theta_{0j}^{(2)} \right) \quad \text{or } h^{(2)} = \phi^{(2)} \left( \theta^{(2)T} h^{(1)} \right), \text{ assume } h_0^{(1)} = 1$$

$$y = \phi^{(3)} \left( \sum_{i=1}^q \theta_i^{(3)} h_i^{(2)} + \theta_{0j}^{(3)} \right) \quad \text{or } y = \phi^{(3)} \left( \theta^{(3)T} h^{(2)} \right), \text{ assume } h_0^{(2)} = 1$$

- (non-linear) activation function for layer  $l$  as  $\phi^{(l)}$
- one weight  $\theta_{ij}^{(l)}$  for each connection  $ij$  (unit  $i$  in layer  $l-1$  layer and unit  $j$  unit in layer  $l$ )



# A Multilayer Perceptron for XOR



$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad \text{and recall: } h_j^{(l)} = \phi^{(l)}\left(\sum_i \theta_{ij}^{(l)} h_i^{(l-1)} + \theta_j^{(l)}\right)$$

Source: [https://www.cs.toronto.edu/~rgrosse/courses/csc321\\_2018/readings/L05%20Multilayer%20Perceptrons.pdf](https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/readings/L05%20Multilayer%20Perceptrons.pdf)

# Multilayer Perceptron I: Inputs

## Feature Engineering

- informative features, e.g., outlook  $\in \{overcast, sunny, rainy\}$ , wind  $\in \{high, low\}$  etc.
- Require **domain knowledge**
- Require feature selection



## Example Classification dataset

Feature engineering on Weather dataset:

Outlook	Temperature	Humidity	Windy	True Label
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
...				

Raw data of Weather dataset:

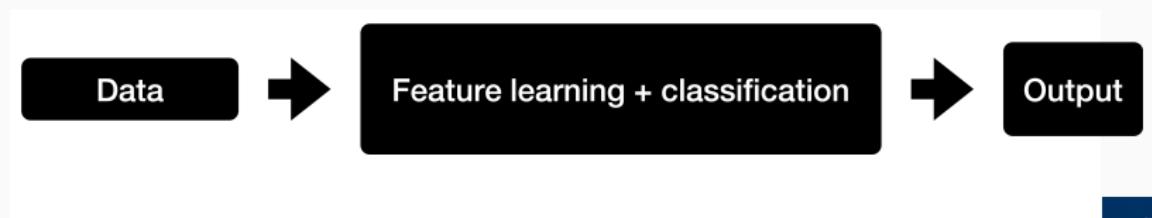
Date	measurements					True Label
01/03/1966	0.4	4.7	1.5	12.7	...	no
01/04/1966	3.4	-0.7	3.8	18.7	...	no
01/05/1966	0.3	8.7	136.9	17	...	yes
01/06/1966	5.5	5.7	65.5	2.7	...	yes



# Multilayer Perceptron: Input layer

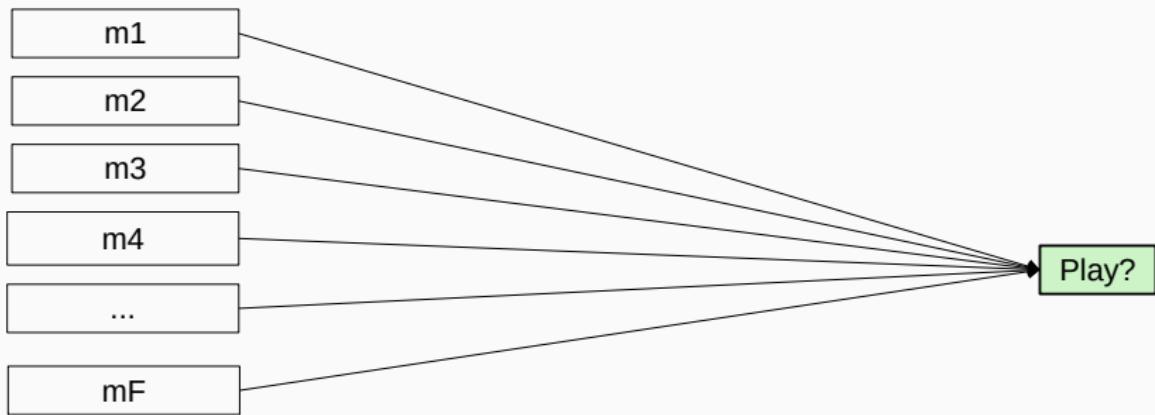
Feature learning:

- Neural networks can take as input 'raw' data
- Neural networks learn features themselves as intermediate representations
- feature engineering is replaced at the cost of additional parameter tuning (layers, activation functions, learning rates, ...)



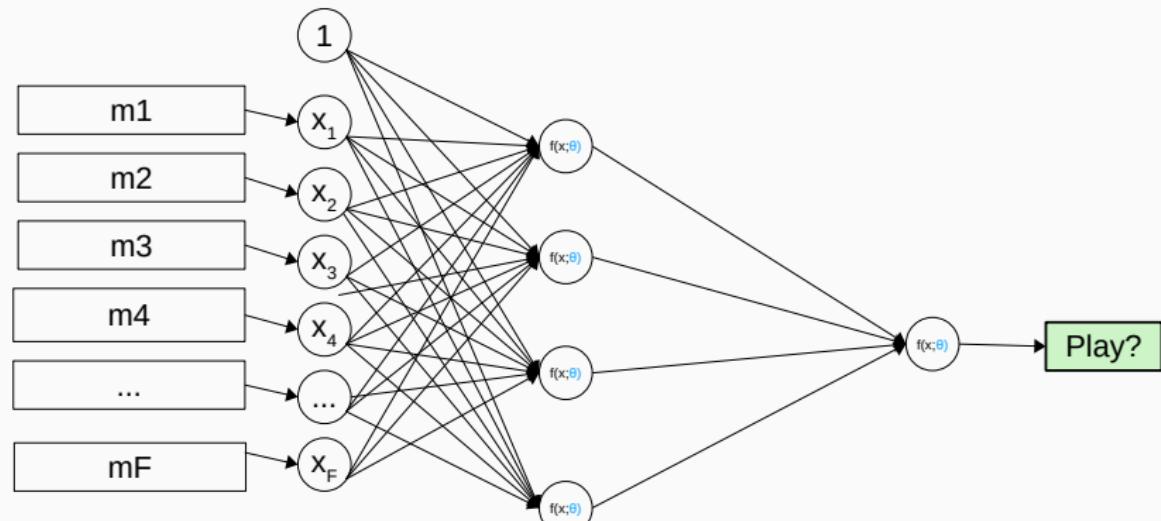
# Multilayer Perceptron for Classification

**Example Problem:** Raw data of Weather Dataset



# Multilayer Perceptron for Classification

**Example Problem:** Raw data of Weather Dataset



Input layer,

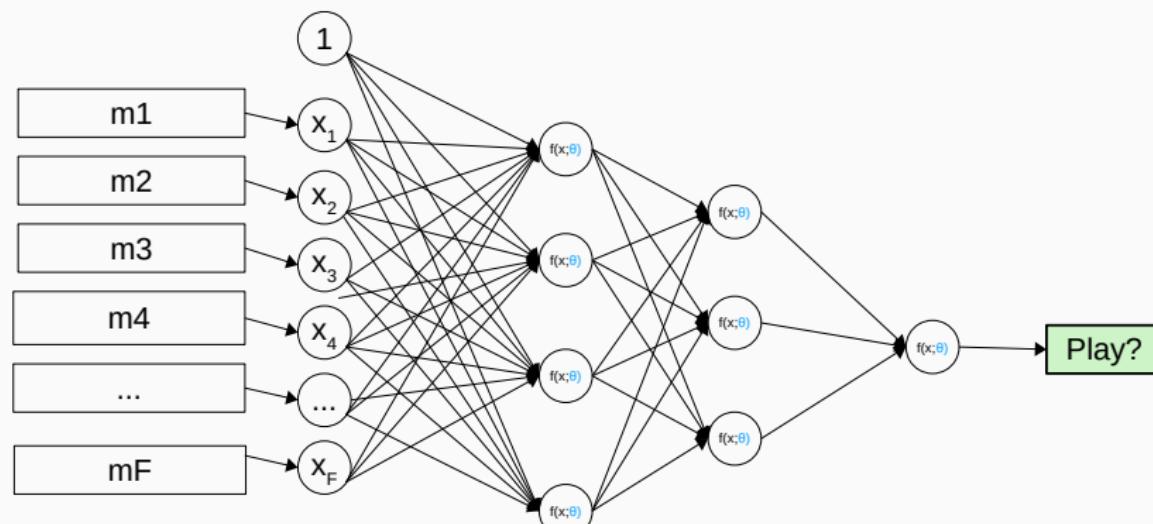
1 hidden layer,

output layer



# Multilayer Perceptron for Classification

**Example Problem:** Raw data of Weather Dataset



Input layer,

2 hidden layers,

output layer



# Multilayer Perceptron I: Inputs

## Inputs and feature functions

- $x$  could be numerical measurements, e.g., {blood pressure, height, age, weight, ...}
- $x$  could be a texts, i.e., a sequence of words
- $x$  could be an image, i.e., a matrix of pixels

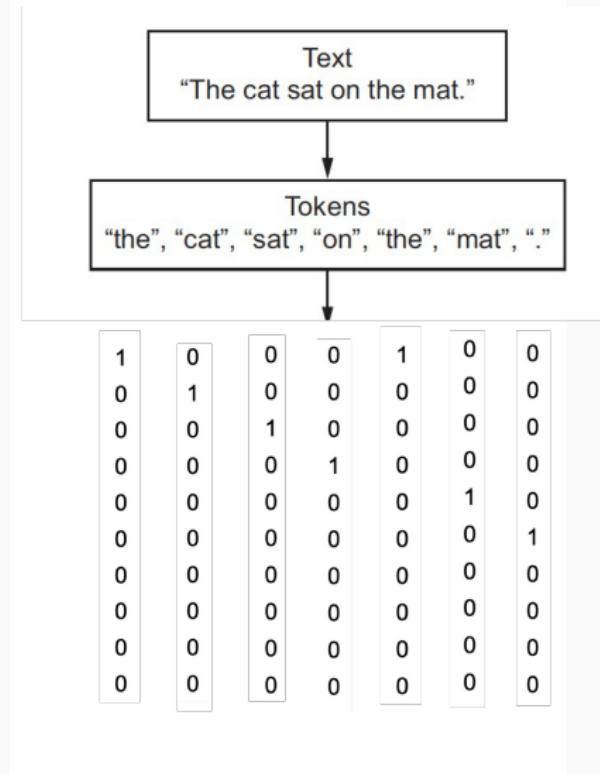
Non-numerical data need to be mapped to numerical

- For language:
  - 1-hot encoding:  $\dim(x) = V$  (words in the vocabulary)
  - pre-trained word embedding vectors:  $\dim(x) = k$  (embedding feature dimension  $k < V$ )
- For pixels, map to RGB, or other visual features



# One-hot Embedding

dic={ “the” , “cat” , “sat” , “on” , “mat” , “.” , “these” , “are” , “other” , “words” }



## Multilayer Perceptron II: Hidden Layer- Activation Functions

- Each layer has an associated activation function (e.g., sigmoid, ReLU, ...)
- Represents the extent to which a neuron is ‘activated’ given an input
- Each hidden layer performs a **non-linear transformation** of the input
- **the activation functions must be non-linear**, as without this, the model is simply a (complex) linear model



# Popular activation functions

1. logistic (aka sigmoid) (" $\sigma$ "):

$$f(x) = \frac{1}{1 + e^{-x}}$$

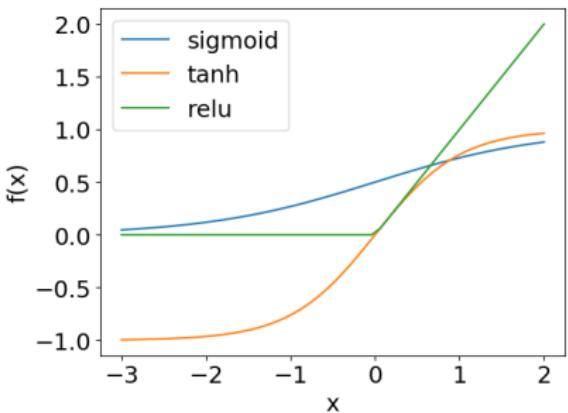
2. hyperbolic tan ("tanh"):

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

3. rectified linear unit ("ReLU"):

$$f(x) = \max(0, x)$$

note not differentiable at  $x = 0$



## Multilayer Perceptron III: Output Functions

Neural networks can learn different concepts: **classification, regression, ...**

The **output function** depends on the concept of interest.

- Binary classification:
  - one neuron, with sigmoid function
- Multiclass classification:
  - typically **softmax** to normalize  $K$  outputs from the pre-final layer into a probability distribution over classes

$$p(y_i = j|x_i; \theta) = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

- Regression:
  - identity function
  - possibly other continuous functions such as sigmoid or tanh



## **Reflections**

---

# Linear vs Non-linear classifiers

## Linear classifier

- Decision boundary is a linear combination of features  $\sum_i \theta_i x_i$
- Cannot learn ‘feature interactions’ naturally
- can solve only linearly separable problems

## Non-linear classifier

- Neural networks with at least 1 hidden layer and non-linear activations are non-linear classifiers
- Decision boundary is a non-linear function of the inputs
- Capture “feature interactions”



# Pros and Cons of Neural Networks

## Pros

- Powerful tool!
- Neural networks with at least 1 hidden layer (with non-linear activation function) can approximate any (continuous) function.
- Automatic feature learning
- Empirically, very good performance for many diverse tasks

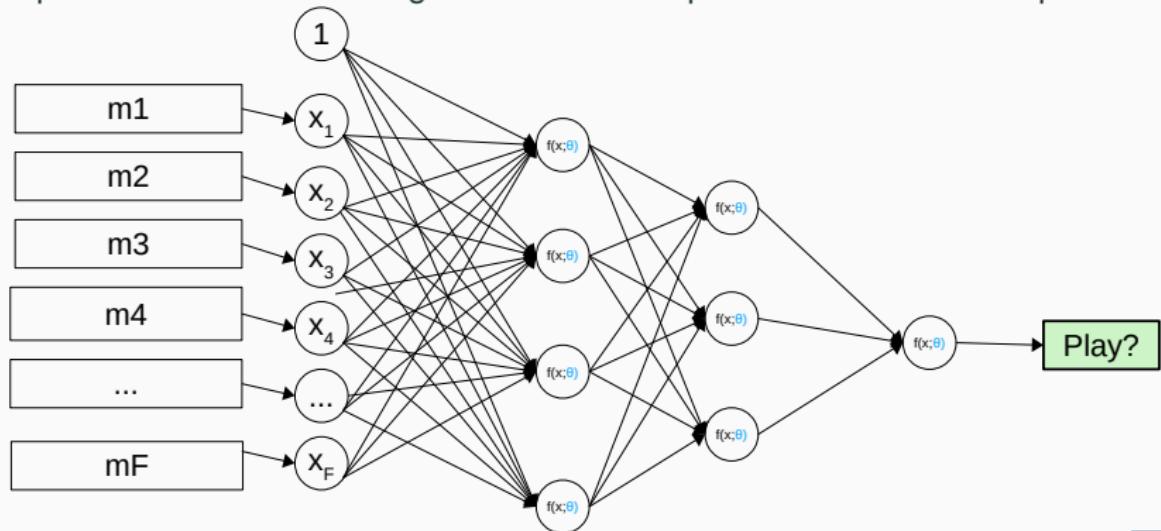
## Cons

- Powerful model increases the danger of ‘overfitting’
- Requires large training data sets
- Often requires powerful compute resources (GPUs)
- Lack of interpretability



# Lack of interpretability

Input units become indistinguishable: What input units lead to the output?



# When is Linear Classification Enough?

- If we know our classes are linearly (approximately) separable
- If the feature space is (very) high-dimensional
  - ...i.e., the number of features exceeds the number of training instances
- If the training set is small
- If *interpretability* is important, i.e., understanding how (combinations of) features explain different predictions



# Neural Networks Structure

## Network Structure

- Sequence of hidden layers  $l_1, \dots, l_L$  for a network of depth  $L$
- Each layer  $l$  has  $K_l$  parallel neurons (breadth)
- Many layers (depth) vs. many neurons per layer (breadth)? Empirical question, theoretically poorly understood.

**Advanced tricks** include allowing for exploiting data structure

- convolutions (convolutional neural networks; CNN), Computer Vision
- recurrencies (recurrent neural networks; RNN), Natural Language Processing
- attention (efficient alternative to recurrencies)
- ...

Beyond the scope of this class.



# Summary

## Today

- From perceptrons to neural networks
- multilayer perceptron prediction
- Input layer (data), hidden layer (activation functions), output layer (output functions)
- features and limitations

## Next Lecture

- Learning parameters of neural networks
- The Backpropagation algorithm



## References

- Jacob Eisenstein (2019). *Natural Language Processing*. MIT Press.  
Chapters 3 (intro), 3.1, 3.2. <https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf>
- Dan Jurafsky and James H. Martin. *Speech and Language Processing*.  
Chapter 7.2, 7.3. Online Draft V3.0.  
<https://web.stanford.edu/~jurafsky/slp3/>



# Lecture 13: Learning Parameters of Multilayer Perceptron with Backpropagation

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne



## Last lecture

- From perceptrons to neural networks
- multilayer perceptron
- input layer, hidden layer, output layer
- features and limitations

## Today

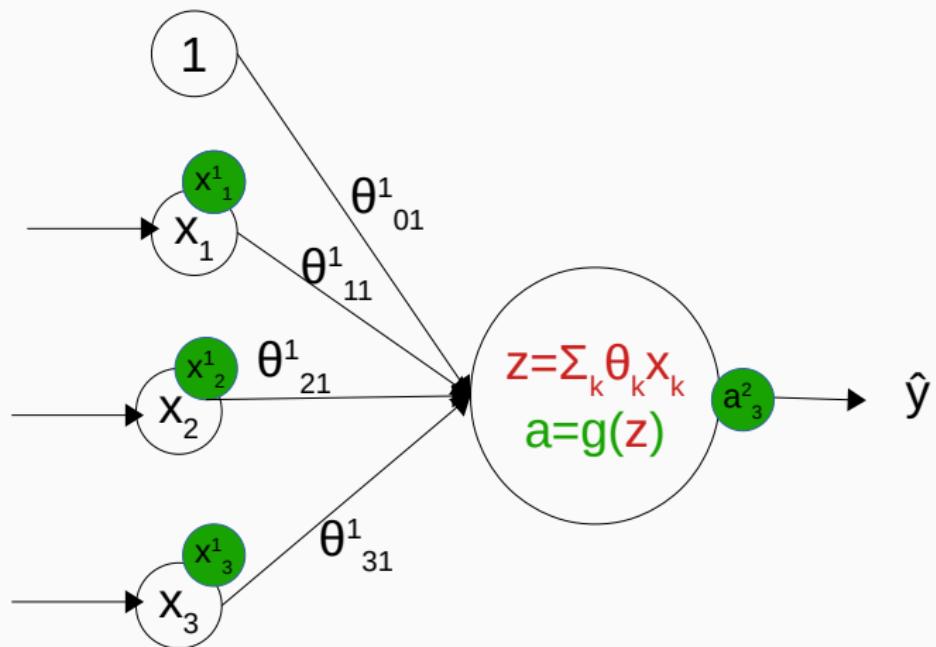
- Training a neural network
- Chaining derivatives: the Backpropagation algorithm



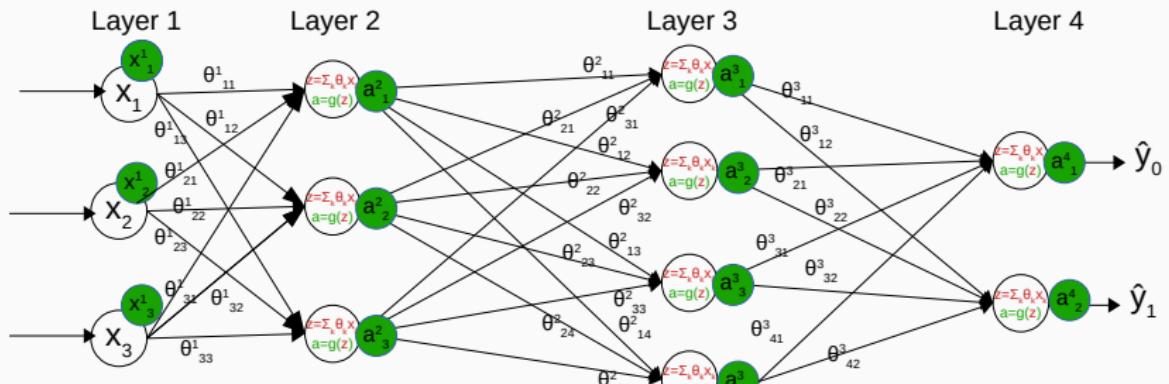
## **Training a neural network**

---

## Recap: perceptron



# Recap: Multilayer perceptron

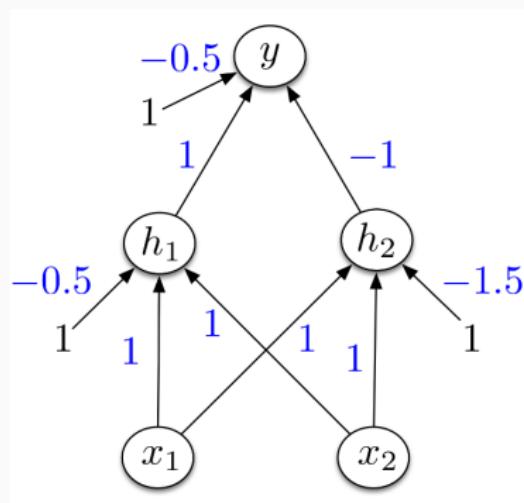


$$a_j^{(l)} = g\left(\sum_i \theta_{ij}^{(l)} a_i^{(l-1)} + \theta_{0j}^{(l)}\right)$$

- non-linearly separable data
- Hidden layers
- Non-linear activation functions

# A Multilayer Perceptron for XOR

How to obtain the optimal parameters?



$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad \text{and recall: } h_j^{(l)} = \phi^{(l)}\left(\sum_i \theta_{ij}^{(l)} h_i^{(l-1)} + \theta_{0j}^{(l)}\right)$$

Source: [https://www.cs.toronto.edu/~rgrosse/courses/csc321\\_2018/readings/L05%20Multilayer%20Perceptrons.pdf](https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/readings/L05%20Multilayer%20Perceptrons.pdf)



# How to train your dragon Network?

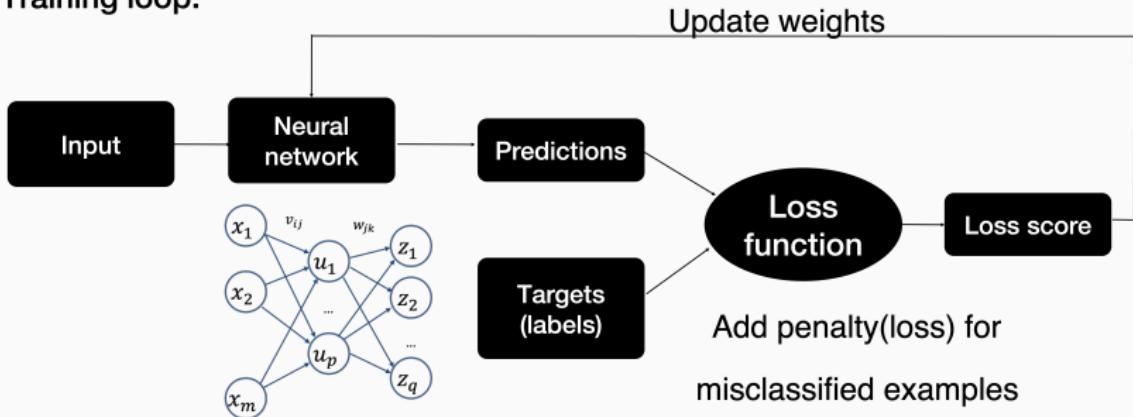


Adapted from Movie Poster from  
Flickr user jdxyw (CC BY-SA 2.0)



# “Training”: adjust weights to minimise loss

Training loop:



# Loss Functions

- Regression Loss: typically mean-squared error (MSE)

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (y^i - \hat{y}^{(i)})^2$$

- Binary classification loss

$$\hat{y}_1^{(i)} = p(y^{(i)} = 1 | x^{(i)}; \theta)$$

$$\mathcal{L} = \sum_i -[y^{(i)} \log(\hat{y}_1^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}_1^{(i)})]$$

- Multiclass classification

$$\hat{y}_j^{(i)} = p(y^{(i)} = j | x^{(i)}; \theta)$$

$$\mathcal{L} = - \sum_i \sum_j y_j^{(i)} \log(\hat{y}_j^{(i)})$$

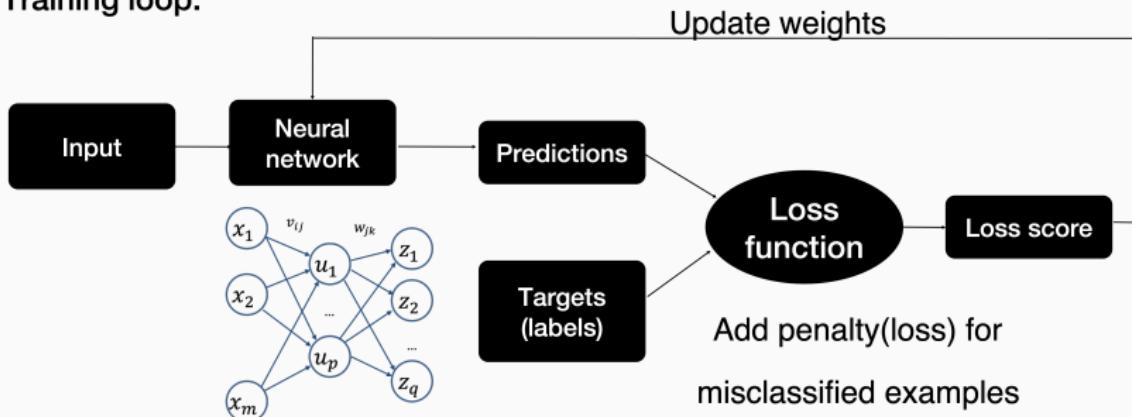
for  $j$  possible labels;  $y_j^{(i)} = 1$  if  $j$  is the true label for instance  $i$ , else 0.



# “Training”: adjust weights to minimise loss

**How?**: Use gradient descend

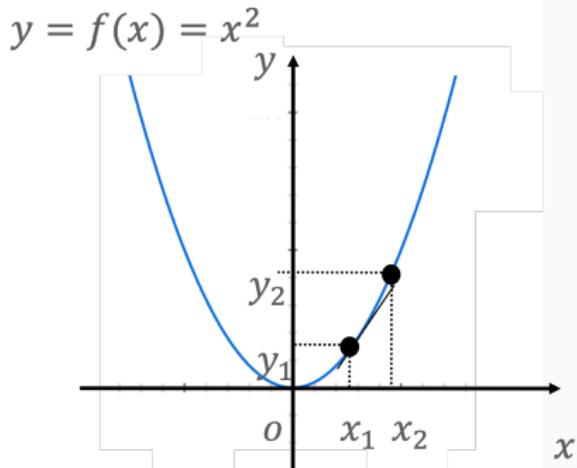
Training loop:



# Derivative

$$\Delta x \rightarrow 0 : f(x_1 + \Delta x) - f(x_1) = a\Delta x$$

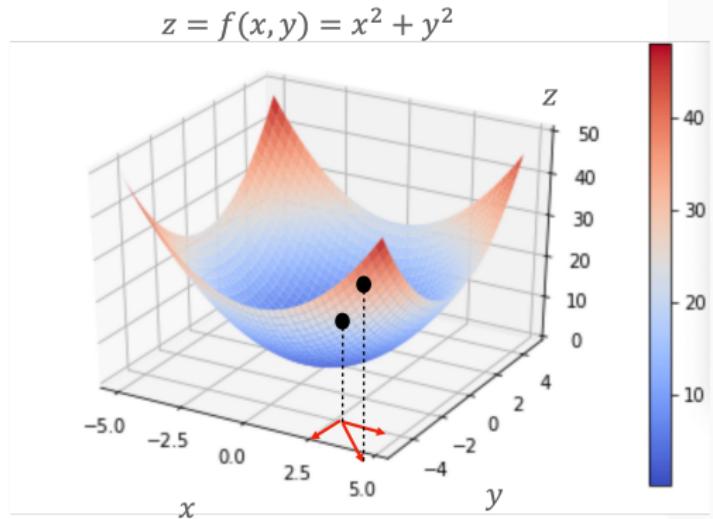
- derivative  $a$ : rate of change (slope) at  $x_1$
- can be treated as a vector from origin along  $x$  dim.
- Move  $x$  along OPPOSITE direction of the vector: decrease  $f(x)$



# Gradient

$$G = \left[ \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right]$$

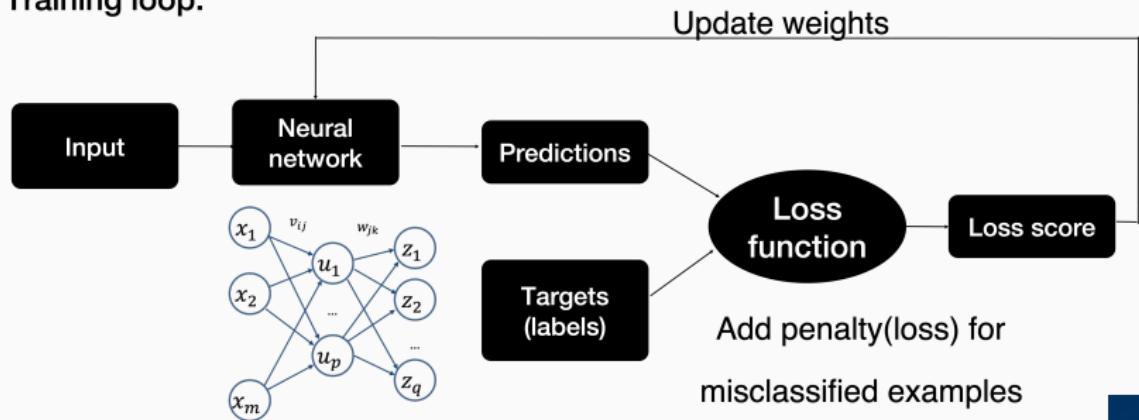
Move input  $(x,y)$  along  
OPPOSITE direction of  
gradient vector:  
decrease output  $f(x,y)$



# “Training”: adjust weights to minimise loss

- Loss = function (weights)
- Gradient descend (opposite direction of gradient vector) to minimize loss

Training loop:



## “Training”: adjust weights to minimise loss

To reduce loss, update weights:

$$w_i^{new} = w_i^{old} - \eta * \Delta L(w_i)$$

$$\Delta L(w_i) = \frac{\partial L}{\partial w_i}$$

$\eta$ : learning rate

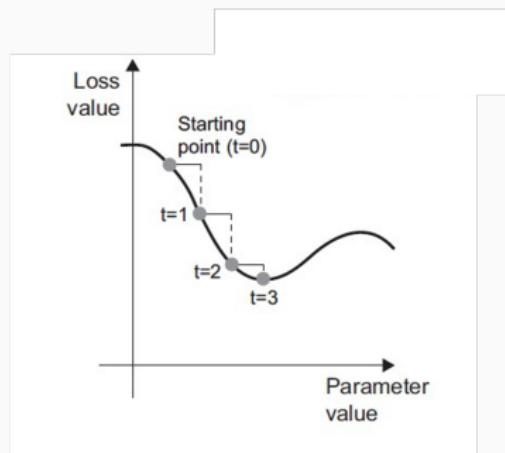


Figure 2.11 in Deep learning with python by Francois Chollet



## “Training”: adjust weights to minimise loss

$\eta$ : learning rate

Small  $\eta$ : local optimal value

Large  $\eta$ : random location

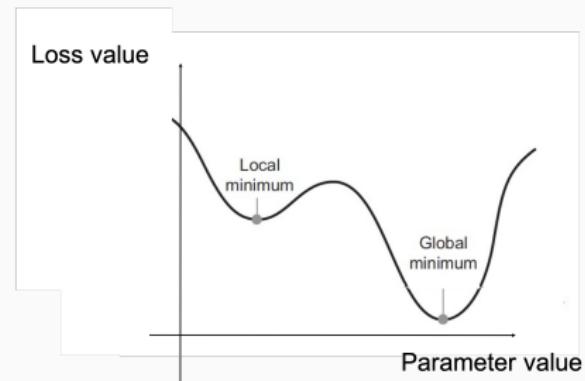


Figure 2.13 in Deep learning with python by Francois Chollet



## Gradient Descent Algorithm

- Randomly shuffle/split all training examples in  $B$  batches
- Choose initial  $\theta^{(1)}$
- For  $i$  from 1 to  $T$
- For  $j$  from 1 to  $B$
- Do gradient descent update using data from batch  $j$
- Advantage of such an approach: computational feasibility for large datasets

Iterations over the entire dataset are called *epochs*



## Stochastic Gradient Descent Algorithm

Choose initial guess  $\mathbf{w}^{(0)}$ ,  $k = 0$

For  $i$  from 1 to  $T$  (epochs)

    For  $j$  from 1 to  $N$  (training examples)

        Consider example  $\{\mathbf{x}_j, y_j\}$

Update\*:  $\mathbf{w}^{(k++)} = \mathbf{w}^{(k)} - \eta \nabla L(\mathbf{w}^{(k)})$



