



# Privacy and Data Collection Notice

This lecture (and all future lectures in this subject) will be recorded. If you choose to turn on your audio or video functions to participate in the class, your audio and/or video image will be recorded.

The recording will be made available to all students in the subject via LMS Lecture Capture until the end of semester.

If you prefer not to be recorded, you can choose **not** to turn on your audio or video function. It's easier for me to teach to faces than to black boxes, but this is a choice that you have. If you wish to ask a question without turning on audio or video, you can ask it in chat.

I do ask that you turn on your audio and video in breakout rooms, which are not recorded. If you do not, you won't be able to enter discussions with your fellow students.

Further information about how the University handles student information can be found in the [Student Privacy Statement](#).

In accordance with the University's [Student Conduct Policy](#) (Section 4.2), **students may not take photographs, video or audio recordings** of lectures, tutorials, rehearsals, performances or practical classes without the express permission of the staff member supervising the activity (or the subject coordinator) and the written permission of any identifiable individuals, or their legal guardians.

# Lecture 1: Introduction and Overview

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



## This lecture

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



## **Intros & Warm-up**

---

# Introductions

## About me

- Lecturer in CIS since 2019
- Research in natural language processing
- PhD from Edinburgh University
- 1.5 years research in industry (Amazon)



# Introductions

## About me

- Lecturer in CIS since 2019
- Research in natural language processing
- PhD from Edinburgh University
- 1.5 years research in industry (Amazon)

## About you

Please go to: [pollev.com/iml2021](http://pollev.com/iml2021)



## 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 & Lecturer	Lea Frermann	lea.frermann@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 subject 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**
- We'll decide together as we go along
- **Live** workshops throughout the semester



## Lectures

---

Lecture 1                    Wed 17:15-18:15

Online; Zoom

---

Lecture 2                    Fri 11:00-12:00

Online; Zoom

---

## Lecture content

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



THE UNIVERSITY OF  
MELBOURNE

## Workshops

- **start from week 2**
- 1 hour per week
- ~ 14 slots, please sign up and stick to one
- Online; live via zoom

## Workshop Content

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



## Coding drop-in sessions

---

Session 1    Wed 12–1 (link via Canvas Zoom)

Session 2    Fri 3:15–4:15 (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)
- Regular 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:** bias and fairness in machine learning



# 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

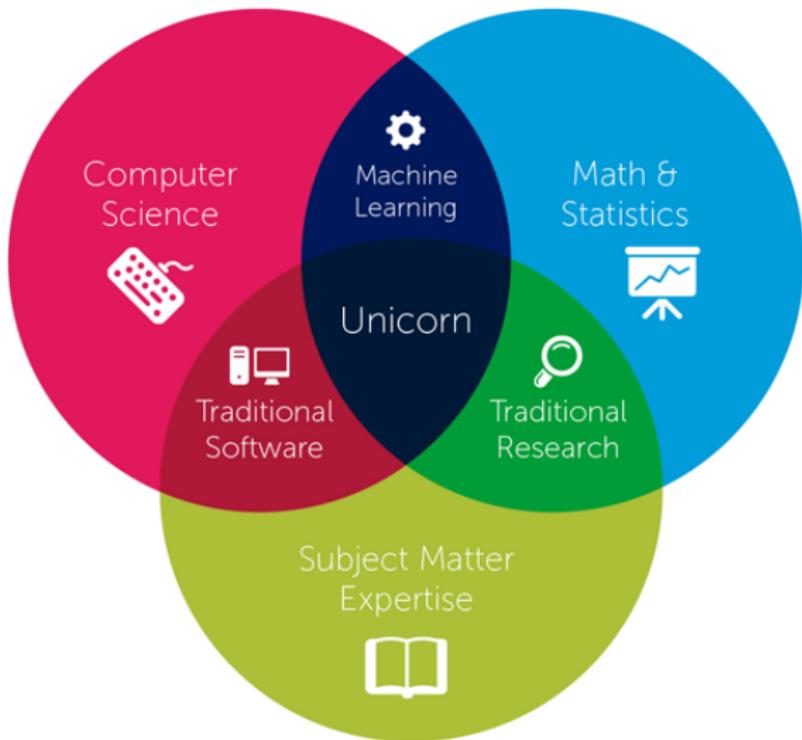
CIS Academic Honesty Training		Complete All Items
<b>Videos</b>		
<input checked="" type="checkbox"/>	Getting help from non-student friends	<input type="radio"/>
	Mark as done	
<input checked="" type="checkbox"/>	Copying the answer from a fellow student	<input type="radio"/>
	Mark as done	
<input checked="" type="checkbox"/>	Getting help from fellow students	<input type="radio"/>
	Mark as done	
<input checked="" type="checkbox"/>	Copying the answer from online sources	
<input 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	
<b>Quiz</b>		
<b>Further information</b>		
<input checked="" type="checkbox"/>	Academic Integrity Principles at Unimelb	
<input checked="" type="checkbox"/>	Further Resources	



## **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.

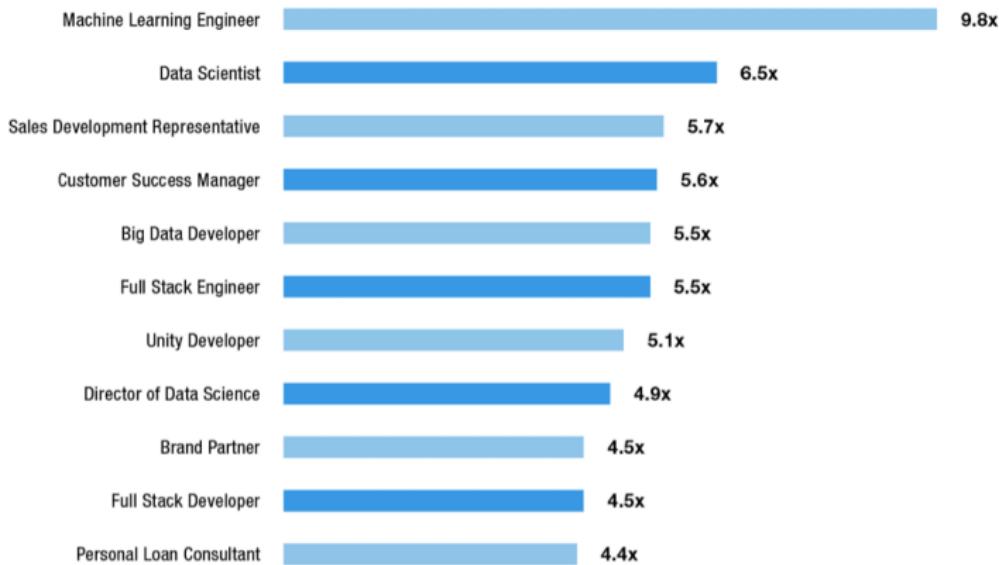
# What is Machine Learning?



<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

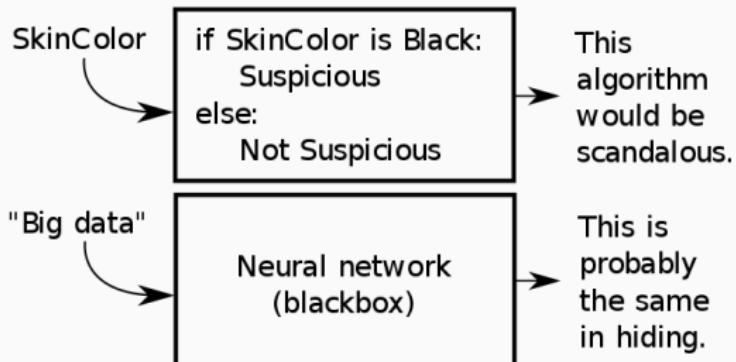


# More Applications

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



# Machine Learning, Ethics, and Transparency



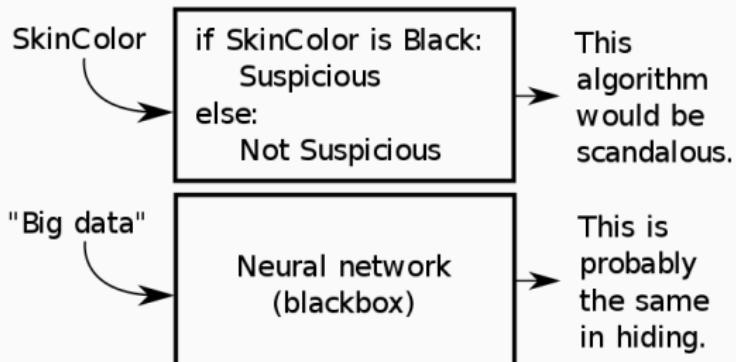
[commons.wikimedia.org/wiki/File:Pseudo-algorithm\\_comparison\\_for\\_my\\_slides\\_on\\_machine\\_learning\\_ethics.svg](https://commons.wikimedia.org/wiki/File:Pseudo-algorithm_comparison_for_my_slides_on_machine_learning_ethics.svg)

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.



# Machine Learning gone wrong...



World Business Markets Breakingviews Video More

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



# 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...



MIT Technology Review

Topics Magazine Newsletters



FRANCISCA BARREIRA

Artificial intelligence

## Predictive policing algorithms are racist. They need to be dismantled.

Lack of transparency and biased training data mean these tools are not fit for purpose. If we can't fix them, we should ditch them.

by **Will Douglas Heaven**

July 17, 2020

**Yeshimabett Milner was in high school the first time she saw kids she knew** getting handcuffed and stuffed into police cars. It was February 29, 2008, and the principal of a nearby school in Miami, with a majority Haitian and African-American population, had put one of his students in a chokehold. The next day several dozen kids staged a peaceful demonstration. It didn't go well.

That night, Miami's NBC 6 News at Six kicked off with a segment called "Chaos on Campus." (There's a [clip on YouTube](#).) "Tensions run high at Edison Senior High after a fight for rights ends in a battle with the law," the



THE UNIVERSITY OF  
MELBOURNE

## 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:](https://www.reuters.com/article/us-microsoft-twitter-bot-idUSKCN0WQ2LA)

//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 racist.

[https:](https://www.reuters.com/article/us-microsoft-twitter-bot-idUSKCN0WQ2LA)

//www.reuters.com/article/us-microsoft-twitter-bot-idUSKCN0WQ2LA



# Lecture 2: Machine Learning Concepts

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



# 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**



## 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
<b>INSTANCE<sub>1</sub></b>	sunny	hot	high	FALSE	no
<b>INSTANCE<sub>2</sub></b>	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	7	7

### The MNIST digit classification data set

- How many **instances** do you see in the dataset above?
- What are these instances?
- What **features** or **attribute** does each instance have?
- What could these features be?



# 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

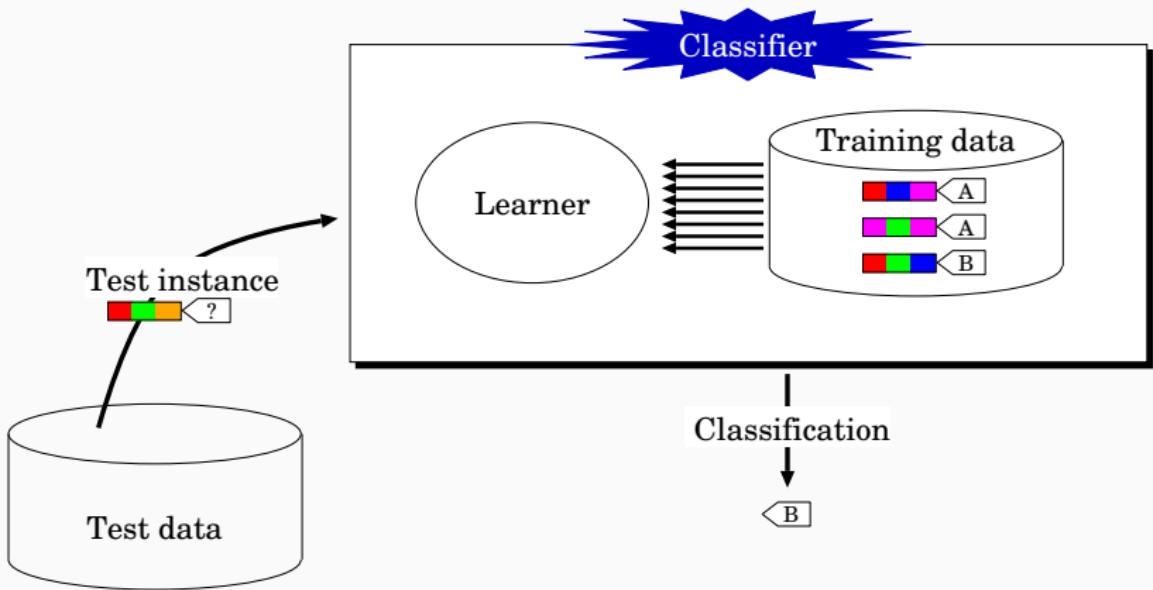
- **Supervised** methods have prior knowledge of a closed set of classes and set out to discover and categorise new instances according to those classes
- **Unsupervised** do not have access to an inventory of classes, and instead discover groups of ‘similar’ examples in a given dataset. Two flavors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

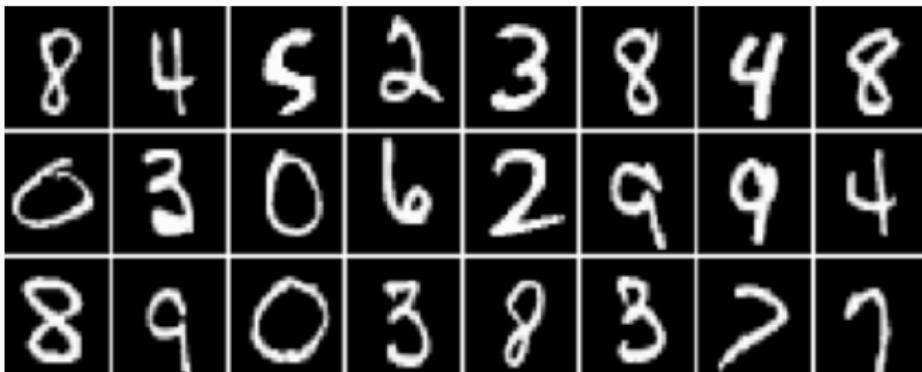


## Example Predictions for weather

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



### 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)

vspace1ex



# 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
- Frequently indicated by out-of-range entries
  - Types: unknown, unrecorded, irrelevant
  - Reasons:
    - malfunctioning equipment
    - changes in experimental design
    - collation of different datasets
    - measurement not possible
- Missing value may have significance in itself (e.g. missing test in a medical examination)
- Most schemes assume that is not the case → 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 1, 2021

Lea Frermann, 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.



## 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 10th March at 7pm!
- Questions: Assignment\_1 discussion board (don't share solutions!)



## **Introduction**

---

## K-Nearest Neighbors: Example

Your 'photographic memory' of all handwritten digits you've every 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



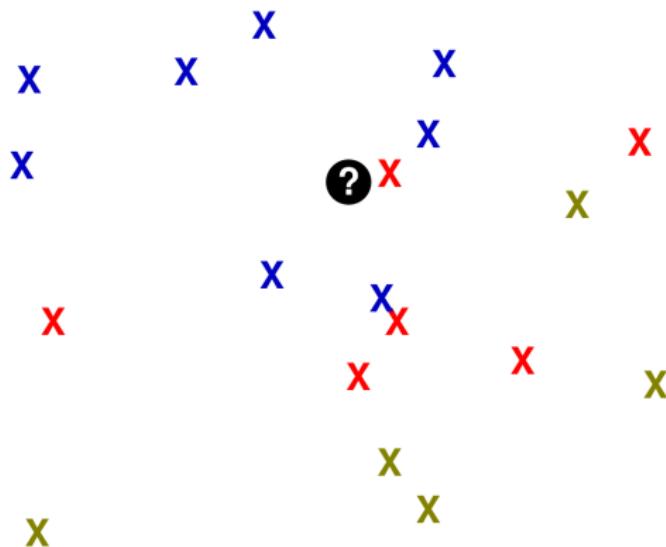
## K-Nearest Neighbors: Example

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

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



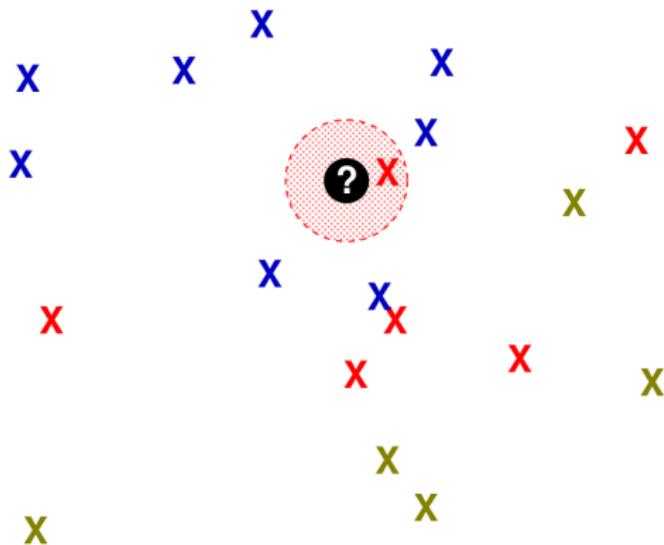
# K-Nearest Neighbors: Visualization



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



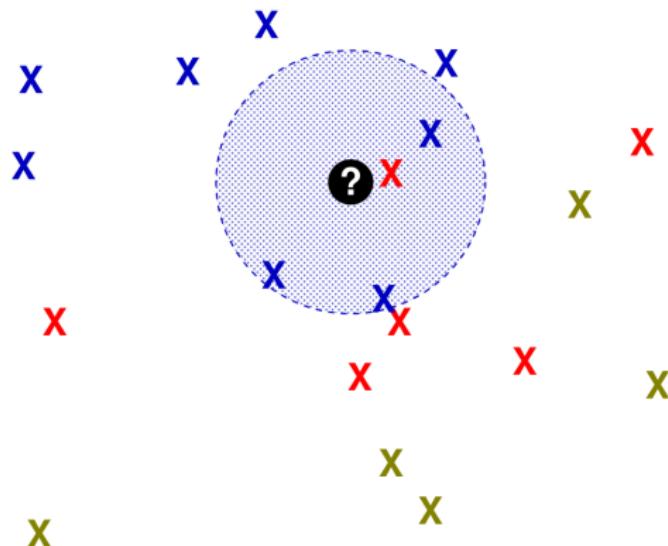
# K-Nearest Neighbors: Visualization



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.