## **Chaining derivatives: the Backpropagation algorithm**

---

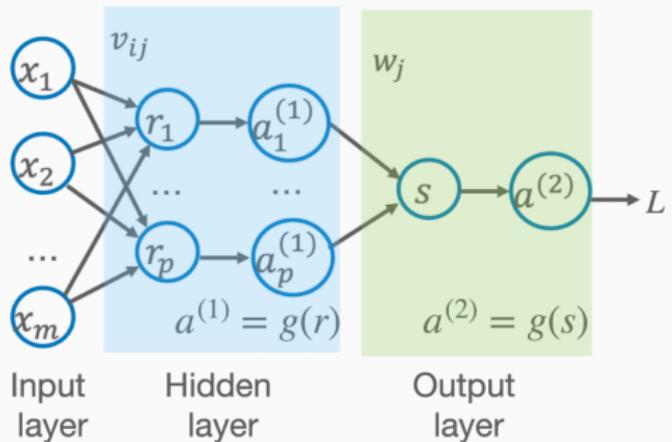
# Multilayer perceptron

$$r_j = \sum_{i=0}^m x_i v_{ij}$$

$$s = \sum_{j=0}^p a_j^{(1)} w_j$$

$a^{(i)}$  : Activation the  $i^{th}$  layer

$g()$  : Activation function



## Chain rule

Given  $z = g(u)$      $u = f(x)$



$$\frac{dz}{dx} = \frac{dz}{du} \frac{du}{dx}$$

Example :  $z = \sin(x^2)$

$$z = \sin(u)$$



$$u = x^2$$

$$\frac{dz}{du} = \cos(u)$$



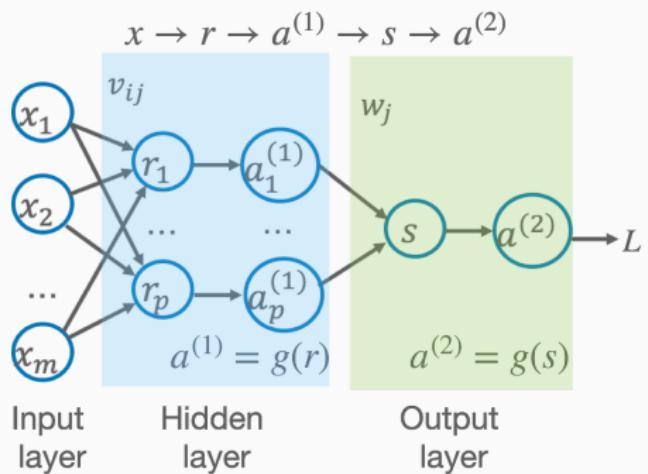
$$\frac{dz}{dx} = \frac{dz}{du} \frac{du}{dx} = 2x \cos(u)$$



# Multilayer perceptron: Forward pass

Forward prediction

$$\begin{aligned} a^{(2)} &= g(s) \\ s &= \sum_{j=0}^p w_j a_j^{(1)} \\ a_j^{(1)} &= g(r_j) \\ r_j &= \sum_{i=0}^m x_i v_{ij} \end{aligned}$$



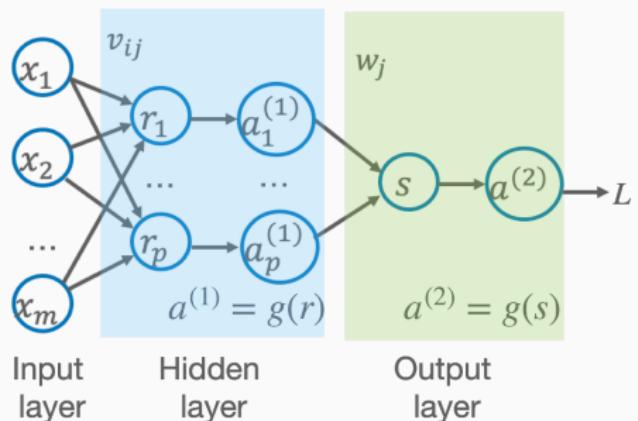
# Multilayer perceptron: Loss

Regression task:

$$L = \frac{1}{2}(a^{(2)} - y)^2$$

$y$ : Ground-truth label

$$x \rightarrow r \rightarrow a^{(1)} \rightarrow s \rightarrow a^{(2)} \rightarrow L$$



# Multilayer perceptron: Gradient I

$$(a^{(2)} - y)$$

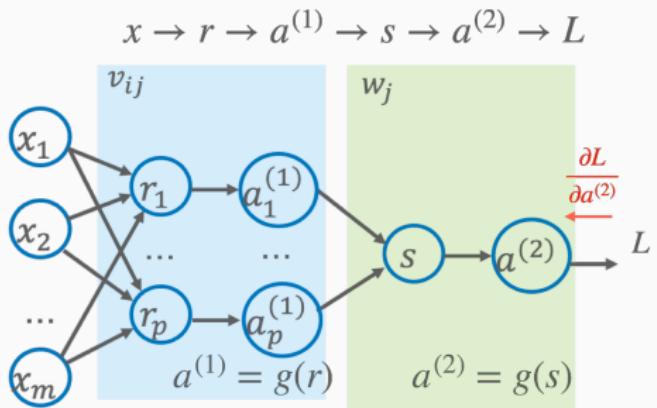
$$\frac{\partial L}{\partial a^{(2)}}$$

$$L = \frac{1}{2}(a^{(2)} - y)^2$$

$$a^{(2)} = g(s)$$

$$s = \sum_{j=0}^p a_j^{(1)} w_j$$

Forward



# Multilayer perceptron: Gradient I

$$(a^{(2)} - y)g'$$

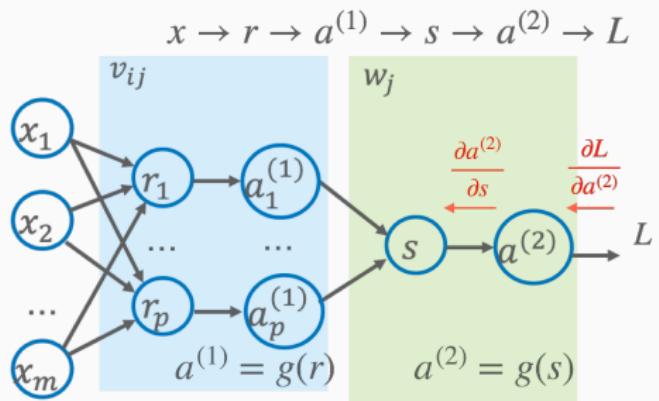
$$\frac{\partial L}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial s}$$

$$L = \frac{1}{2}(a^{(2)} - y)^2$$

$$a^{(2)} = g(s)$$

$$s = \sum_{j=0}^p a_j^{(1)} w_j$$

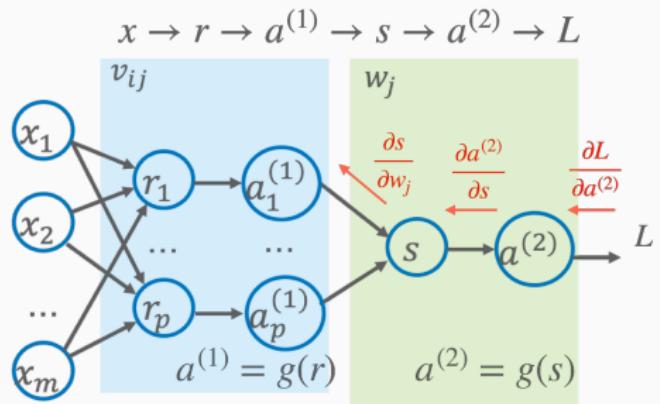
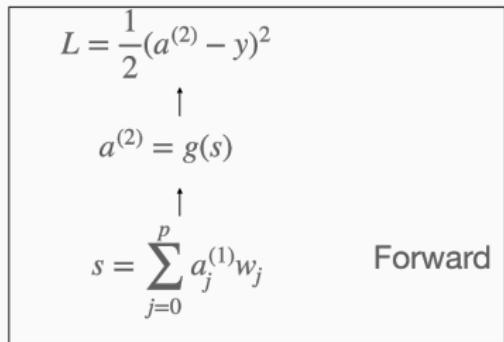
Forward



THE UNIVERSITY OF  
MELBOURNE

# Multilayer perceptron: Gradient I

$$(a^{(2)} - y)g' a_j^{(1)}$$
$$\frac{\partial L}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial s} \frac{\partial s}{\partial w_j}$$



# Multilayer perceptron: Gradient I

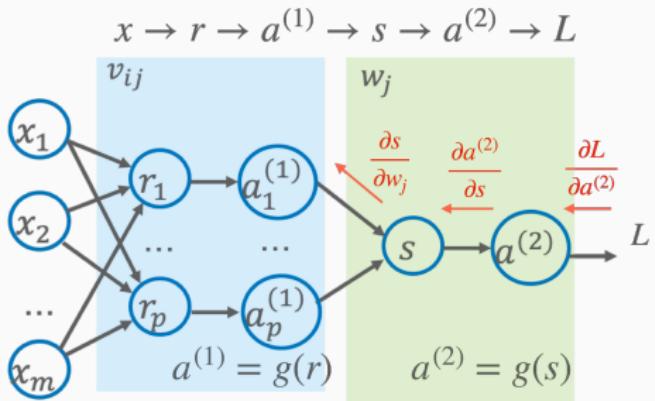
$$\begin{aligned}\frac{\partial L}{\partial w_j} &= (a^{(2)} - y) g' a_j^{(1)} \\ &= \frac{\partial L}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial s} \frac{\partial s}{\partial w_j}\end{aligned}$$

$$L = \frac{1}{2}(a^{(2)} - y)^2$$

$$\begin{array}{c} \uparrow \\ a^{(2)} = g(s) \end{array}$$

$$s = \sum_{j=0}^p a_j^{(1)} w_j$$

Forward



# Multilayer perceptron: Update weights

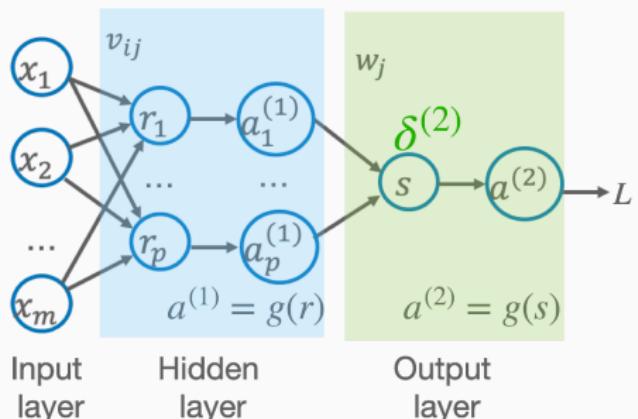
To reduce loss, update each weight in each layer:

$$\frac{\partial L}{\partial w_j} = (a^{(2)} - y)g' a_j^{(1)} = \delta^{(2)} a_j^{(1)}$$

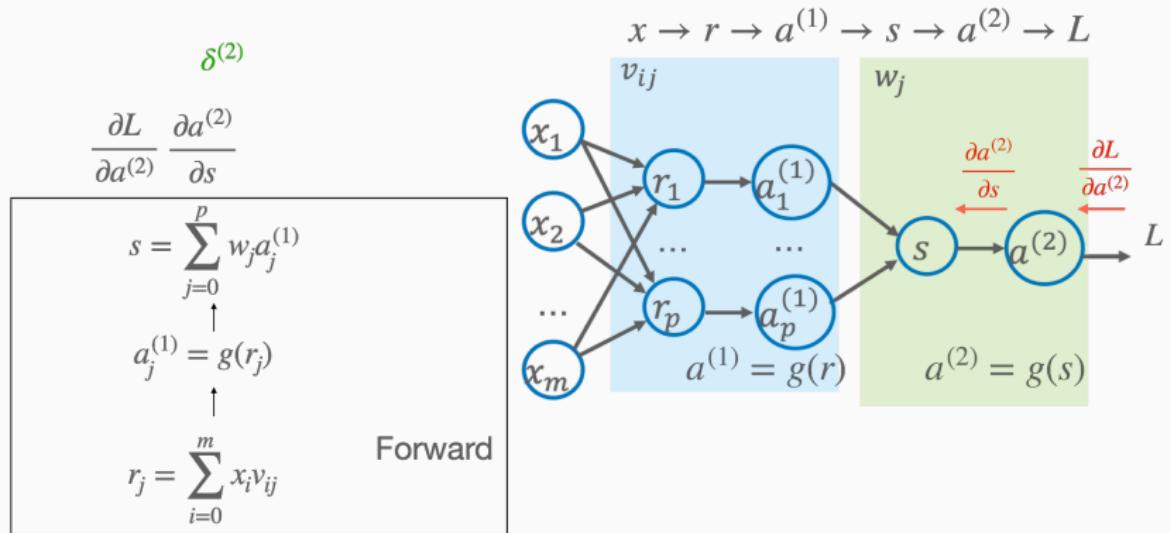
$$\begin{aligned} w_j^{new} &= w_j^{old} - \eta * \frac{\partial L}{\partial w_j} \\ &= w_j^{old} - \eta \delta^{(2)} a_j^{(1)} \end{aligned}$$

$$\delta^{(2)} = \frac{\partial L}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial s} = (a^{(2)} - y)g'$$

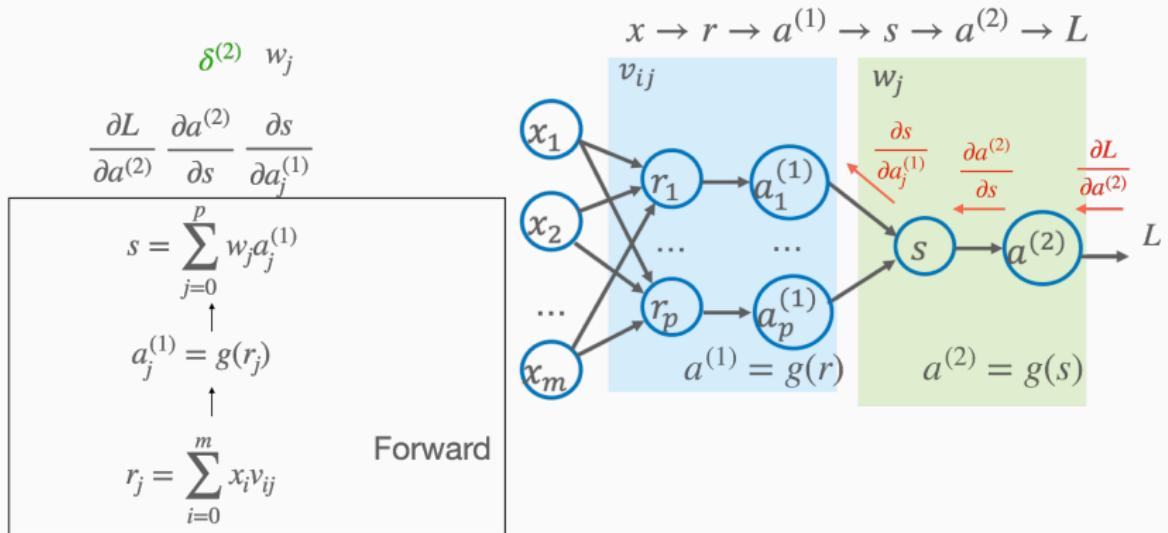
(Error at the output layer)



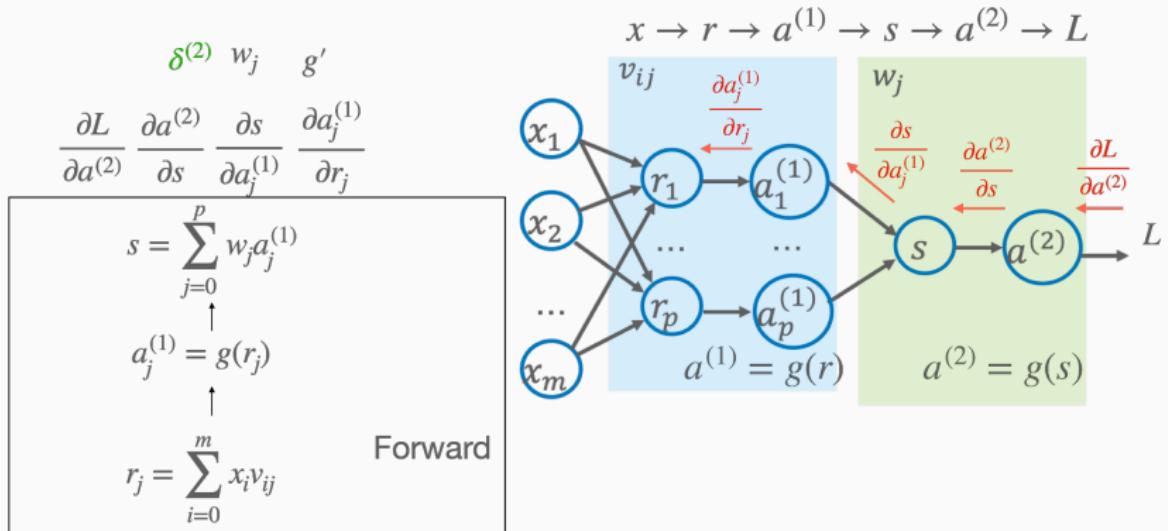
# Multilayer perceptron: Gradient II



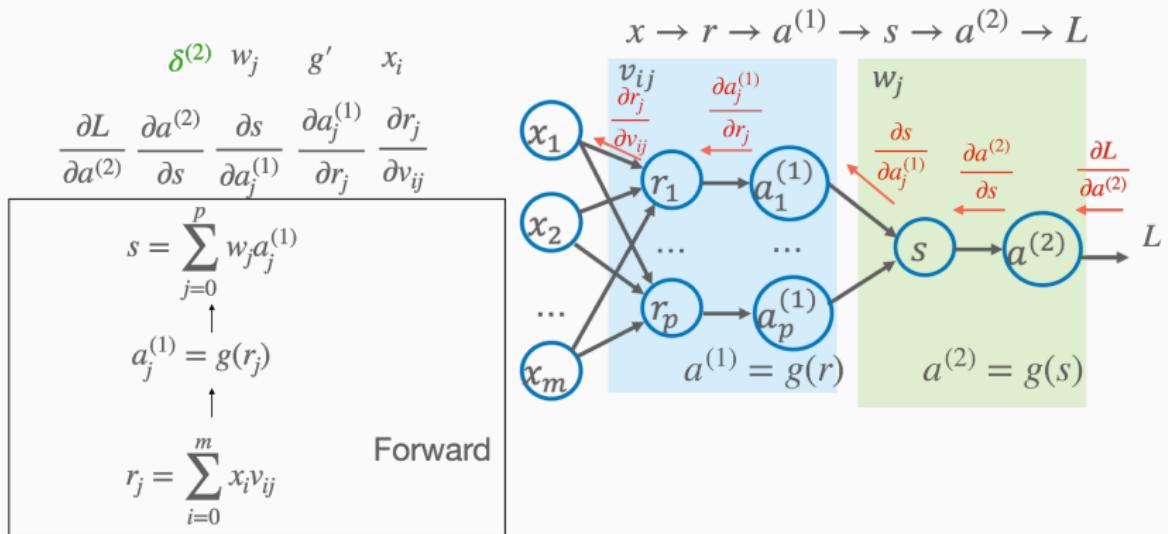
# Multilayer perceptron: Gradient II



# Multilayer perceptron: Gradient II



# Multilayer perceptron: Gradient II



# Multilayer perceptron: Gradient II

$$\frac{\partial L}{\partial v_{ij}} = \delta^{(2)} w_j g' x_i$$

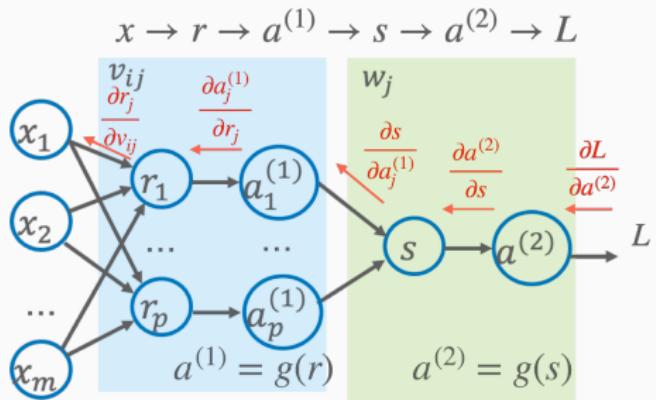
$$= \frac{\partial L}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial s} \frac{\partial s}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial r_j} \frac{\partial r_j}{\partial v_{ij}}$$

Forward

$$s = \sum_{j=0}^p w_j a_j^{(1)}$$

$$a_j^{(1)} = g(r_j)$$

$$r_j = \sum_{i=0}^m x_i v_{ij}$$



# Multilayer perceptron: Update weights

To reduce loss, update each weight in each layer:

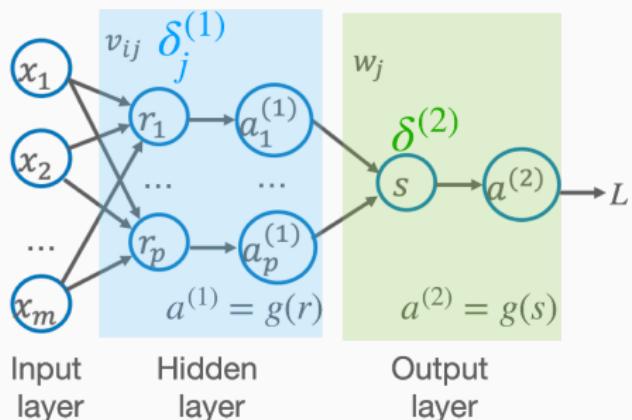
$$\frac{\partial L}{\partial v_{ij}} = \delta_j^{(2)} w_j g' x_i = \delta_j^{(1)} x_i$$

$$v_{ij}^{new} = v_{ij}^{old} - \eta * \frac{\partial L}{\partial v_{ij}}$$

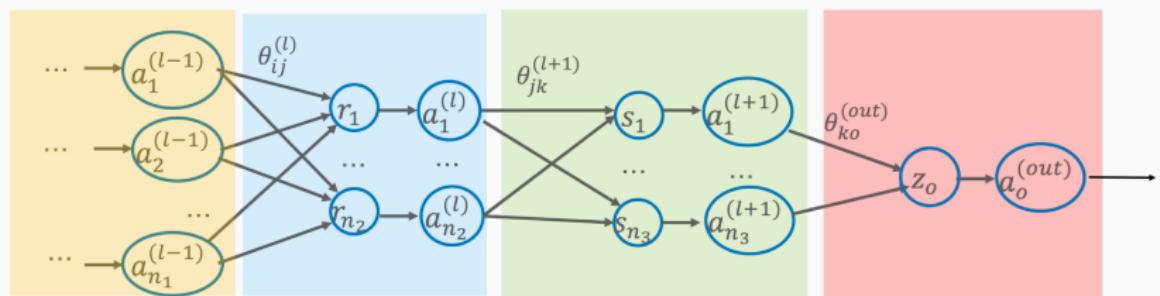
$$= v_{ij}^{old} - \eta \delta_j^{(1)} x_i$$

$$\delta^{(2)} = \frac{\partial L}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial s} = (a^{(2)} - y) g'$$

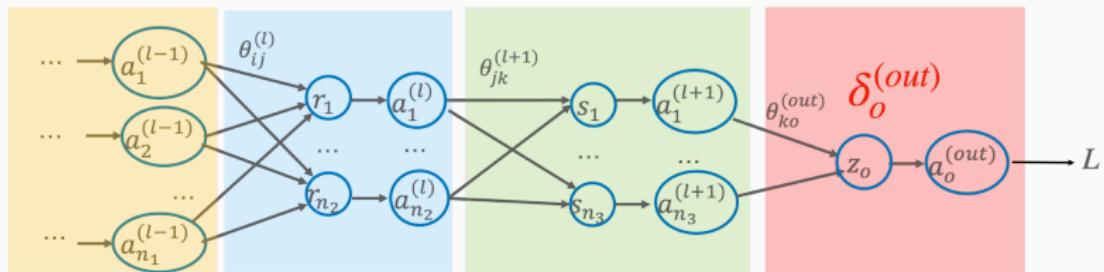
$$\delta_j^{(1)} = \delta^{(2)} w_j g' \quad (\text{error backpropagation})$$



## Multilayer perceptron: more hidden layers



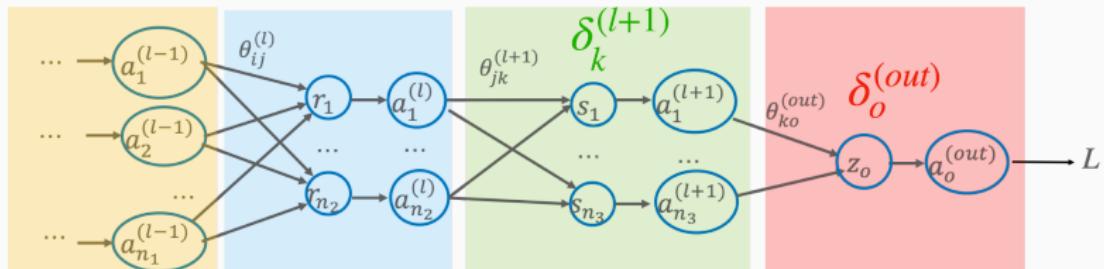
## Backpropagation: error of output layer



$$\delta_o^{out} = \frac{\partial L}{\partial a_o^{(out)}} \frac{\partial a^{(out)_o}}{\partial z_o} = \frac{\partial L}{\partial a_o^{(out)}} g'$$

$$\theta_{ko}^{(out_{new})} = \theta_{ko}^{(out_{old})} - \eta \delta_o^{(out)} a_k^{(l+1)}$$

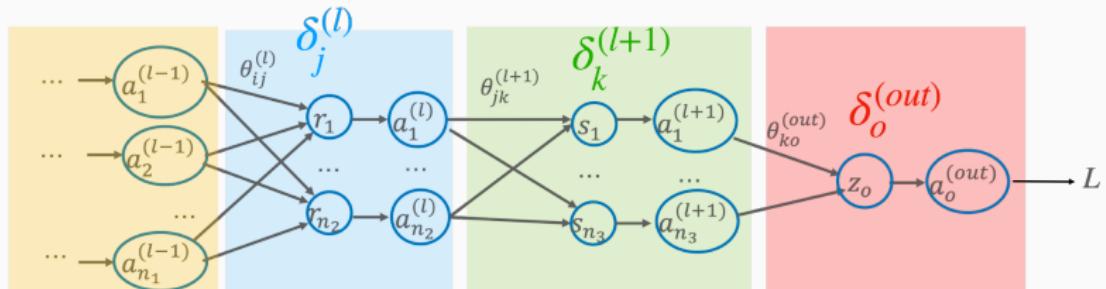
## Backpropagation: error of hidden layer



$$\delta_k^{(l+1)} = \sum_o \delta_o^{(out)} \theta_{ko}^{(out)} g' \quad \delta_o^{out} = \frac{\partial L}{\partial a_o^{(out)}} \frac{\partial a^{(out)_o}}{\partial z_o} = \frac{\partial L}{\partial a_o^{(out)}} g'$$

$$\theta_{jk}^{(l+1_{new})} = \theta_{jk}^{(l+1_{old})} - \eta \delta_k^{(l+1)} a_j^{(l)} \quad \theta_{ko}^{(out_{new})} = \theta_{ko}^{(out_{old})} - \eta \delta_o^{(out)} a_k^{(l+1)}$$

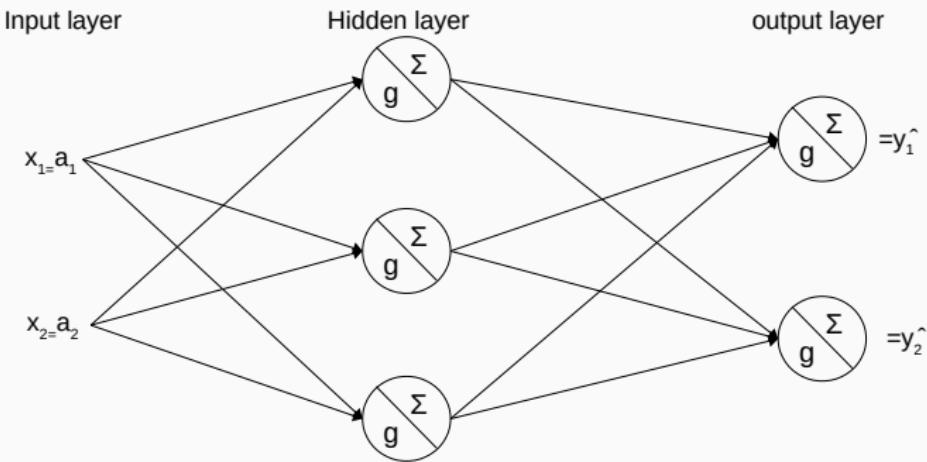
# Backpropagation: error of hidden layer



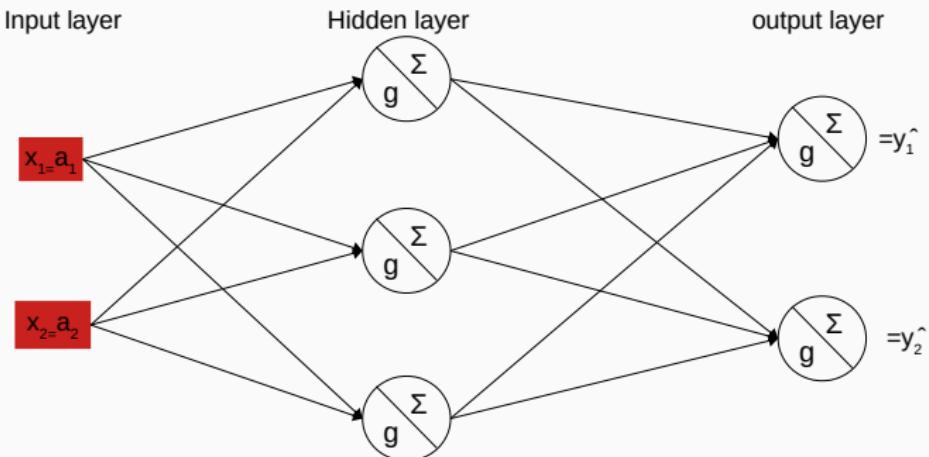
$$\delta_j^{(l)} = \sum_{k=1}^{n_3} \delta_k^{(l+1)} \theta_{jk}^{(l+1)} g' \quad \delta_k^{(l+1)} = \sum_o \delta_o^{(out)} \theta_{ko}^{(out)} g' \quad \delta_o^{(out)} = \frac{\partial L}{\partial a_o^{(out)}} - \frac{\partial a_o^{(out)} \circ}{\partial z_o} = \frac{\partial L}{\partial a_o^{(out)}} g'$$

$$\theta_{ij}^{(l_{new})} = \theta_{ij}^{(l_{old})} - \eta \delta_j^{(l)} a_i^{(l-1)} \quad \theta_{jk}^{(l+1_{new})} = \theta_{jk}^{(l+1_{old})} - \eta \delta_k^{(l+1)} a_j^{(l)} \quad \theta_{ko}^{(out_{new})} = \theta_{ko}^{(out_{old})} - \eta \delta_o^{(out)} a_k^{(l+1)}$$

# Backpropagation: Demo

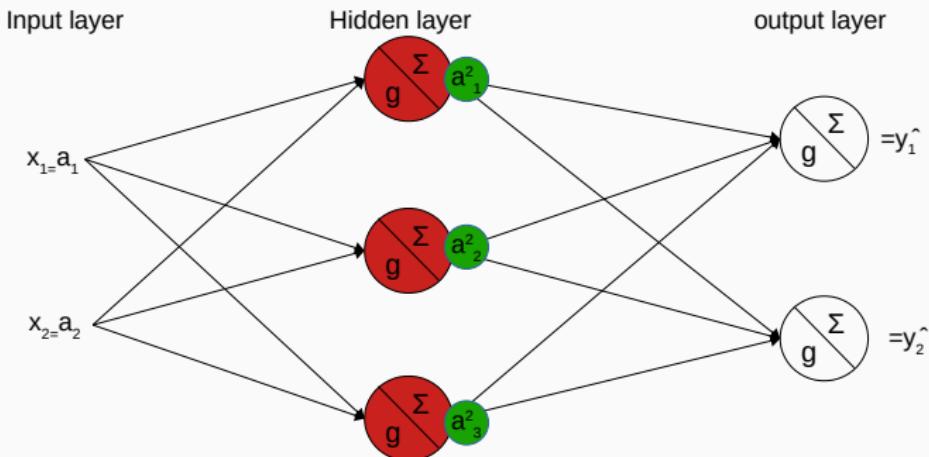


# Backpropagation: Demo



- Receive input

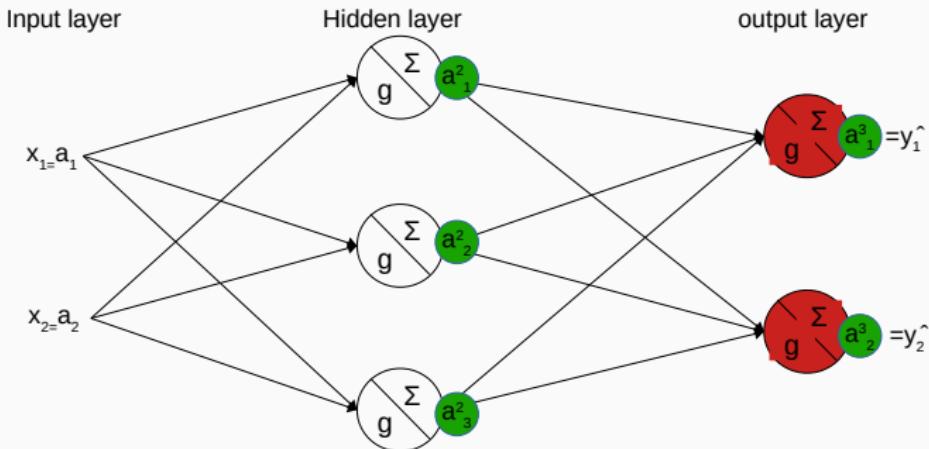
# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network

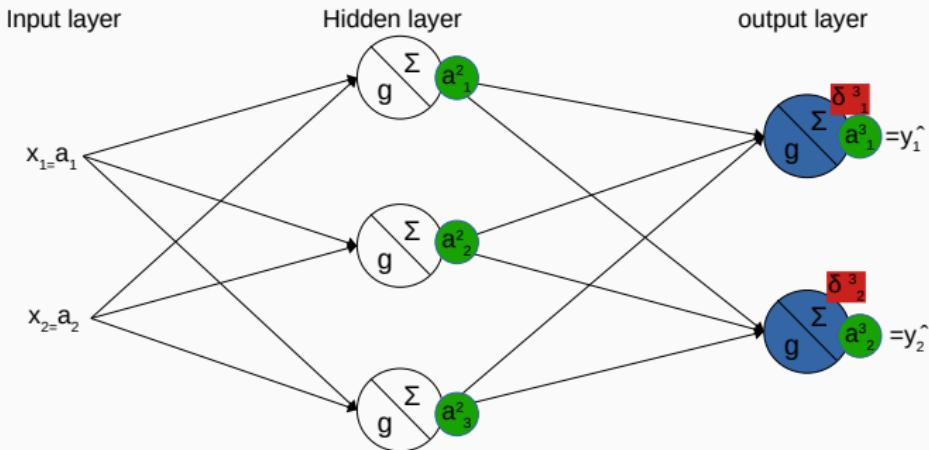


# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network

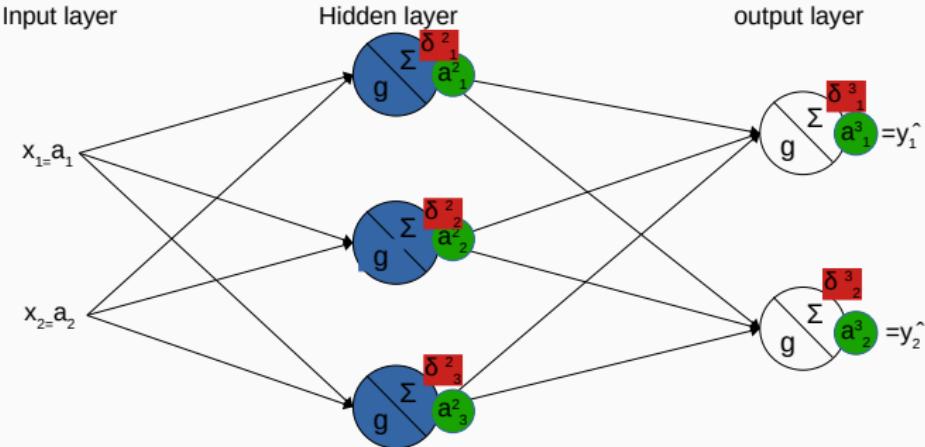
# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error of output layer:  $\delta_o^{out} = \frac{\partial L}{\partial a_o^{(out)}} g'$

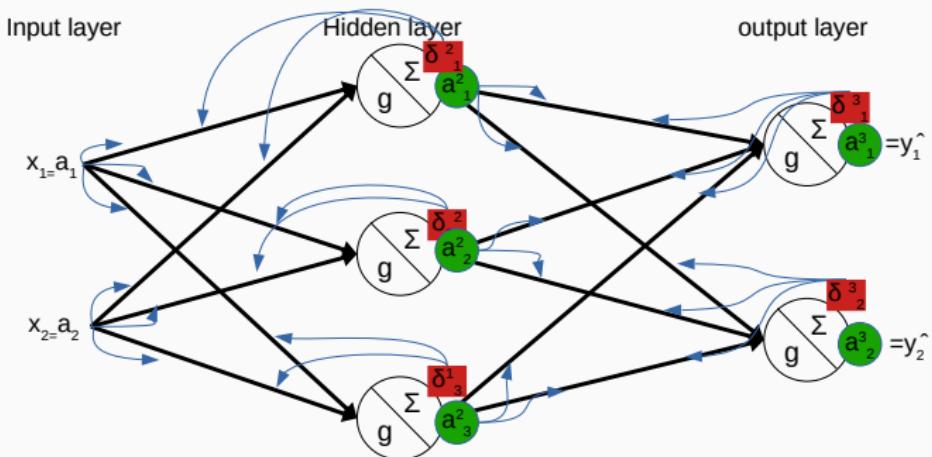


# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error of output layer:  $\delta_o^{out} = \frac{\partial L}{\partial a_o^{(out)}} g'$
- Backward pass: propagate error terms through the network  
$$\delta_j^{(l)} = \sum_{k=1}^{n_{l+1}} \delta_k^{(l+1)} \theta_{jk}^{(l+1)} g'$$

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error of output layer:  $\delta_o^{out} = \frac{\partial L}{\partial a_o^{(out)}} g'$
- Backward pass: propagate error terms through the network
  - $\delta_j^{(l)} = \sum_{k=1}^{n_{l+1}} \delta_k^{(l+1)} \theta_{jk}^{(l+1)} g'$
  - Calculate  $\frac{\partial L}{\partial \theta_{ij}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)}$  for all  $\theta_{ij}^{(l)}$
- Update weights  $\theta_{ij}^{(l)} \leftarrow \theta_{ij}^{(l)} - \eta \frac{\partial L}{\partial \theta_{ij}^{(l)}}$

# Backpropagation Algorithm

---

Design your neural network

Initialize parameters  $\theta$

**repeat**

**for** training instance  $x_i$  **do**

1. **Forward pass** the instance through the network, compute activations, determine output

2. Compute the **error**  $\delta_o^{out} = \frac{\partial L}{\partial a_o^{(out)}} g'$

3. Propagate error **back** through the network, and compute for all weights between nodes  $ij$  in all layers  $I$

$$\delta_j^{(I)} = \sum_{k=1}^{n_{I+1}} \delta_k^{(I+1)} \theta_{jk}^{(I+1)} g'$$

4. Update **all** parameters **at once**

$$\theta_{ij}^{(I)} \leftarrow \theta_{ij}^{(I)} - \eta \frac{\partial L}{\partial \theta_{ij}^{(I)}} = \theta_{ij}^{(I)} - \eta \delta_j^{(I)} a_i^{(I-1)}$$

**until** stopping criteria reached.



## After this lecture, you be able to understand

- How to use Gradient Descent to optimize the parameters for neural network
- How Backpropagation allows us to efficiently compute the gradients of all weights wrt. the error in Multilayer perceptron



# Lecture 14: Decision Trees

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne

Acknowledgement: Jeremy Nicholson, Tim Baldwin & Karin Verspoor



# Roadmap

So far:

- Naive Bayes: Features are conditionally independent.
- Logistic Regression: Linear classifier
- Neural Network: Lack of interpretability
- KNN: time-consuming during testing

Today:

- Decision Trees
- ID3 Algorithm
  - Information Gain
  - Gain Ratio
- Properties of Decision Trees



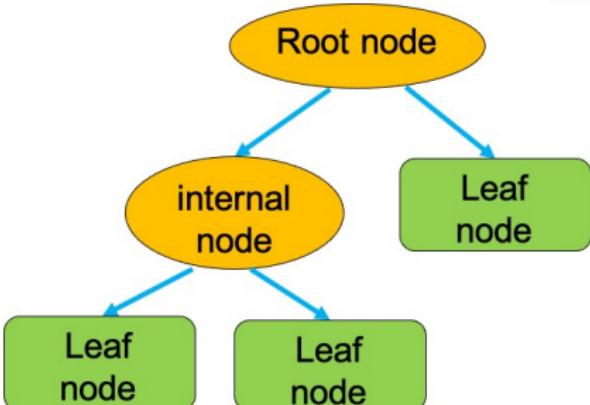
## **Decision Trees**

---

# What Are Decision Trees

Tree-like graphical representation:

- Root/Internal Node: a test on an attribute
- Branch: outcome of the test
- Leaf: class



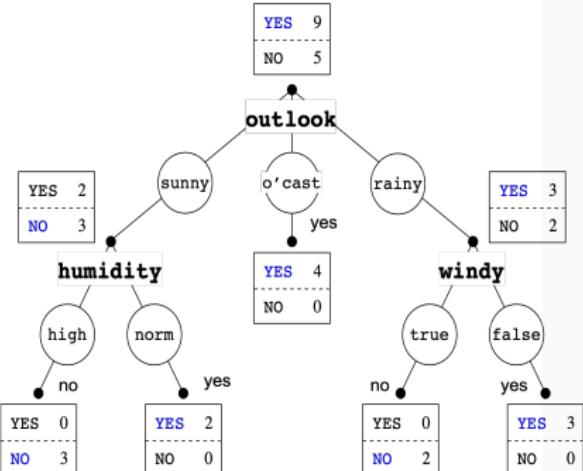
## Example

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no
g	overcast	cool	normal	TRUE	yes
h	sunny	mild	high	FALSE	no
i	sunny	cool	normal	FALSE	yes
j	rainy	mild	normal	FALSE	yes
k	sunny	mild	normal	TRUE	yes
l	overcast	mild	high	TRUE	yes
m	overcast	hot	normal	FALSE	yes
n	rainy	mild	high	TRUE	no



## Example

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no
g	overcast	cool	normal	TRUE	yes
h	sunny	mild	high	FALSE	no
i	sunny	cool	normal	FALSE	yes
j	rainy	mild	normal	FALSE	yes
k	sunny	mild	normal	TRUE	yes
l	overcast	mild	high	TRUE	yes
m	overcast	hot	normal	FALSE	yes
n	rainy	mild	high	TRUE	no

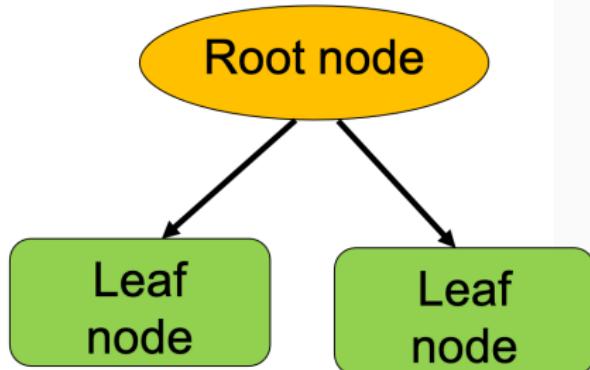


## Majority voting to assign class



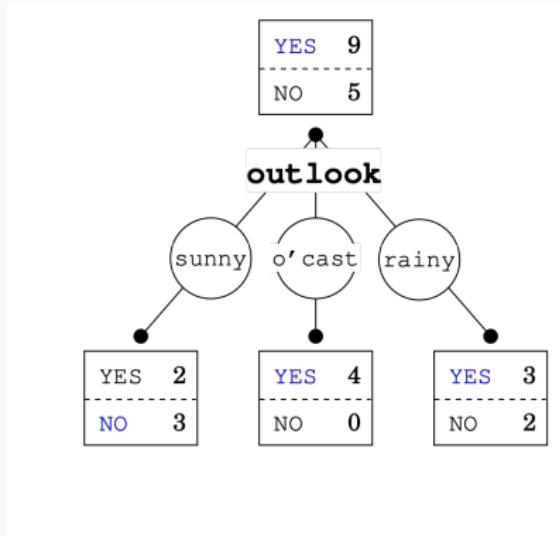
## Decision Stump

One-level decision tree: leaves immediately connect to the root



## Example

	Outlook	Play
a	sunny	no
b	sunny	no
c	overcast	yes
d	rainy	yes
e	rainy	yes
f	rainy	no
g	overcast	yes
h	sunny	no
i	sunny	yes
j	rainy	yes
k	sunny	yes
l	overcast	yes
m	overcast	yes
n	rainy	no



# How to Test Decision Trees

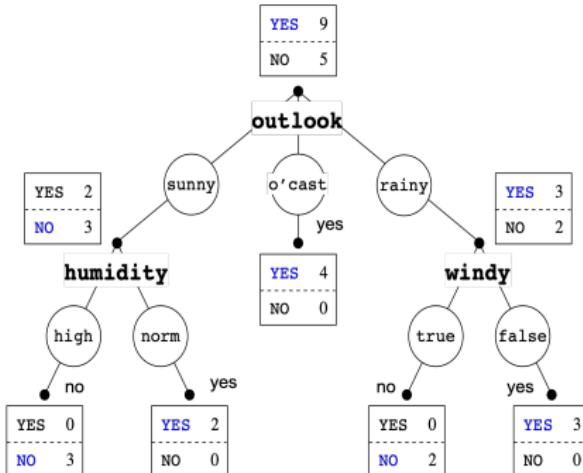
Traverse from root to leaf. Example:

- (sunny, hot, normal, false)
- (rainy, hot, low, false)

Missing values: try all attribute values, then majority voting.

Example:

- (?, cool, high, true)
- (?, hot, norm, false)



# Applications

Examples:

- Business management: predict customer's intention to use e-commerce (Lee et al. 2007)
- Engineering: predict electricity energy consumption (Tso et al. 2007)
- Healthcare Management: predict breast cancer survivability. (Delen et al. 2005)

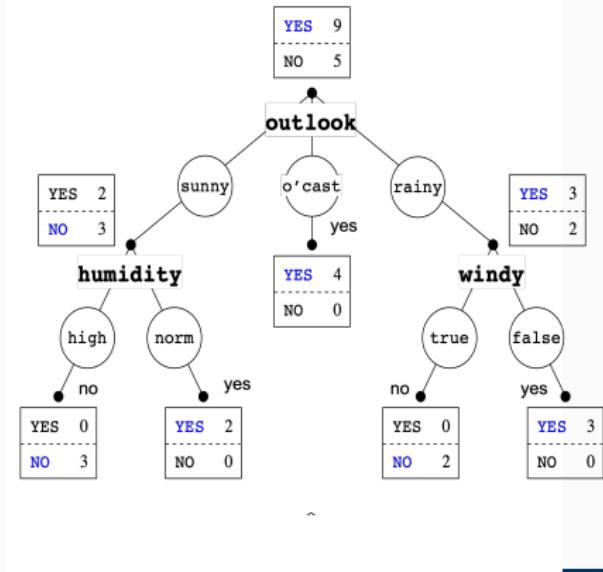


## **ID3 Algorithm**

---

# Constructing Decision Trees

1. Select a new attribute.
2. Create a branch for each attribute value to partition the node instances.
3. Check if all instances at sub-nodes have same class or all attributes are run out:
  - if so, stop.
  - If not, go back to step 1 and repeat the process.

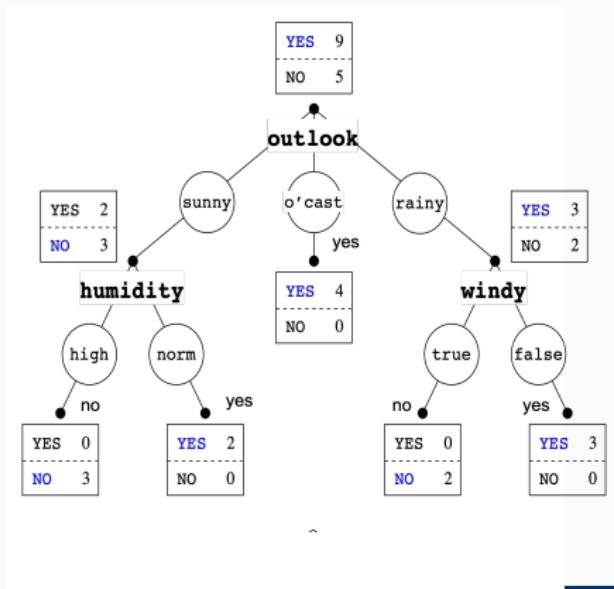


# Constructing Decision Trees

1. Select a new attribute.
2. Create a branch for each attribute value to partition the node instances.
3. Check if all instances at sub-nodes have same class or all attributes are run out:
  - if so, stop.
  - If not, go back to step 1 and repeat the process.

## How to select new attribute?

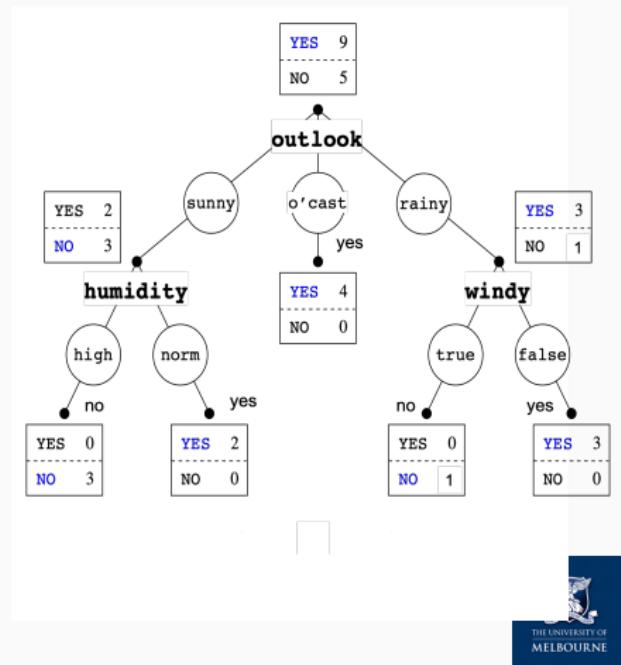
Choose attributes that can lead to a small tree



# Smaller Trees vs Larger Trees

A smaller tree that fits the data generalizes better.

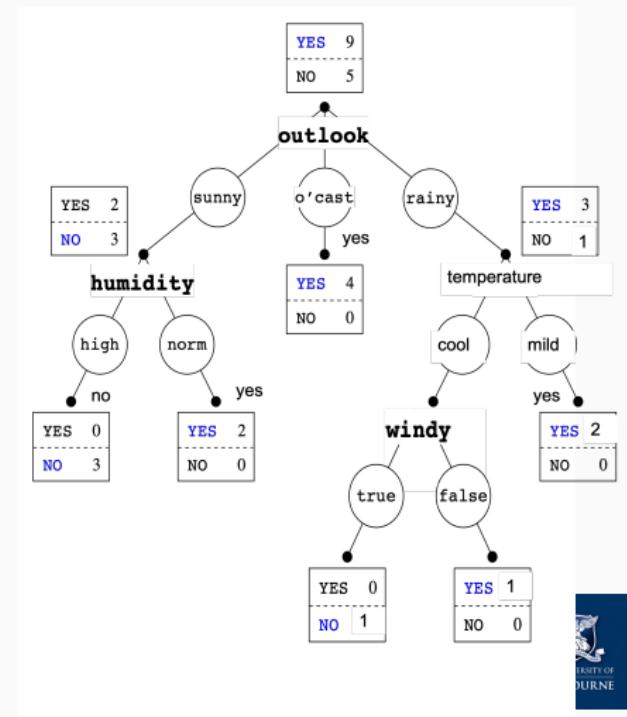
Train/ test	label	Outlook	Temperature	Humidity	Windy	Play
train	a	sunny	hot	high	FALSE	no
train	b	sunny	hot	high	TRUE	no
train	c	overcast	hot	high	FALSE	yes
train	d	rainy	mild	high	FALSE	yes
train	e	rainy	cool	normal	FALSE	yes
train	f	rainy	cool	normal	TRUE	no
train	g	overcast	cool	normal	TRUE	yes
train	h	sunny	mild	high	FALSE	no
train	i	sunny	cool	normal	FALSE	yes
train	j	rainy	mild	normal	FALSE	yes
train	k	sunny	mild	normal	TRUE	yes
train	l	overcast	mild	high	TRUE	yes
train	m	overcast	hot	normal	FALSE	yes
test	n	rainy	mild	high	TRUE	no



# Smaller Trees vs Larger Trees

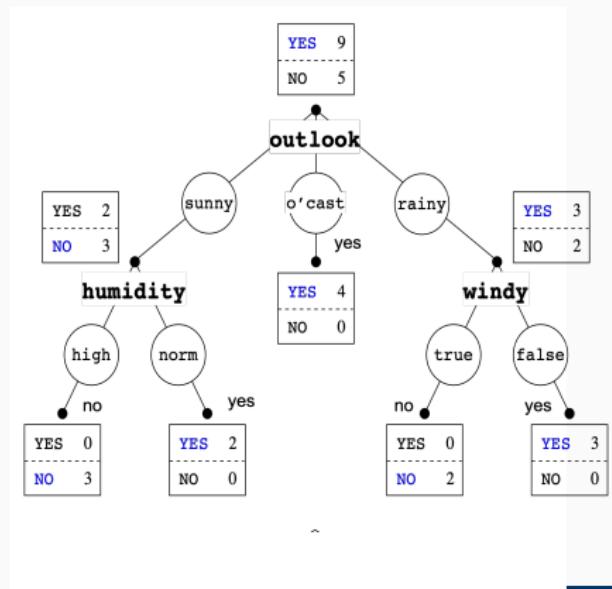
A smaller tree that fits the data generalizes better.

Train/ test	label	Outlook	Temperature	Humidity	Windy	Play
train	a	sunny	hot	high	FALSE	no
train	b	sunny	hot	high	TRUE	no
train	c	overcast	hot	high	FALSE	yes
train	d	rainy	mild	high	FALSE	yes
train	e	rainy	cool	normal	FALSE	yes
train	f	rainy	cool	normal	TRUE	no
train	g	overcast	cool	normal	TRUE	yes
train	h	sunny	mild	high	FALSE	no
train	i	sunny	cool	normal	FALSE	yes
train	j	rainy	mild	normal	FALSE	yes
train	k	sunny	mild	normal	TRUE	yes
train	l	overcast	mild	high	TRUE	yes
train	m	overcast	hot	normal	FALSE	yes
test	n	rainy	mild	high	TRUE	no



# Criterion for Attribute Selection

Choose an attribute to partition the data at the node such that each partition is as homogeneous (least impure) as possible to build small trees.

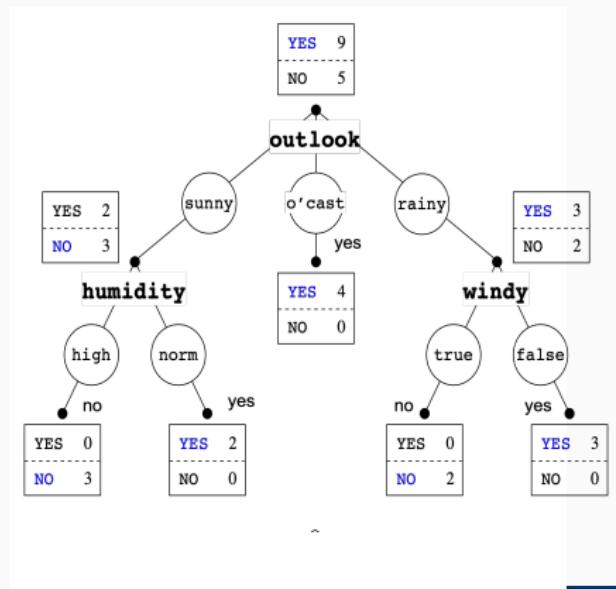


# Criterion for Attribute Selection

Choose an attribute to partition the data at the node such that each partition is as homogeneous (least impure) as possible to build small trees.

How to measure the purity of classes?

Use Entropy!



# Entropy

Entropy is a measure of unpredictability:

- pure class distribution: low entropy
- evenly distributed distribution: high entropy

$$\begin{aligned} H &= \sum_i^n P(i) \text{self\_information}(i) \\ &= \sum_i^n P(i) \log_2 \frac{1}{P(i)} \\ &= - \sum_i^n P(i) \log_2 P(i) \end{aligned}$$

Example:

- Flip a normal coin 100 times, 52 heads, 48 tails:  $H \approx 1$
- Flip a two-headed coin 100 times,  $H = 0$



## Criterion for Attribute Selection: Information Gain

- Select the attribute that has largest information gain
- Information Gain: Reduction of entropy before and after the data is partitioned using an attribute.

$$H(R_A|R) = H(R) - \text{MeanInfo}(x_1, \dots, x_m)$$

$$\text{MeanInfo}(x_1, \dots, x_m) = \sum_i^m w(x_i)H(x_i)$$

- $H(R)$ : entropy of the instances in node R.
- MeanInfo: weighted entropy of the sub-nodes.
- $H(x_i)$ : entropy of the instances in sub-node  $x_i$ .
- $w_i$ : weight (proportion ratio) of each sub-node



## Example

$$H(\text{root}) = - \left( \frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14} \right)$$

$\text{MeanInfo}(\text{outlook})$

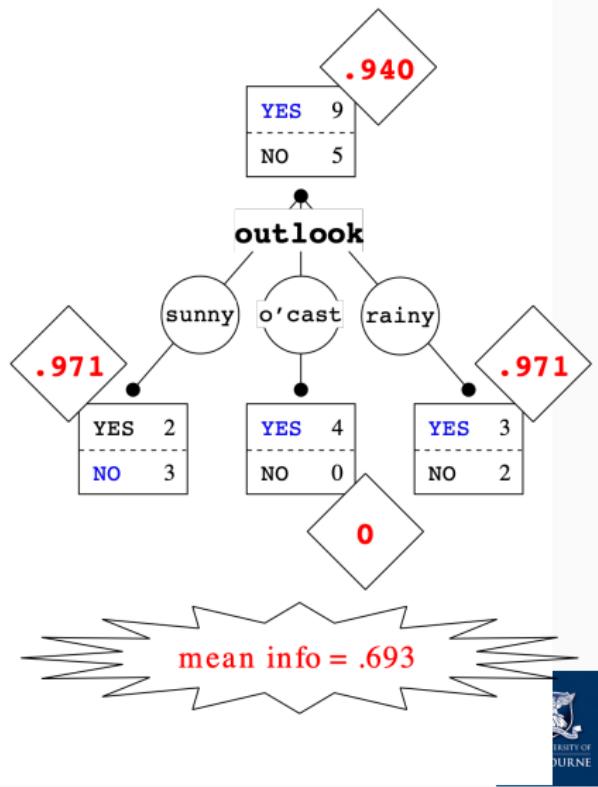
$$= \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971$$

$IG(\text{outlook}|R)$

$$= H(\text{root}) - \text{MeanInfo}(\text{outlook})$$

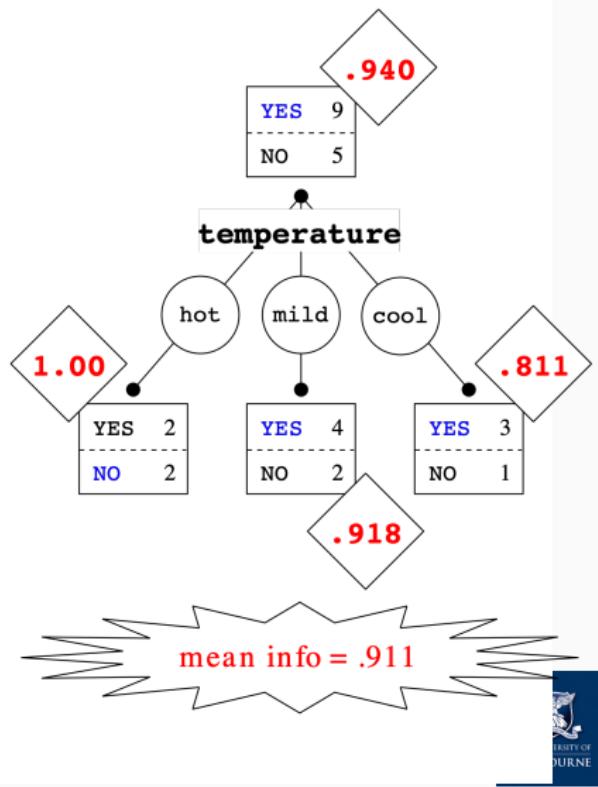
$$= 0.940 - 0.693$$

$$= 0.247$$



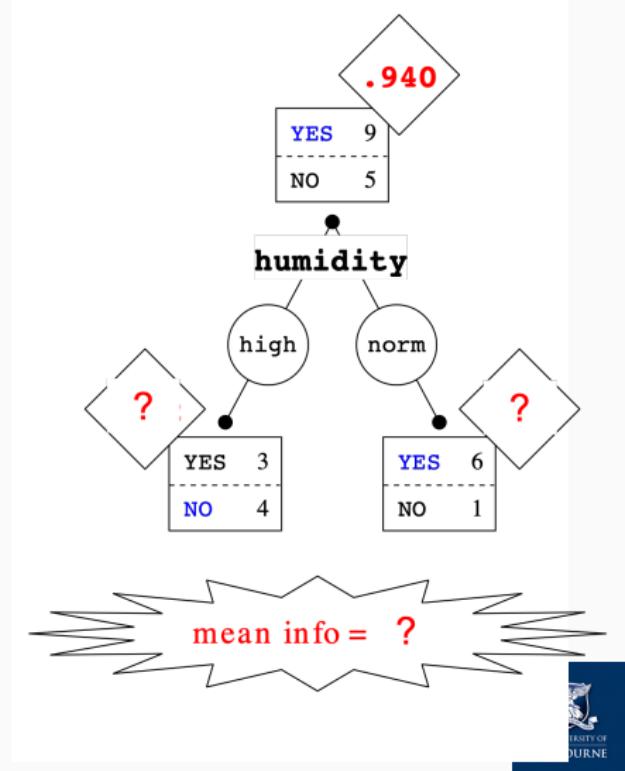
## Example

$$\begin{aligned} & IG(temperature|R) \\ &= H(\text{root}) - \text{MeanInfo}(temperature) \\ &= 0.940 - 0.911 \\ &= 0.029 \end{aligned}$$



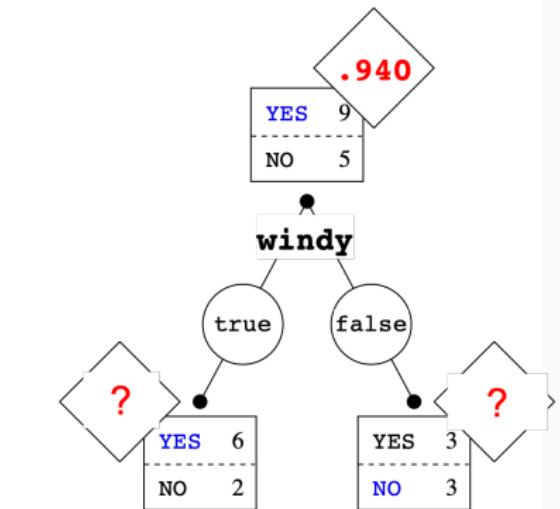
## Example

$$\begin{aligned} & IG(humidity|R) \\ & = H(\text{root}) - \text{MeanInfo}(humidity) \\ &=? \end{aligned}$$



## Example

$$\begin{aligned} & IG(windy|R) \\ &= H(\text{root}) - \text{MeanInfo}(windy) \\ &=? \end{aligned}$$



mean info = . ?



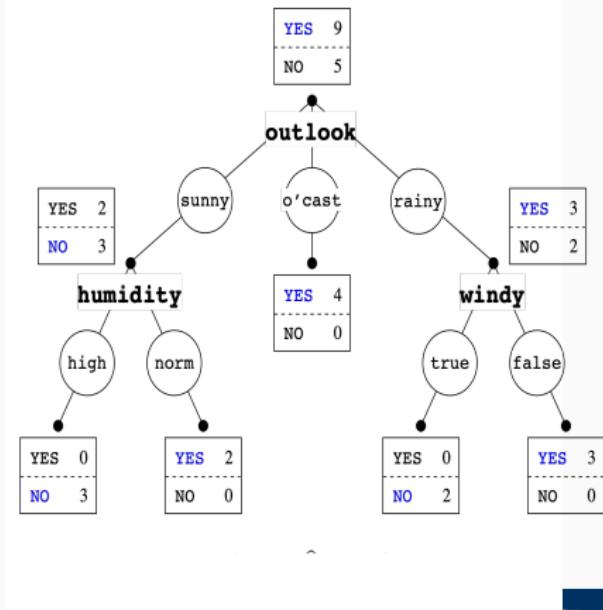
## Example

$$H(\text{sunny}) = 0.971$$

$$IG(\text{humidity}|\text{sunny}) = 0.971$$

$$IG(\text{temperature}|\text{sunny}) = 0.571$$

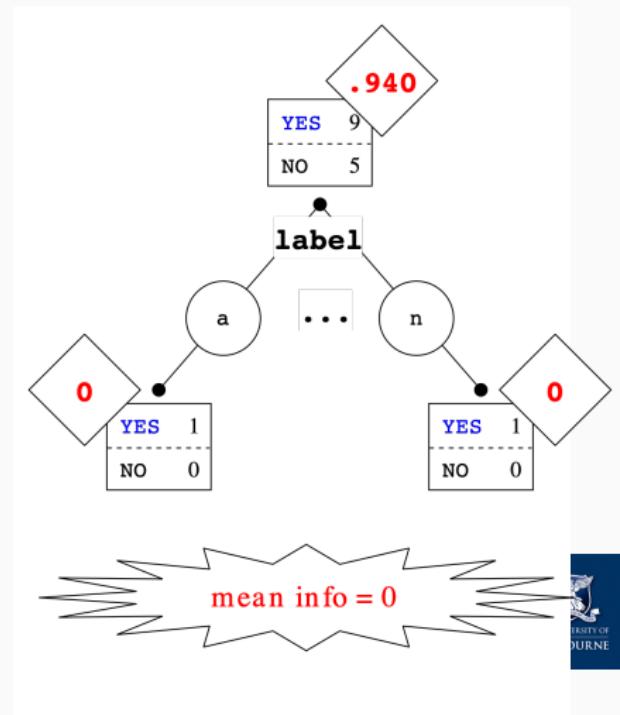
$$IG(\text{windy}|\text{sunny}) = 0.020$$



# Shortcoming of Information Gain

- Information Gain tends to prefer highly-branching attributes
- It may result in overfitting problem.