$$\begin{aligned}d &= 1 - J \\&= 1 - \frac{|A \cap B|}{|A \cup B|} \\&= 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}\end{aligned}$$



## 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.

$$\begin{aligned}d &= 1 - J \\&= 1 - \frac{|A \cap B|}{|A \cup B|} \\&= 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}\end{aligned}$$

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



# 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

$$\begin{aligned} d(a, b) &= |2.0 - 1.0| + |1.4 - 2.4| + |4.6 - 6.6| + |5.5 - 2.5| \\ &= 2 + 1 + 2 + 3 \\ &= 8 \end{aligned}$$



# 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

$$\begin{aligned} d(a, b) &= \sqrt{(2.0 - 1.0)^2 + (1.4 - 2.4)^2 + (4.6 - 6.6)^2 + (5.5 - 2.5)^2} \\ &= \sqrt{1 + 4 + 4 + 9} = \sqrt{18} \\ &= 4.24 \end{aligned}$$



# 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

$$\cos(\text{doc1}, \text{doc2}) = \frac{200 \times 300 + 300 \times 300 + 200 \times 100}{\sqrt{200^2 + 300^2 + 200^2} \sqrt{300^2 + 200^2 + 100^2}} = 0.91$$

$$d(\text{doc1}, \text{doc3}) = 0.09$$

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

$$d(\text{doc1}, \text{doc2}) = 0.01$$



# Comparing Ordinal Feature Vectors

## Normalized Ranks

- sort values, and return a rank  $r \in \{1 \dots m\}$
- 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  $r \in \{1 \dots m\}$
- 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



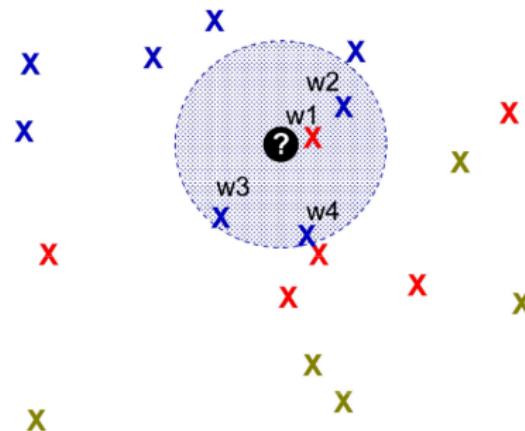
## Four problems

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)



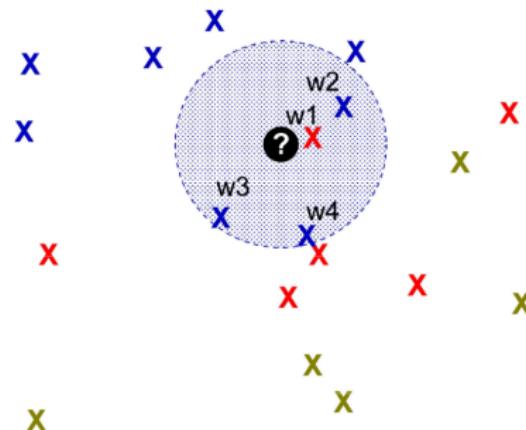
Voting Example (k=4)

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



# Majority Voting

Equal weights (=majority vote)



## Voting Example ( $k=4$ )

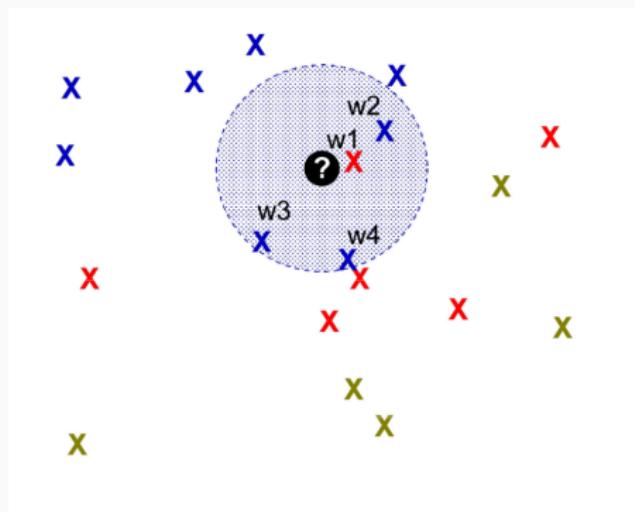
- $w_1 = w_2 = w_3 = w_4 = 1$
- red: 1
- blue:  $1+1+1=3$

# 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)

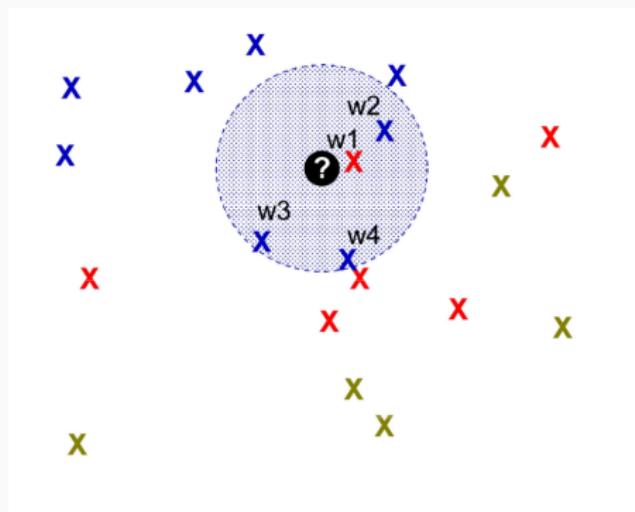
- $d_1=0$ ;  $d_2=1$ ;  $d_3=d_4=1.5$
- $\epsilon = 1e - 5$

# 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

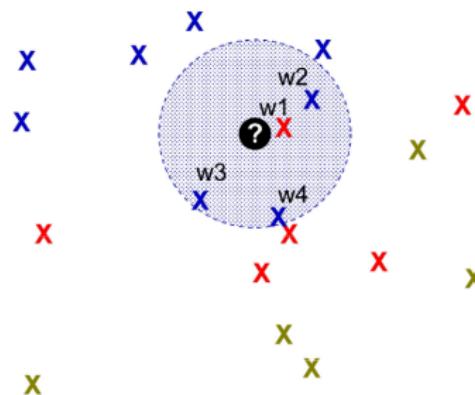
Inverse Linear distance

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

$d_1$  = 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

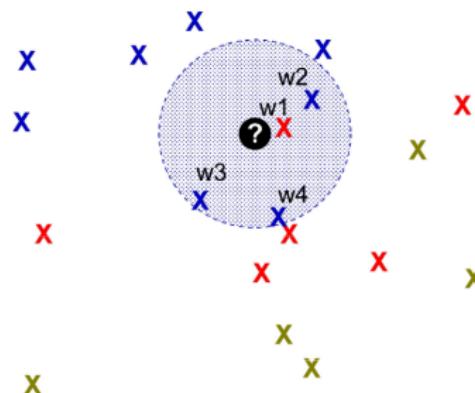
Inverse Linear distance

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

$d_1$  = 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

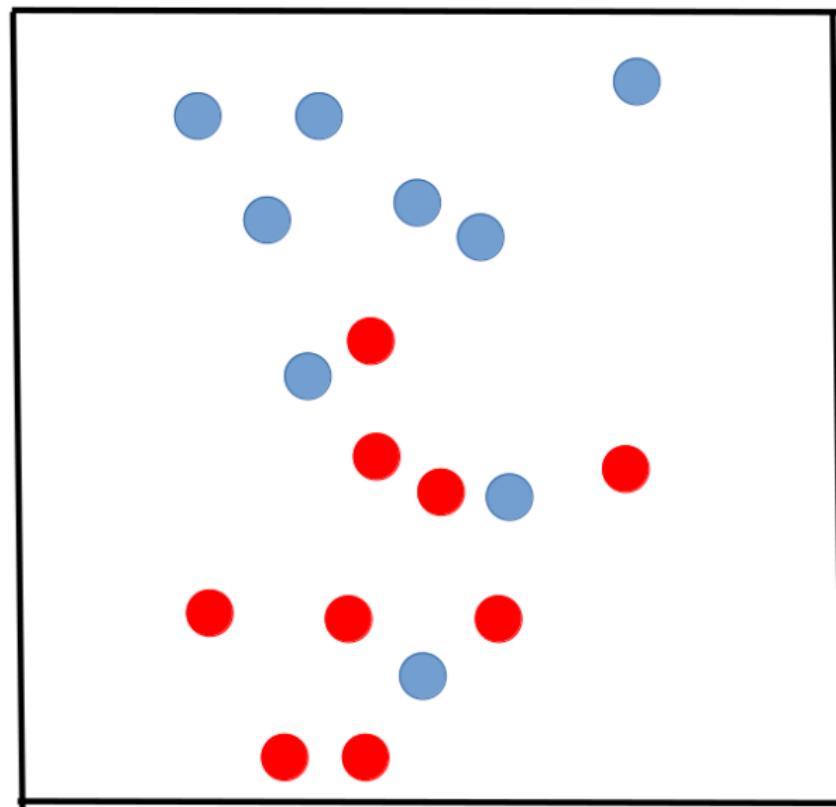
## Four problems

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  $K$ ?**



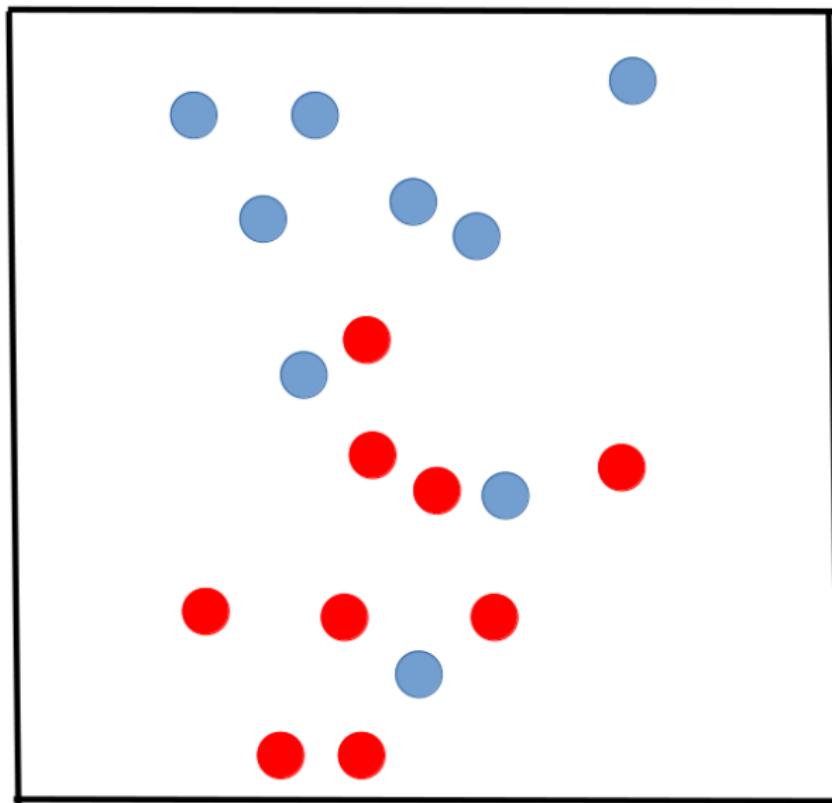
## Selecting the value of $K$

$K=1$



## 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/iml2021](http://pollev.com/iml2021)



# Why K-NN?

## Pros

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

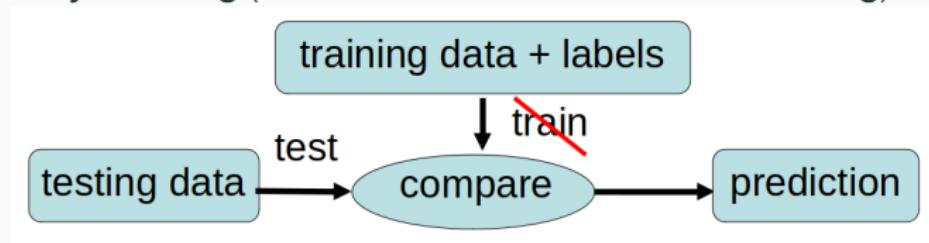
## Cons

- 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)

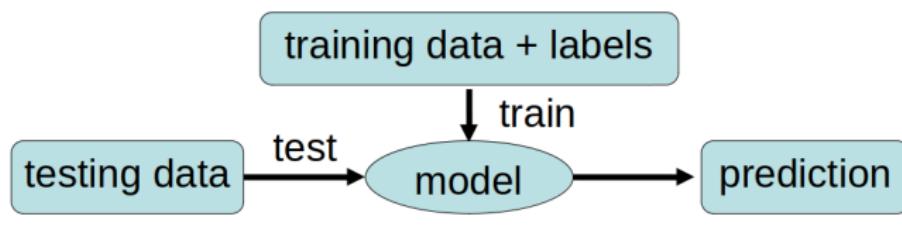


- **store** the training data
- **fixed** distance function
- **fixed** prediction rule (majority, weighting, ...)
- **compare** test instances with stored instances
- **no learning**



# 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 1, 2021

Lea Frermann, 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.



## 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

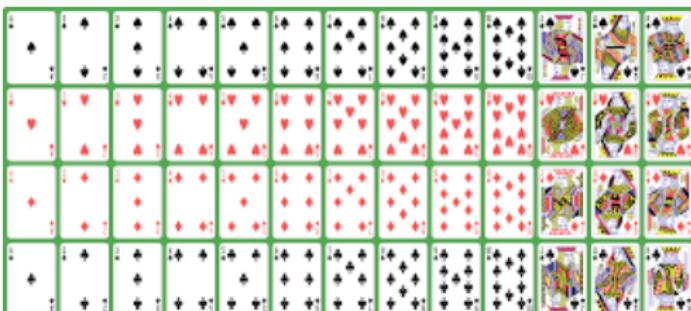
$$0 \leq P(A) \leq 1$$

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

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

**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  $P(A=a)$ .

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

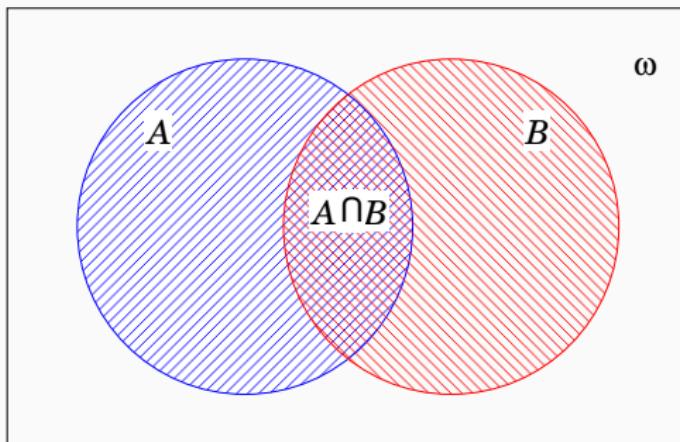
$$P(A = \text{red}) = \frac{1}{2}$$

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



# Basics of Probability Theory

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



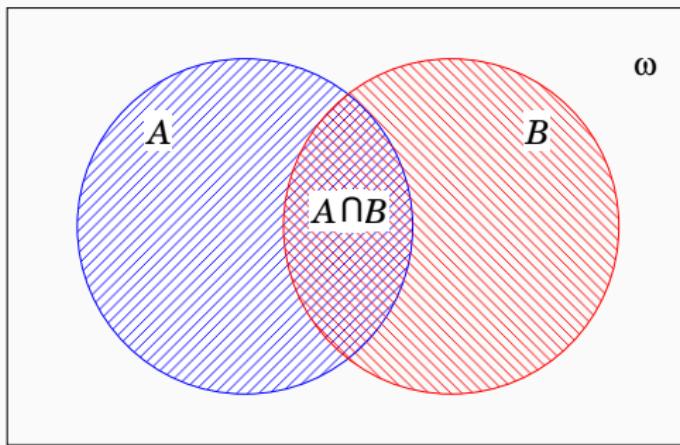
$$P(A = \text{ace}, B = \text{heart}) = ?$$