	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	FALSE	no
b	sunny	hot	high	TRUE	no
c	overcast	hot	high	FALSE	yes
d	rainy	mild	high	FALSE	yes
e	rainy	cool	normal	FALSE	yes
f	rainy	cool	normal	TRUE	no
g	overcast	cool	normal	TRUE	yes
h	sunny	mild	high	FALSE	no
i	sunny	cool	normal	FALSE	yes
j	rainy	mild	normal	FALSE	yes
k	sunny	mild	normal	TRUE	yes
l	overcast	mild	high	TRUE	yes
m	overcast	hot	normal	FALSE	yes
n	rainy	mild	high	TRUE	no



## Solution: Gain Ratio

$$GR(R_A|R) = IG(R_A|R)/SI(R_A|R)$$

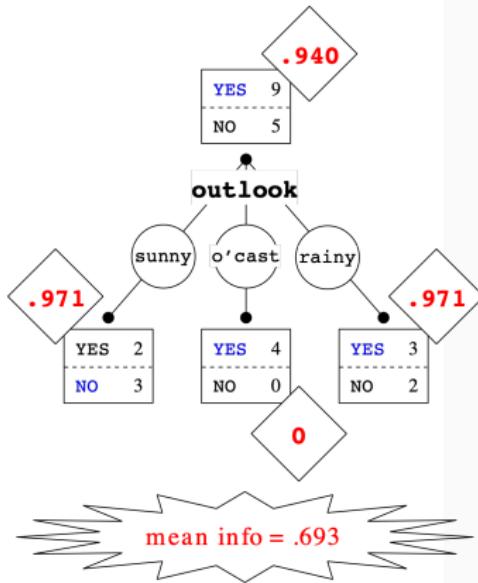
$$SI(R_A|R) = - \sum_i^m w(x_i) \log_2 w(x_i)$$

Split info (SI): entropy of a given split (evenness of split).

$w(x_i)$ : weight (proportion ratio) of each sub-node.

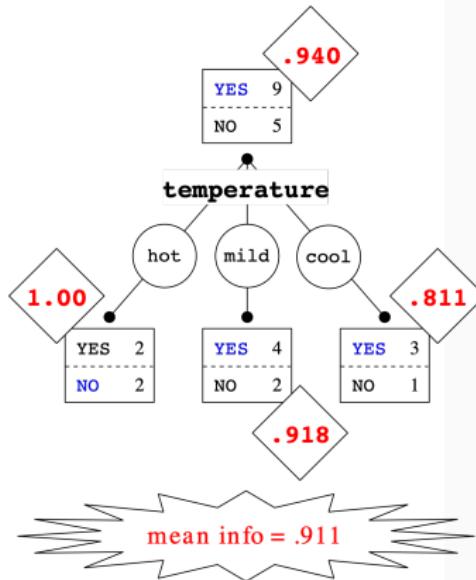


## Example



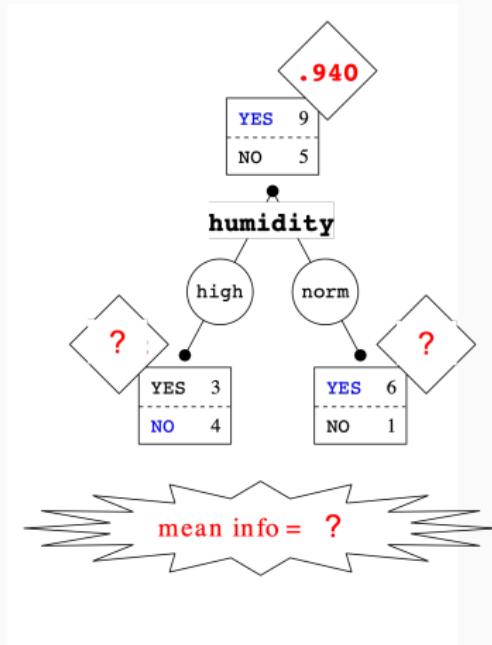
$$SI(outlook|R) = -\left(\frac{5}{14} \times \log_2 \frac{5}{14} + \frac{4}{14} \times \log_2 \frac{4}{14} + \frac{5}{14} \times \log_2 \frac{5}{14}\right)$$
$$GR(outlook|R) = IG(outlook|R)/SI(outlook|R) = 0.247/1.577 = 0.157$$

## Example



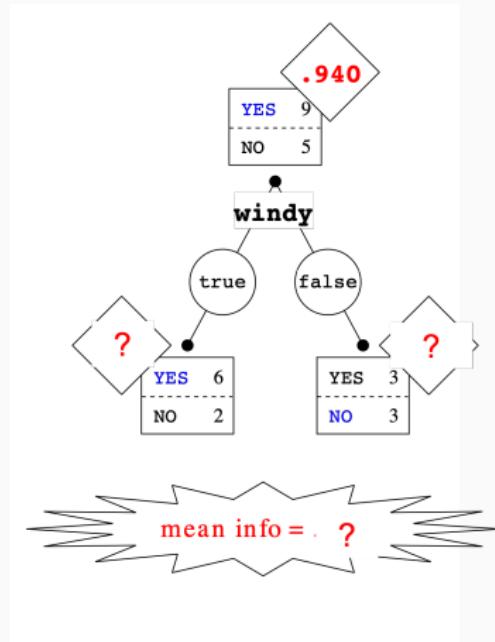
$$GR(\text{temperature}|R) = ?$$

## Example



$$GR(\text{humidity}|R) = ?$$

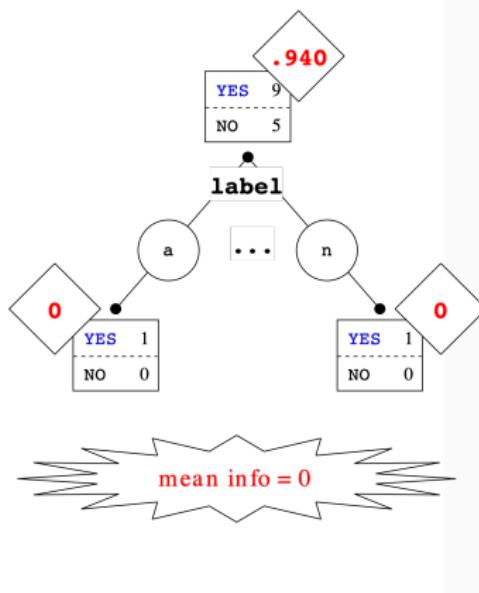
## Example



$$GR(windy|R) = ?$$



## Example



$$GR(windy|R) = ?$$



## **Properties of Decision Trees**

---

## Advantages

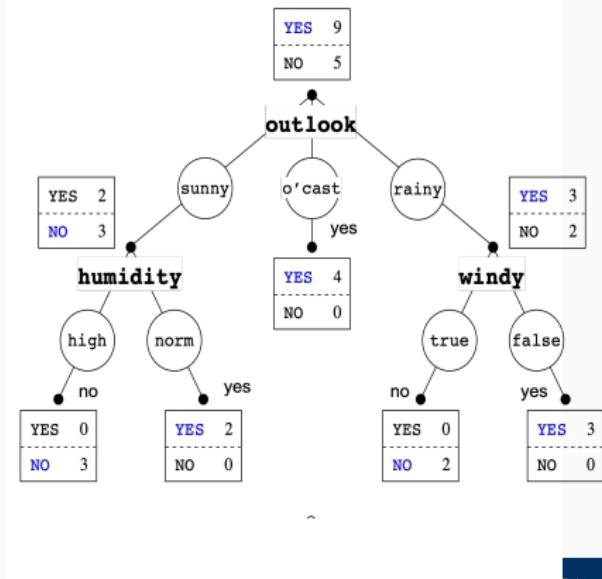
- Easy to read and understand.
- Requires less data preparation.
- Can handle feature with missing values.
- Can be expressed as disjunctive descriptions.  
i.e.,  $(\dots \wedge \dots \wedge \dots) \vee (\dots \wedge \dots \wedge \dots) \vee \dots$



# Disjunctive descriptions

disjunctive descriptions - yes:

(outlook=sunny  $\wedge$  humidity=normal)  $\vee$   
(outlook=overcast)  $\vee$  (outlook=rainy  $\wedge$   
windy=false)



# Disadvantages

- Loss of information for continuous variables.
- No guarantee to return the globally optimal decision:
  - Simple-to-complex, hill-climbing search through hypothesis space to construct trees.
  - It does not do back tracking on selected attributes.
- Information gain has bias for attributes with greater no. of values.
- Overfitting
  - New stopping criteria: choose best attribute only if IG/GR is greater than some threshold
  - Pruning: post-process the tree to remove undesirable branches (with few instances, or small IG/GR improvements)



## Example Code

```
#https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y)

estimator = DecisionTreeClassifier(criterion='entropy',max_depth=1) #decision stump
estimator.fit(X_train, y_train)
estimator.score(X_test,y_test)
```

```
0.5526315789473685
```

```
estimator = DecisionTreeClassifier(criterion='entropy')
estimator.fit(X_train, y_train)
estimator.score(X_test,y_test)
```

```
0.9736842105263158
```

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>



# Summary

- How to build decision trees using IG/GR?
- How to test decision trees?
- What are the advantages/disadvantages of decision trees?



## References

- Mitchell, Tom (1997). Machine Learning. Chapter 3: Decision Tree Learning.
- Tan et al (2006) Introduction to Data Mining. Section 4.3, pp 150-171.
- Quinlan, J. R. 1986. Induction of Decision Trees. *Mach. Learn.* 1, 1 (Mar. 1986), 81–106
- Lee, S., Lee, S., Park, Y. (2007). A prediction model for success of services in e-commerce using decision tree: E-customer's attitude towards online service. *Expert Systems with Applications*, 33(3), 572-581.
- Delen, D., Walker, G., Kadam, A. (2005). Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial intelligence in medicine*, 34(2), 113-127.
- Tso, G. K., Yau, K. K. (2007). Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. *Energy*, 32(9), 1761-1768.

# Lecture 16: Evaluation Part 2

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne

Acknowledgement: Jeremy Nicholson, Tim Baldwin & Karin Verspoor



# Roadmap

So far

- Supervised learning algorithms: Naive Bayes, Logistic Regression, Neural Networks, Decision trees ...
- Evaluation Part 1: Assess the effectiveness of the classifier

Today

- What's the problem of the model? Underfitting (high model bias) and overfitting (high model variance)
- How to know which problem the model suffers from? Learning curve
- How to correct the problems? Remedies for underfitting and overfitting
- Evaluation bias and variance



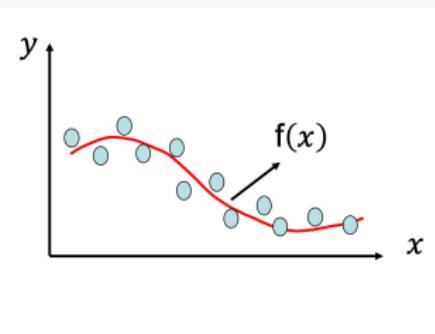
## **Underfitting and Overfitting**

---

## Generalization Error I

Given a training dataset  $D = \{x_i, y_i\}, i = 1 \dots n$  and  $y \in \mathbb{R}$ :

$$y = f(x) + \epsilon$$

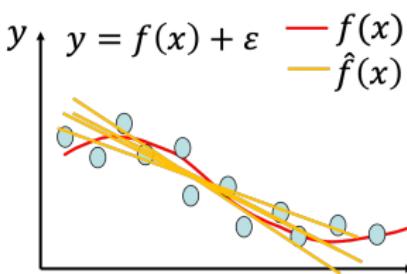


- $f(\cdot)$ : true function to generate data
- $\epsilon \in \mathcal{N}(0, \sigma)$ : data noise



## Generalization Error II

$$Err(x) = E \left[ (y - \hat{f}(x))^2 \right]$$



- $\hat{f}()$ : estimation of  $f()$
- Use multiple models (trained on different training sets) to remove data dependency
- $E$ : expectation (average) operator over all possible training sets



## Generalization Error III

- The generalization error can be decomposed to:

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E \left[ \left( \hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$

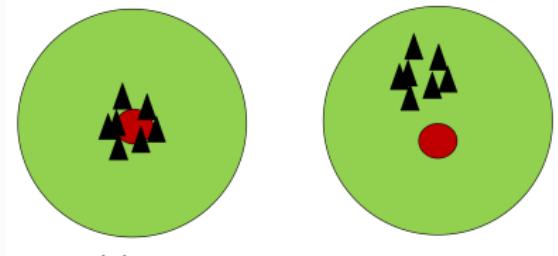
- Or simply written as:

$$Err(x) = \text{Model Bias}^2 + \text{Model Variance} + \text{Irreducible Error}$$



# Bias Example

Which one has low bias?



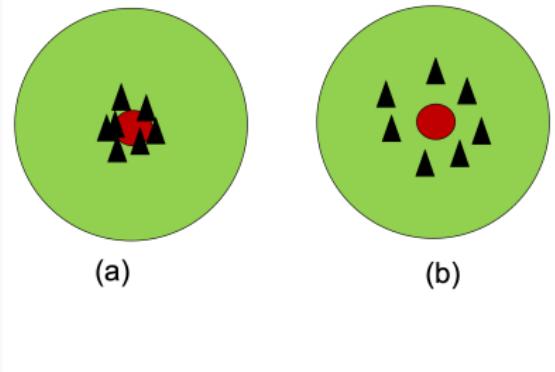
(a)

(b)



# Variance Example

Which one has low variance?



# Underfitting I

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E \left[ \left( \hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$

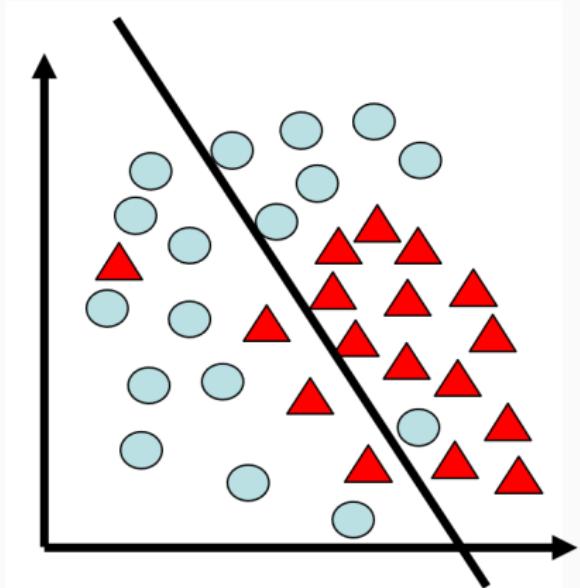
Lazy model  $\hat{f}(x) = c$ : Extreme underfitting

- Model Variance: zero,
- Model Bias: large



## Underfitting II

- does not fit the data
- Bad performance on training data
- Does not generalize to new data.



# Overfitting I

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E \left[ \left( \hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$

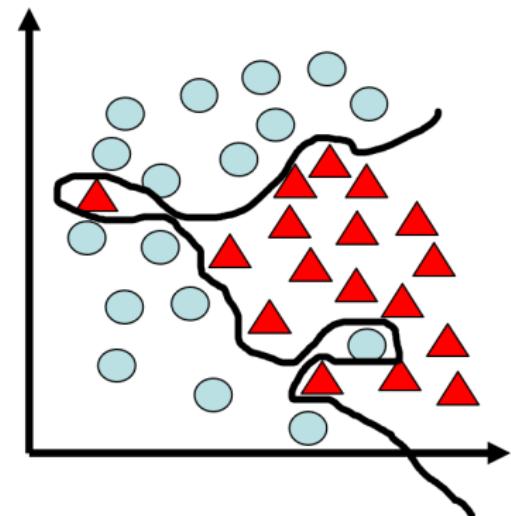
Hard-working model  $\hat{f}(x) = y = f(x) + \epsilon$ : Extreme Overfitting

- Bias: zero,
- Variance: large



## Overfitting II

- Fit the training data too well
- Good performance on training data
- Unreliable Generalization.



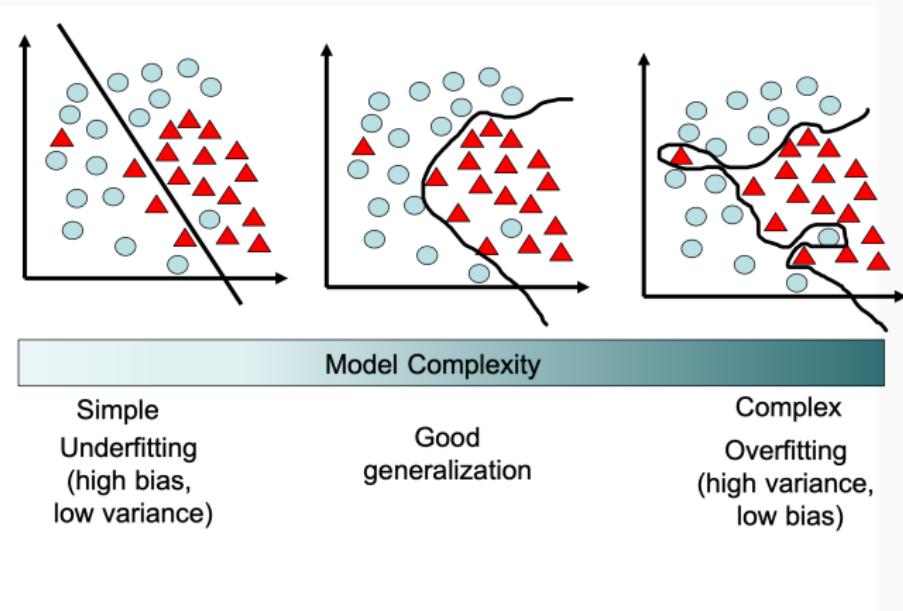
## Overfitting III

Which baseline has low variance?

- (a) Weighted random classifier
- (b) 0-R (majority classifier)



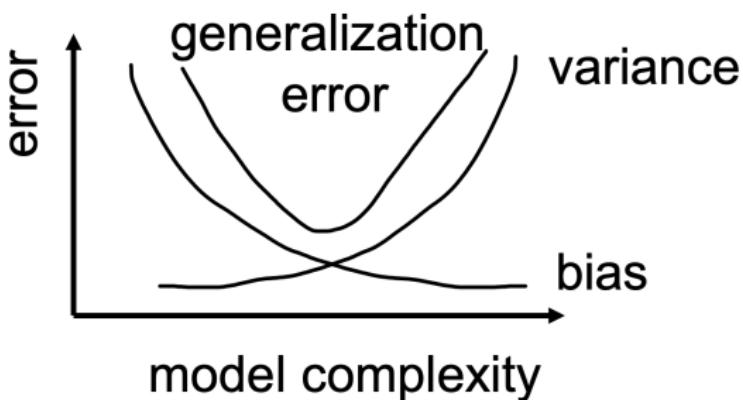
# Underfitting vs Overfitting



Can we have a model with minimum bias and variance?



## Bias-Variance Tradeoff



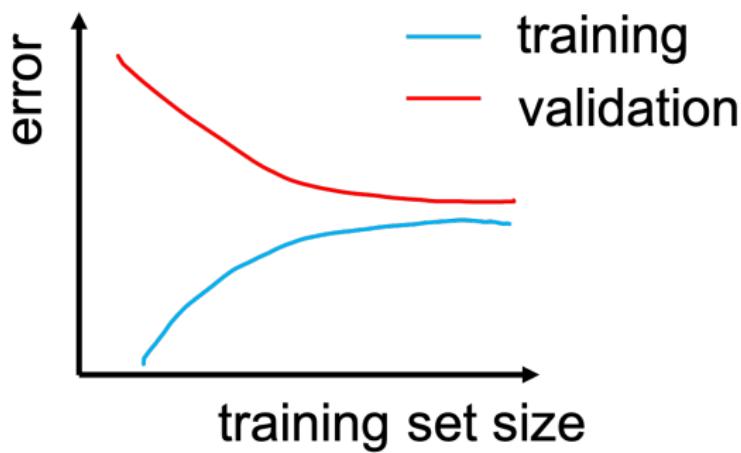
## **Learning Curve**

---

## Learning Curve

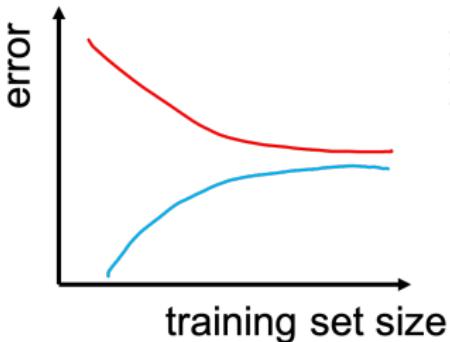
Learning curve: plot of learning performance over increasing size of training dataset.

- x-axis: increasing number of training examples
- y-axis: scores like accuracy, error...

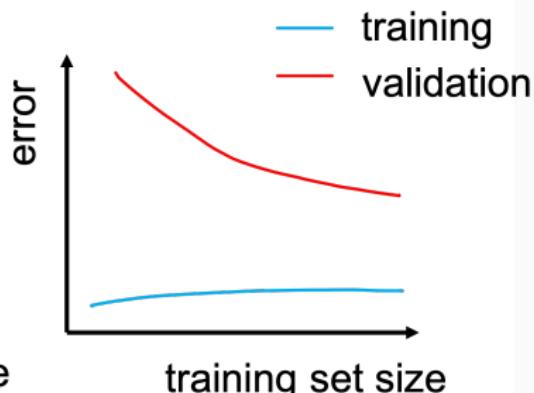


# Underfitting vs Overfitting

Which is underfitting ? Which is overfitting?

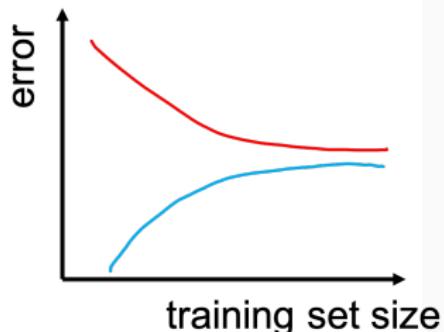


(a)



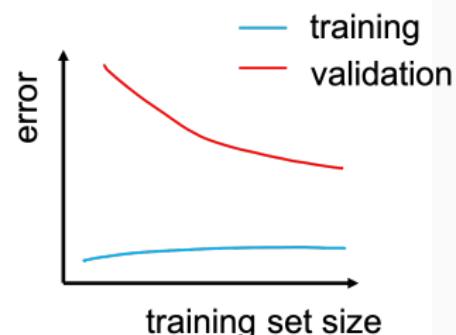
(b)

# Underfitting vs Overfitting



## Underfitting

- high training and validation error
- increasing data does not help



## Overfitting

- low training error, high validation error
- increasing data can help



# How to Plot Learning Curve

Learning curve:

- Choose various split sizes, and calculate effectiveness
  - For example: 90-10, 80-20, 70-30, 46-40, 50-50, 40-60, 30-70, 20-80, 10-90 (9 points)
  - Might need to average multiple runs per split size
- Plot % of training data vs training/test Accuracy (or other metric)



# Example Code

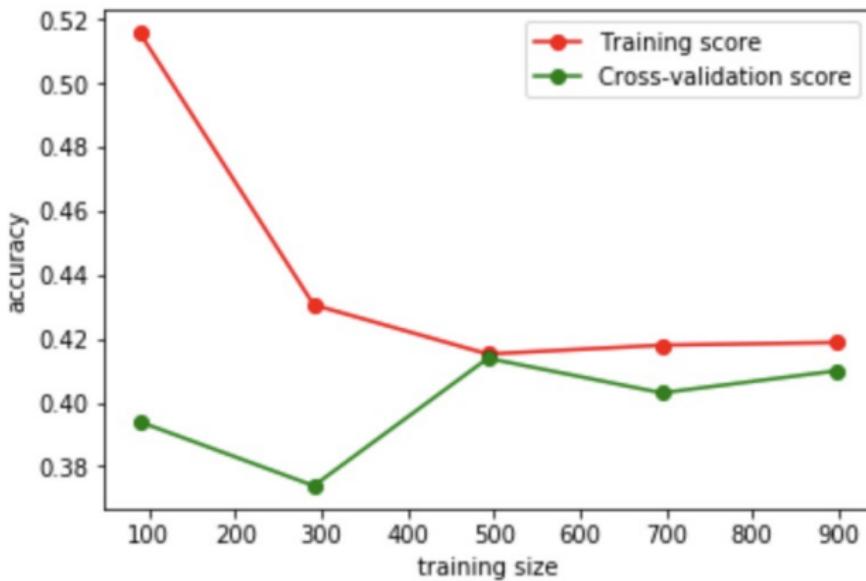
```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html#
#https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import StratifiedKFold

estimator = tree.DecisionTreeClassifier(max_depth=2)
train_sizes, train_scores, valid_scores = \
    learning_curve(estimator, X_train, y_train, scoring='accuracy', cv=StratifiedKFold(10),
                   train_sizes=np.linspace(.1, 1.0, 5))
plt.figure()
plt.xlabel("training size")
plt.ylabel("accuracy")
plt.plot(train_sizes, np.mean(train_scores, axis=1), 'o-', color="r",
         label="Training score")
plt.plot(train_sizes, np.mean(valid_scores, axis=1), 'o-', color="g",
         label="Cross-validation score")
plt.legend(loc="best")
plt.show()
```



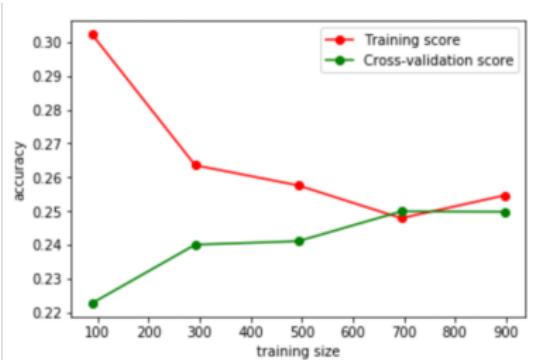
THE UNIVERSITY OF  
MELBOURNE

## Learning Curve Example I

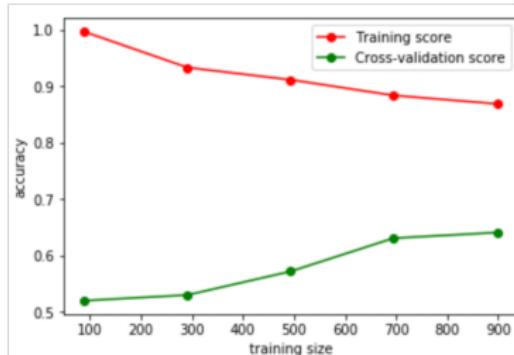


## Learning Curve Example II

Which of the following is the learning curve of decision stump?



(a)



(b)

## **Remedy for Underfitting and Overfitting**

---

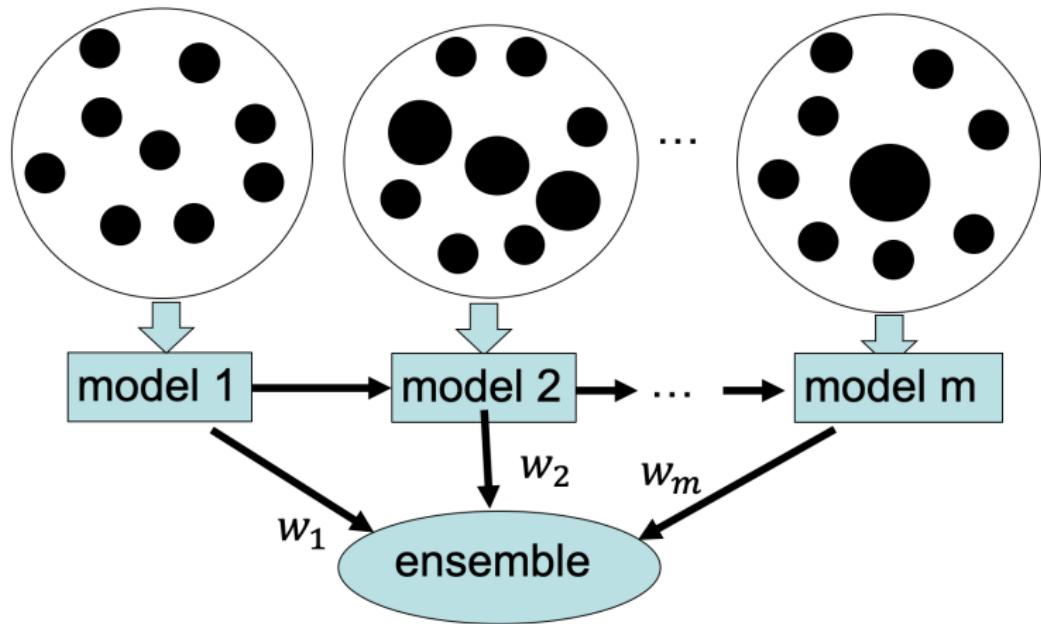
## Underfitting Remedy

- Use more complex model (e.g. use nonlinear models)
- Add features
- Boosting



## Boosting

- training data: different weights (probabilities to be selected)
- Use multiple weak models → a stronger model; reduces bias (improves performance)



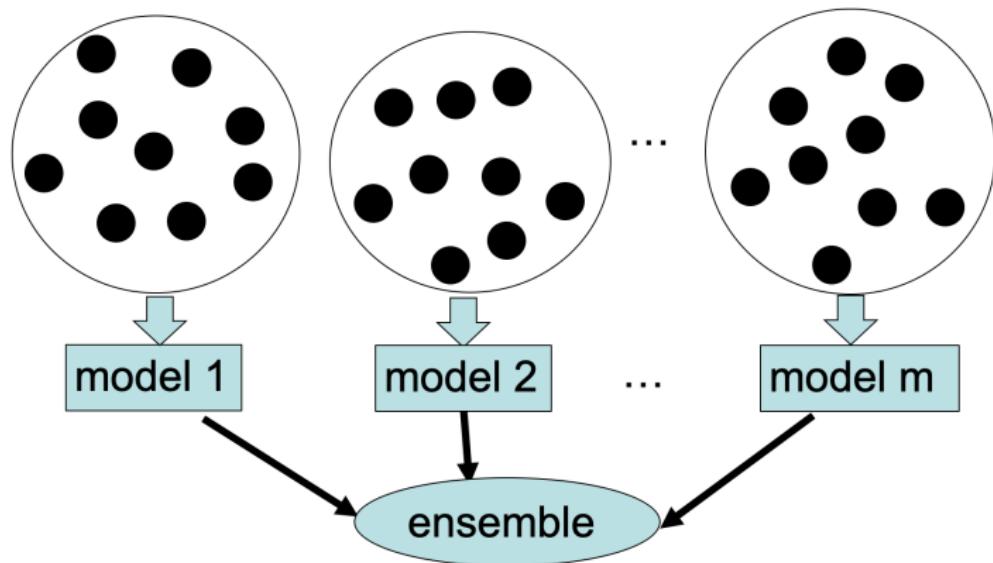
# Overfitting Remedy

- Add more training data
- Reduce features
- Reduce model complexity – complex models are prone to high variance
- Bagging



# Bagging

- Construct new datasets: randomly select the training data with replacement
- Combining multiple models → predictions are more stable; reduces variance of individual model.



## **Evaluation Bias and Variance**

---

# Evaluation Bias and Variance I

- We want to know the “true” error rate of a classifier, but we only have an estimate of the error rate, subject to some particular set of evaluation instances
  - **High evaluation Bias:** Our estimate of the effectiveness of a model is systematically too high/low
  - **High evaluation Variance:** Our estimate of the effectiveness of a model changes a lot, as we alter the instances in the evaluation set



## Evaluation Bias and Variance II

How do we control bias and variance in evaluation?

- Holdout partition size
  - More training data: more evaluation variance
  - Less training (more test) data: less evaluation variance
- Repeated random subsampling and K-fold Cross-Validation
  - Less variance than Holdout
- Stratification: less evaluation bias
- Leave-one-out Cross-Validation
  - No sampling bias, lowest bias/variance in general



## **Summary**

---

# Summary

- How are underfitting and overfitting different?
- How are model bias and variance different?
- how to diagnose underfitting and overfitting using learning curve?
- How do we try to control for model bias and variance
- What is evaluation bias and variance?
- How do we try to control for bias and variance in evaluation?



## References

- Sammut, Claude; Webb, Geoffrey I., eds. (2011). Bias Variance Decomposition. Encyclopedia of Machine Learning. Springer. pp. 100–101.
- Luxburg, Ulrike V.; Schölkopf, B. (2011). Statistical learning theory: Models, concepts, and results. Handbook of the History of Logic. 10: Section 2.4.
- Vijayakumar, Sethu (2007). The Bias–Variance Tradeoff. University of Edinburgh. Retrieved 19 August 2014.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). New York: springer. Chapter 2.
- Jeremy Nicholson & Tim Baldwin & Karin Verspoor: Machine Learning



# Lecture 17: Ensemble Learning

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne

Acknowledgement: Jeremy Nicholson, Tim Baldwin & Karin Verspoor



# Classification

So far:

- Classification algorithms in isolation
- Training and testing one classifier
- Remedies for Overfitting and underfitting

Today:

- Introduction of Ensemble learning
- Stacking
- Bagging
- Boosting

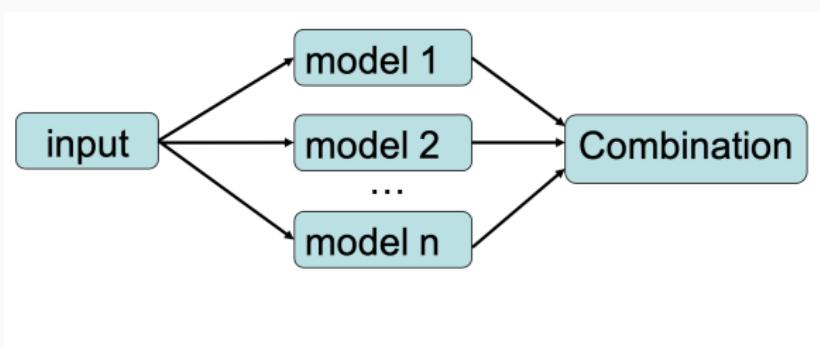


## **Introduction of Ensemble learning**

---

# What is Ensemble Learning

Ensemble learning (aka. Classifier combination): constructs a set of base classifiers from a given set of training data and aggregates the outputs (e.g., using majority voting) into a single meta-classifier.



# Approaches for Ensemble Learning

- **Instance manipulation:** generate multiple training datasets through sampling, and train a base classifier over each dataset
- **Feature manipulation:** generate multiple training datasets through different feature subsets, and train a base classifier over each dataset
- **Class label manipulation:** generate multiple training datasets by manipulating the class labels in a reversible manner
- **Algorithm manipulation:** semi-randomly “tweak” internal parameters within a given algorithm to generate multiple base classifiers over a given dataset



# Why Ensemble Learning

- **Intuition 1:** the combination of lots of weak classifiers can be at least as good as one strong classifier
- **Intuition 2:** the combination of a selection of strong classifiers is (usually) at least as good as the best of the base classifiers



# When does ensemble learning work? I

- The base classifiers should not make the same mistakes
- The base classifiers are reasonably accurate

	$t_1$	$t_2$	$t_3$
$C_1$	✓	✓	✗
$C_2$	✗	✓	✓
$C_3$	✓	✗	✓
$C^*$	✓	✓	✓

	$t_1$	$t_2$	$t_3$
$C_1$	✓	✓	✗
$C_2$	✓	✓	✗
$C_3$	✓	✓	✗
$C^*$	✓	✓	✗

	$t_1$	$t_2$	$t_3$
$C_1$	✓	✗	✗
$C_2$	✗	✓	✗
$C_3$	✗	✗	✓
$C^*$	✗	✗	✗

## When does ensemble learning work? II

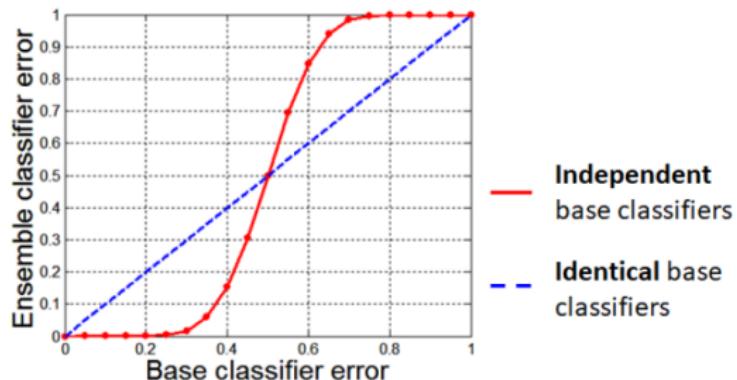
- Given 25 binary base classifiers, each with an error rate of  $\epsilon = 0.35$ .
- Ensemble by majority voting
  - if the base classifiers are identical, after ensemble,  $\epsilon = 0.35$ .
  - If the base classifiers are independent, after ensemble,

$$\sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} \approx 0.06$$



## When does ensemble learning work? II

- When does ensemble learning work?



## Quiz

Which of the following statement(s) are TRUE about ensemble learning?

- (a) An ensemble of classifiers may not be able to outperform any of its individual base learners.
- (b) Combining significantly diverse base learners (suppose each produces meaningful predictions) typically yields bad results.

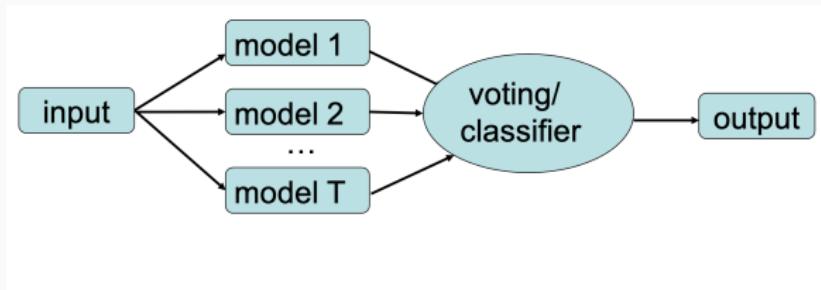


## **Stacking**

---

# Stacking

- **Intuition:** “smooth” errors over a range of algorithms with different biases
- **Method:** use different algorithms to train multiple base classifiers on the dataset.



- **Inputs for second-level classifier (meta-learner):** use base classifiers to generate predictions on unseen samples (using cross-validation).

## Stacking II

- Mathematically simple but computationally expensive method
- Able to combine heterogeneous classifiers with varying performance
- Generally, stacking results in as good or better results than the best of the base classifiers

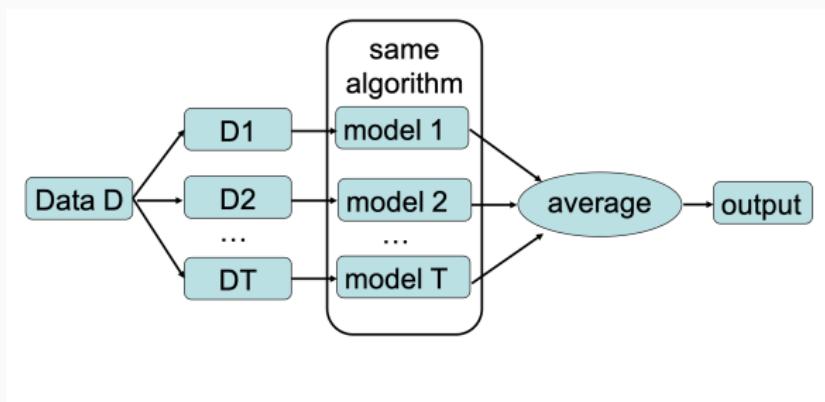


## **Bagging**

---

# Bagging I

- **Intuition:** Average multiple models can lower the model variance.
- **Method:** Create multiple new training sets for training multiple classifiers base on the same algorithm and average the predictions.



## Bagging II

**Dataset generation:** randomly sample the original dataset (N instances) N times, with replacement. Any individual instance is absent with probability  $(1 - \frac{1}{N})^N$

Example:

- Original dataset:

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

- Bootstrap Samples

7	2	6	7	5	4	8	8	1	10
---	---	---	---	---	---	---	---	---	----

1	3	8	10	3	5	8	10	1	9
---	---	---	----	---	---	---	----	---	---

2	9	4	2	7	9	3	10	1	10
---	---	---	---	---	---	---	----	---	----

:



## Bagging III

- Possibility to parallelise computation of individual base classifiers
- Highly effective over noisy datasets (outliers may vanish)
- Generally produces the best results on unstable models that have high variance and low bias



# Random Forest I

- A “Random Forest” is an ensemble of Random Trees (many trees = forest)
- A “Random Tree” is a Decision Tree where at each node, only some of the possible attributes are considered
- Use random trees instead of decision trees to increase diversity of base classifier



## Random Forest II

Practical Properties of Random Forests:

- Embarrassingly parallelisable
- Robust to overfitting
- Interpretability sacrificed



# Quiz

Which of the following statement(s) are TRUE about Random Forest?

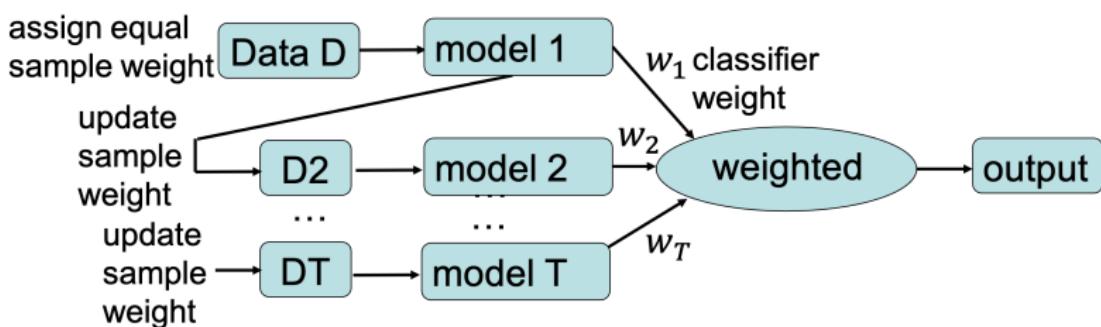
- (a) Random Forest provides higher interpretability over the logic behind the predictions than a single random tree.
- (b) Random Forest adopts both feature manipulation and instance manipulation approaches.
- (c) Random Forest minimizes the bias by having multiple random trees trained on different versions of the dataset.



## **Boosting**

---

- **Intuition:** Build a strong model from several weak models to reduce model bias.
- **Method:** Iteratively change the weights of training instances to train next base classifier and combine the base classifiers via weighted voting



## Boosting II

- Original dataset:

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

- Boosting samples:

*Iteration 1:*

7	2	6	7	5	4	8	8	1	10
---	---	---	---	---	---	---	---	---	----

*Iteration 2:*

1	3	8	4	3	5	4	10	1	4
---	---	---	---	---	---	---	----	---	---

*Iteration 3:*

4	9	4	2	4	4	3	10	1	4
---	---	---	---	---	---	---	----	---	---

⋮



- Input: Training instances  $(x_j, y_j) | j = 1, 2, \dots, N$
- Initial equal sample weights  $w_j^{(0)} = \frac{1}{N} | j = 1, 2, \dots, N$
- For  $i = 1 \dots T$ 
  - Construct classifier  $C_i$  in iteration  $i$ :
    - apply sample weights to the loss or
    - use the weights to re-sample data to train model
  - Calculate weight of the classifier  $\alpha_i$
  - Update the sample weights
- Final classification via weighted voting: multiply vote of each classifier with its weight.



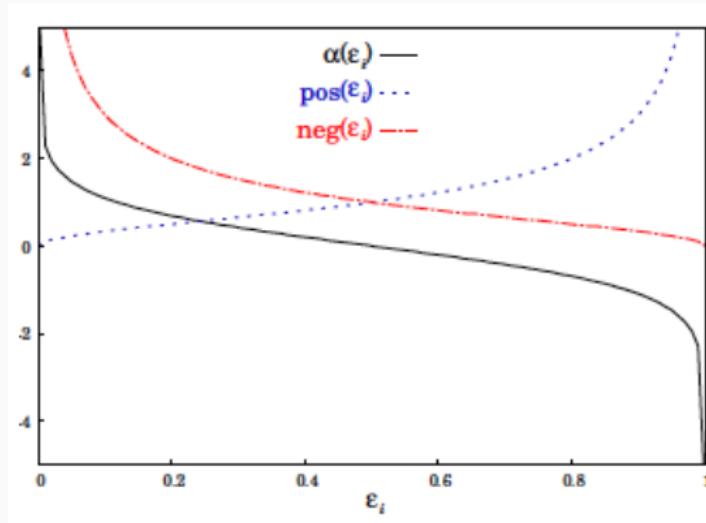
## AdaBoost II

- Error rate for  $C_i$ :

$$\epsilon_i = \sum_{j=1}^N w_j^{(i)} \delta(C_i(x_j) \neq y_j)$$

- “Importance” of  $C_i$  (i.e. the weight associated with the classifiers’ votes):

$$\alpha_i = \frac{1}{2} \log_e \frac{1 - \epsilon_i}{\epsilon_i}$$



## AdaBoost III

- If  $\alpha_i > 0$ , adjust weights for instance  $j$  ( $i > 0$ ):

$$w_j^{(i+1)} = w_j^{(i)} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ e^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

$$Z_i = \sum_{j=1}^N w_j^{(i+1)}$$

$$w_j^{(i+1)} = w_j^{(i+1)} / Z_i$$



- Classification:

$$C^*(x) = \underset{y}{\operatorname{argmax}} \sum_{i=1}^T \alpha_i \delta(C_i(x) = y)$$

- Base classification algorithm: decision stumps (OneR) or decision trees



## Quiz

Which of the following statement(s) are TRUE about Boosting?

- (a) Boosting adopts feature manipulation approach to train multiple base learners
- (b) Boosting assigns higher weights to better-performing base learners
- (c) Boosting iteratively learns base learners while emphasizing the samples that can be easily classified



## Bagging vs. Boosting

Bagging/Random Forests	Boosting/AdaBoost
Builds base models in parallel	Builds base models sequentially
Parallel sampling: Resamples data points with replacement	Iterative sampling: Reweights data points (modifies their distribution)
Base classifiers have the same weight	Base classifiers have the different weight
Reduce variance	Reduce bias
Not prone to overfitting	Prone to overfitting



## **Summary**

---

- What is classifier combination?
- What is bagging and what is the basic thinking behind it?
- What is boosting and what is the basic thinking behind it?
- What is stacking and what is the basic thinking behind it?
- How do bagging and boosting compare?



# Quiz

What are the techniques that use instance manipulation approach to combine classifiers?

- (a) Bagging
- (b) Boosting
- (c) Random Forest
- (d) Stacking



## References

- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.
- Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, USA, second edition, 2005.



# Lecture 18: Unsupervised Learning

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne

Acknowledgement: Jeremy Nicholson, Tim Baldwin & Karin Verspoor



# Roadmap

So far:

- Supervised machine learning algorithms
- Train and evaluation the performance of the classifiers

Today:

- Introduction of clustering (unsupervised learning)
- K-means clustering
- Hierarchical clustering
- Evaluate clustering performance

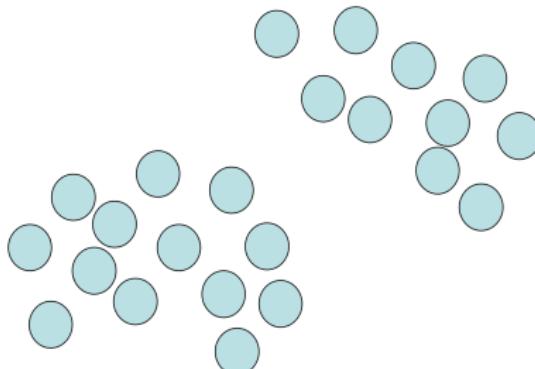


## **Introduction**

---

## What is clustering

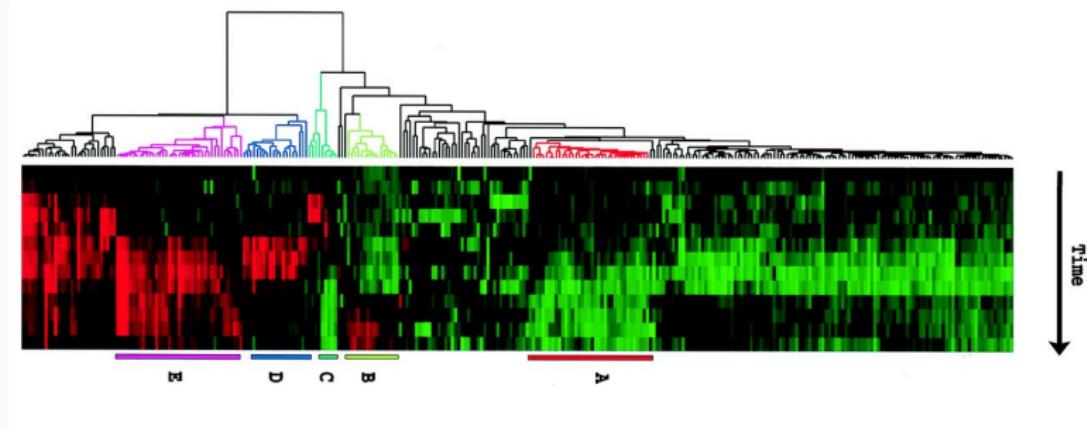
- Unsupervised learning: The class of an example is not known (or at least not used)
- Goal: find groups of similar data points
- “Similar” measure: e.g., small Euclidean distance.



# Why clustering I

Applications:

- Gene expression data analysis



Eisen, M.B. et al. Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences, 95(25), 1998, pp.14863-14868.

# Why clustering II

Applications:

- Image segmentation



(a) original image



(b) segmentation  
output (2 clusters)

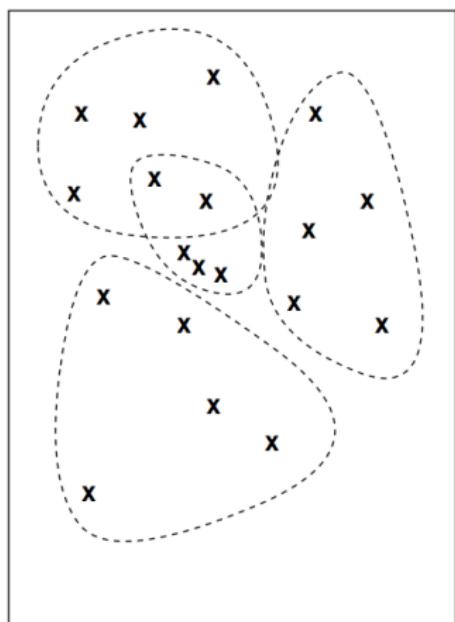
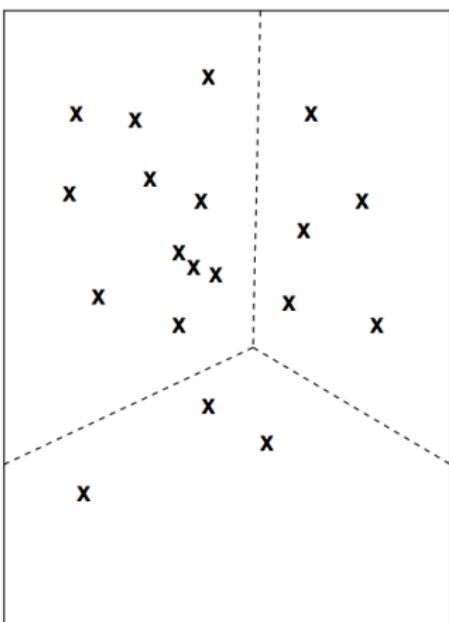


(c) segmentation  
output (3 clusters)

- Identify groups of customer

# Exclusive vs. overlapping clustering

- Can an item be in more than one cluster?



## Deterministic vs. probabilistic clustering

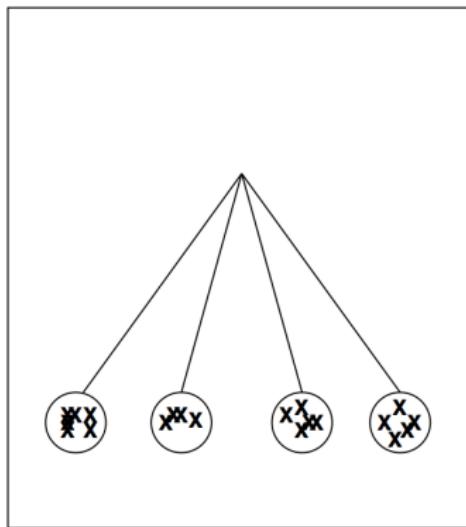
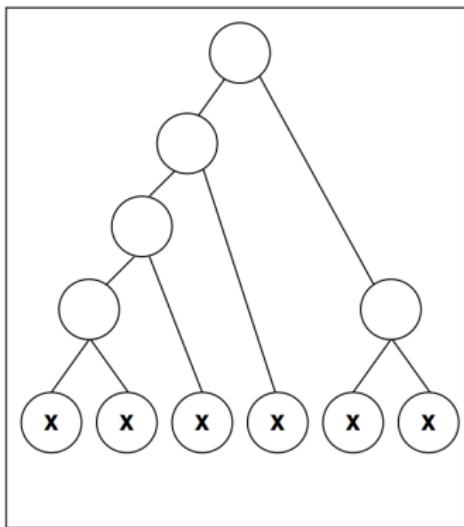
- Can an item be partially or weakly in a cluster?

<i>Instance</i>	<i>Cluster</i>		<i>Cluster</i>			
		<i>Instance</i>	1	2	3	4
1	2	1	0.01	0.87	0.12	0.00
2	3	2	0.05	0.25	0.67	0.03
3	2	3	0.00	0.98	0.02	0.00
4	1	4	0.45	0.39	0.08	0.08
5	2	5	0.01	0.99	0.00	0.00
6	2	6	0.07	0.75	0.08	0.10
7	4	7	0.23	0.10	0.20	0.47
:	:	:	:	:	:	



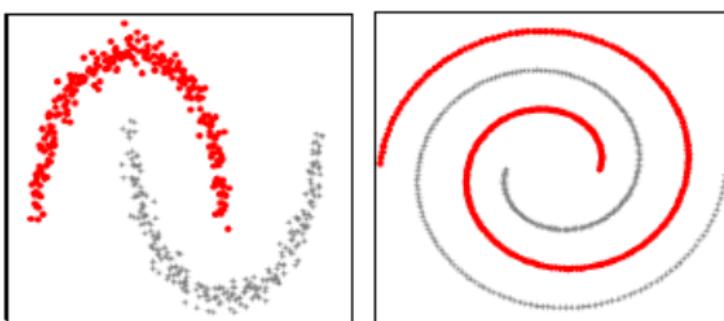
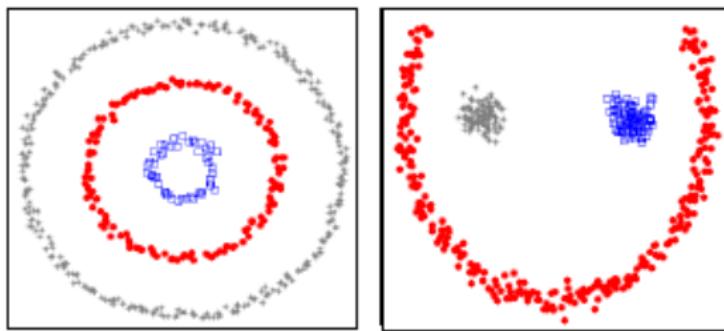
## Hierarchical vs. partitioning clustering

- Do the clusters have subset relationships between them? e.g. nested in a tree?



## Heterogenous vs. homogenous clustering

- Clusters of widely different sizes, shapes, and densities



# Clustering, basic contrasts

- Exclusive vs. overlapping clustering
  - Can an item be in more than one cluster?
- Deterministic vs. probabilistic clustering (Hard vs. soft clustering)
  - Can an item be partially or weakly in a cluster?
- Hierarchical vs. partitioning clustering
  - Do the clusters have subset relationships between them? e.g. nested in a tree?
- Heterogenous vs. homogenous
  - Clusters of widely different sizes, shapes, and densities
- Partial vs. complete clustering
  - In some cases, we only want to cluster some of the data
- Incremental vs. batch clustering
  - Is the whole set of items clustered in one go?



## ***k*-means**

---

## *k*-means Clustering

Given  $k$  (the number of clusters), the  $k$ -means algorithm is implemented in four steps:

1. Initialize: Select random  $k$  points to act as seed cluster centroids
2. **Iterate:**
  - Step 1 - Cluster assignment: Assign each instance to the cluster with the **nearest centroid**
  - Step 2 - Recompute the cluster centroids: Update centroid of each cluster to the average of its assigned instances
3. **Until** the centroids don't change
  - Exclusive, deterministic, partitioning, batch clustering method



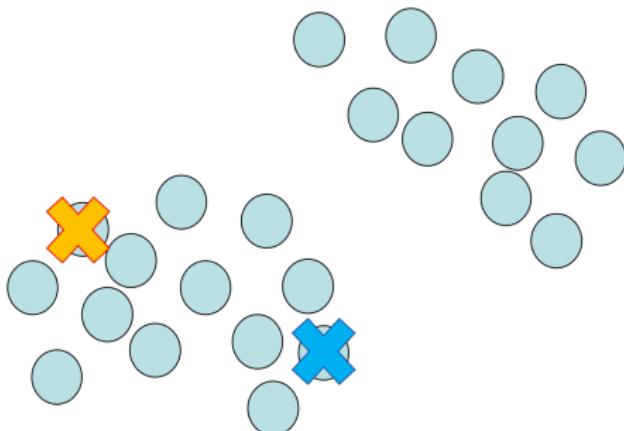
# Measuring Similarity / Proximity / Closeness

- Data points in Euclidean space
  - Euclidean distance
  - Manhattan (L1) distance
- Discrete values
  - Hamming distance: discrepancy between the bit strings
- Other measures
  - Cosine similarity
  - Jaccard measure



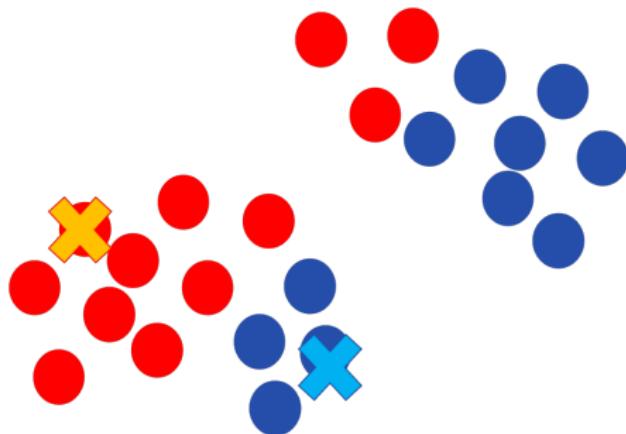
## Example 1

Pick 2 ( $k = 2$ ) random points as cluster centroids.



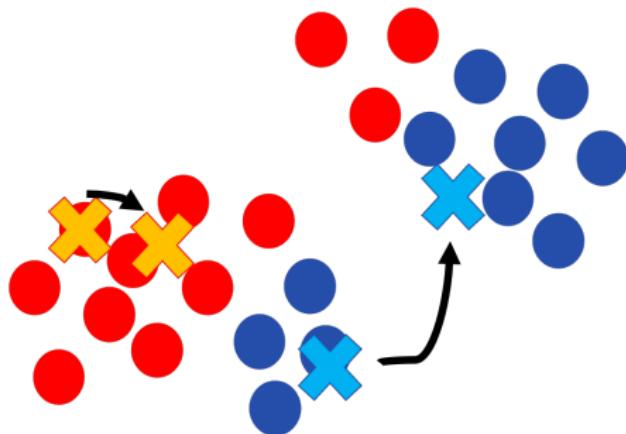
## Example 1

(Iteration 1) Step 1: Cluster assignment



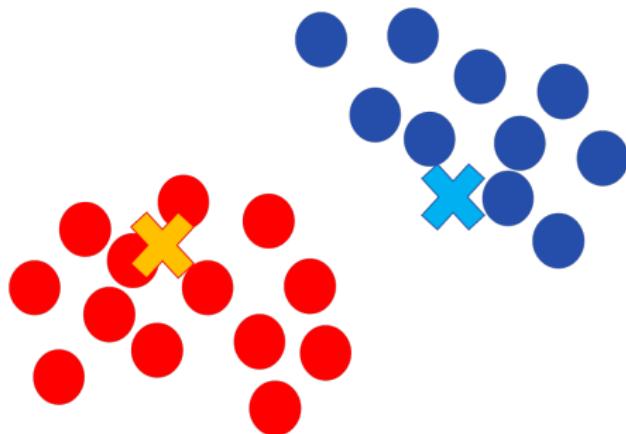
## Example 1

(Iteration 1) Step 2: Recompute the cluster centroids



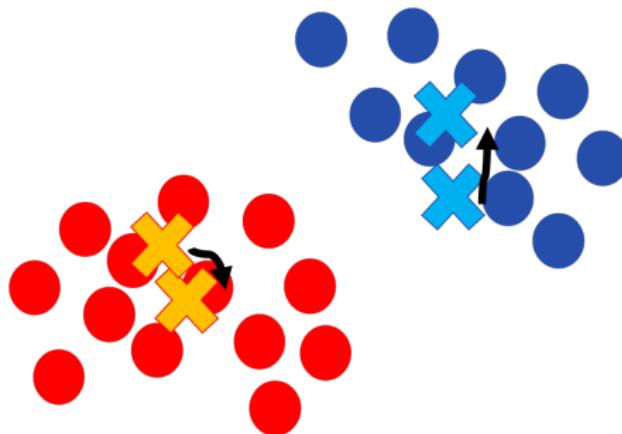
## Example I

(Iteration 2) Step 1: Cluster assignment

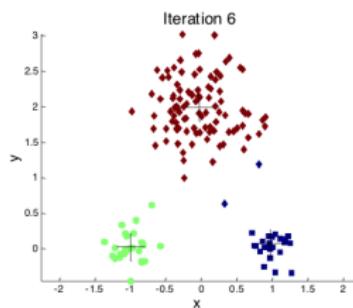
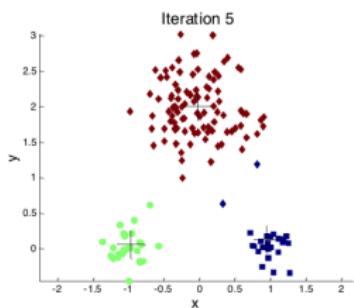
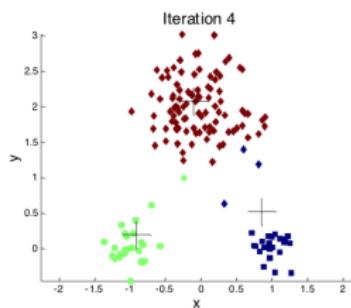
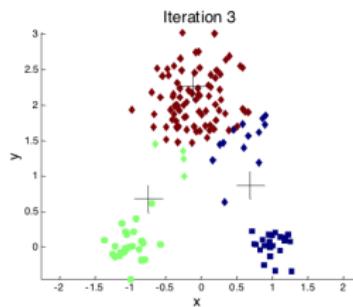
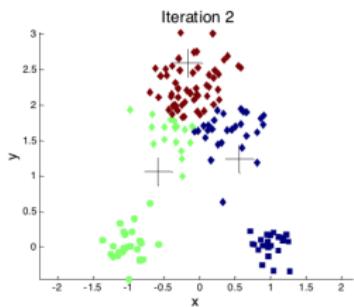
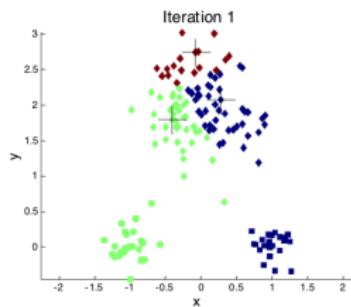


## Example 1

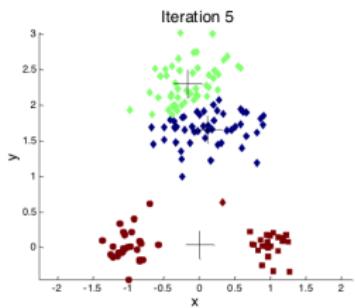
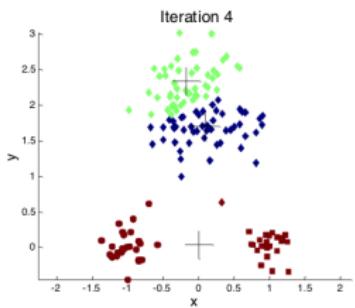
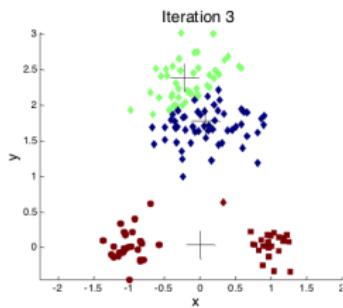
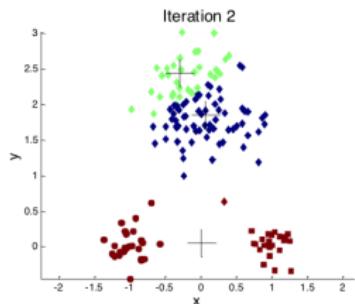
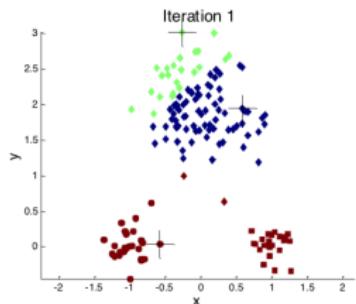
(Iteration 2) Step 2: Recompute the cluster centroids



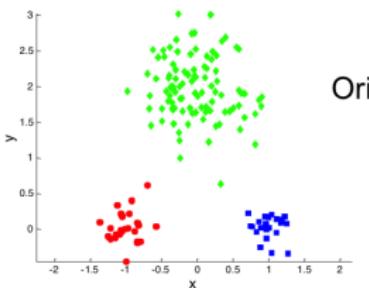
# Example, Iterations



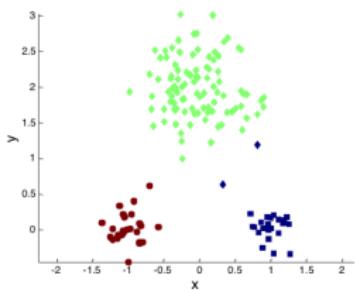
## Example, Impact of initial seeds



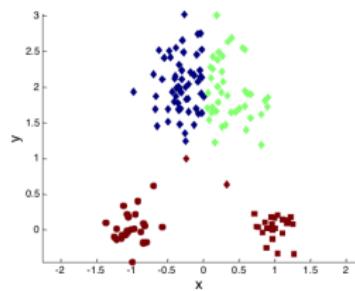
## Example, Different outcomes



Original Points



Optimal Clustering



Sub-optimal Clustering



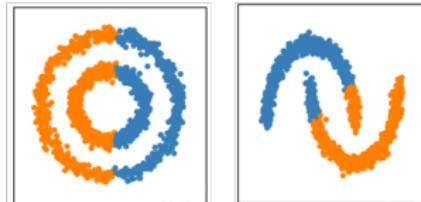
## Pros of $k$ -means

- relatively efficient:
  - $O(ndki)$ , where  $n$  is no. instances,  $d$  is no. attributes,  $k$  is no. clusters, and  $i$  is no. iterations; normally  $k, i \ll n$
  - Unfortunately we cannot a priori know the value of  $i$ !
- can be extended to hierarchical clustering



## Cons of $k$ -means

- results sensitive to random centroid selection:
  - try multiple iterations with different seeds
  - try better initialization method ( $k$ -means++)
- “mean” ill-defined for nominal or categorical attributes
- may not work well when the data contains outliers
- not able to handle non-convex clusters, or clusters of differing densities or sizes



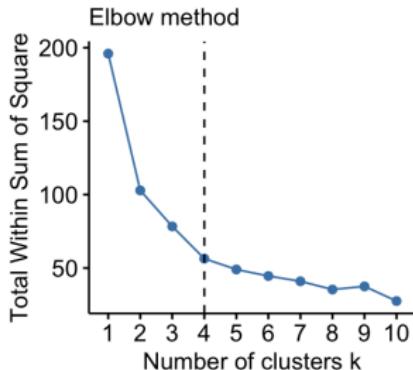
- need to specify  $k$  in advance.