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



# Basics of Probability Theory

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

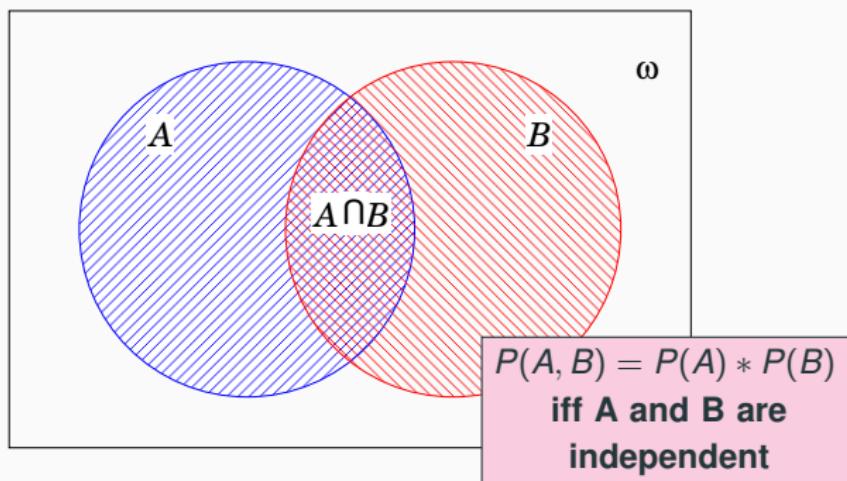


$$P(A = \text{ace}, B = \text{heart}) = \frac{1}{52}$$

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

# Basics of Probability Theory

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



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

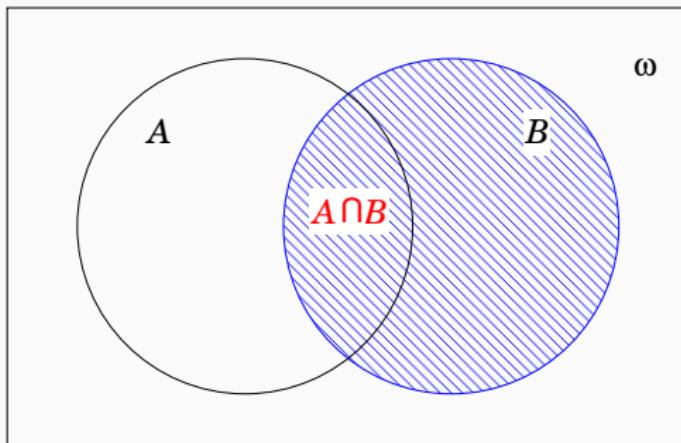
$$P(A = \text{heart}, B = \text{red}) = \frac{1}{4}$$



# Conditional Probability

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

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



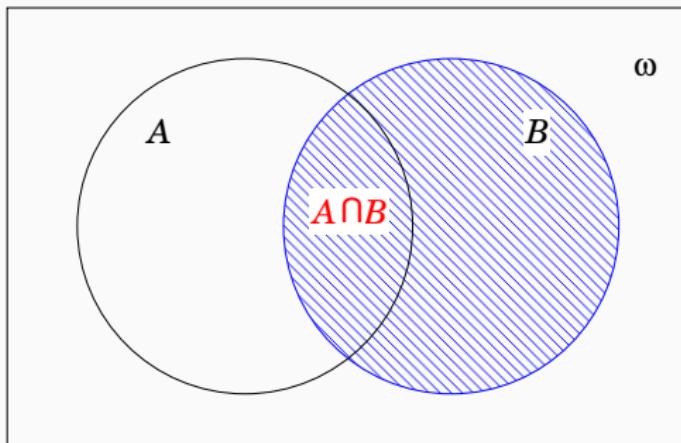
$$P(A = \text{ace} | B = \text{heart}) = ?$$

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

# Conditional Probability

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

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



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

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

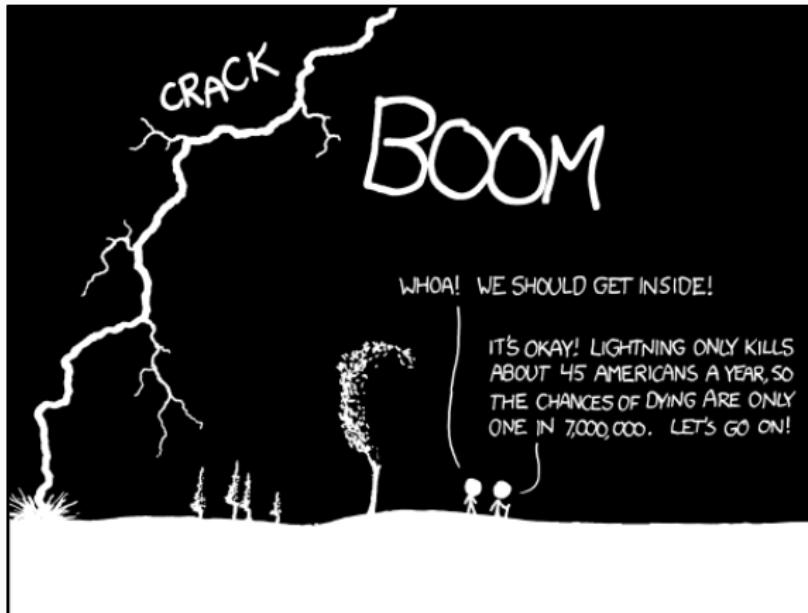


# Notation

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:  $A$ =draw an ace;  $B$ =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:  $A=$  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 ( $\langle H, T, H, H, \dots, T \rangle$ )*



# 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/iml2021>

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 permits: 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 permits: 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 permits: 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?

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



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.



# **Lecture 5: Introduction to Optimization**

---

**COMP90049**  
**Introduction to Machine Learning**  
Semester 1, 2021



# Roadmap

## Last time... Probability

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



# Roadmap

## 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**



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**

**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.

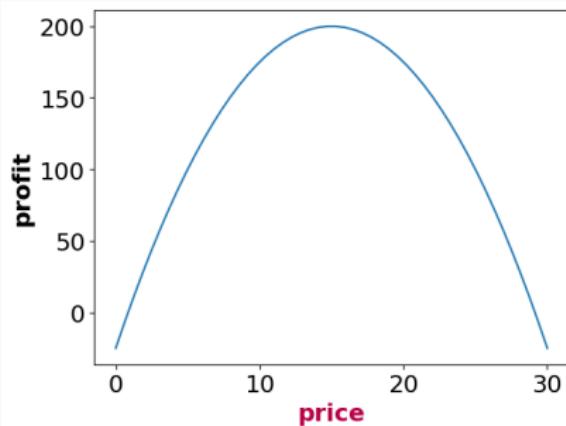


# Finding Optimal Points I

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**

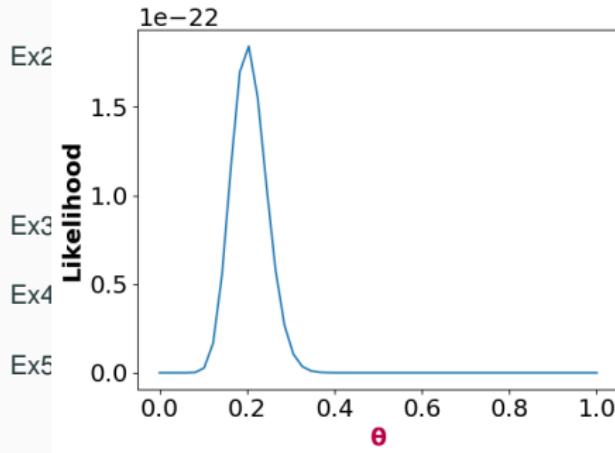
Ex5: Find the **parameters  $\theta$**  of a spam classifier which lead to the **highest data log likelihood**



# 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 profit of our shoe shop.

Ex3 combination of house age and house location

Ex4 which lead to the **lowest error**

Ex5 which lead to the **highest data log likelihood**



# 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**)

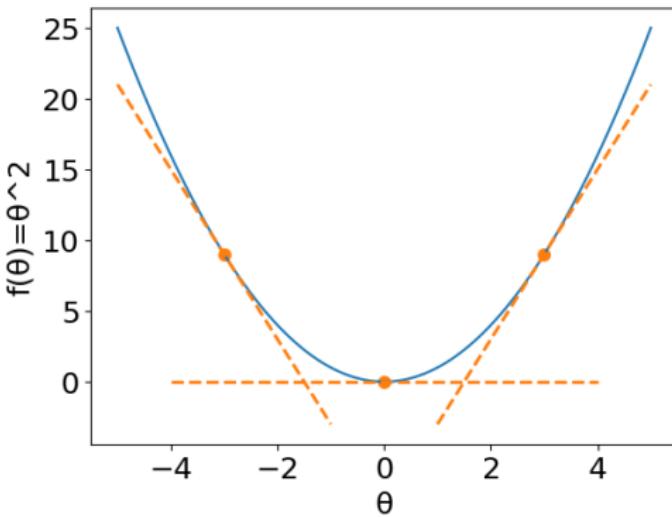
E.g., the **smallest** possible classification error

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



# 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  $\frac{\partial f}{\partial \theta}$

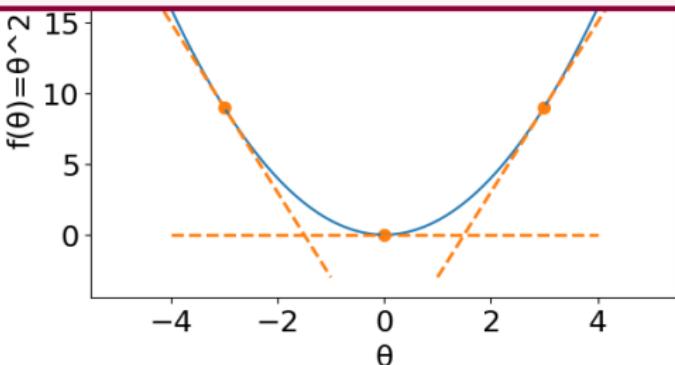


# Finding extreme points of a function

- 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

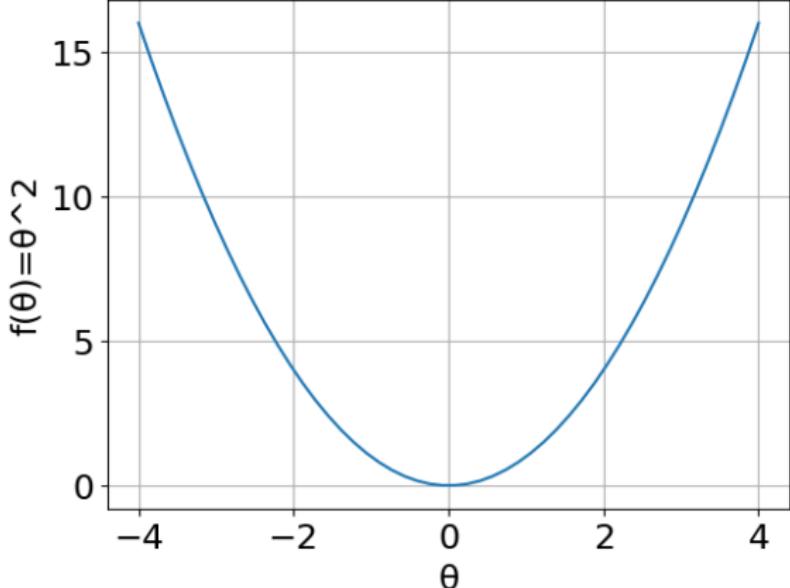
### Example

- For our function, with a parabola:

Take the derivative

We want to find the point

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

## 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)

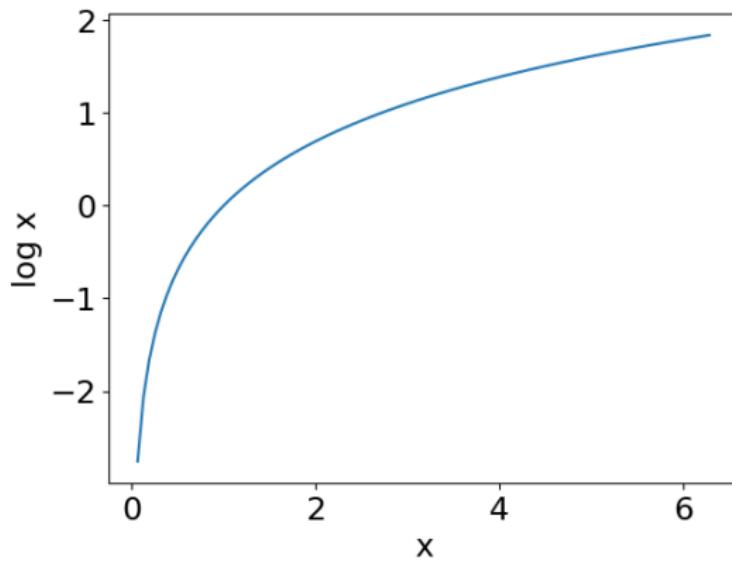
$$\log \mathcal{L}(\theta) = x \log \theta + (N - x) \log(1 - \theta)$$



# 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 our 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

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

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 1, 2021

Lea Frermann, 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.



# 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 3)
- Finding the optimal parameters (drawing on foundations in lecture 4)
- 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**  $\hat{y} = \text{argmax}_y p(y|x)$



# Maximum Likelihood Estimation for Categorical Naive Bayes

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!



# Maximum Likelihood Estimation for Categorical Naive Bayes

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?

$P(\{$

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  $x$

$$P(Flu) = 3/5$$

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

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

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

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

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

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

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

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

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

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

$$P(Cold) = 2/5$$

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

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

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

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

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

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

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

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

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

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



## 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:

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

Flu:

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



## 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



## 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:

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

- 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 1, 2021

Lea Frermann, CIS

(parts adapted from slides by Karin Verspoor and Tim Baldwin)

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.



## 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

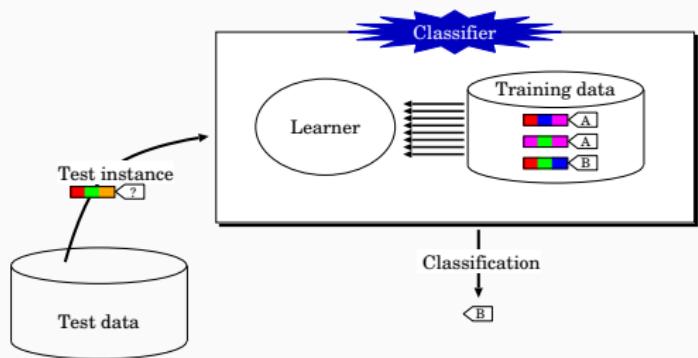


## **Classification Evaluation**

---

# 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

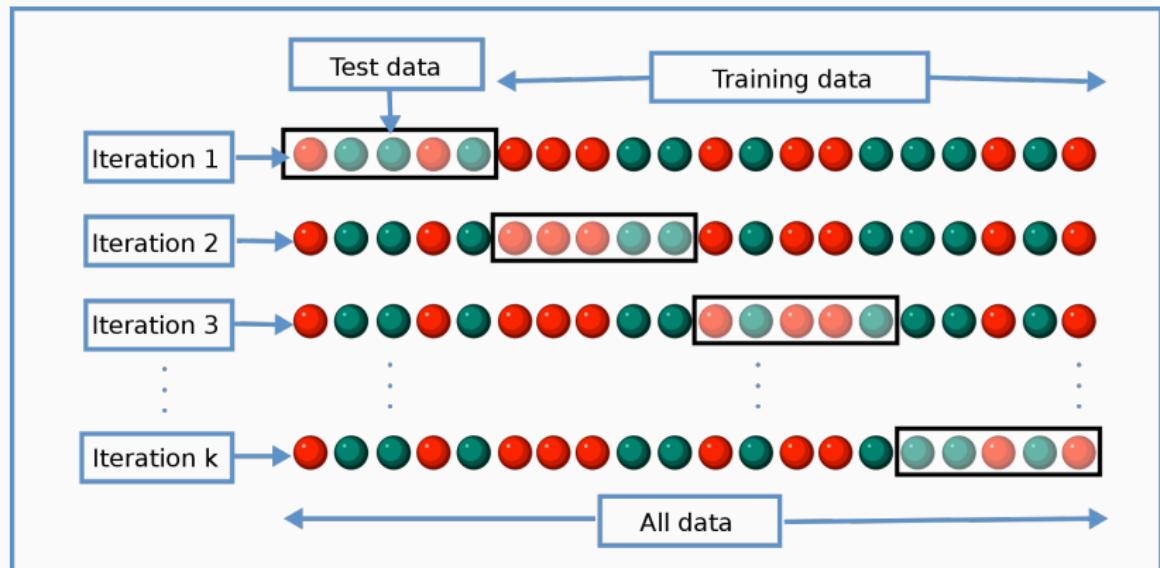
Usually preferred alternative: **Cross–Validation**

- 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"*

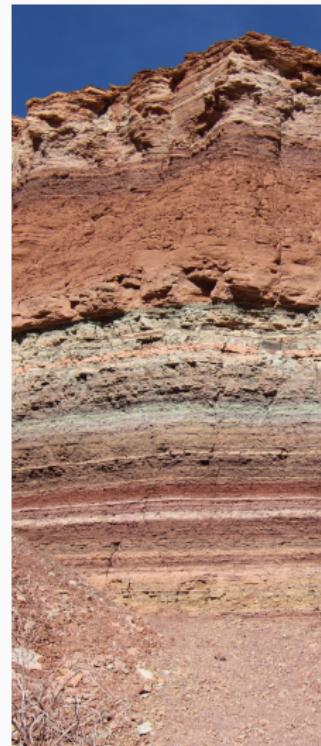
[Mitchell, 1997, Ch. 2.7.3]

- **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**:

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

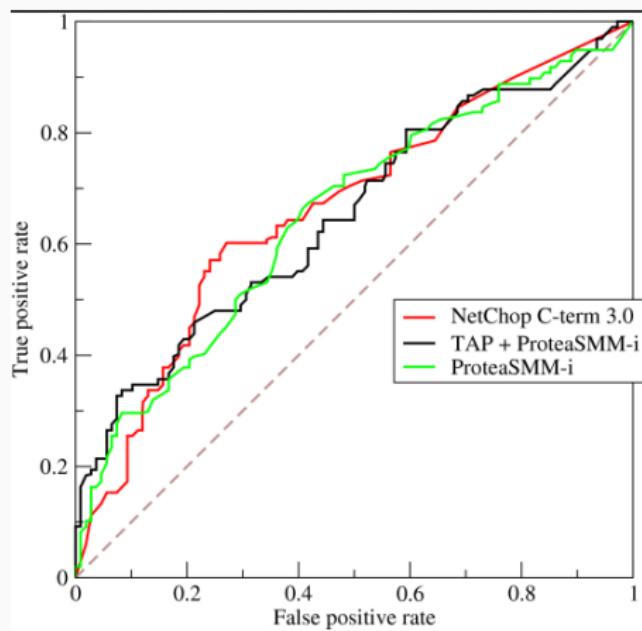
$$F_1 = \frac{2PR}{P + R}$$



# 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, probability of detection $= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	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, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$
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>U1</i>	<i>U2</i>	...
<i>Actual</i>	<i>I</i>	TP	FN	FN	...
	<i>U1</i>	FP	TN	TN	...
	<i>U2</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

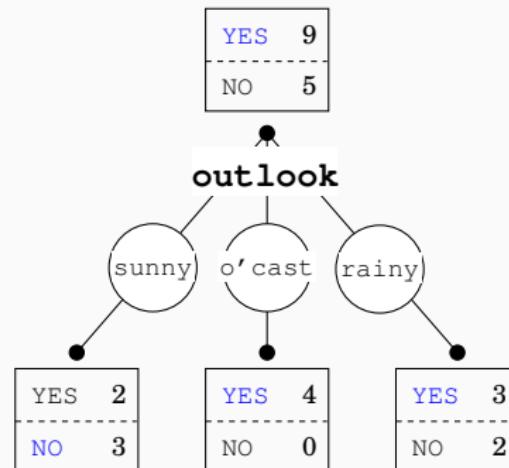


## Weather dataset

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

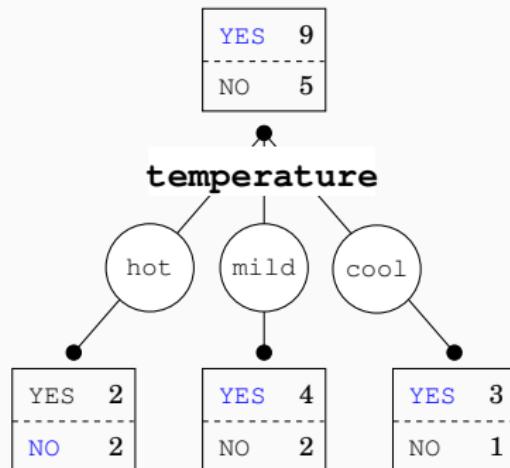


## Attribute Option 1 (outlook)



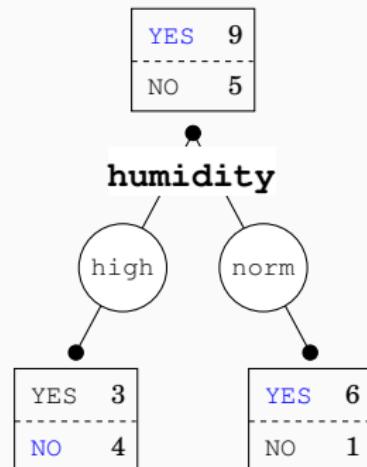
Total errors =  $\frac{4}{14}$

## Attribute Option 2 (temperature)



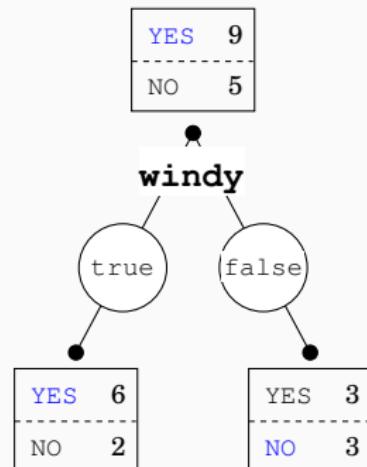
Total errors =  $\frac{5}{14}$

## 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

- Optimization (part 2)
- Logistic Regression



## 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, 1:67–82, 1997.

Jacob Eisenstein. Natural Language Processing. Chapter 4.4



# Lecture 8: Feature Selection and Analysis

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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

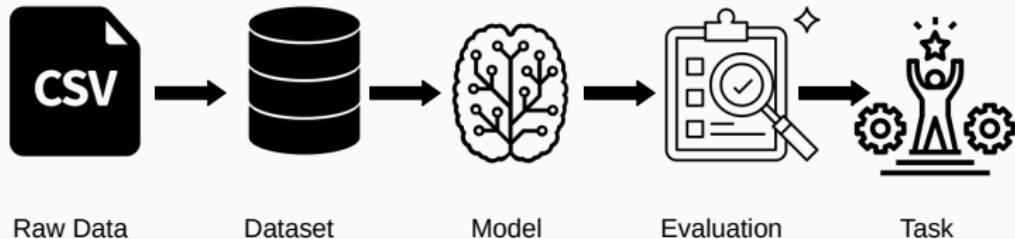


## **Features in Machine Learning**

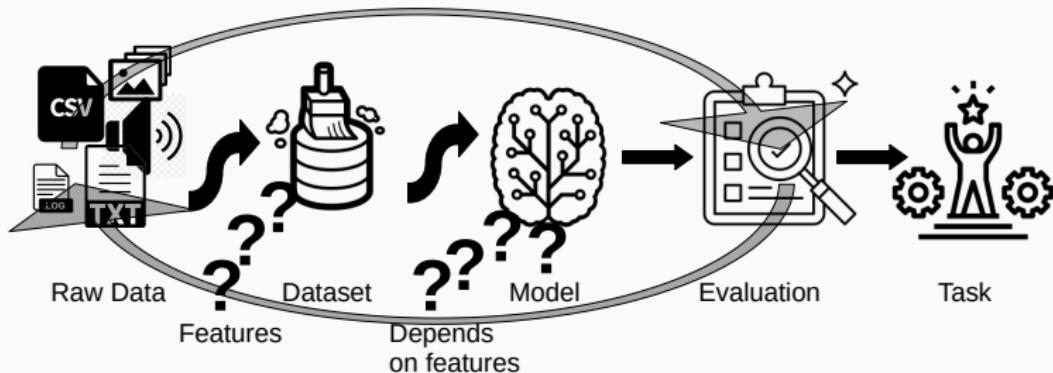
---

# 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



# 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  $LHS \sim 0$ , attribute and class occur together as often as we would expect from random chance
- If  $LHS << 0$ , 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)

0	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	http://bitlyczmk	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
angeles	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;

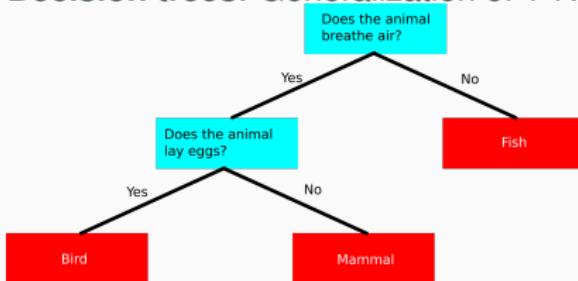
$|D|$ =number of documents in  $D$



# 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 next four lectures will be pre-recorded, as sequences of shorter videos. There will be no meeting next Wednesday.



## 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 (part 2): Logistic Regression

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



# 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



THE UNIVERSITY OF  
MELBOURNE

## 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

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:

$$\frac{1/6}{1 - (1/6)} = \frac{0.17}{0.83} = 0.2$$



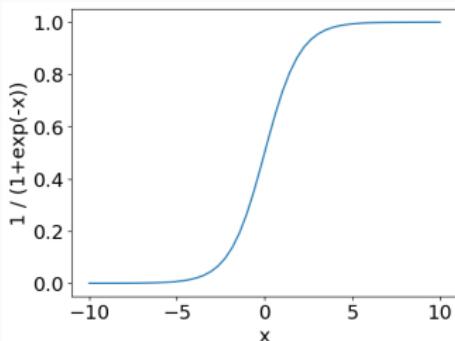
## Logistic Regression: Derivation III

$$\log \frac{P(x)}{1 - P(x)} = \theta_0 + \theta_1 x_1 + \dots + \theta_F x_F$$

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

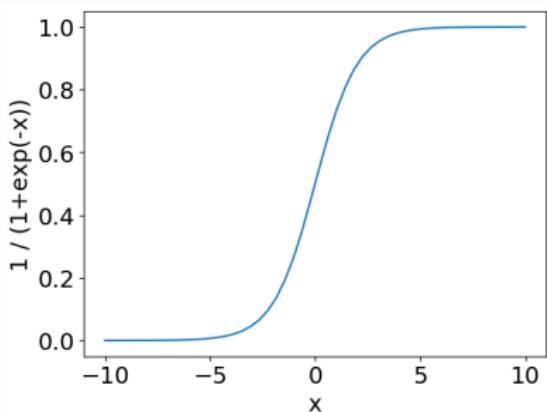


# Logistic Regression: Interpretation

$$P(y|x; \theta) = \frac{1}{1 + \exp(-(\theta_0 + \sum_{f=1}^F \theta_f x_f))}$$

## 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  $y = 0$



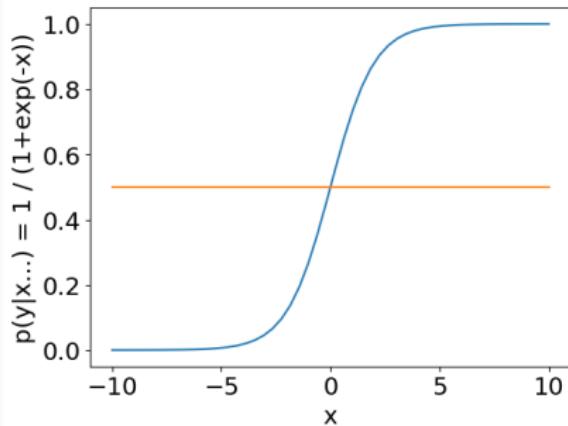
## Logistic Regression: Prediction

- 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

$$\frac{\log \mathcal{L}(\theta)}{\partial \theta_j} = \sum_{i=1}^N \left( \sigma(\theta^T x^i) - y^i \right) x_j^i$$

- 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**



## 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:  $y \in \{\text{play}, \text{not\_play}\}$ )

$$P(y = 1|x; \theta) = \sigma(\theta^T x) = \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)}$$
$$P(y = 0|x; \theta) = 1 - \sigma(\theta^T x) = 1 - \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)}$$



## 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

$$p(y = c|x; \theta) = \frac{\exp(\theta_c x)}{\sum_k \exp(\theta_k x)}$$

### (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

### Model parameters

$$\theta_{c0} = [0.1, 0.7, 0.2, -3.5, -3.5, -3.5, 0.7, 2.1]$$

$$\theta_{c1} = [0.6, 0.1, 0.9, 2.5, 2.5, 2.5, 2.7, -2.1]$$

$$\theta_{c2} = [3.1, 3.4, 4.1, 1.5, 1.5, 1.5, 0.7, 3.6]$$

### 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 10: The Perceptron

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



## **Introduction**

---

## So far... Naive Bayes and Logistic Regression

- Probabilistic models
- Maximum likelihood estimation
- Examples and code



## So far... Naive Bayes and Logistic Regression

- 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



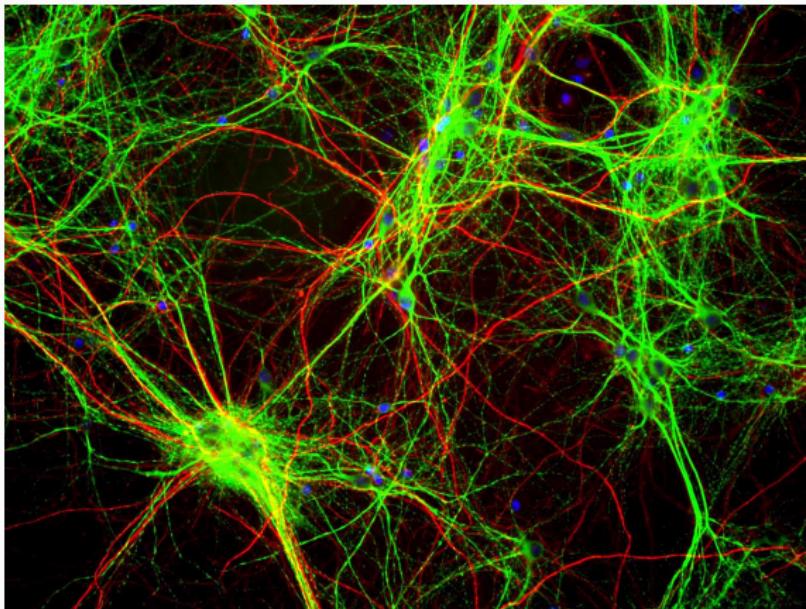
# Introduction: Neurons in the Brain

- Humans are the best learners we know
- Can we take inspiration from human learning
- → the brain!

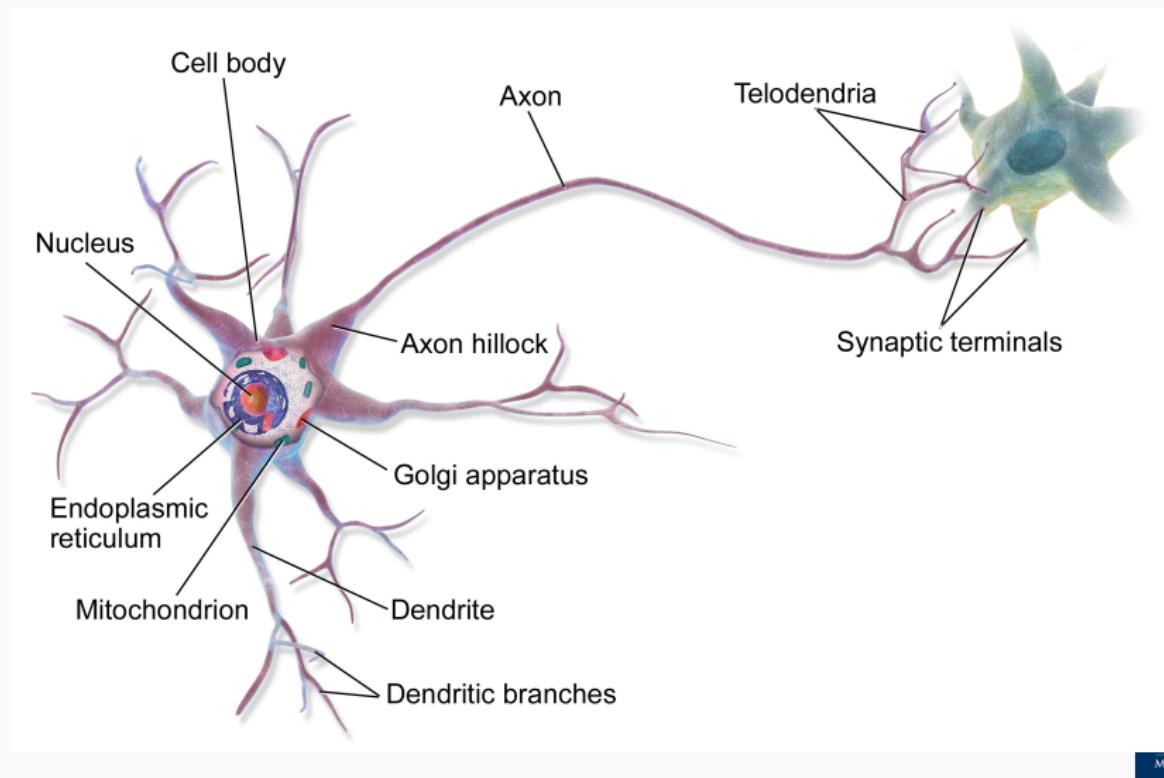


# Introduction: Neurons in the Brain

<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 ]].

**NEW NAVY DEVICE  
LEARNS BY DOING**

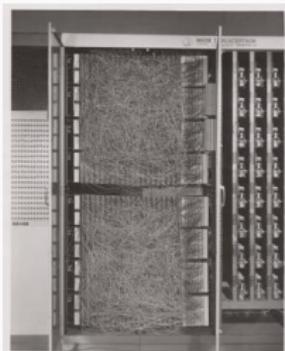
Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the key to making a thinking computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The key—the Weather Bureau's \$2,600,000 "T-4" computer—learned to read and identify right and left after fifty attempts in the Navy's demonstration room here.