## How to choose the number of clusters?

- calculate within-cluster Sum of Squared Error  $SSE_w$  for different number of clusters  $K$

$$SSE_w = \sum_{i=1}^K \sum_{x \in C_i} (x - m_i)^2$$

- $x$  is a data point in cluster  $C_i$  and  $m_i$  is the centroid of cluster  $C_i$ .
- As  $K$  increases, we will have a smaller number of instances in each cluster  $\rightarrow SSE_w$  decreases
- Elbow method:  $K$  increases to  $K + 1$ , the drop of  $SSE_w$  starts to diminish



## Hierarchical Clustering

---

# Hierarchical Clustering

Bottom-up (= agglomerative) clustering

- Start with single-instance clusters
- At each step, join the two “closest” clusters (in terms of margin between clusters, distance between mean, ...)

Top-down (= divisive) clustering

- Start with one universal cluster
- Find two partitioning clusters
- Proceed recursively on each subset

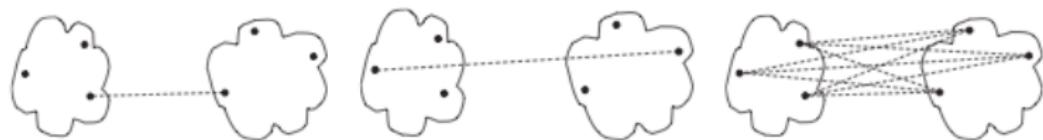


## Bottom-up (Agglomerative) Clustering

1. Each point starts as a cluster. Compute the proximity matrix.
2. **repeat**
3.     Merge the closest two clusters
4.     Update the proximity matrix to reflect the proximity between the new cluster and the original clusters
5. **until** Only one cluster remains



## Graph-based measure of Proximity



(a) MIN (single link.)

(b) MAX (complete link.)

(c) Group average.

Updating the proximity matrix:

- Single Link: *Minimum* distance between any two points in the two clusters. (most similar members)
- Complete Link: *Maximum* distance between any two points in the two clusters. (most dissimilar members)
- Group Average: *Average* distance between all points (pairwise).



# Agglomerative Clustering Example

	1	2	3	4	5
1	1.00	0.90	0.10	0.65	0.20
2	0.90	1.00	0.70	0.60	0.50
3	0.10	0.70	1.00	0.40	0.30
4	0.65	0.60	0.40	1.00	0.80
5	0.20	0.50	0.30	0.80	1.00

What are the two closest points?



## Agglomerative Clustering Example

	1	2	3	4	5
1	1.00	0.90	0.10	0.65	0.20
2	0.90	1.00	0.70	0.60	0.50
3	0.10	0.70	1.00	0.40	0.30
4	0.65	0.60	0.40	1.00	0.80
5	0.20	0.50	0.30	0.80	1.00

Merge points 1 & 2 into a new cluster: 6

Update (single link):

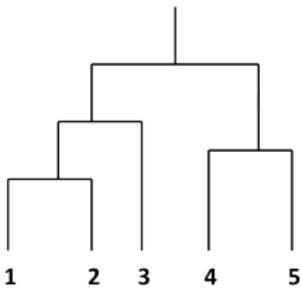
	1	2	3	4	5	6
6	—	—	0.70	0.65	0.50	1.00

Update (complete link):

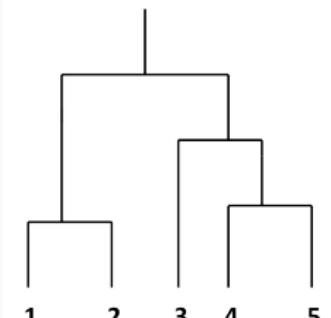
	1	2	3	4	5	6
6	—	—	0.10	0.60	0.20	1.00



	1	2	3	4	5
1	1.00	0.90	0.10	0.65	0.20
2	0.90	1.00	0.70	0.60	0.50
3	0.10	0.70	1.00	0.40	0.30
4	0.65	0.60	0.40	1.00	0.80
5	0.20	0.50	0.30	0.80	1.00



Single link

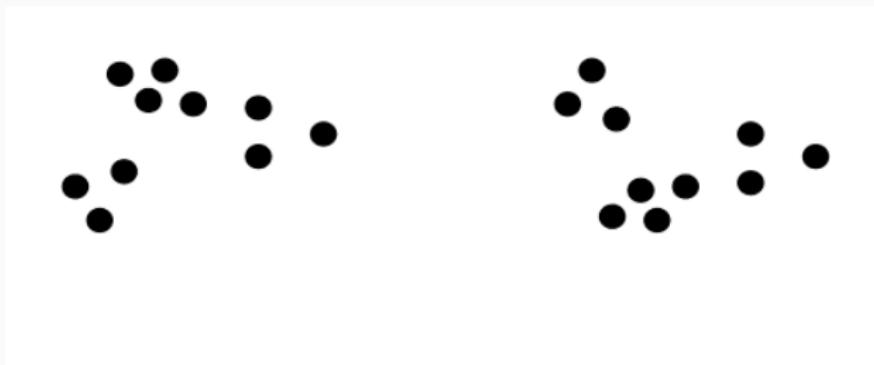


Complete link

## Evaluation

---

# What is a good clustering?



## Two clusters?



## Four clusters?



## Six clusters?



# Types of Evaluation

Unsupervised:

- cluster cohesion: compactness, tightness
- cluster separation: isolation, distinctiveness.

Supervised: measure how well cluster labels match externally supplied class labels.

- entropy
- purity



A “good” cluster should have one or both of:

- **High Cluster Cohesion:** instances in a given cluster should be closely related to each other
- **High Cluster Separation** instances in different clusters should be distinct from each other



## Unsupervised Evaluation II

Within-cluster Sum of Squared Error  $SSE_w$ : the smaller, the better

$$SSE_w = \sum_{i=1}^K \sum_{x \in C_i} (x - m_i)^2$$

- $x$ : a data point in cluster  $C_i$
- $m_i$ : the centroid for cluster  $C_i$ .

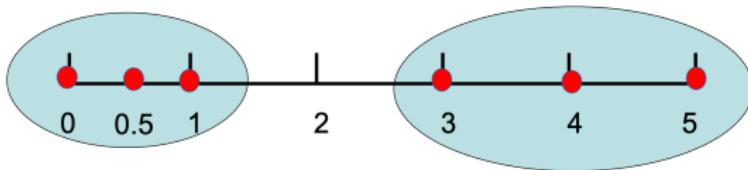
Between-cluster Sum of Squared Error  $SSE_b$ : the larger, the better

$$SSE_b = \sum_{i=1}^K n_i(m_i - m)^2$$

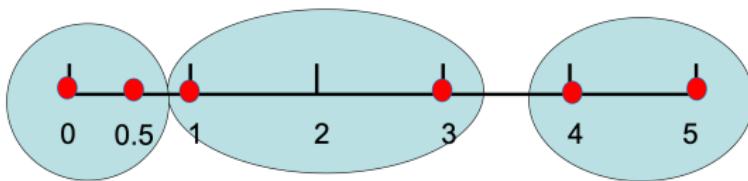
- $m$ : mean of all data points in the dataset
- $m_i$ : centroid for cluster  $C_i$ .
- $n_i$ : number of instances in cluster  $C_i$ .



## Example



$$SSE_w = 2.5, SSE_b = 18.375$$



$$SSE_w = ?, SSE_b = ?$$

## Supervised Evaluation

Entropy: the smaller, the better

$$\text{entropy} = \sum_{i=1}^k \frac{|n_i|}{N} H_i$$

Purity: the larger, the better

$$\text{purity} = \sum_{i=1}^k \frac{|n_i|}{N} \max_j P_i(j)$$

- $n_i$ : number of instances in cluster  $C_i$ .
- $N$ : total number of instances in all clusters.
- $P_i(j)$ : probability of the class  $j$  in the cluster  $i$ .
- $H_i$ : entropy of class distribution in cluster  $i$ .

$$H_i = - \sum_j P_i(j) \log_2 P_i(j)$$



## Supervised Evaluation Example I

- Calculate the entropy and purity of the following cluster output

Cluster	Play = yes	Play = no
1	4	0
2	4	4

$$\text{entropy}_1 = -1 \times \log(1) - 0 \times \log(0) = 0$$

$$\text{entropy}_2 = -0.5 \times \log(0.5) - 0.5 \times \log(0.5) = 1$$

$$\text{purity}_1 = \max(1, 0) = 1$$

$$\text{purity}_2 = \max(0.5, 0.5) = 0.5$$

$$\text{entropy} = \frac{4}{12} \times 0 + \frac{8}{12} \times 1 = 0.67$$

$$\text{purity} = \frac{4}{12} \times 1 + \frac{8}{12} \times 0.5 = 0.67$$



## Supervised Evaluation Example II

- Calculate the entropy and purity of the following cluster output

Cluster	Play = yes	Play = no
1	2	0
2	6	4

*entropy* =?

*purity* =?



# Summary

- What basic contrasts are there in different clustering methods?
- How does  $k$ -means operate, and what are its strengths and weaknesses?
- What is hierarchical clustering, and how does it differ from partitioning clustering?
- How to evaluation the clustering performance?



## References

- Tan, Steinbach, Kumar (2006) Introduction to Data Mining. Chapter 8, Cluster Analysis  
<http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf>
- Jain, Dubes (1988) Algorithms for Clustering Data. [http://homepages.inf.ed.ac.uk/rbf/BOOKS/JAIN/Clustering\\_Jain\\_Dubes.pdf](http://homepages.inf.ed.ac.uk/rbf/BOOKS/JAIN/Clustering_Jain_Dubes.pdf)



# Active Learning and Semi-Supervised Learning

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne

Acknowledgement: Tim Baldwin, Karin Verspoor & Kris Ehinger



# Roadmap

So far:

- Supervised learning
- Unsupervised learning

Today:

- Active learning
  - Query Scenarios
  - Query Strategies
- Semi-supervised learning
  - Combine unsupervised and supervised algorithm
  - Self-training



# Why bother?

- (Labelled) data is a bottleneck for machine learning
  - labeling is time-consuming
  - labeling may require special devices
- Unlabelled data is often cheap and in large quantity

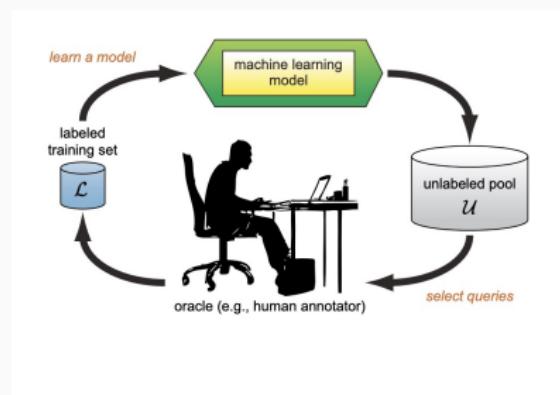


## **Active Learning**

---

# Active Learning

- Motivation: labelling is a finite resource, which should be expended in a way which optimises machine learning effectiveness.
- Key idea: the learner
  - has access to raw unlabeled data,
  - make a query about the label to an **oracle** (e.g. a human annotator)
- Goal: train a good classifier with reduced annotation cost.



## Toy Example: 1D classifier



- Input: Unlabeled data, labels are all 0 then all 1 (left to right)
- Goal: find classifier (threshold function between 0 and 1)
- Naive method: annotate all data points
- Better method: use binary search to reduce annotations



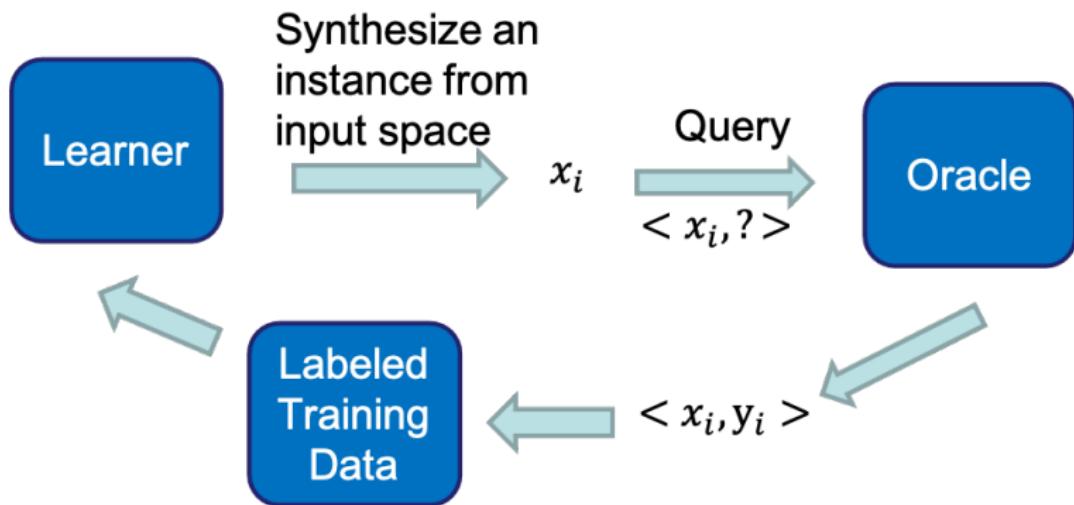
# Query Scenarios

- Query Synthesis
- Stream-based Sampling
- Pool-based Sampling



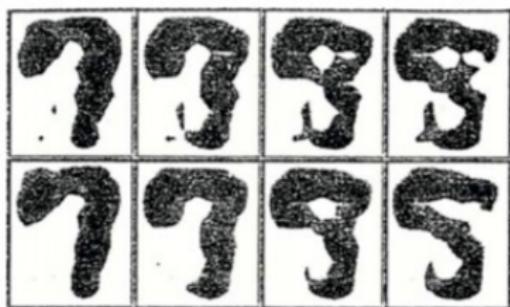
# Query Synthesis I

Learner constructs an instance from input space or distribution from scratch



## Query Synthesis II

- Problem: Human annotator might not recognize the pseudo instance

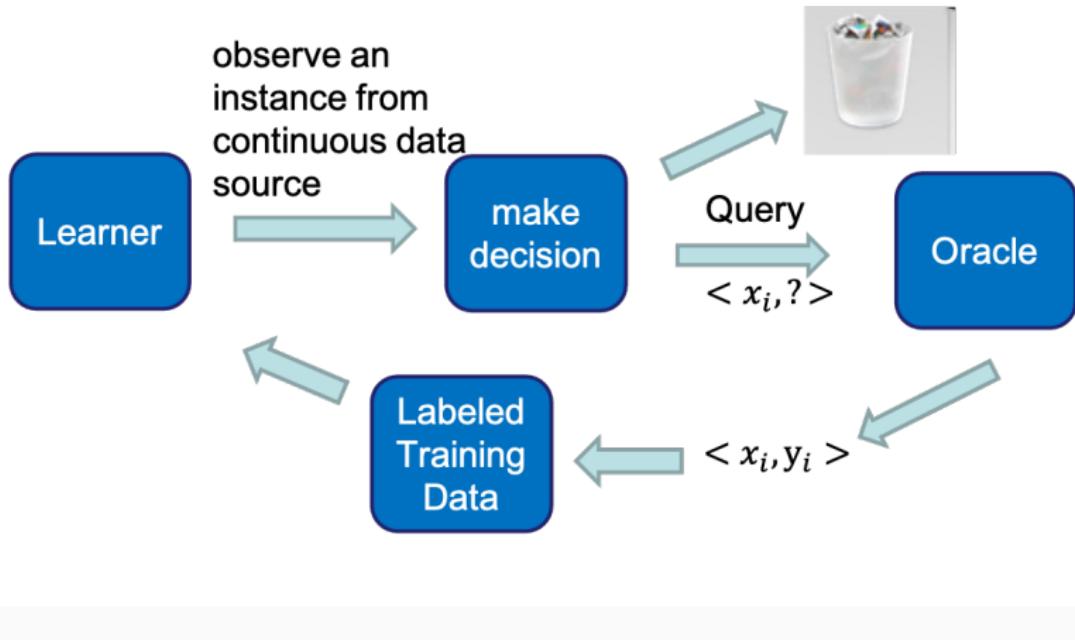


Source: Kevin J. Lang and Eric B Baum. Query Learning Can Work Poorly when a Human Oracle is Used, 1992



# Stream-based Sampling I

Learner decides query or ignore the observed instance from the continuous data source



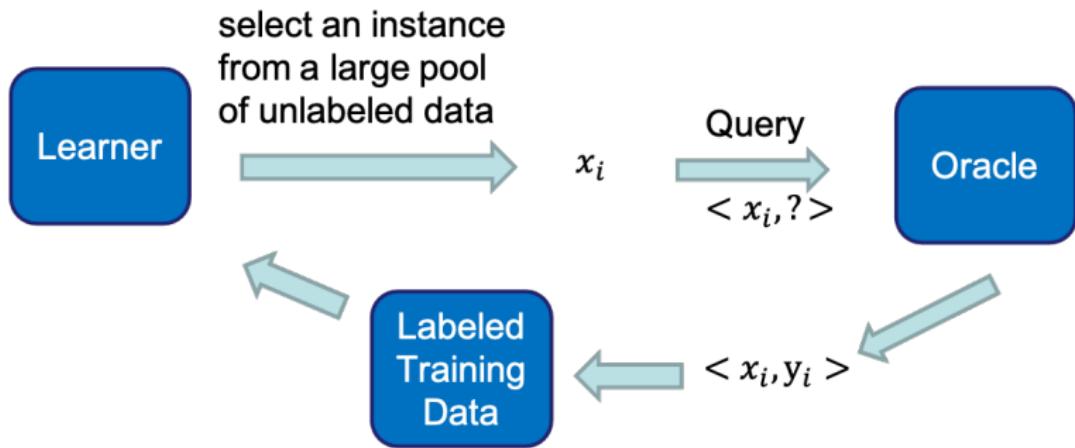
## Stream-based Sampling II

- Query data from true distribution
- Useful if the dataset is too large to load
- Assumption: drawing an instance is less expensive than labeling, e.g., downloading video vs annotating actions



## Pool-based Sampling I

Learner chooses the best instance from a large pool of unlabeled examples to query.



## Stream-based vs Pool-based Sampling

- Stream-based: the learner observes one instance at a time and makes the decision individually.
- Pool-based: the learner observes whole dataset and choose the best one.



# Query Strategies

- Uncertainty Sampling
- Query by Committee



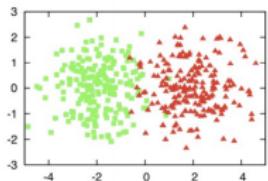
# Uncertainty Sampling

- Initial labeled instances  $L = \{x_i, y_i\}_{i=1}^l$
- Unlabeled instances  $U = \{x_i\}_{i=l+1}^{l+u}$
- Repeat
  - Supervised learning: Train a model  $f$  on  $L$
  - Prediction:  $y = f(U)$
  - Select  $x^*$  with the most uncertain prediction
  - query the oracle to obtain label  $y^*$
  - add  $\langle x^*, y^* \rangle$  to  $L$
  - Remove  $x^*$  from  $U$

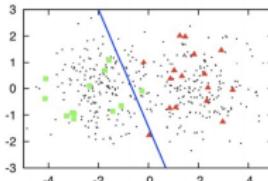


# Uncertainty Sampling

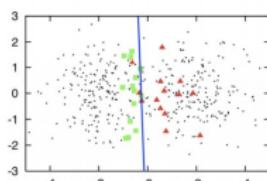
Query the instance the learner is most uncertain about



400 instances sampled  
from 2 class Gaussians



random sampling  
30 labeled instances  
(accuracy=0.7)



uncertainty sampling  
30 labeled instances  
(accuracy=0.9)



# Query Strategies I: Uncertainty Sampling

- Least Confident
- Margin Sampling
- Entropy Sampling



## Uncertainty Sampling I: Least Confident

- Query instances where the classifier is least confident of the classification

$$x^* = \underset{x}{\operatorname{argmin}}(P_\theta(\hat{y}|x))$$

where

$$\hat{y} = \underset{y}{\operatorname{argmax}}(P_\theta(y|x))$$

- Example: select instance 2 as the query

	$y_1$	$y_2$	$y_3$
Instance 1	0.01	0.9	0.09
Instance 2	0.5	0.3	0.2



## Uncertainty Sampling II: Margin Sampling

- Selects queries where the classifier is least able to distinguish between the first and second most probable categories, e.g.:

$$x = \underset{x}{\operatorname{argmin}}(P_{\theta}(\hat{y}_1|x) - P_{\theta}(\hat{y}_2|x))$$

- where  $\hat{y}_1$  and  $\hat{y}_2$  are the first- and second-most-likely labels for  $x$
- Example: select instance 2 as the query

	$y_1$	$y_2$	$y_3$
Instance 1	0.25	0.5	0.25
Instance 2	0.5	0.4	0.1



## Uncertainty Sampling III: Entropy Sampling

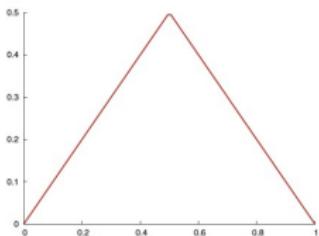
- Use entropy as an uncertainty measure to utilize all the possible class probabilities:

$$x = \underset{x}{\operatorname{argmax}} - \sum_i P_{\theta}(\hat{y}_i|x) \log_2 P_{\theta}(\hat{y}_i|x)$$

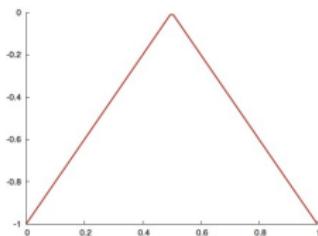


# Binary-Class Uncertainty

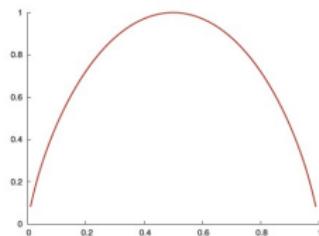
For binary tasks, these are equivalent.



(a) least confident – binary



(b) margin – binary

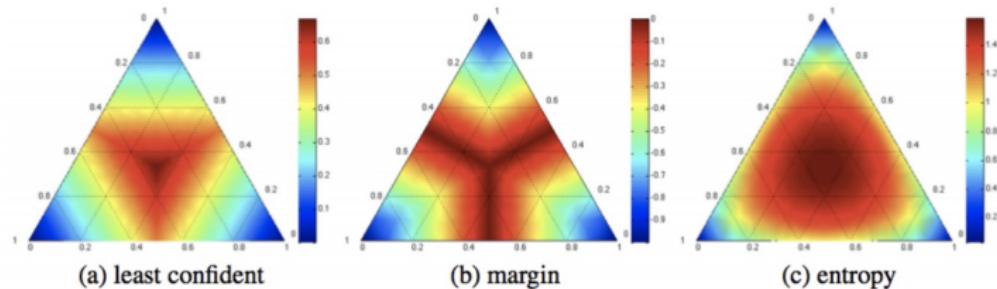


(c) entropy – binary



# Multi-Class Uncertainty

For multi-class tasks, these are not equivalent.



## Example

- Which instance should be the query based on the strategy of least confidence?
- Which instance should be the query based on the strategy of margin sampling?
- Which instance should be the query based on the strategy of entropy?

	$y_1$	$y_2$	$y_3$	$y_4$
Instance 1	0.2	0.4	0.2	0.2
Instance 2	0.5	0.35	0.1	0.05

# Applications & Other Uncertainty Sampling Methods

- 1 Speech Recognition
  - 2 Machine Translation
  - 3 Text Classification
  - 4 Word Segmentation: classifier margin
- 
- 1 Hakkani-Tür, Dilek, Giuseppe Riccardi, and Allen Gorin. "Active learning for automatic speech recognition." ICASP 2002.
  - 2 Haffari, Gholamreza, Maxim Roy, and Anoop Sarkar. "Active learning for statistical phrase-based machine translation." ACL 2009.
  - 3 Lewis, David D., and William A. Gale. "A sequential algorithm for training text classifiers." SIGIR 1994.
  - 4 Sassano, Manabu. "An empirical study of active learning with support vector machines for Japanese word segmentation." ACL 2002.



## Query Strategies II: Query by Committee

- Use multiple classifiers to predict on unlabelled data, and select instances with the highest disagreement between classifiers
- Assumes that all the classifiers learn something different, so can provide different information
- Disagreement can be measured by:
  - Vote entropy
  - KL divergence



## Disagreement Measure I: Vote Entropy

Select instance with highest vote entropy for query:

$$x = \operatorname{argmax}_x - \sum_{y_i} \left( \frac{V(y_i)}{N} \right) \log_2 \left( \frac{V(y_i)}{N} \right)$$

- $V(y_i)$ : number of “votes” that label  $y_i$  receives.
- $N$ : total number of “votes” (classifiers).



## Example

$$V(y_1) = 0, V(y_2) = 4, V(y_3) = 0$$

$$H = 0$$

classifier	$y_1$	$y_2$	$y_3$
$C_1$	0.2	0.7	0.1
$C_2$	0.2	0.6	0.2
$C_3$	0.05	0.9	0.05
$C_4$	0.1	0.8	0.1



## Example

$H = ?$

classifier	$y_1$	$y_2$	$y_3$
$C_1$	0.2	0.7	0.1
$C_2$	0.1	0.3	0.6
$C_3$	0.8	0.1	0.1
$C_4$	0.3	0.5	0.2



## Disagreement Measure II: KL Divergence

$$x = \underset{x}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^N D(P_i || P_m)$$

- $P_m$ : mean probability distribution of all the  $N$  models.
- Kullback Leibler (KL) divergence (relative entropy)

$$D(P_i || P_m) = - \sum_{j=1}^{n_c} P_i(j) [\log_2 P_m(j) - \log_2 P_i(j)] = \sum_{j=1}^{n_c} P_i(j) \log_2 \frac{P_i(j)}{P_m(j)}$$

- $P_i = [P_i(1), P_i(2), \dots, P_i(n_c)]$
- $P_m = [P_m(1), P_m(2), \dots, P_m(n_c)]$
- $P_i(j)$ : probability of the  $j^{th}$  class in the probability distribution  $P_i$
- $P_m(j)$ : probability of the  $j^{th}$  class in the probability distribution  $P_m$



## **Semi-supervised learning**

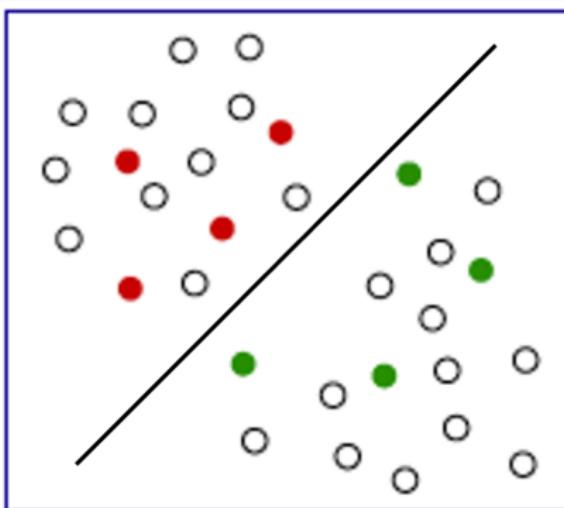
---

- Semi-supervised learning is learning from both labelled and unlabelled data
- **Semi-supervised classification:**
  - $L$  is the set of labelled training instances  $\{x_i, y_i\}_{i=1}^l$
  - $U$  is the set of unlabelled training instances  $\{x_i\}_{i=l+1}^{l+u}$
  - Often  $u \gg l$
  - Goal: learn a better classifier from  $L \cup U$  than is possible from  $L$  alone



## Semi-Supervised Learning Approach I

- A simple approach: combine a supervised and unsupervised model
- e.g., Find clusters, choose a label for each (most common label?) and apply it to the unlabelled cluster members



## Self-Training (Also known as “Bootstrapping”)

- Assume you have  $L = \{x_i, y_i\}_{i=1}^l$  labelled and  $U = \{x_i\}_{i=l+1}^{l+u}$  unlabelled training instances
- Repeat
  - Supervised learning: Train a model  $f$  on  $L$
  - Prediction:  $y = f(U)$  to predict the labels on each instance in  $U$
  - Identify a subset  $U'$  of  $U$  with “high confidence” labels
  - $L \leftarrow L \cup \langle U', f(U') \rangle$
  - $U \leftarrow U \setminus U'$
  - Until  $L$  does not change



# Active Learning vs Semi-supervised learning

- Same goal: reduce human annotation effort
- semi-supervised learning:
  - Learner produce labels automatically (e.g., on the data with high confidence)
- active learning:
  - Learner select unlabeled data (e.g., with low confidence/high uncertainty) to make a query
  - Oracle annotates the query



## **Summary**

---

# Summary

- What is active learning?
- What are the main sampling strategies in active learning?
- Outline a selection of query strategies in active learning.
- What is semi-supervised learning?
- What is self-training, and how does it operate?