The service said it would use this principle to build the first of its Perceptrons thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

The New York Times, July 8 1958

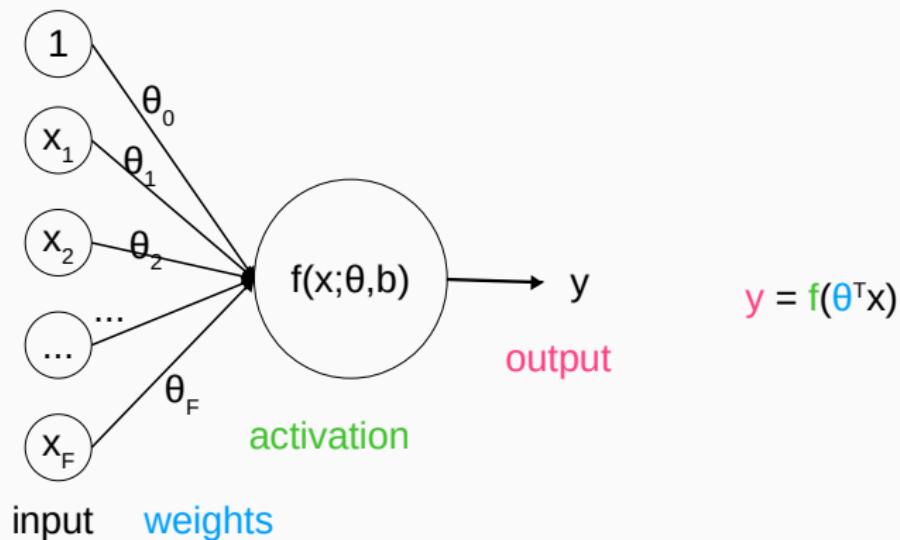


## The AI winter

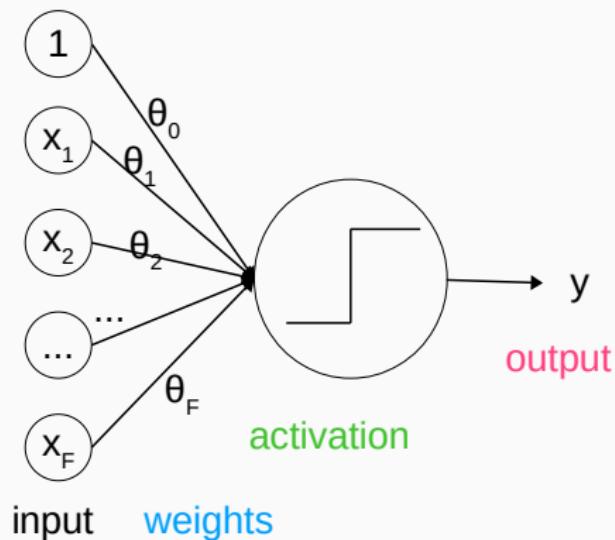
- A few years later Misky and Papert (too?) successfully pointed out the fundamental limitations of the perceptron.
- As a result, research on artificial neural networks stopped until the mid-1980s
- But the limitations can be overcome by combining multiple perceptrons into **Artificial Neural Networks**
- 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:

$$y^i = f \left( \left[ \sum_j \theta_j x_j^i \right] \right) = f(\theta^T x^i)$$



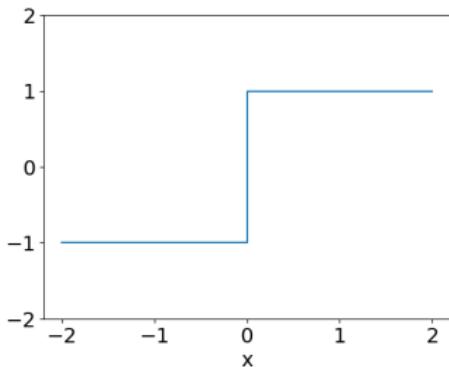
THE UNIVERSITY OF  
MELBOURNE

## 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:

$\langle x_1, x_2 \rangle$	$\theta_1 \cdot 1 + \theta_2 \cdot x_1 + \theta_3 \cdot x_2$	$\hat{y}_i^{(1)}$	$y_i$
$\langle 1, 1 \rangle$	$0 + 1 \times 0 + 1 \times 0 = 0$	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:

$\langle x_1, x_2 \rangle$	$\theta_1 \cdot 1 + \theta_2 \cdot x_1 + \theta_3 \cdot x_2$	$\hat{y}_i^{(2)}$	$y_i$
$\langle 1, 1 \rangle$	$-2 + 1 \times 0 + 1 \times 0 = -2$	-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:

$\langle x_1, x_2 \rangle$	$\theta_1 \cdot 1 + \theta_2 \cdot x_1 + \theta_3 \cdot x_2$	$\hat{y}_i^{(3)}$	$y_i$
$\langle 1, 1 \rangle$	$-2 + 1 \times 2 + 1 \times 2 = 2$	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

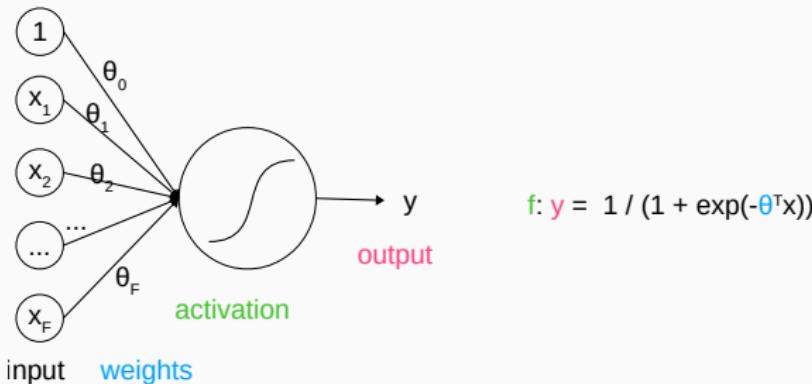
$$\theta_j^{(t+1)} \leftarrow \theta_j^{(t)} + \eta(y_i - \hat{y}^i)x_j^i$$

## 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 11: Neural Networks

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



## So far ... Classification and Evaluation

- KNN, Naive Bayes, Logistic Regression, Perceptron
- Probabilistic models
- Loss functions, and estimation
- Evaluation



## So far ... Classification and Evaluation

- KNN, Naive Bayes, Logistic Regression, Perceptron
- Probabilistic models
- Loss functions, and estimation
- Evaluation

## Today... Neural Networks

- Multilayer Perceptron
- Motivation and architecture
- Linear vs. non-linear classifiers



## **Introduction**

---

# Classifier Recap

## Perceptron

$$\hat{y} = f(\theta \cdot x) = \begin{cases} 1 & \text{if } \theta \cdot x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Single processing ‘unit’
- Inspired by neurons in the brain
- Activation: step-function (discrete, non-differentiable)



# Classifier Recap

## Perceptron

$$\hat{y} = f(\theta \cdot x) = \begin{cases} 1 & \text{if } \theta \cdot x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Single processing ‘unit’
- Inspired by neurons in the brain
- Activation: step-function (discrete, non-differentiable)

## Logistic Regression

$$P(y = 1|x; \theta) = \frac{1}{1 + \exp(-(\sum_{f=0}^F \theta_f x_f))}$$

- View 1: Model of  $P(y = 1|x)$ , maximizing the data log likelihood
- View 2: Single processing ‘unit’
- Activation: sigmoid (continuous, differentiable)



## Neural Networks

- Connected sets of many such units
- Units must have continuous activation functions
- Connected into many layers → **Deep** Learning

## Multi-layer Perceptron

- This lecture!
- One specific type of neural network
- Feed-forward
- Fully connected
- Supervised learner



# Neural Networks and Deep Learning

## Neural Networks

- Connected sets of many such units
- Units must have continuous activation functions
- Connected into many layers → **Deep** Learning

## Multi-layer Perceptron

- This lecture!
- One specific type of neural network
- Feed-forward
- Fully connected
- Supervised learner

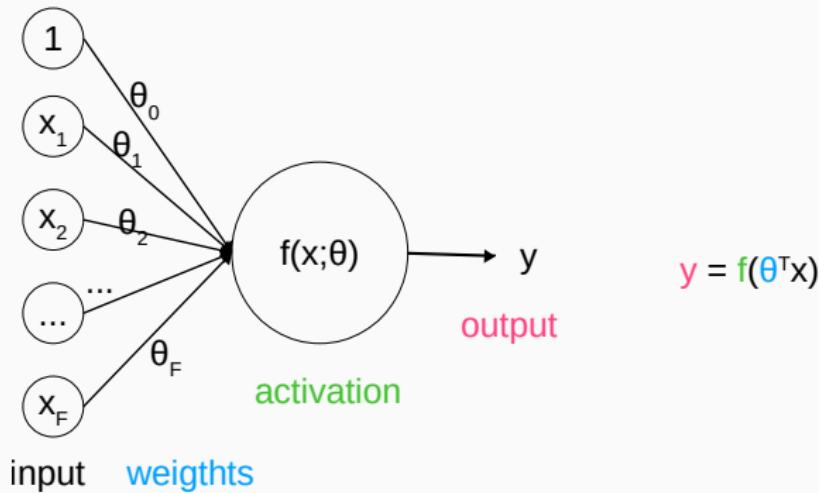
## Other types of neural networks

- Convolutional neural networks
- Recurrent neural networks
- Autoencoder (unsupervised)



# Perceptron Unit (recap)

A single processing unit



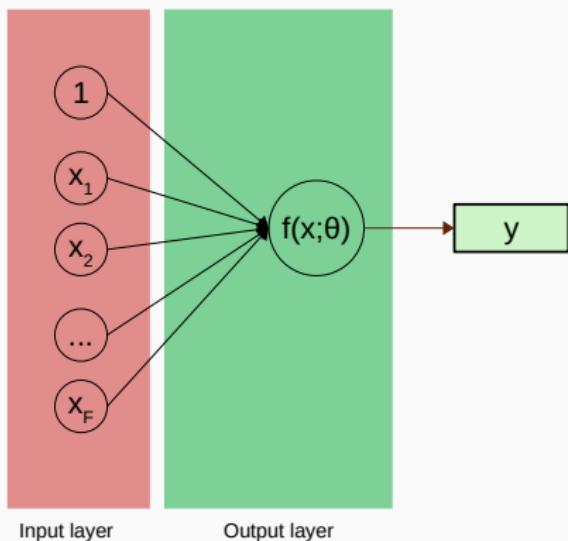
A **neural network** is a combination of lots of these units.



# Multi-layer Perceptron (schematic)

## Three Types of layers

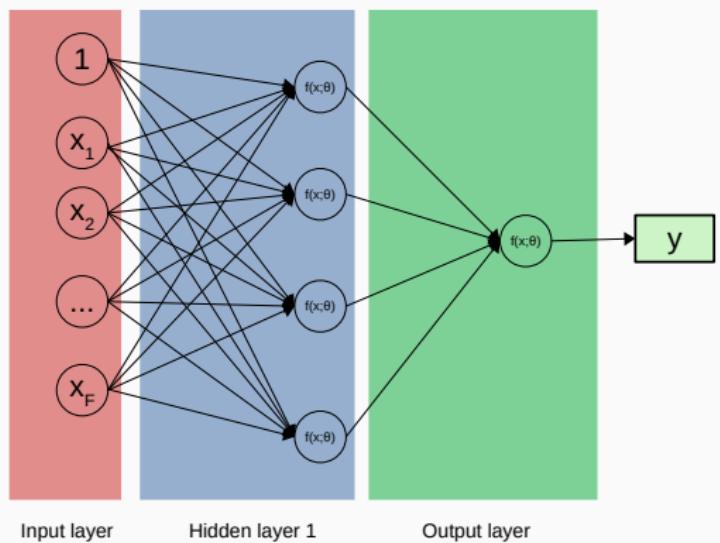
- **Input layer** with input units  $x$ : the first layer, takes features  $x$  as inputs
- **Output layer** with output units  $y$ : the last layer, has one unit per possible output (e.g., 1 unit for binary classification)
- **Hidden layers** with hidden units  $h$ : all layers in between.



# Multi-layer Perceptron (schematic)

## Three Types of layers

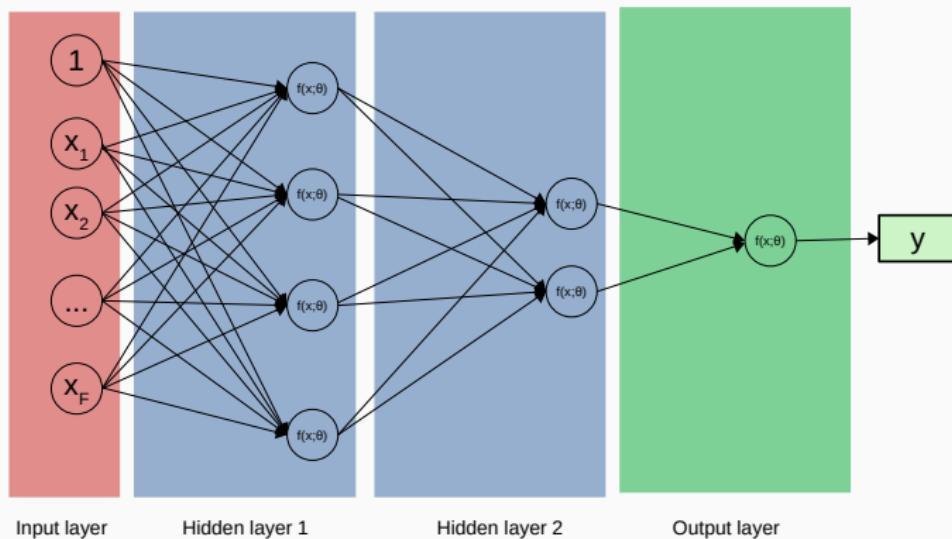
- **Input layer** with input units  $x$ : the first layer, takes features  $x$  as inputs
- **Output layer** with output units  $y$ : the last layer, has one unit per possible output (e.g., 1 unit for binary classification)
- **Hidden layers** with hidden units  $h$ : all layers in between.



# Multi-layer Perceptron (schematic)

## Three Types of layers

- **Input layer** with input units  $x$ : the first layer, takes features  $x$  as inputs
- **Output layer** with output units  $y$ : the last layer, has one unit per possible output (e.g., 1 unit for binary classification)
- **Hidden layers** with hidden units  $h$ : all layers in between.



# Why the Hype? I

## Linear classification

- The perceptron, naive bayes, logistic regression are linear classifiers
- Decision boundary is a linear combination of features  $\sum_i \theta_i x_i$
- Cannot learn ‘feature interactions’ naturally
- Perceptron can solve only linearly separable problems

## Non-linear classification

- 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’



# Why the Hype? II

## Feature Engineering

- (more next week!)
- The perceptron, naive Bayes and logistic regression require a fixed set of informative **features**
- e.g., outlook  $\in \{overcast, sunny, rainy\}$ , wind  $\in \{high, low\}$  etc
- Requiring **domain knowledge**

## Feature learning

- Neural networks take as input ‘raw’ data
- They learn features themselves as intermediate representations
- They learn features as part of their target task (e.g., classification)
- ‘Representation learning’: learning representations (or features) of the data that are useful for the target task
- Note: often feature engineering is replaced at the cost of additional parameter tuning (layers, activations, learning rates, ...)