## References

- Burr Settles. Active learning literature survey. Technical report, Department of Computer Sciences, University of Wisconsin, Madison, 2010.
- Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison, 2005.
- Xiaojin Zhu. Tutorial on semi-supervised learning.
- Edith Law. Introduction to machine learning-active learning



# Anomaly Detection

---

**COMP90049**

Semester 2, 2021

QiuHong Ke, CIS

©2021 The University of Melbourne

Acknowledgement: Jeremy Nicholson, Tim Baldwin Karin Verspoor



# Roadmap

So far:

- Supervised learning
- Unsupervised learning
- Active learning
- Semi-supervised learning

Today: Anomaly Detection

- Anomaly
  - Definition
  - Types
- Anomaly Detection Algorithms
  - Statistical
  - Proximity-based
  - Density-based
  - Clustering-based

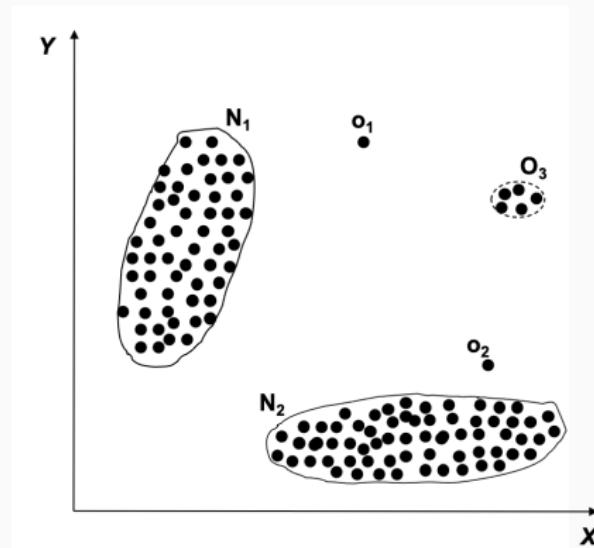


## Anomaly

---

# What is Outlier/Anomaly?

A pattern in the data that does not conform to the normal/standard/expected behavior



# Why Anomaly Detection?

Anomalous events are rare but can lead to dramatic (and often negative) consequence

Applications:

- Fraud Detection: odd credit card charges
- Ecosystem Disturbances: floods, droughts, heat waves
- Medicine and public health: influenza outbreaks
- Aviation Safety: abnormal pilot behavior or aircraft sequence of events



# Type of Anomaly

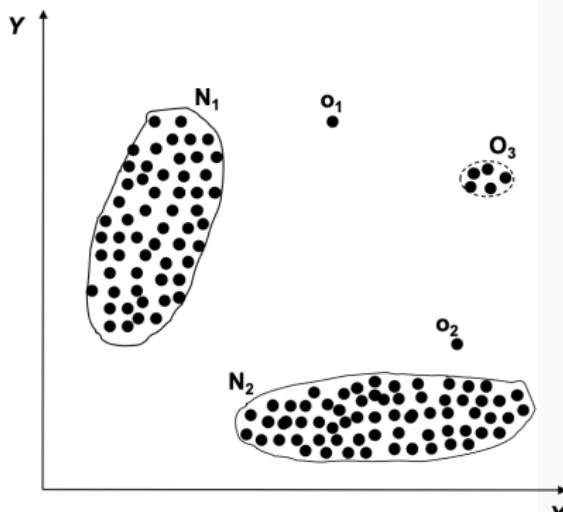
- Point/global anomalies
- Contextual/conditional anomalies
- Collective anomalies



## Point/global anomalies

An individual data instance is anomalous w.r.t. the data (deviate significantly from the entirety of the data set)

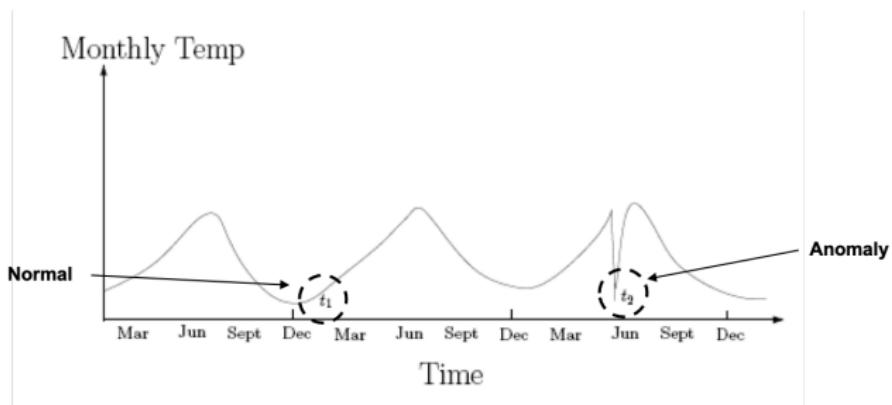
- Example: credit card fraud based on "amount spent."
- Detection: Find an appropriate measurement of deviation



## Contextual/conditional anomalies

An individual data instance is anomalous within a context ((deviate significantly from the rest of data points in the same context)

- Example:
  - 150 heart rate is normal during exercise, but may be odd at rest.
  - Temperature in Paris:



## Contextual/conditional anomalies

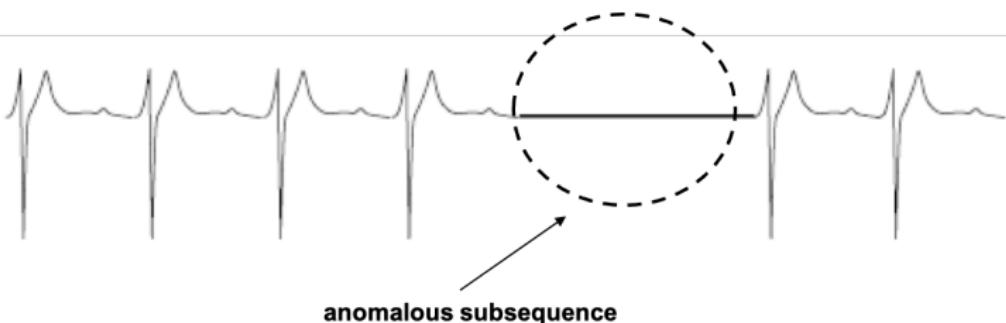
- Attributes of data objects should be divided into two groups
  - Contextual attributes: defines the context, e.g., time
  - Behavioral attributes: characteristics of the object, used in anomaly evaluation, e.g., temperature
- Detection: How to define or formulate meaningful context?



## Collective anomalies

A subset of data points is anomalous (deviate significantly from the entire data set)

- The individual instances within a collective anomaly are not anomalous by themselves
- Example:
  - cyber intrusion: Repeated failed login attempts
  - Heart rate signal:



# Collective anomalies

Detection:

- Consider behavior of groups of objects
- Requires a relationship among data instances
  - Sequential data
  - Spatial data
  - Graph data



# Anomaly vs Noise

- Anomalies are different from noise
  - Noise is random error
    - Label annotated incorrectly
    - Feature measured incorrectly
  - Noise is not necessarily interesting
  - Noise should be removed before anomaly detection
- Anomalies are interesting:
  - They violate the mechanism that generates the normal data
  - translate to significant (often critical) real life entities (e.g., cyber intrusions, credit card fraud)



## Anomaly Detection Algorithms

---

# Supervised Anomaly Detection

- Labels available for both normal data and anomalies
- Build classifier to distinguish between normal and known anomalies
- Challenges
  - Requires labels for both normal data and anomalies
  - Imbalanced classes,
  - Cannot detect unknown and emerging anomalies



## Semi-supervised Anomaly Detection

- Labels available only for normal data
- Model normal objects and report those not matching the model as outliers
- Challenges:
  - Require labels from normal class
  - Possible high false alarm rate - previously unseen (yet legitimate) data records may be recognized as anomalies



## Unsupervised point anomaly detection

- Statistical methods (model-based methods)
- Proximity-based: the nearest neighbors of outliers are far away
- Density-based: Outliers are objects in regions of low density
- Clustering-based Normal data belong to large and dense clusters



# Statistical anomaly detection I

Anomalies are objects that are fit poorly by a statistical model.

- Assumption: normal data is generated by a parametric distribution
- Idea:
  - Estimate the parameters probability density function (PDF) of the distribution
  - Identify the instances in low probability regions of the distribution as anomalies
- Challenges of Statistical testing:
  - highly depends on whether the assumption of statistical model holds in the real data



# Univariate Data I

Assumption: Gaussian distribution

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

$$mean : \mu = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$variance : \sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$



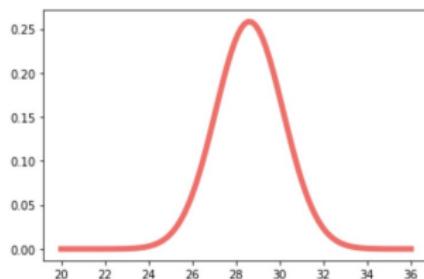
# Univariate Data I

temp.: 24.0, 28.9, 28.9, 29.0, 29.1, 29.1, 29.2, 29.2, 29.3, 29.4

- Assumption: Gaussian distribution

$$\mu = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = 28.61$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 = 1.51$$



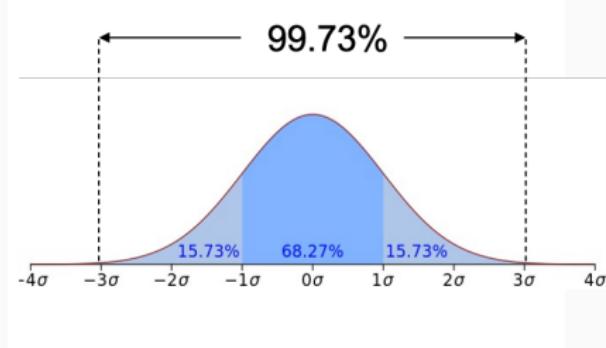
- Calculate probability using probability density function
- Outlier: low probabilities



## Univariate Data II

temp.: 24.0, 28.9, 28.9, 29.0, 29.1, 29.1, 29.2, 29.2, 29.3, 29.4

- set a normal limit:  $\mu \pm 3\sigma$  (the region contains 99.73% data)
- Then 24 is an outlier since:  $(24 - 28.61)/1.51 = -3.04 < -3$



# Multivariate Data I

Multivariate Gaussian distribution

$$f(x) = \frac{1}{\sqrt{(2\pi)^k \det S}} \exp \left( -\frac{1}{2} (x - \mu)^T S^{-1} (x - \mu) \right)$$

$\mu$ : the mean.

$k$ : dim of feature space.

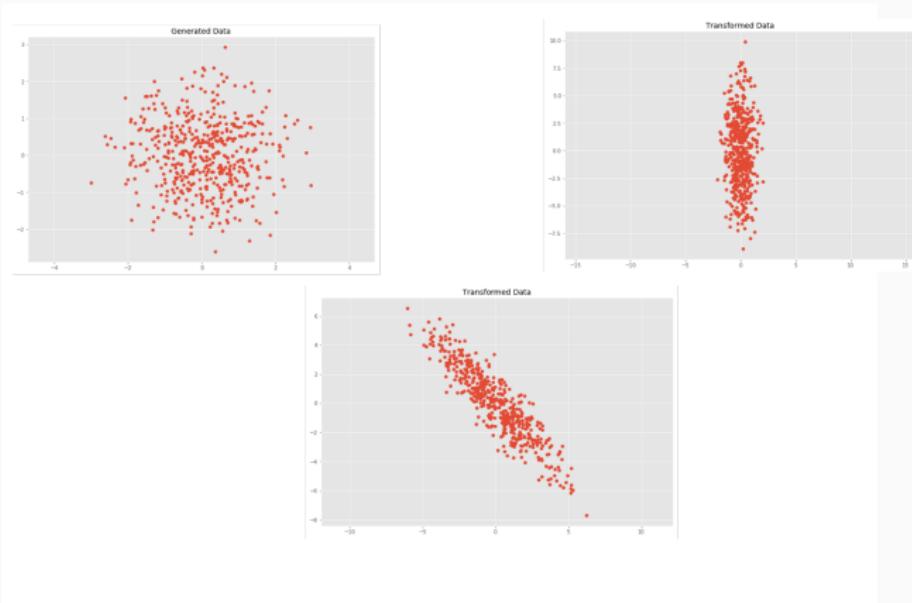
$S$ : covariance matrix.

For a 2-dimensional data:

$$S = \begin{bmatrix} \sigma^2(x, x) & \sigma^2(x, y) \\ \sigma^2(y, x) & \sigma^2(y, y) \end{bmatrix}$$



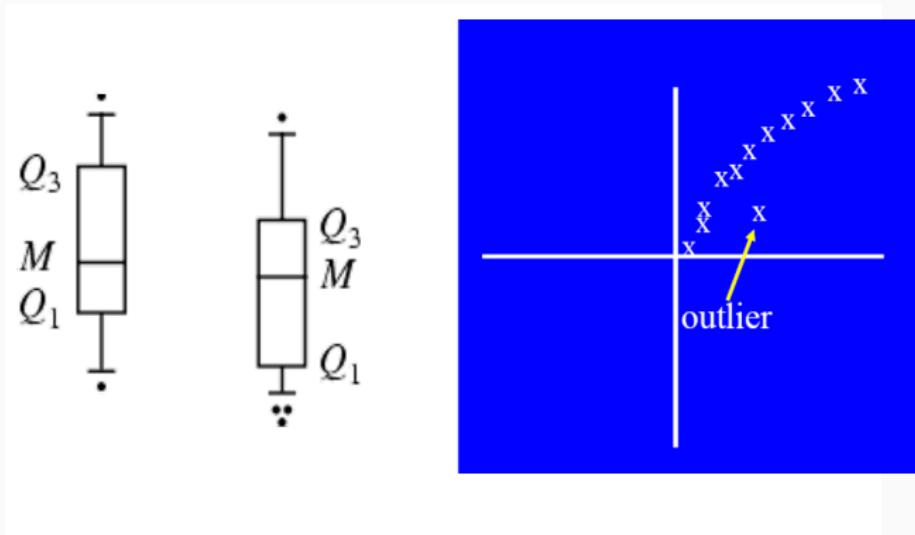
## Multivariate Data II



# Statistical anomaly detection II

## Graphical Approaches

- Boxplot (1-D), Scatter plot (2-D)
- Limitations
  - Time consuming
  - Subjective



## Example

temp.: 24.0, 28.9, 28.9, 29.0, 29.1, 29.1, 29.2, 29.2, 29.3, 29.4

- Median: 29.1
- Q1: 28.9
- Q3: 29.2
- IQR:  $29.2 - 28.9 = 0.4$
- Minimum:  $Q1 - 1.5 \cdot IQR = 28.3$
- Maximum:  $Q3 + 1.5 \cdot IQR = 29.8$
- $24.0 < \text{Minimum}$ : outlier



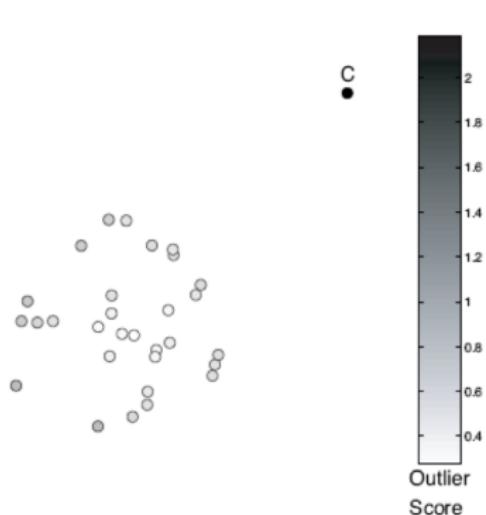
## Proximity-based Anomaly detection

An object is an anomaly if the nearest neighbors of the object are far away,

- Compute the distance between every pair of data points
- To determine outliers:
  - Data points for which there are fewer than  $p$  neighboring points within a distance  $D$
  - The top  $n$  data points whose distance to the  $k$ th nearest neighbor is greatest
  - The top  $n$  data points whose average distance to the  $k$  nearest neighbors is greatest



## Proximity-based Anomaly detection



**Figure 10.4.** Outlier score based on the distance to fifth nearest neighbor.



## Proximity-based (Nearest-Neighbor based) Anomaly detection

- Pros:
  - Easier to define a proximity measure for a dataset than determine its statistical distribution.
  - Quantitative measure of degree to which object is an outlier.
- Cons:
  - $O(n^2)$  complexity.
  - outlier score is sensitive to choice of  $k$ .
  - Does not work well if data has widely variable density.



## Proximity-based Anomaly detection

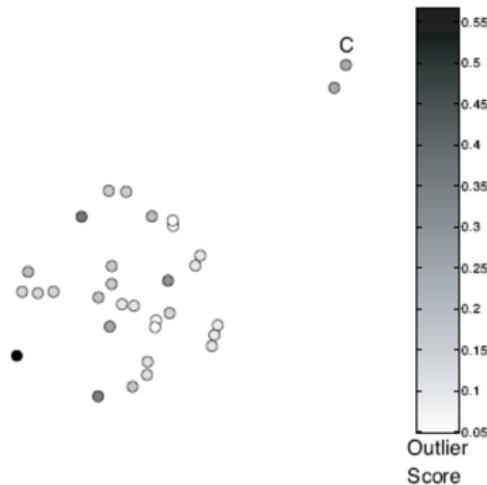
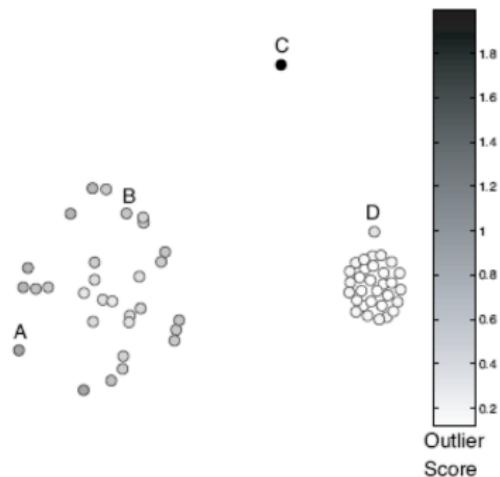


Figure 10.5. Outlier score based on the distance to the first nearest neighbor. Nearby outliers have low outlier scores.



## Proximity-based Anomaly detection



**Figure 10.7.** Outlier score based on the distance to the fifth nearest neighbor. Clusters of differing density.



## Density-based outlier detection

Outliers are objects in regions of low density

- Outlier score is inverse of density around object.
- Density scores usually based on proximities. Example density scores:
  - Number of objects within fixed radius  $d$ .
  - inverse of average distance to  $k$  nearest neighbors:

$$\text{density}(x, k) = \frac{1}{\frac{1}{k} \sum_{y \in N(x, k)} \text{distance}(x, y)}$$

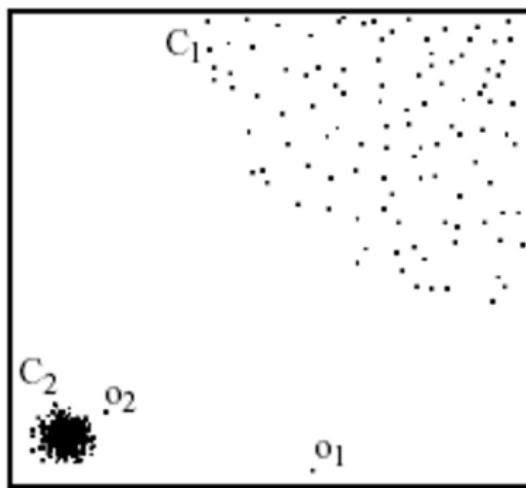
- These above two example scores work poorly if data has variable density.
- Relative density outlier score (Local Outlier Factor, LOF):

$$\text{relative density}(x, k) = \frac{\text{density}(x, k)}{\frac{1}{k} \sum_{y \in N(x, k)} \text{density}(y, k)}$$



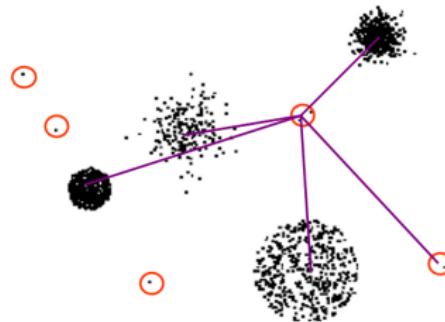
## Example

How do you compare Proximity (Nearest-Neighbor) based and LOF in finding outliers?



# Cluster-based Outlier Detection

Outliers are objects that do not belong strongly to any cluster



Approaches:

- Assess degree to which object belongs to any cluster.
- Eliminate object(s) to improve objective function.
- Discard small clusters far from other clusters.
- Issue: Outliers may affect initial formation of clusters.



## Cluster-based outlier detection

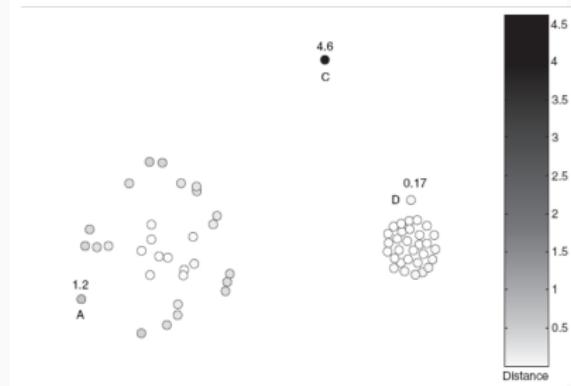
Assess degree to which object belongs to any cluster:

- For prototype-based clustering (e.g. k-means), use distance to cluster centers.
- To deal with variable density clusters, use relative distance:

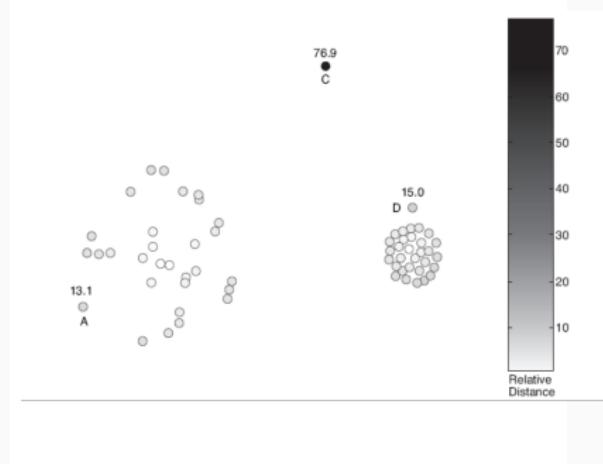
$$\frac{\text{distance}(\mathbf{x}, \text{centroid}_C)}{\text{median}(\{\forall_{x' \in C} \text{distance}(\mathbf{x}', \text{centroid}_C)\})}$$



# Cluster-based outlier detection



# Cluster-based outlier detection



## Cluster-based outlier detection

Pro:

- Some clustering techniques have  $O(n)$  complexity.
- Extends concept of outlier from single objects to groups of objects.

Cons:

- Requires thresholds for the distance.
- Sensitive to number of clusters chosen.
- Outliers may affect initial formation of clusters.



## **Summary**

---

# Summary

- Types of outliers: global, contextual collective outliers
- Outlier detection: supervised, semi-supervised, or unsupervised
  - Statistical (or model-based) approaches
  - Proximity-base approaches
  - Density-based approaches
  - Clustering-base approaches



## References

- Tan et al (2006) Introduction to Data Mining. Section 4.3, pp 150-171. (Chapter 10)
- V. Chandola, A. Banerjee, and V. Kumar, (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.
- A. Banerjee, et al (2008). Tutorial session on anomaly detection. The SIAM Data Mining Conference (SDM08)



# Ethics in Machine Learning: Measuring and Mitigating Algorithmic Bias

---

**COMP90049**

**Introduction to Machine Learning**

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



# Roadmap

## So far... ML nuts and bolts

- Supervised learning / classification
- Unsupervised learning
- Features selection
- Evaluation



## So far... ML nuts and bolts

- Supervised learning / classification
- Unsupervised learning
- Features selection
- Evaluation

## Today... ML in the world

- What is Bias and where does it come from?
- Algorithmic Fairness
- How to make ML algorithms fairer



## **Introduction**

---

# Applications

- medical diagnoses
- language generation
- hate speech detection
- stock market prediction
- insurance policy suggestion
- search and recommendation
- spam / malware detection
- predictive policing
- ...



# Applications

- medical diagnoses
- language generation
- hate speech detection
- stock market prediction
- insurance policy suggestion
- search and recommendation
- spam / malware detection
- predictive policing
- ...

**What can possibly go wrong?**



# A quick (and biased) press review

News | Opinion | Sport | Culture | Lifestyle | More ▾

Australia World AU politics Environment Football Indigenous Australia Immigration Media Business Science Tech

**Artificial intelligence (AI)** • This article is more than 1 month old

## AI expert calls for end to UK use of 'racially biased' algorithms

Prof Noel Sharkey says systems so infected with biases they cannot be trusted

Henry McDonald  
Fri 13 Dec 2019 01.07 AEDT

221

▲ Facial recognition technology has also come under scrutiny. Photograph: Fanatic Studio/Gary Waters/Getty /Collection Mix: Subjects RF

An expert on artificial intelligence has called for all algorithms that make life-changing decisions - in areas from job applications to immigration into the UK - to be halted immediately.

Prof Noel Sharkey, who is also a leading figure in a global campaign against "killer robots", said algorithms were so "infected with biases" that their decision-making processes could not be fair or trusted.

A moratorium must be imposed on all "life-changing decision-making algorithms" in Britain, he said.

### Read The Guardian without interruption on all your devices

Subscribe now

most viewed in Australia

- Live** Oscars 2020: Parasite wins best picture award - live!
- Live** Oscar winners 2020: the full awards list - live!
- Live** NSW floods and weather: rain eases but chaos continues - live
- Live** Morrison questioned over help for bushfire-affected businesses - politics live
- Live** Coronavirus live updates: WHO sends experts to China as cases exceed 40,000 - latest news

UNIVERSITY OF ELBOURNE

# A quick (and biased) press review

[News](#)[Opinion](#)[Sport](#)[Culture](#)[Lifestyle](#)[More](#) ▾

World ▶ Europe US Americas Asia Australia Middle East Africa Inequality Cities Global development

## Artificial intelligence (AI)

Jon Henley and Robert Booth

The 6 Feb 2020 00.18 AEDT



1,303

## Welfare surveillance system violates human rights, Dutch court rules

Government told to halt use of AI to detect fraud in decision hailed by privacy campaigners



▲ People in Rotterdam, the Netherlands. The Dutch system aimed to predict the likelihood of an individual committing benefit or tax fraud, or violating labour laws. Photograph: Geography Photos/UIG via Getty Images

A Dutch court has ordered the immediate halt of an automated surveillance system for detecting welfare fraud because it violates human rights, in a judgment likely to resonate well beyond the [Netherlands](#).

The case was seen as an important legal challenge to the controversial but growing use by governments around the world of artificial intelligence (AI) and risk modelling in administering welfare benefits and other core services.

Campaigners say such "digital welfare states" - developed often without consultation, and operated secretly and without adequate oversight -

Read The Guardian without interruption on all your devices

Subscribe now



### most viewed in Australia



[Live Oscars 2020: Parasite wins best picture award - live!](#)



[Oscar winners 2020: the full awards list - live!](#)



[Live NSW floods and weather: rain eases but chaos continues - live](#)



[Live Morrison questioned over help for bushfire-affected businesses - politics live](#)



[Live Coronavirus live updates: WHO sends experts to China as cases exceed 40,000 - latest news](#)



# A quick (and biased) press review

News   Opinion   Sport   Culture   Lifestyle   More ▾

Australia World AU politics Environment Football Indigenous Australia Immigration Media Business Science Tech

**Artificial intelligence (AI)**

## New AI fake text generator may be too dangerous to release, say creators

The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse

Alex Hern  
@alexhrn  
Fri 15 Feb 2019 04.00 AEDT

6,686 572



▲ The AI wrote a new passage of fiction set in China after being fed the opening line of Nineteen Eighty-Four by George Orwell (pictured). Photograph: Mondadori/Getty Images

The creators of a revolutionary AI system that can write news stories and works of fiction - dubbed "deepfakes for text" - have taken the unusual step of not releasing their research publicly, for fear of potential misuse.

OpenAI, an nonprofit research company backed by Elon Musk, Reid Hoffman, Sam Altman, and others, says its new AI model, called GPT2 is so good and the risk of malicious use so high that it is breaking from its normal practice of releasing the full research to the public in order to allow more time to discuss the ramifications of the technological breakthrough.

**Read The Guardian without interruption on all your devices**

**Subscribe now**



most viewed in Australia

**Live** Oscars 2020: Parasite wins best picture award - live!

**Live** Oscar winners 2020: the full awards list - live!

**Live** NSW floods and weather: rain eases but chaos continues - live

**Live** Morrison questioned over help for bushfire-affected businesses - politics live

**Live** Coronavirus live updates: WHO sends experts to China as cases exceed 40,000 - latest news



# A quick (and biased) press review

News    Opinion    Sport    Culture    Lifestyle    More ▾

Columnists Cartoons Indigenous Editorials Letters

**Opinion**  
Computing

## Can the planet really afford the exorbitant power demands of machine learning?

*John Naughton*

Sun 17 Nov 2019 03.00 AEDT

270 348

The environmental impact of such technological advances can be huge



Only huge firms such as Facebook can house the number of processors that machine learning requires.  
Photograph: Jim Thompson/Zuma Press/eyevine

**T**here is, alas, no such thing as a free lunch. This simple and obvious truth is invariably forgotten whenever irrational exuberance teams up with digital technology in the latest quest to "change the world". A case in point was the [bitcoin frenzy](#), where one could apparently become insanely rich by "mining" for the elusive coins. All you needed was to get a computer to solve a complicated mathematical puzzle and - lo! - you

**Read The Guardian without interruption on all your devices**

**Subscribe now**



most viewed in Australia

**Live** Oscars 2020: Parasite wins best picture award - live!

**Oscar winners 2020: the full awards list - live!**

**Live** NSW floods and weather: rain eases but chaos continues - live

**Live** Morrison questioned over help for bushfire-affected businesses - politics live

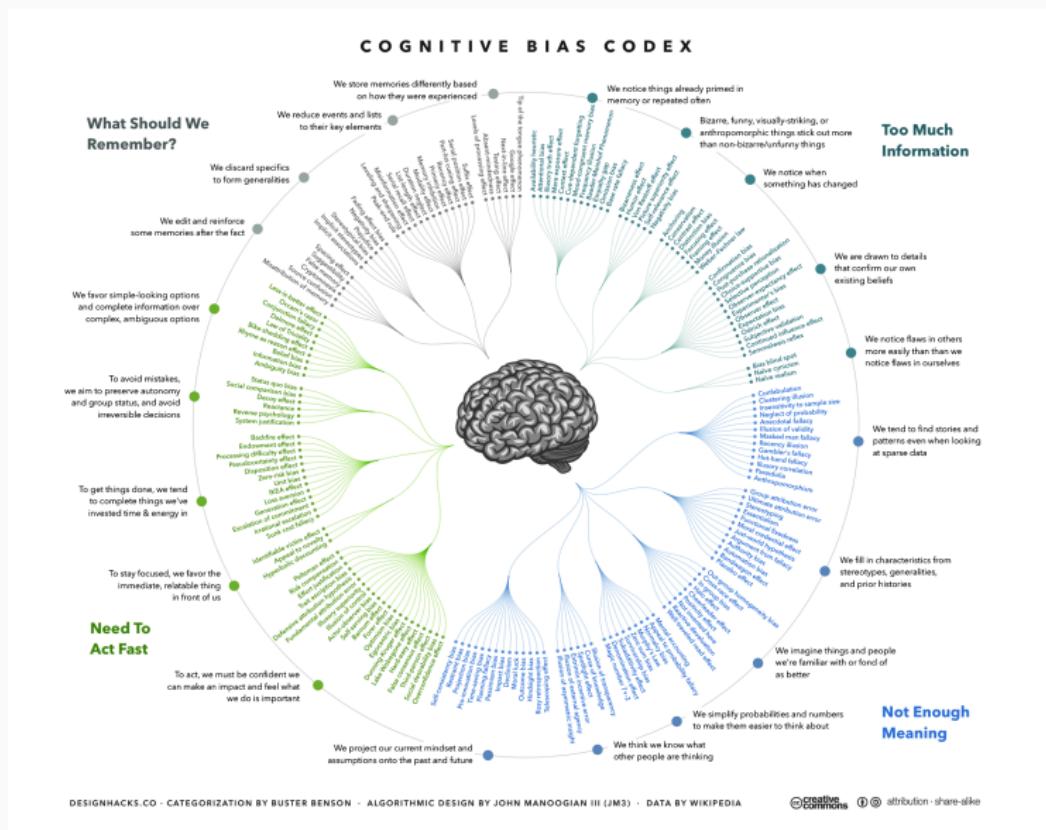
**Live** Coronavirus live updates: WHO sends experts to China as cases exceed 40,000 - latest news



## **Sources and types of bias in ML**

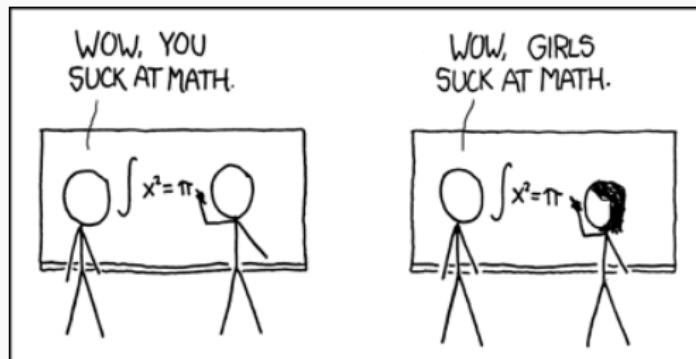
---

# Humans are biased



# Humans are biased

## Out-group homogeneity bias (Stereotypes/Prejudice)

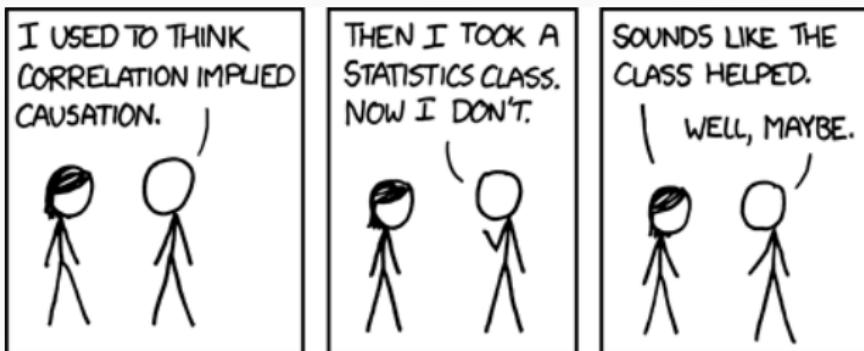


<https://xkcd.com/385/>

Humans tend to perceive out-group members as less nuanced than in-group members.

# Humans are biased

## Correlation Fallacy



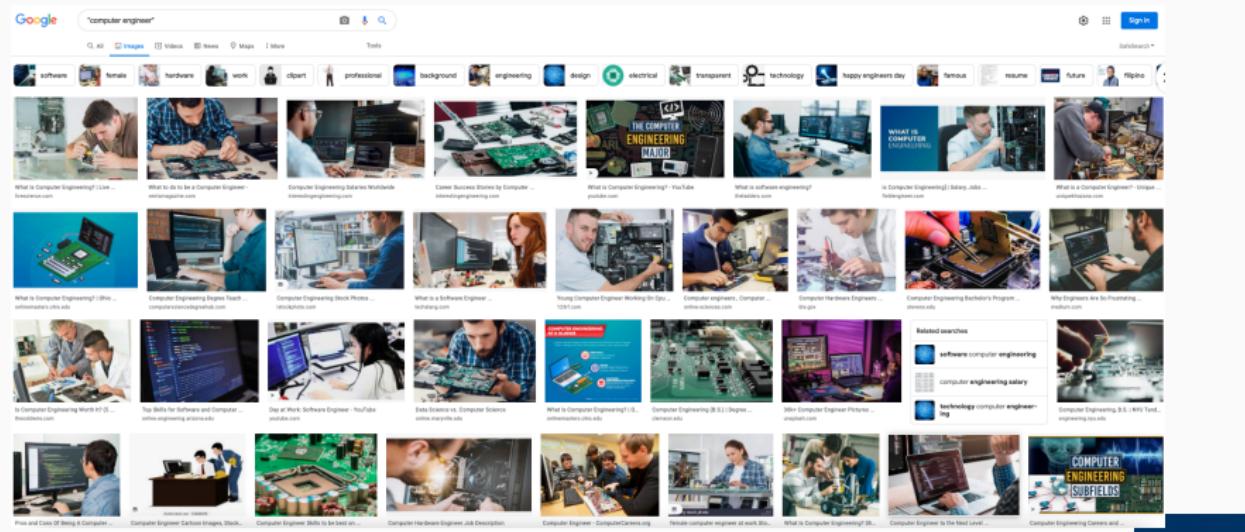
<https://xkcd.com/552/>

Humans have a tendency to mistake correlation (two co-incidentally co-occurring events) with causation.