# Multilayer Perceptron: Motivation I

## Example Classification 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
...				

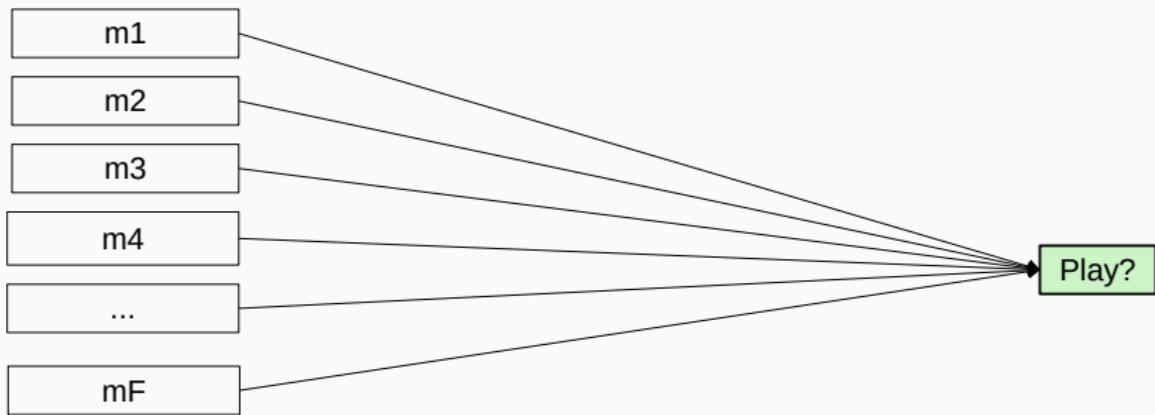
We really observe raw data

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



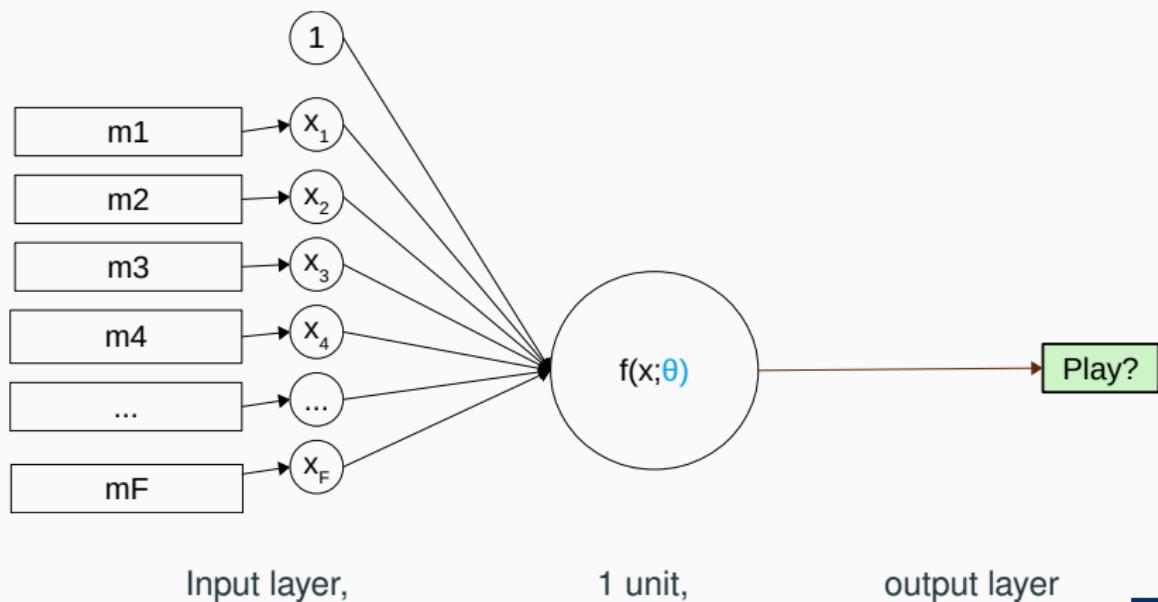
# Multilayer Perceptron: Motivation II

## Example Problem: Weather Dataset



# Multilayer Perceptron: Motivation II

## Example Problem: Weather Dataset



Input layer,

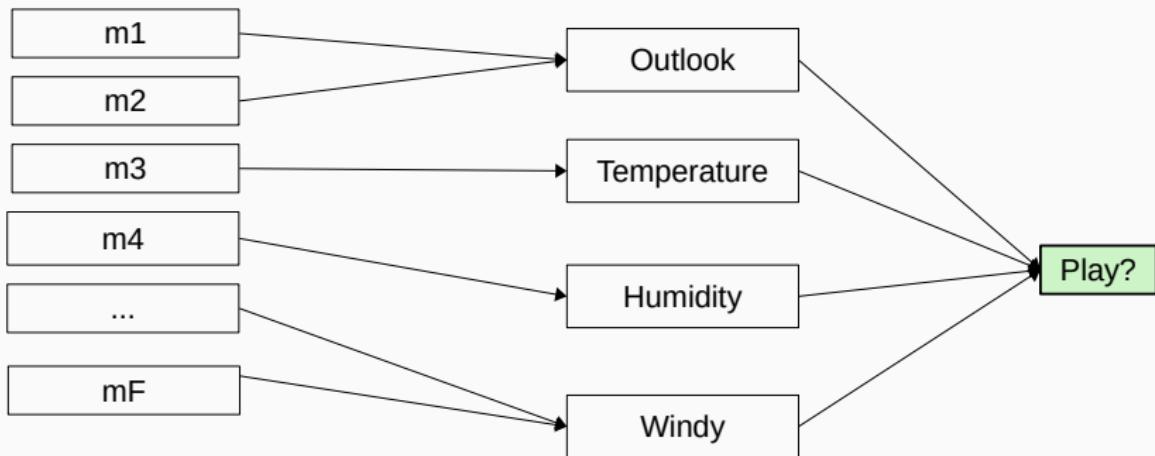
1 unit,

output layer



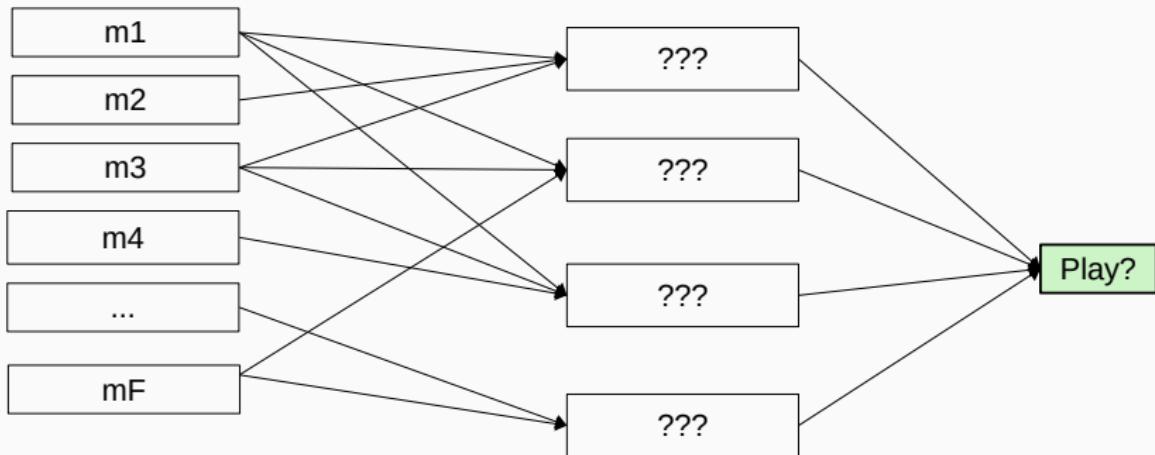
# Multilayer Perceptron: Motivation II

## Example Problem: Weather Dataset



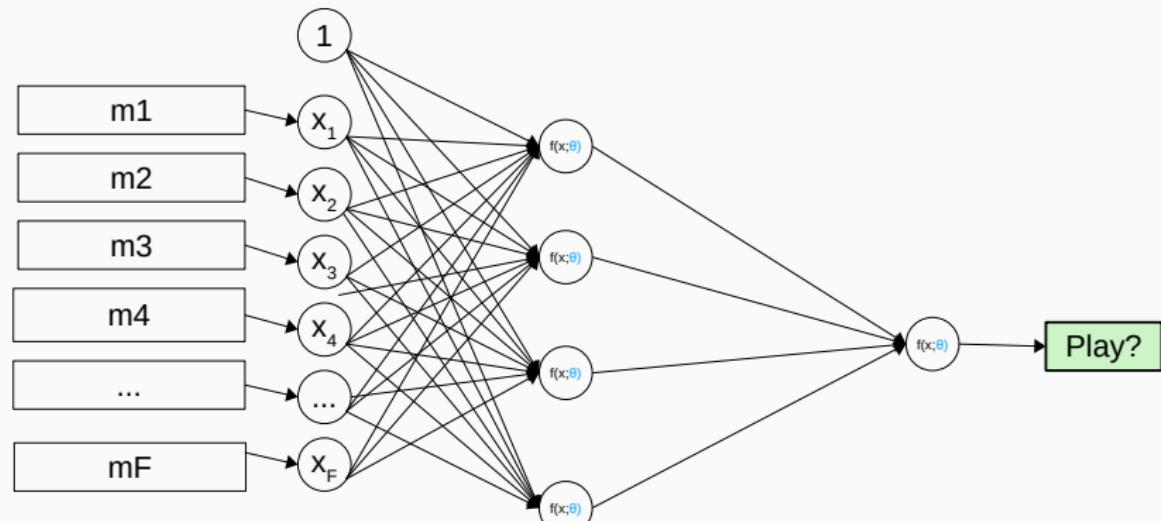
# Multilayer Perceptron: Motivation II

## Example Problem: Weather Dataset



# Multilayer Perceptron: Motivation II

## Example Problem: Weather Dataset



Input layer,

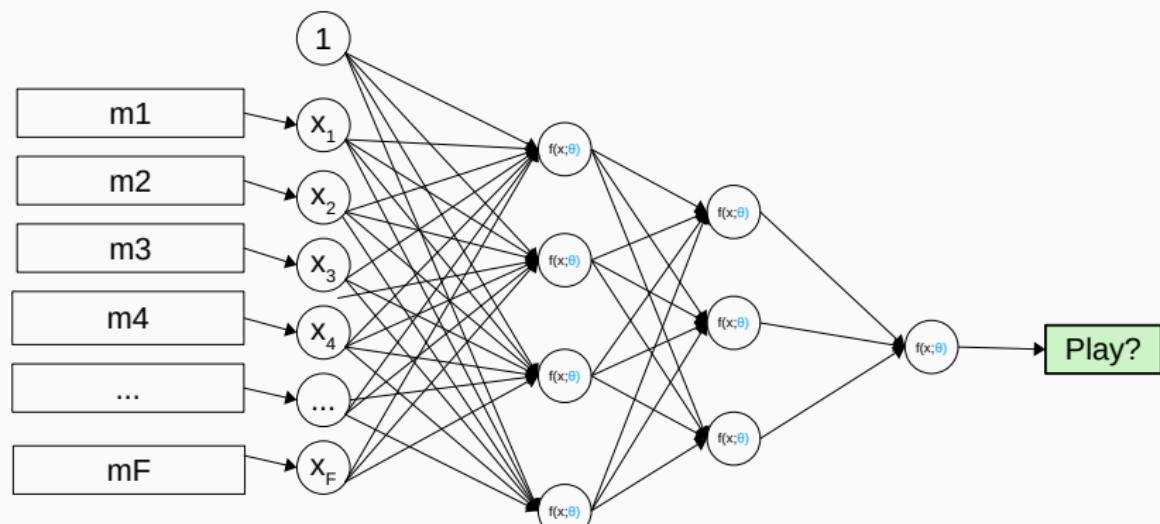
1 hidden layer,

output layer



# Multilayer Perceptron: Motivation II

## Example Problem: Weather Dataset



Input layer,

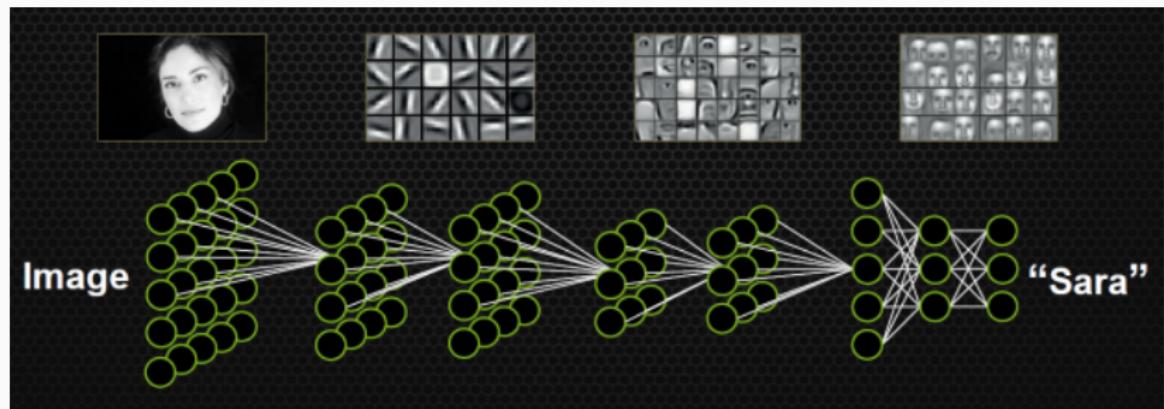
2 hidden layer,

output layer



## Another Example: Face Recognition

- the **hidden layers** learn increasingly high-level feature representations
- e.g., given an image, predict the person:



Source: <https://devblogs.nvidia.com/accelerate-machine-learning-cudnn-deep-neural-network-library/>



# Multilayer Perceptron

## Terminology

- input units  $x_j$ , one per feature  $j$
- Multiple **layers**  $l = 1 \dots L$  of nodes.  $L$  is the **depth** of the network.
- Each layer  $l$  has a number of units  $K_l$ .  $K_l$  is the **width** of layer  $l$ .
- The width can vary from layer to layer
- output unit  $y$
- Each layer  $l$  is **fully connected** to its neighboring layers  $l - 1$  and  $l + 1$
- one weight  $\theta_{ij}^{(l)}$  for each connection  $ij$  (including 'bias'  $\theta_0$ )
- non-linear activation function for layer  $l$  as  $\phi^{(l)}$



# Prediction with a feedforward Network

Passing an input through a neural network with 2 hidden layers

$$h_i^{(1)} = \phi^{(1)} \left( \sum_j \theta_{ij}^{(1)} x_j \right)$$

$$h_i^{(2)} = \phi^{(2)} \left( \sum_j \theta_{ij}^{(2)} h_j^{(1)} \right)$$

$$y_i = \phi^{(3)} \left( \sum_j \theta_{ij}^{(3)} h_j^{(2)} \right)$$



# Prediction with a feedforward Network

Passing an input through a neural network with 2 hidden layers

$$h_i^{(1)} = \phi^{(1)} \left( \sum_j \theta_{ij}^{(1)} x_j \right)$$

$$h_i^{(2)} = \phi^{(2)} \left( \sum_j \theta_{ij}^{(2)} h_j^{(1)} \right)$$

$$y_i = \phi^{(3)} \left( \sum_j \theta_{ij}^{(3)} h_j^{(2)} \right)$$

Or in vectorized form

$$h^{(1)} = \phi^{(1)} \left( \theta^{(1)T} x \right)$$

$$h^{(2)} = \phi^{(2)} \left( \theta^{(2)T} h^{(1)} \right)$$

$$y = \phi^{(3)} \left( \theta^{(3)T} h^{(2)} \right)$$

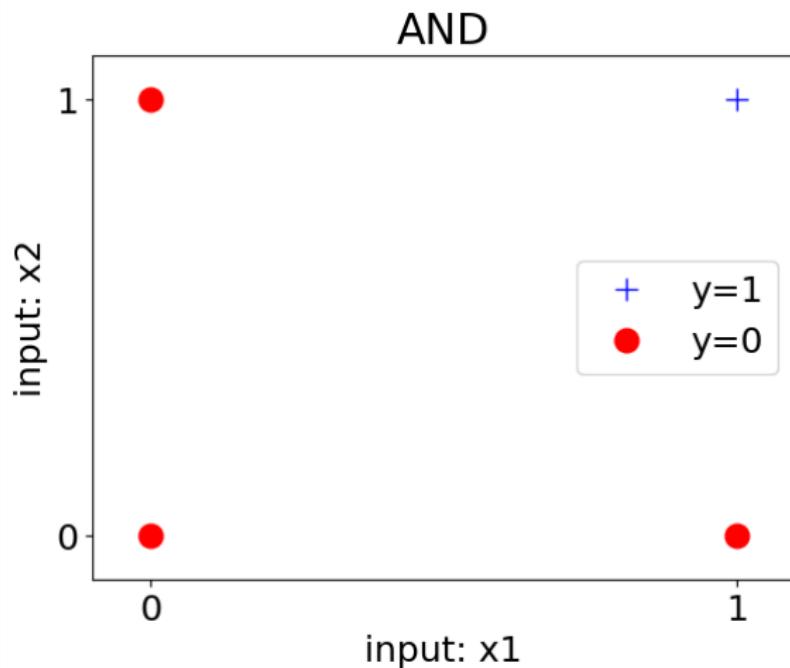
where the activation functions  $\phi^{(l)}$  are applied **element-wise** to all entries



# Boolean Functions

1. Can the **perceptron** learn this function? Why (not)?
2. Can a **multilayer perceptron** learn this function? Why (not)?

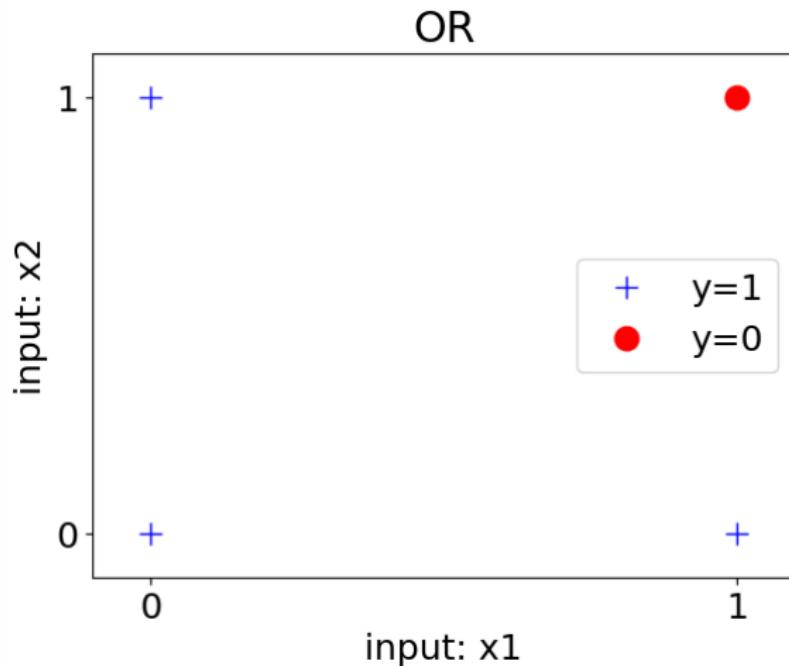
$x_1$	$x_2$	$y$
1	1	1
1	0	0
0	1	0
0	0	0



# Boolean Functions

1. Can the **perceptron** learn this function? Why (not)?
2. Can a **multilayer perceptron** learn this function? Why (not)?

$x_1$	$x_2$	$y$
1	1	0
1	0	1
0	1	1
0	0	1

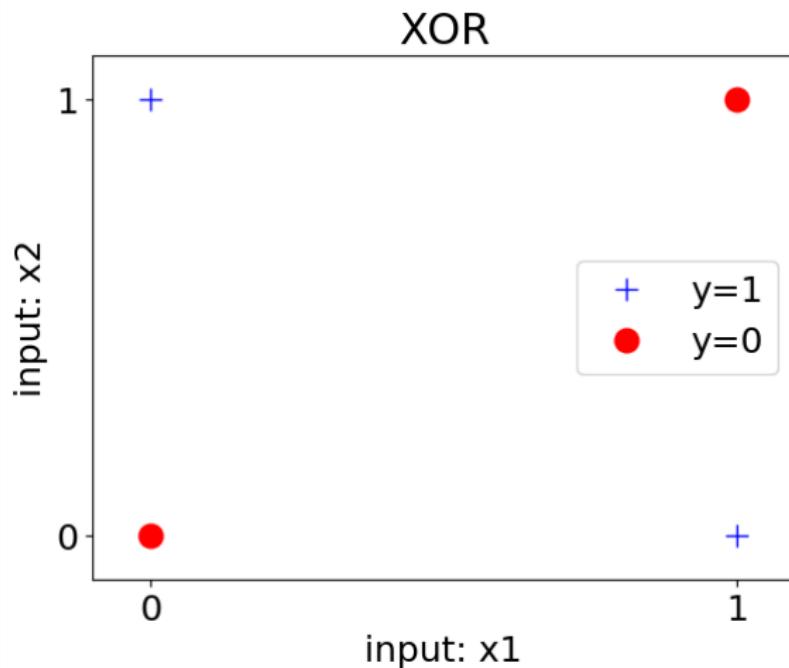


THE UNIVERSITY OF  
MELBOURNE

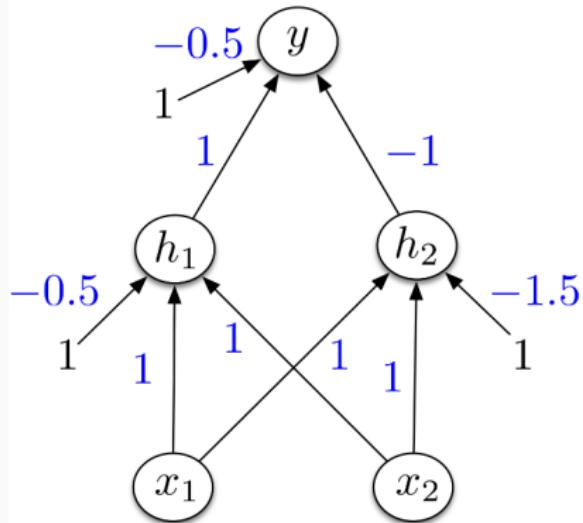
# Boolean Functions

1. Can the **perceptron** learn this function? Why (not)?
2. Can a **multilayer perceptron** learn this function? Why (not)?

$x_1$	$x_2$	$y$
1	1	0
1	0	1
0	1	1
0	0	0



# A Multilayer Perceptron for XOR



$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad \text{and recall: } h_i^{(l)} = \phi^{(l)}\left(\sum_j \theta_{ij}^{(l)} h_j^{(l-1)} + b_i^{(l)}\right)$$

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)

# Designing Neural Networks I: Inputs

## Inputs and feature functions

- $x$  could be a patient with features {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 features need to be mapped to numerical

- For language, typical to map words to **pre-trained embedding vectors**
  - for 1-hot:  $\dim(x) = V$  (words in the vocabulary)
  - for embedding:  $\dim(x) = k$ , dimensionality of embedding vectors
- Alternative: **1-hot encoding**
- For pixels, map to RGB, or other visual features



## Designing Neural Networks II: Activation Functions

- Each layer has an associated activation function (e.g., sigmode, ReLU, ...)
- Represents the extent to which a neuron is ‘activated’ given an input
- Each hidden layer performs a **non-linear transformation** of the input
- Popular choices include



## Designing Neural Networks II: Activation Functions

- Each layer has an associated activation function (e.g., sigmode, ReLU, ...)
- Represents the extent to which a neuron is ‘activated’ given an input
- Each hidden layer performs a **non-linear transformation** of the input
- Popular choices include

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”):

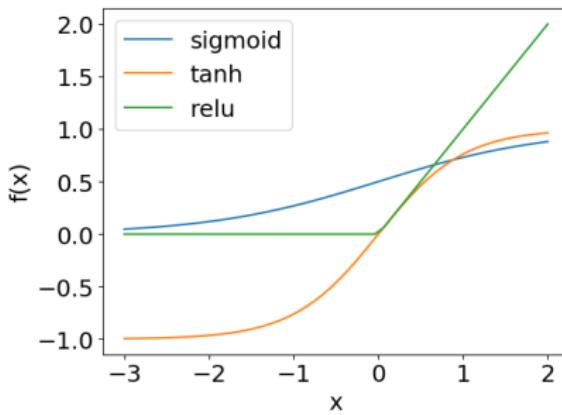
$$f(x) = \max(0, x)$$

note not differentiable at  $x = 0$

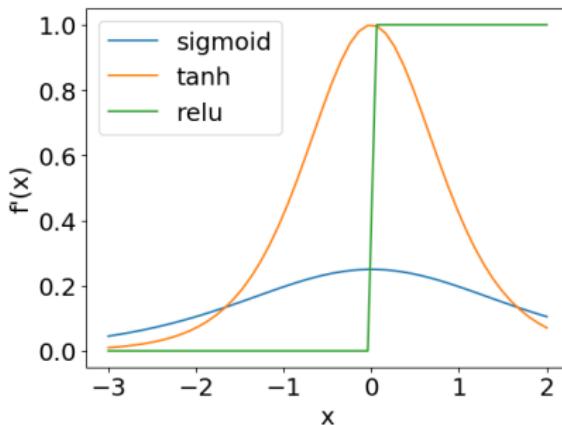


# Designing Neural Networks II: Activation Functions

function values:



derivatives:



# Designing Neural Networks III: 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.



## Designing Neural Networks IV: Output Function

Neural networks can learn different concepts: **classification, regression, ...**

The **output function** depends on the concept of interest.

- Binary classification:
  - one neuron, with step function (as in the perceptron)
- 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



## Designing Neural Networks V: Loss Functions

**Classification Loss:** typically negative conditional log-likelihood (cross-entropy)

$$\mathcal{L}^i = -\log p(y^{(i)}|x^{(i)}; \theta) \quad \text{for a single instance } i$$

$$\mathcal{L} = -\sum_i \log p(y^{(i)}|x^{(i)}; \theta) \quad \text{for all instances}$$

- 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.



## Designing Neural Networks V: Loss Functions

### Regression Loss: typically mean-squared error (MSE)

- Here, the output, as well as the target are real-valued numbers

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (y^i - \hat{y}^{(i)})^2$$



# Representational Power of Neural Nets

- The **universal approximation theorem** states that a feed-forward neural network with a single hidden layer (and finite neurons) is able to approximate any continuous function on  $\mathbb{R}^n$
- Note that **the activation functions must be non-linear**, as without this, the model is simply a (complex) linear model



## How to Train a NN with Hidden Layers

- Unfortunately, the perceptron algorithm can't be used to train neural nets with hidden layers, as we can't directly observe the labels
- Instead, train neural nets with **back propagation**. Intuitively:
  - compute errors at the output layer wrt each weight using partial differentiation
  - propagate those errors back to each of the input layers
- Essentially just gradient descent, but using the chain rule to make the calculations more efficient

**Next lecture: Backpropagation for training neural networks**



## **Reflections**

---

# 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



# Pros and Cons of Neural Networks

## Pros

- Powerful tool!
- Neural networks with at least 1 hidden layer can approximate any (continuous) function. They are **universal approximators**
- 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



# Summary

## Today

- From perceptrons to neural networks
- multilayer perceptron
- some examples
- features and limitations

## Next Lecture

- Learning parameters of neural networks
- The Backpropagation algorithm



THE UNIVERSITY OF  
MELBOURNE

## References

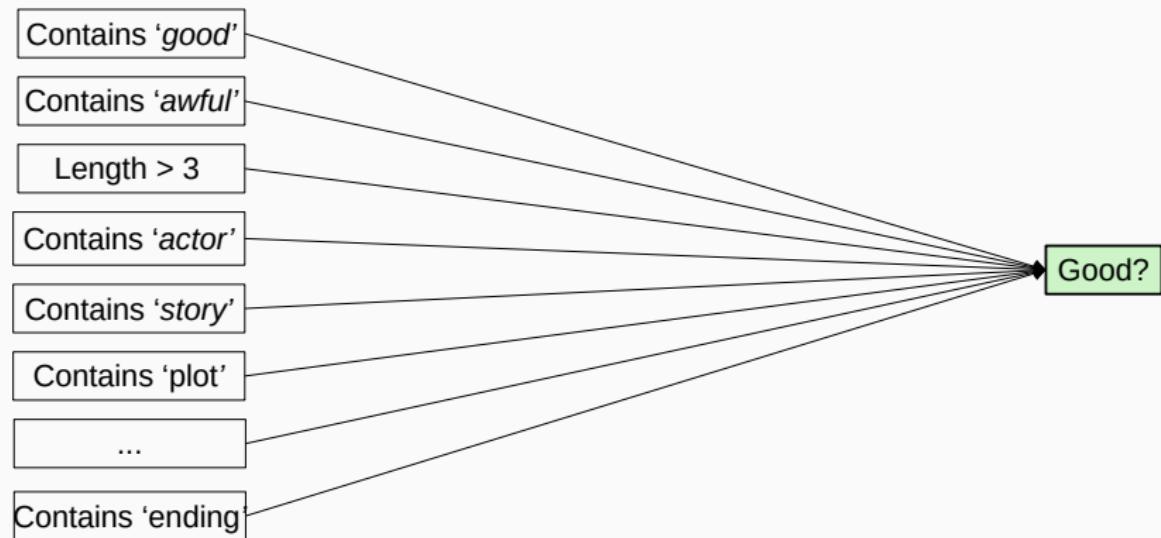
Jacob Eisenstein (2019). *Natural Language Processing*. MIT Press.  
Chapters 3 (intro), 3.1, 3.2. <https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf>

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/>



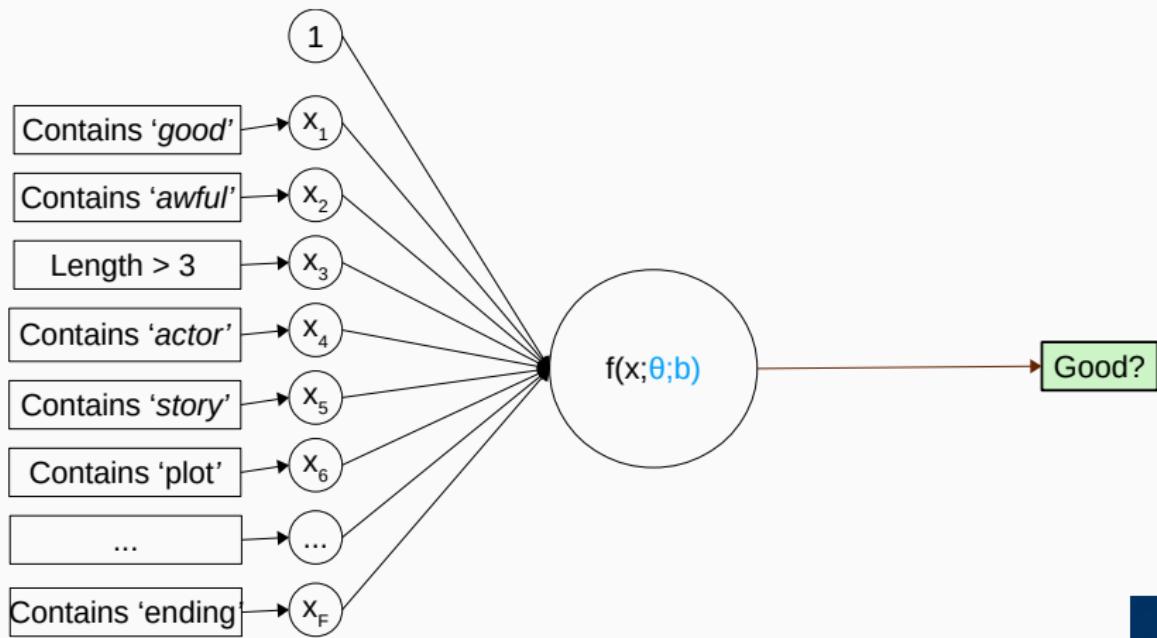
# Multilayer Perceptron: Motivation II

**Another Example Problem:** Sentiment analysis of movie reviews



# Multilayer Perceptron: Motivation II

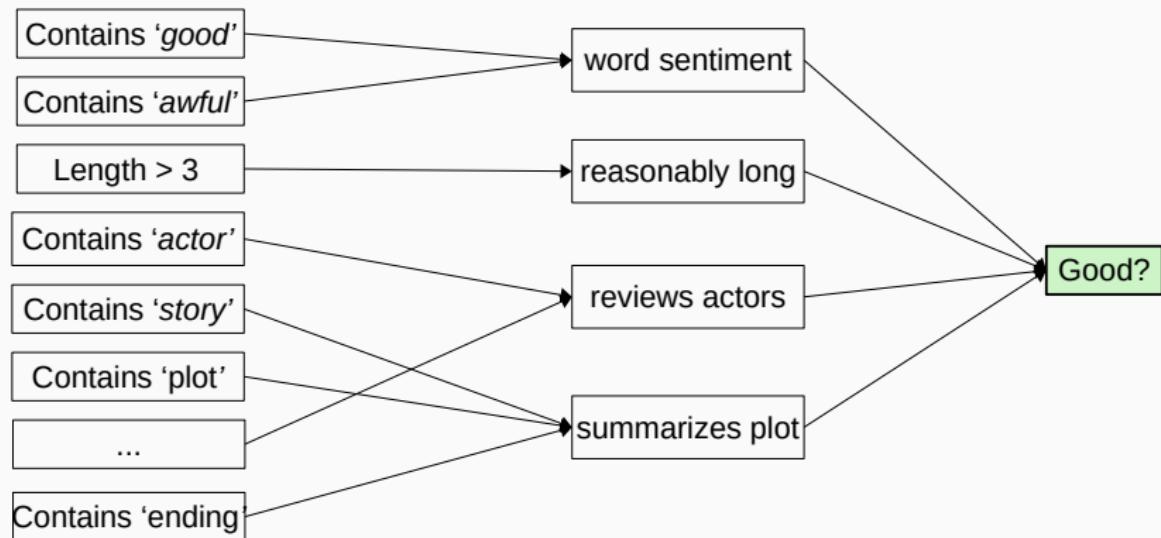
**Another Example Problem:** Sentiment analysis of movie reviews



Input layer, 1 unit, output layer

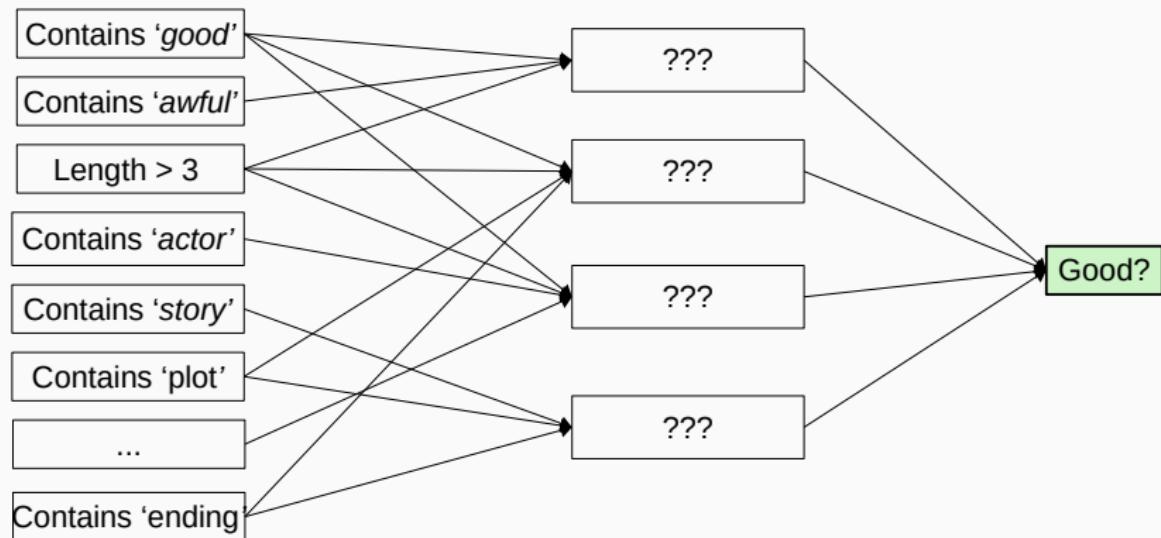
# Multilayer Perceptron: Motivation II

**Another Example Problem:** Sentiment analysis of movie reviews



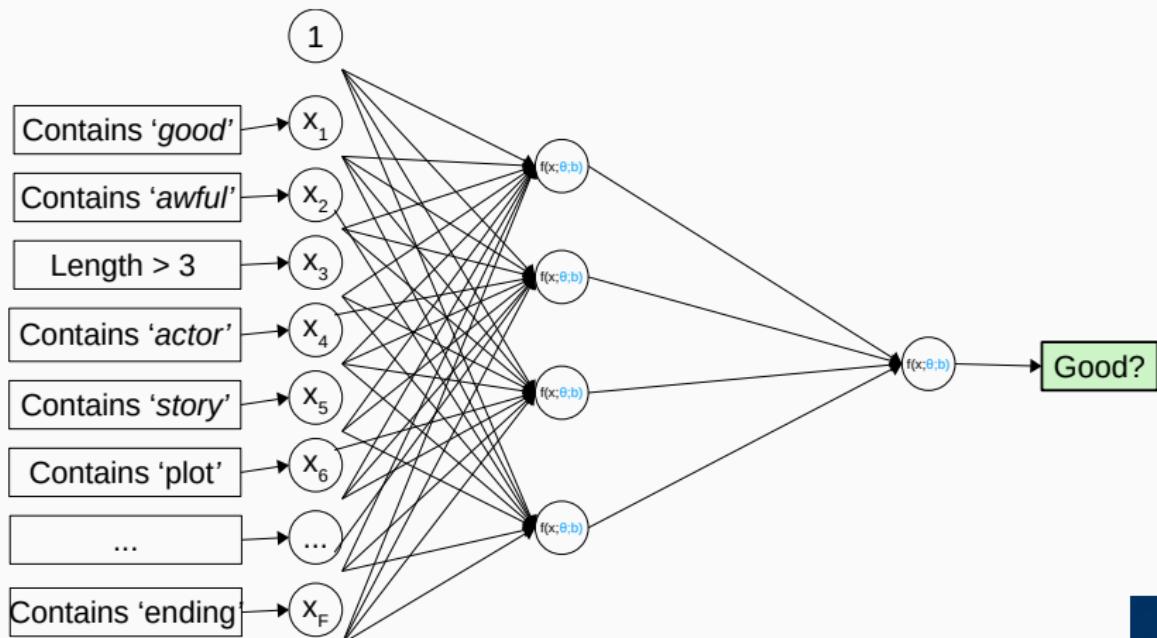
# Multilayer Perceptron: Motivation II

**Another Example Problem:** Sentiment analysis of movie reviews



# Multilayer Perceptron: Motivation II

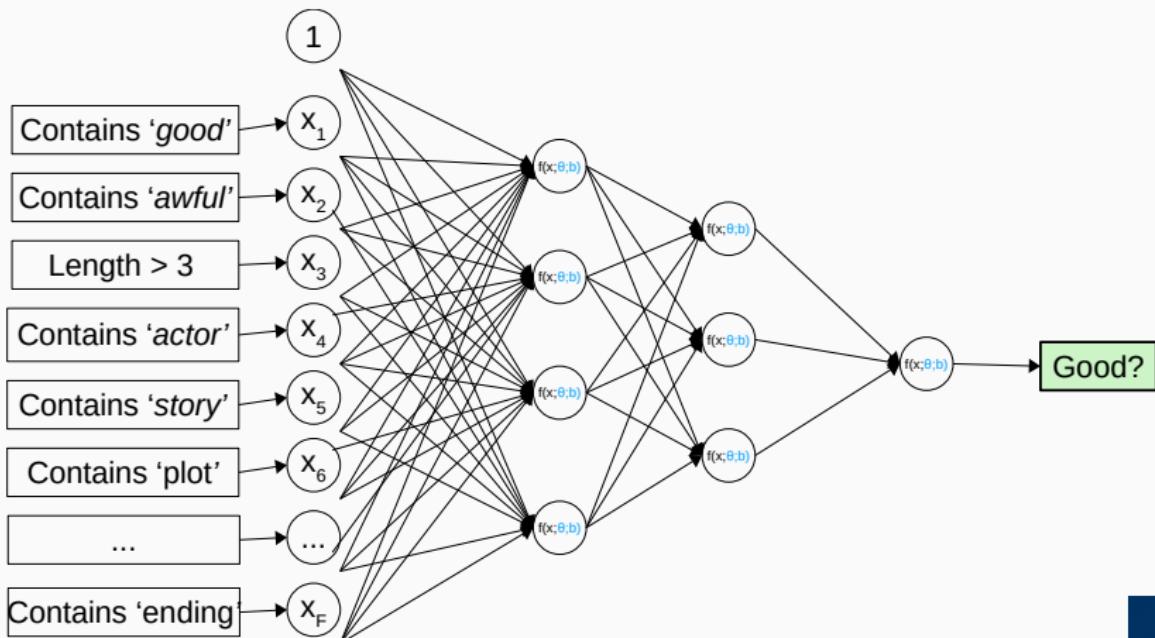
**Another Example Problem:** Sentiment analysis of movie reviews