# Data is biased

**Historical Bias:** A randomly sampled data set, reflects the world *as it was* including existing biases which should not be carried forward



# Data is biased

## Representation bias / Reporting bias

- The data sets do not faithfully represent the whole population
- Minority groups are underrepresented
- Obvious facts are underrepresented. Anomalies are overemphasized.

Word	Teraword	Word	Teraword
spoke	11,577,917	hugged	610,040
laughed	3,904,519	blinked	390,692
murdered	2,843,529	was late	368,922
inhaled	984,613	exhaled	168,985
breathed	725,034	was punctual	5,045

From: Gordon and van Durme (2013)



# Data is biased

## Measurement bias

1. Noisy measurement → errors or missing data points which **are not randomly distributed**
  - e.g., records of police arrests differ in level of detail across postcode areas
2. Mistaking a **(noisy) proxy** for a label of interest
  - e.g., ‘hiring decision’ as a proxy for ‘applicant quality’. **(why noisy?)**
3. **Oversimplification** of the quantity of interest
  - e.g., classifying political leaning into: ‘Democrat’ vs. ‘Republican’ (USA); binarizing gender into: ‘Male’ vs. ‘Female’



# Data is biased

## Measurement bias

1. Noisy measurement → errors or missing data points which **are not randomly distributed**
  - e.g., records of police arrests differ in level of detail across postcode areas
2. Mistaking a **(noisy) proxy** for a label of interest
  - e.g., ‘hiring decision’ as a proxy for ‘applicant quality’. **(why noisy?)**
3. **Oversimplification** of the quantity of interest
  - e.g., classifying political leaning into: ‘Democrat’ vs. ‘Republican’ (USA); binarizing gender into: ‘Male’ vs. ‘Female’

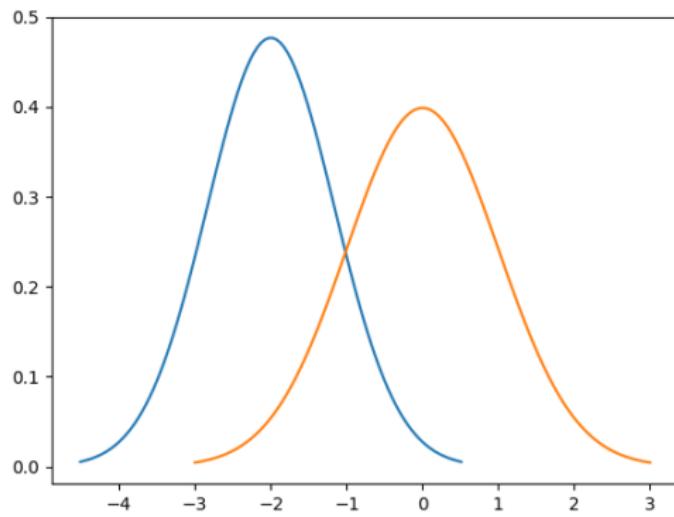
1. Know your domain
2. Know your task
3. Know your data



# Models are Biased

## Model Fit

- Weak models: high bias – low variance
- Unjustified model assumptions



# Models are Biased

## Biased Loss Function

- Blind to certain types of errors
- E.g., 0/1 loss will tend to tolerate errors in the minority class for highly imbalanced data



# Models are Biased

## Biased Loss Function

- Blind to certain types of errors
- E.g., 0/1 loss will tend to tolerate errors in the minority class for highly imbalanced data

1. Carefully consider model assumptions
2. Carefully choose loss functions
3. Model groups separately (e.g., multi-task learning)
4. Represent groups fairly in the data



# Bias in Evaluation or Deployment

## Evaluation bias

- Test set not representative of target population
- Overfit to a *test* set. Widely used benchmark data sets can reinforce the problem.
- Evaluation metrics may not capture all quantities of interest (disregard minority groups or average effects). E.g.,
  - Accuracy?
  - Face recognition models largely trained/evaluated on images of ethnically white people.

## Deployment bias

- Use of systems in ways they weren't intended to use. Lack of education of end-users.



# Bias in Evaluation or Deployment

## Evaluation bias

- Test set not representative of target population
- Overfit to a *test* set. Widely used benchmark data sets can reinforce the problem.
- Evaluation metrics may not capture all quantities of interest (disregard

1. Carefully select your evaluation metrics

2. Use multiple evaluation metrics

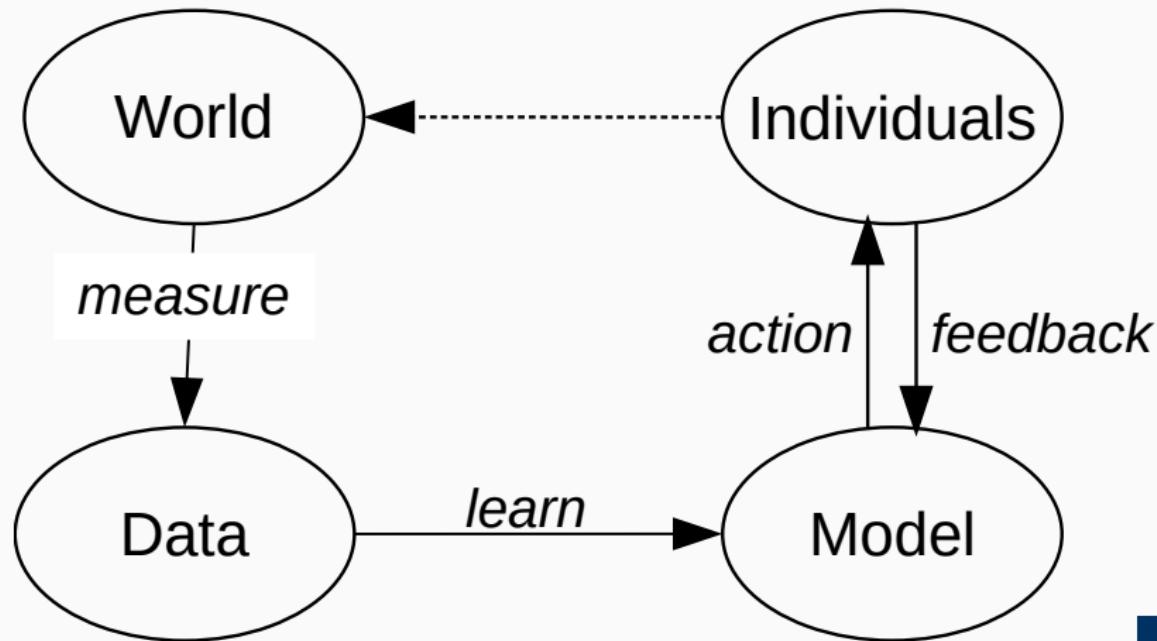
3. Carefully select your test sets and benchmarks

D 4. Document your models to ensure they are used correctly

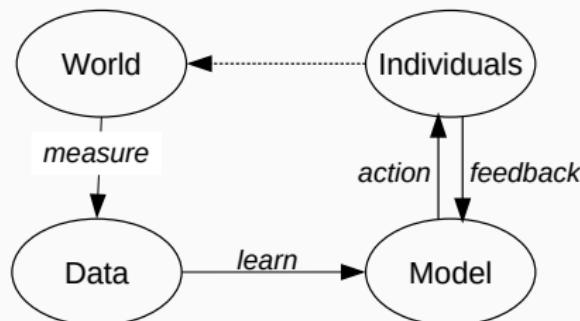
- Use of systems in ways they weren't intended to use. Lack of education of end-users.



# The Machine Learning Pipeline



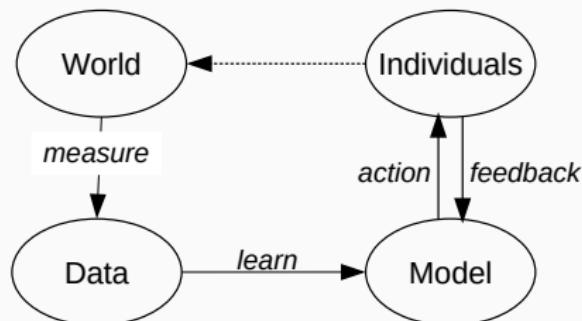
# The Machine Learning Pipeline



## Measurement

- Define your variables of interests
- Define your target variable
- Especially critical, if target variable is not measured explicitly.  
E.g., *hiring decision* → *applicant quality* or *income* → *creditworthiness*

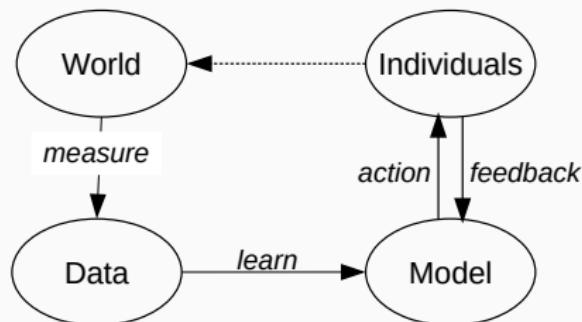
# The Machine Learning Pipeline



## Learning

- Models are faithful to the data (by design!)
- Data contains “knowledge” (*smoking causes cancer*)
- Data contains “Stereotypes” (*boys like blue, girls like pink*)
- What’s the difference? Based on social norms, no clear line!

# The Machine Learning Pipeline

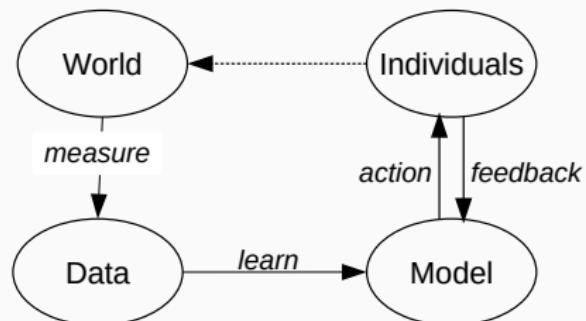


## Action

- ML concept: regression, classification, information retrieval, ...
- Resulting action: Class prediction (Spam, credit granted), search results, hotel recommendation, ...



# The Machine Learning Pipeline



## Feedback

- Approximated from user behavior
- Ex: click-rates



# Demographic Disparity / Sample Size Disparity

**Demographic groups will be differently represented in our samples**

- Historical bias
- Minority groups
- ...

**What does this mean for model fit?**

- Models will work better for majorities (ex: dialects, speech recognition)
- Models will **generalize** based on majorities

**Effects on society**

- Minorities may adopt technology more slowly (increase the gap)
- The danger of feedback loops: (ex: predictive policing → more arrests → re-enforce model signal ...)



# Demographic Disparity / Sample Size Disparity

## Two questions

- Is any disparity **justified**?
- Is any disparity **harmful**?

## Case study

- Amazon same-day delivery by zip-code (USA)
- Areas with largely black population are left out
- Amazon's objective: min cost / max efficiency
- Is the system biased?
- Is discrimination happening?
- If so: is the discrimination justified? Is it harmful?



# Sensitive Attributes

## Notation:

---

- $X$  non-sensitive features
  - $A$  sensitive attributes with discrete labels (male/female, old/young, ...)
  - $Y$  true labels
  - $\hat{Y}$  classifier score (predicted label, for our purposes)
- 

Very often instances have a mix of useful, uncontroversial attributes, and *sensitive attributes* based on which we do not want to make classification decisions.

Different attributes lead to different demographic groups of the population

It is rarely clear which attributes are sensitive and which are not. Choice can have profound impacts.



## Approach

- Hide all sensitive features from the classifier. Only train on  $X$  and remove  $A$ .

$$P(\hat{Y}_n|X_n, A_n) \approx P(\hat{Y}_n|X_n)$$

## Another case study

- A bank which serves both **humans** and **martians** wants a classifier predicting whether an applicant should get a credit or not. Assume access to features (credit history, education, ...) for all applicants.
  - What are  $X$ ,  $A$ ,  $Y$  and  $\hat{Y}$ ?
  - Apply “fairness through unawareness”. Would the model be fair?

## Problem

- General features may be strongly correlated with sensitive features.  
Example..?

Consequently, this approach does **not generally result in a fair model**



## **Formal Fairness Criteria**

---

## Quick recap of metrics

	$\hat{y} = 1$	$\hat{y} = 0$
$y = 1$	true positive (TP)	false negative (FN)
$y = 0$	false positive (FP)	true negative (TN)

**Positive Predictive Value (PPV)** (also: precision)

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

**True Positive Rate (TPR)** (also: Recall)

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

**False Negative Rate (FNR):**

$$\frac{FN}{TP + FN} = 1 - TPR$$

**Accuracy**

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



## Example Problem: Credit scoring

- We trained a classifier to predict a binary credit score: should an applicant be granted a credit or not?
- Assume a version of the Adult data set as our training data (UCI Machine Learning Repository), which covers *humans* and *martians*. It includes actual credit scoring information

age	workclass	marital-status	race	class
39	State-gov	Never-married	White	$\leq 50K$
49	Self-emp-inc	Married-civ-spouse	White	$> 50K$
28	Private	Married-civ-spouse	Other	$\leq 50K$
35	Private	Divorced	White	$> 50K$
38	Private	Divorced	White	$\leq 50K$
53	Local-gov	Never-married	White	$\leq 50K$
28	Private	Married-civ-spouse	Black	$\leq 50K$
37	Private	Married-civ-spouse	Black	$> 50K$

- We consider *species* to be the protected attribute: our classifier should make *fair* decisions for both human and martian applicants.



**How can we measure fairness?**

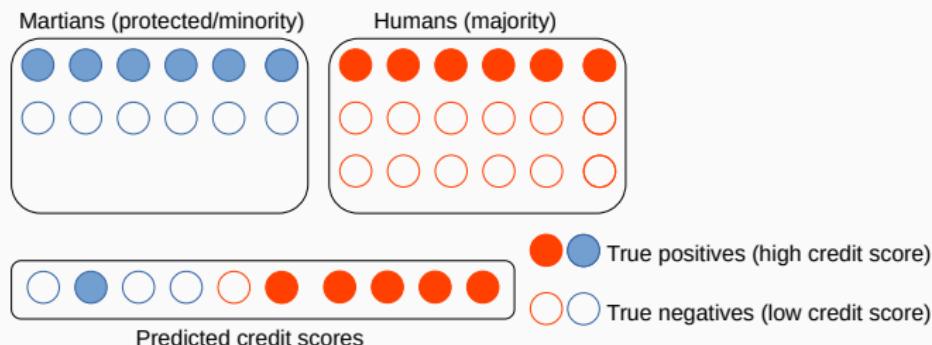
# Fairness Criteria I: Group Fairness (Demographic Parity)

The sensitive attribute shall be statistically independent of the prediction

$$\hat{Y} \perp A$$

For our classifier this would imply that:

$$P(\hat{Y} = 1 | A = m) = P(\hat{Y} = 1 | A = h)$$



## Fairness Criteria I: Group Fairness (Demographic Parity)

The sensitive attribute shall be statistically independent of the prediction

$$\hat{Y} \perp A$$

For our classifier this would imply that:

$$P(\hat{Y} = 1 | A = m) = P(\hat{Y} = 1 | A = h)$$

Goal: same chance to get a positive credit score for all applicants, regardless of their species.

- Simple and intuitive
- This is **independent** of the ground truth label  $Y$
- We can predict *good* instances for majority class, but *bad* instances for minority class.
- Danger to further harm reputation of minority class

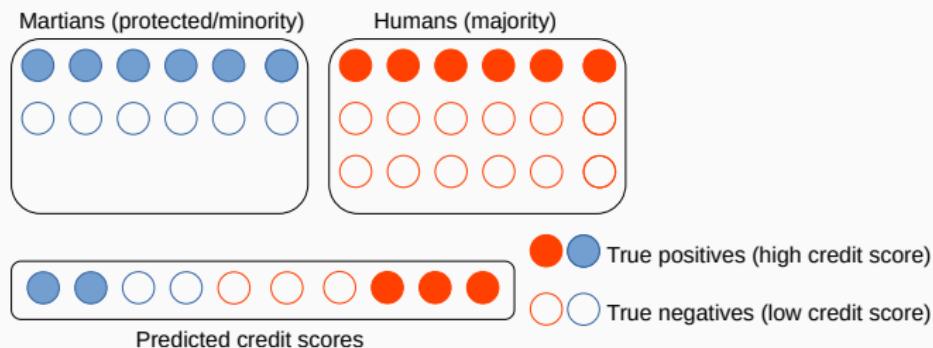


## Fairness Criteria II: Predictive Parity

All groups shall have the same **PPV** (precision): i.e., probability of a predicted positive to be truly positive.

For our classifier this would imply that:

$$P(Y = 1 | \hat{Y} = 1, A = m) = P(Y = 1 | \hat{Y} = 1, A = h)$$



## Fairness Criteria II: Predictive Parity

All groups shall have the same **PPV** (precision): i.e., probability of a predicted positive to be truly positive.

For our classifier this would imply that:

$$P(Y = 1 | \hat{Y} = 1, A = m) = P(Y = 1 | \hat{Y} = 1, A = h)$$

- The chance to **correctly** get a positive credit score should be the same for both human and martian applicants
- Now, we take the ground truth  $Y$  into account
- **Limitation:** Accept (and amplify) possible unfairness in the ground truth:  
If humans are more likely to have a good credit score in the data, then the classifier may predict good scores for humans with a higher probability than for martians in the first place.



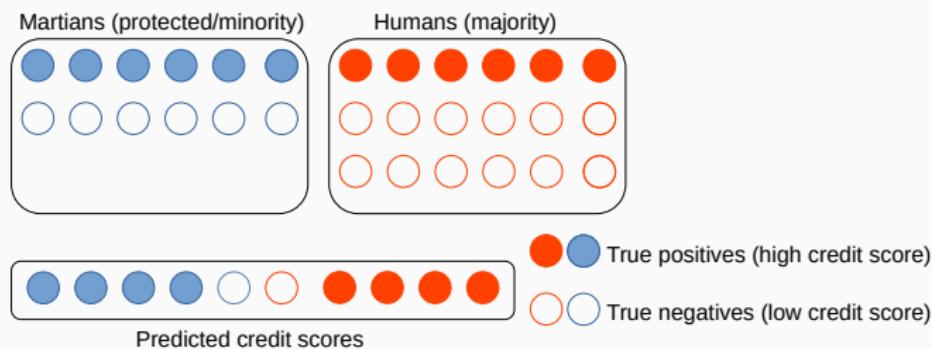
## Fairness Criteria III: Equal Opportunity

All groups shall have the same **FNR** (and TPR): i.e., probability of a truly positive instance to be predicted negative.

For our classifier this would imply that:

$$P(\hat{Y}=0|Y=1, A=m) = P(\hat{Y}=0|Y=1, A=h) \text{ or equivalently,}$$

$$P(\hat{Y}=1|Y=1, A=m) = P(\hat{Y}=1|Y=1, A=h)$$



## Fairness Criteria III: Equal Opportunity

All groups shall have the same **FNR** (and TPR): i.e., probability of a truly positive instance to be predicted negative.

For our classifier this would imply that:

$$P(\hat{Y}=0|Y=1, A=m) = P(\hat{Y}=0|Y=1, A=h) \text{ or equivalently,}$$
$$P(\hat{Y}=1|Y=1, A=m) = P(\hat{Y}=1|Y=1, A=h)$$

- Our classifier should make similar prediction for humans and martians with truly good credit scores
- We take the ground truth  $Y$  into account
- **Limitation:** Similar as for “Predictive Parity”, we accept (and amplify) possible unfairness in the ground truth.



## Fairness Criteria IV: Individual Fairness

Rather than balancing by group (human, martian), compare *individual* applicants directly.

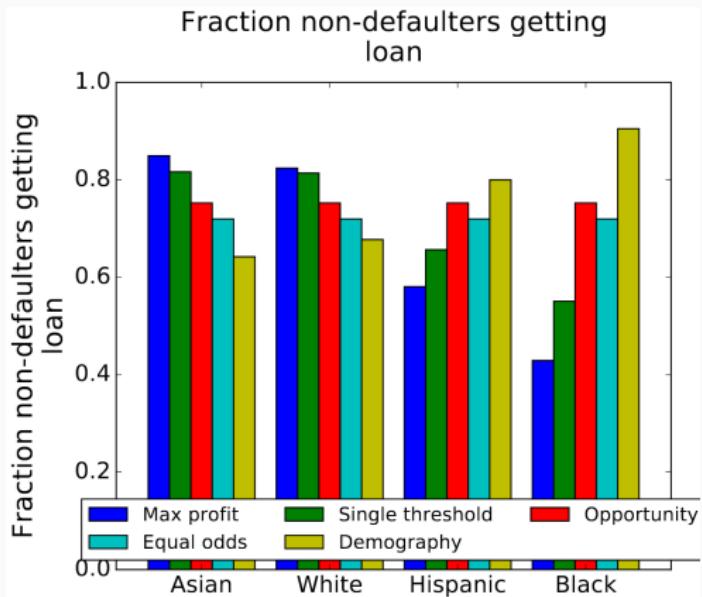
$$P(\hat{Y}_i=1|A_i, X_i) \approx P(\hat{Y}_j=1|A_j, X_j) \quad \text{if} \quad sim(X_i, X_j) > \theta$$



- Individuals which have **similar** features  $X$  (job, education, ...) should receive similar classifier scores
- Need to define a similarity function  $sim()$  (often non-trivial)
- Need to select a similarity threshold  $\theta$

# No Fair Free Lunch!

- Many more criteria exist. Many cannot be simultaneously satisfied. And many limit the maximum performance that is achievable.



Source: Hardt et al. (2016)

# No Fair Free Lunch!

- Many more criteria exist. Many cannot be simultaneously satisfied. And many limit the maximum performance that is achievable.
- **Long-term impacts:** “Group fairness” enforces equal rates of credit loans to males and females even though females are statistically less likely to return. This further disadvantages the already poor (as well as the bank).
- Fairness criteria as **soft constraints**, not hard rules
- Fairness criteria as **diagnostic tools** rather than constraints: analyzing classifiers through the lens of fairness criteria can highlight social impacts once the system is deployed
- All criteria we discussed are **observational**, i.e., correlations. They do not allow us to argue about **causality**.



## **Classifier Evaluation Revisited**

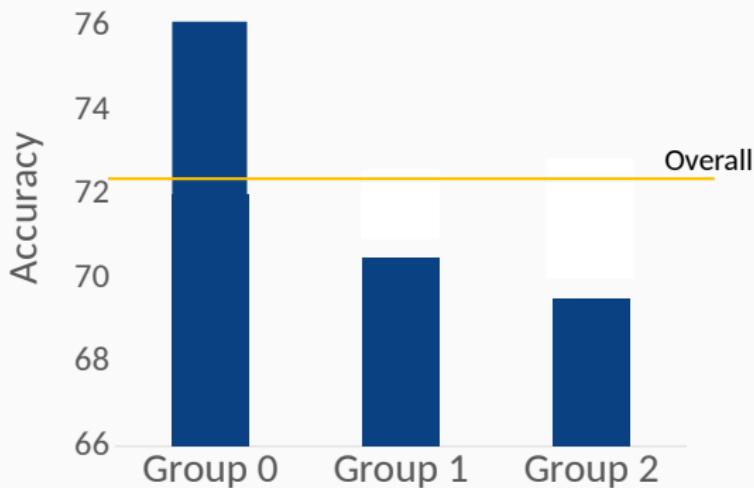
---

# Fairness Evaluation

**GAP measures:** Measure the deviation of performance any group  $\phi_g$  from the global average performance  $\phi$

- Average GAP:  $\text{GAP}_{avg} = \frac{1}{G} \sum_{g=1}^G |\phi_g - \phi|$
- Maximum GAP:  $\text{GAP}_{max} = \max_{g \in G} |\phi_g - \phi|$

## Accuracy GAP

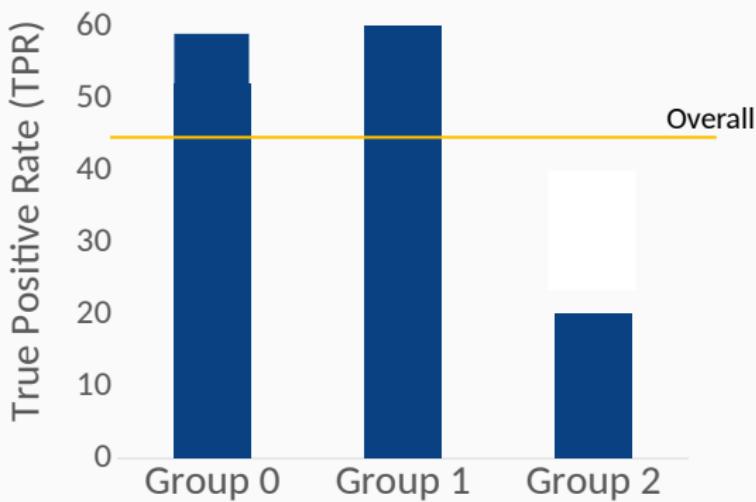


# Fairness Evaluation

**GAP measures:** Measure the deviation of performance any group  $\phi_g$  from the global average performance  $\phi$

- Average GAP:  $\text{GAP}_{avg} = \frac{1}{G} \sum_{g=1}^G |\phi_g - \phi|$
- Maximum GAP:  $\text{GAP}_{max} = \max_{g \in G} |\phi_g - \phi|$

**True positive rate (TPR) GAP:** Equal opportunity

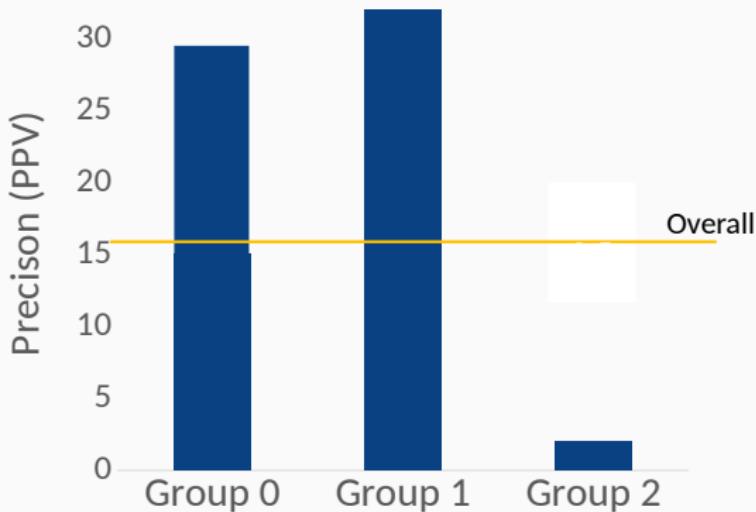


## Fairness Evaluation

**GAP measures:** Measure the deviation of performance any group  $\phi_g$  from the global average performance  $\phi$

- Average GAP:  $\text{GAP}_{avg} = \frac{1}{G} \sum_{g=1}^G |\phi_g - \phi|$
- Maximum GAP:  $\text{GAP}_{max} = \max_{g \in G} |\phi_g - \phi|$

**Positive Predictive Value (PPV) GAP:** Predictive Parity



## **Creating Fairer Classifiers**

---

## Now we know

- Where bias can arise (data, model, ...)
- How we can statistically define fairness in classification
- How we can diagnose (un)fairness in evaluation
- **What can we do – practically – to achieve better fairness?**

## We can improve fairness in

1. Pre-processing
2. Training / Optimization
3. Post-processing



# 1. Pre-processing

## Balancing the data set

- Up-sample the minority group (*martians*)
- Down-sample the majority group (*humans*)



# 1. Pre-processing

## Re-weighting data instances

Expected distribution (if  $A \perp Y$ )

$$P_{exp}(A=a, Y=1) = P(A=a) \times P(Y=1) = \frac{\#(A=a)}{|D|} \times \frac{\#(Y=1)}{|D|}$$

Observed distribution

$$P_{obs}(A=a, Y=1) = \frac{\#(Y=1, A=a)}{|D|}$$

Weigh each instance by

$$W(X_i=\{x_i, a_i, y_i\}) = \frac{P_{exp}(A=a_i, Y=y_i)}{P_{obs}(A=a_i, Y=y_i)}$$



## 2. Model training / optimization

Add constraints to the optimization function.

$$\begin{array}{ll} \text{minimize} & \mathcal{L}(f(X, \theta), Y) \quad \text{the overall loss} \\ \text{subject to} & \forall g \in G \underbrace{|\phi_g - \phi|}_{\psi_g} < \alpha \quad \text{fairness constraints (e.g., GAP)} \end{array}$$

Incorporate using a Lagrangian (cf. Lecture 2: constrained optimization!)

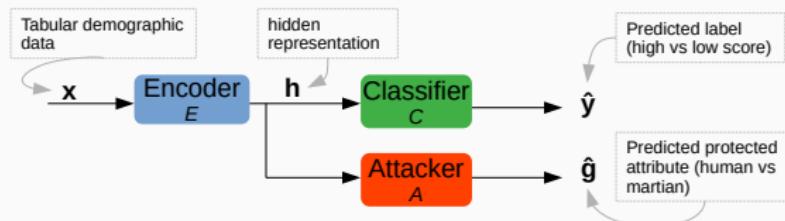
$$\mathcal{L}_{final}(\theta) = \mathcal{L}(f(X, \theta), Y) + \sum_{g=1}^G \lambda_g \psi_g$$



## 2. Model training / optimization

### Adversarial Training (taster)

- Learn a classifier that predicts credit scores while being agnostic to the *species* of the applicant.



- $E$  maps input to latent representation  $h$
- $C$  uses  $h$  to predict target label:  $h$  should be **good at** predicting  $\hat{y}$ .
- $A$  uses  $h$  to predict protected attribute:  $h$  should be **bad at** predicting  $\hat{g}$ .

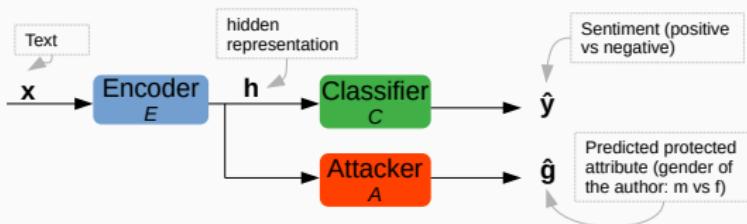
$$\mathcal{L} = \underbrace{\mathcal{L}_C(\hat{y}_i, y_i)}_{\text{minimize loss}} + \underbrace{\mathcal{L}_A(\hat{g}_i, g)}_{\text{maximize loss}}$$



## 2. Model training / optimization

### Adversarial Training (taster)

- Learn a classifier that predicts the sentiment of a movie review (positive, negative) while being agnostic to the gender of the author.



### 3. Post-processing

#### Modify the classifier predictions (scores $s$ or labels $\hat{y}$ )

- E.g., decide on individual thresholds per group, such that:

$$\hat{y}_i = 1 \text{ if } s_i > \theta_i$$

- Come up with a special strategy for “difficult” instances, i.e., instances where  $P(\hat{y}_i) \approx 0.5$

#### Pros

- Model-independent
- Even works with proprietary / black-box models

#### Cons

- Needs access to protected attribute at test time



## Some (optional, but excellent) talks

**Predictive Policing:** The danger of predictive algorithms in criminal justice

<https://www.youtube.com/watch?v=p-82YeUPQh0>

**Impacts of Machine Learning:** Humans Need Not Apply

<https://www.youtube.com/watch?v=7Pq-S557XQU&feature=youtu.be>

**Tutorial on Fairness in ML** (Far beyond the scope of this subject)

<https://fairmlbook.org/tutorial1.html>

**Netflix Documentary:** Coded bias

<https://www.netflix.com/au/title/81328723>



# Summary

## Today... Fair machine learning

- What is Bias and where does it come from?
- Algorithmic Fairness
- How to make ML algorithms fairer

## Next up

- Guest lecture by Dr. Marc Cheong (ML Ethicist)

*“Social Media Sentiment Matters: data science, social data & their ethics”*



## **References**

---

Harini Suresh and John V. Guttag (2019). *A Framework for Understanding Unintended Consequences of Machine Learning*. arXiv preprint arXiv:1901.10002/

Gordon, Jonathan, and Benjamin Van Durme. "Reporting bias and knowledge acquisition." Proceedings of the 2013 workshop on Automated knowledge base construction. 2013.

Verma, Sahil, and Julia Rubin. "Fairness definitions explained." 2018 ieee/acm international workshop on software fairness (fairware). IEEE, 2018.

Hardt, Moritz, Eric Price, and Nathan Srebro. "Equality of opportunity in supervised learning." arXiv preprint arXiv:1610.02413 (2016).

Solon Barocas and Moritz Hardt and Arvind Narayanan (2019). "Fairness and Machine Learning". <http://www.fairmlbook.org>