Input layer, 1 hidden layer, output layer

# Multilayer Perceptron: Motivation II

**Another Example Problem:** Sentiment analysis of movie reviews



Input layer, 2 hidden layer, output layer

# Lecture 12: Learning Parameters of Multi-layer Perceptrons with Backpropagation

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



# Roadmap

## Last lecture

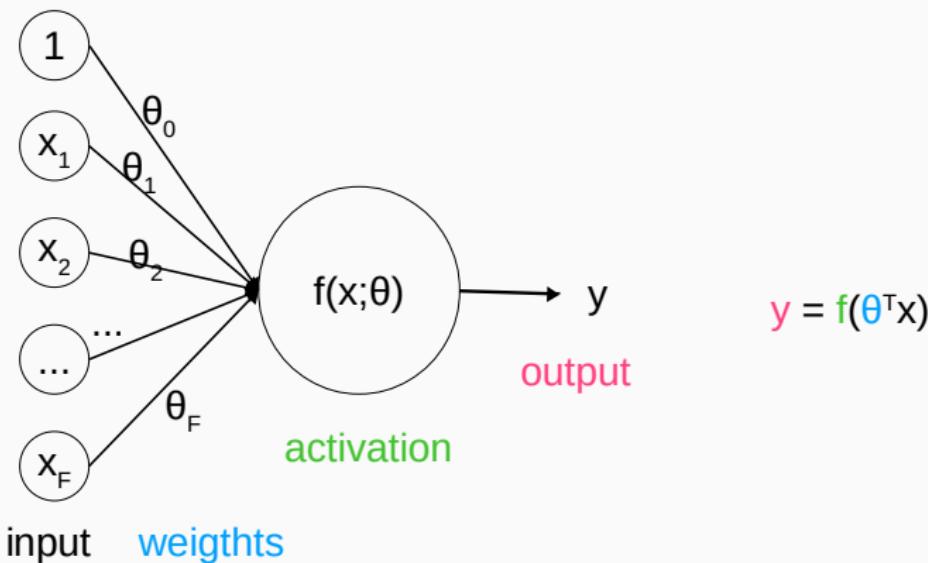
- From perceptrons to neural networks
- multilayer perceptron
- some examples
- features and limitations

## Today

- Learning parameters of neural networks
- The Backpropagation algorithm



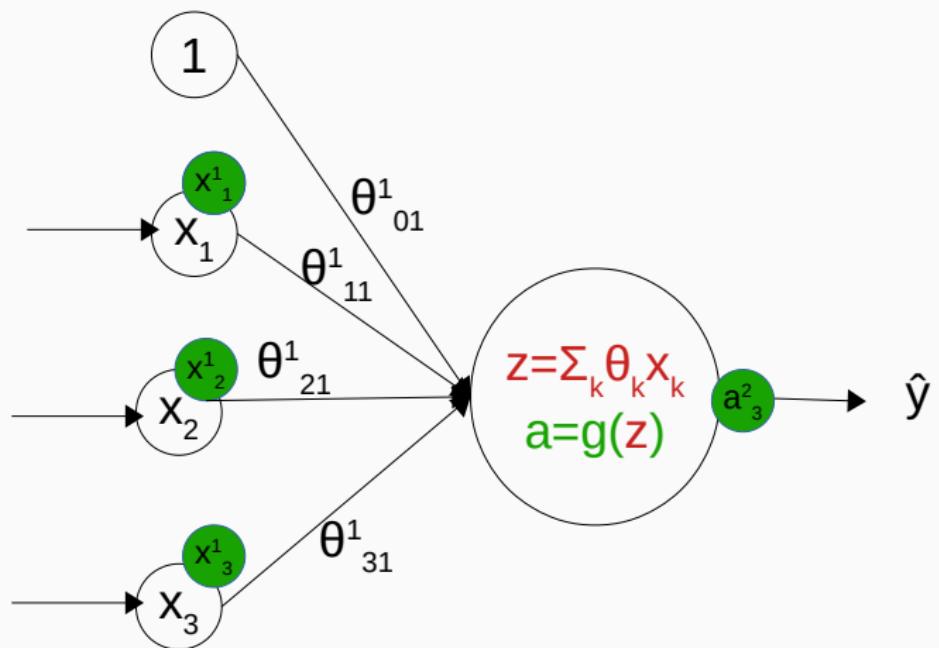
## Recap: Multi-layer perceptrons



- Linearly separable data
- Perceptron learning rule

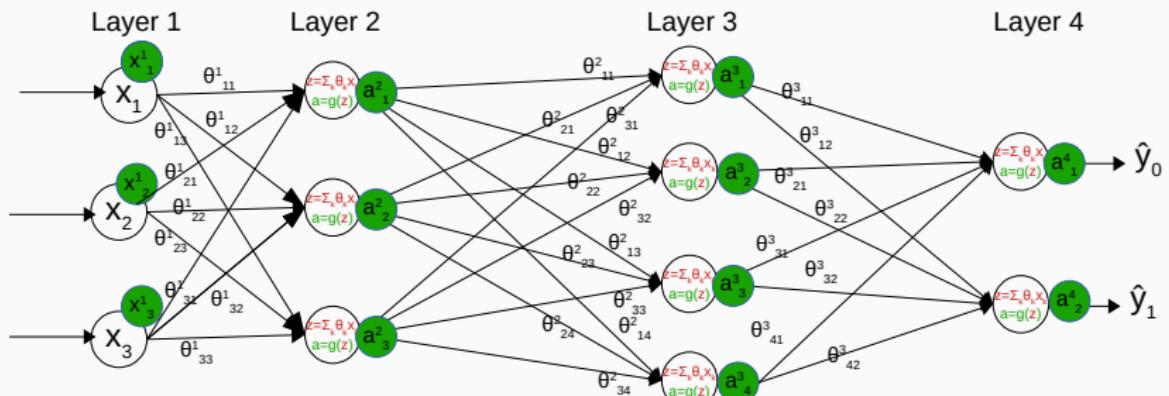


## Recap: Multi-layer perceptrons

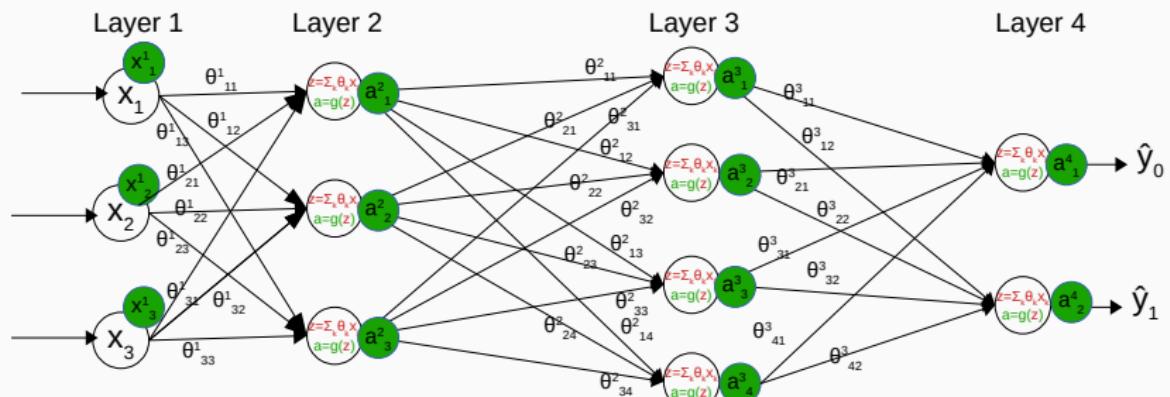


- Linearly separable data
- Perceptron learning rule

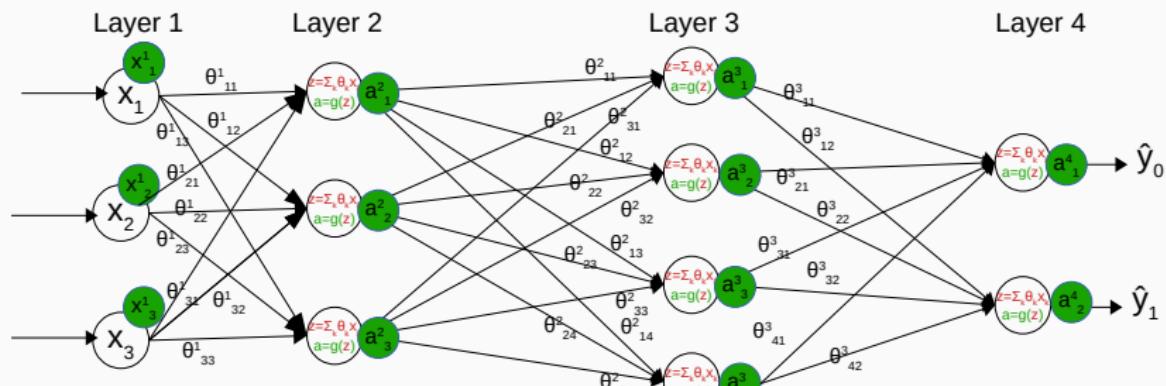
# Recap: Multi-layer perceptrons



# Recall: Supervised learning



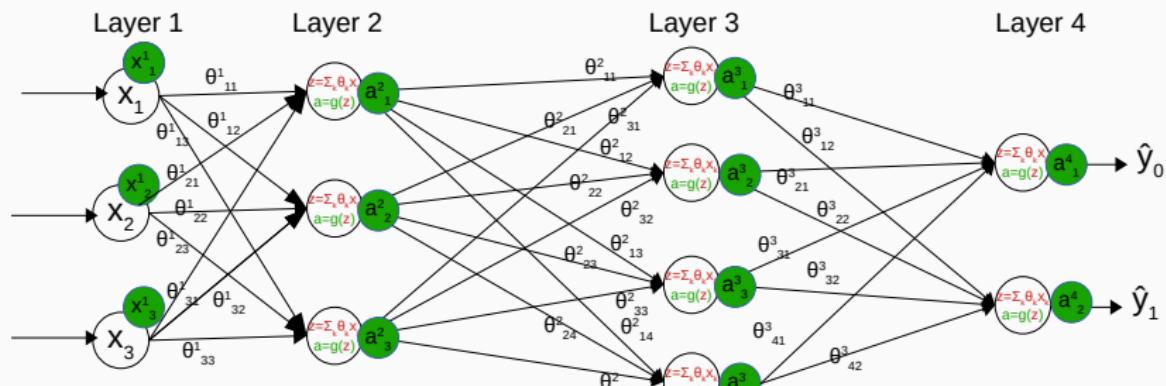
# Recall: Supervised learning



## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP

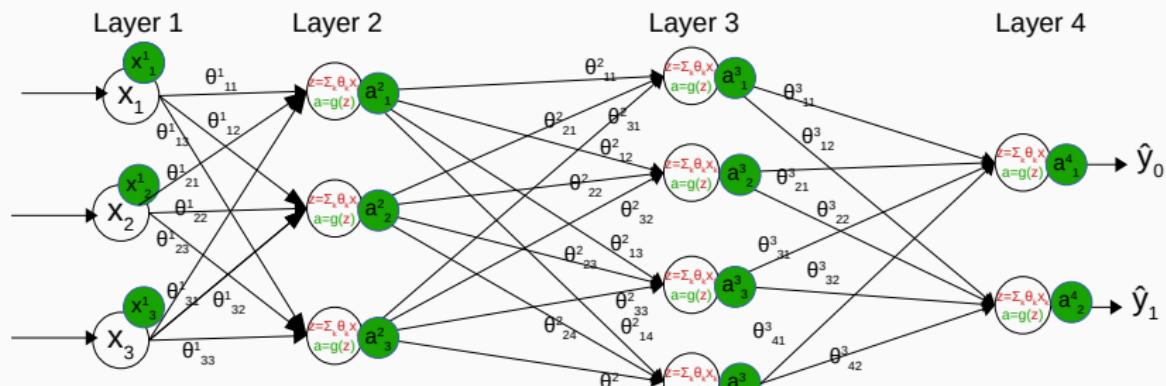
# Recall: Supervised learning



## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP
3. Compare predicted output  $\hat{y}$  against true output  $y$ ; compute the **error**

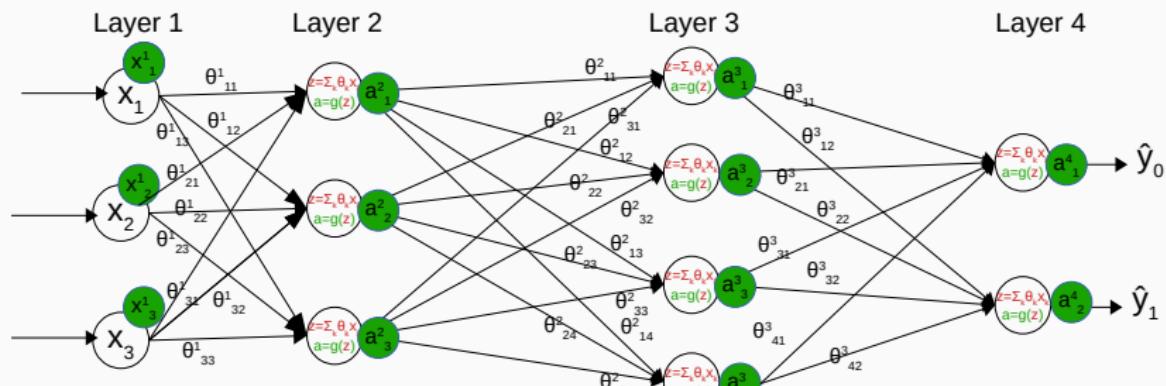
# Recall: Supervised learning



## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP
3. Compare predicted output  $\hat{y}$  against true output  $y$ ; compute the **error**
4. **Modify each weight** such that the error decreases in future predictions (e.g., by applying **gradient descent**)

# Recall: Supervised learning

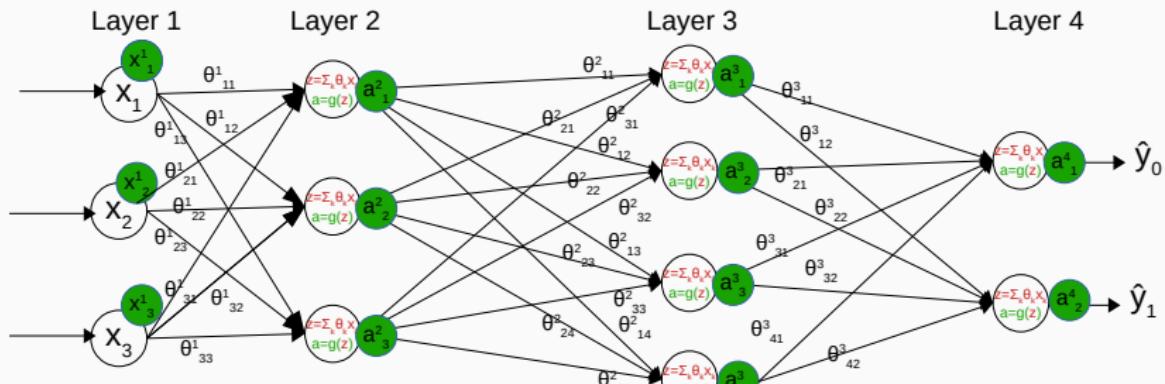


## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP
3. Compare predicted output  $\hat{y}$  against true output  $y$ ; compute the **error**
4. **Modify each weight** such that the error decreases in future predictions (e.g., by applying **gradient descent**)
5. Repeat.



# Recall: Optimization with Gradient Descent



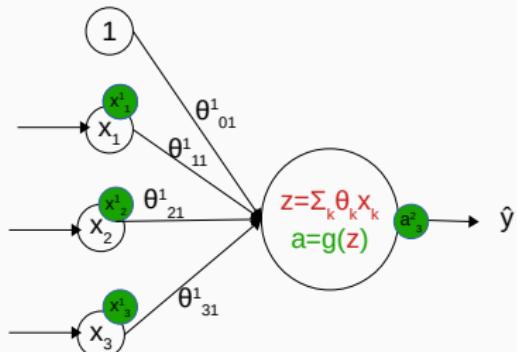
We want to

1. Find the best parameters, which lead to the smallest error  $E$
2. Optimize each model parameter  $\theta_{ij}^l$
3. We will use **gradient descent** to achieve that
4.  $\theta_{ij}^{l,(t+1)} \leftarrow \theta_{ij}^{l,(t)} + \Delta \theta_{ij}^l$

# Towards Backpropagation

Recall Perceptron learning:

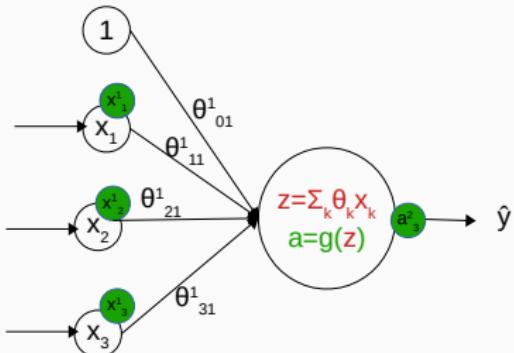
- Pass an input through and compute  $\hat{y}$
- Compare  $\hat{y}$  against  $y$
- Weight update  $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$



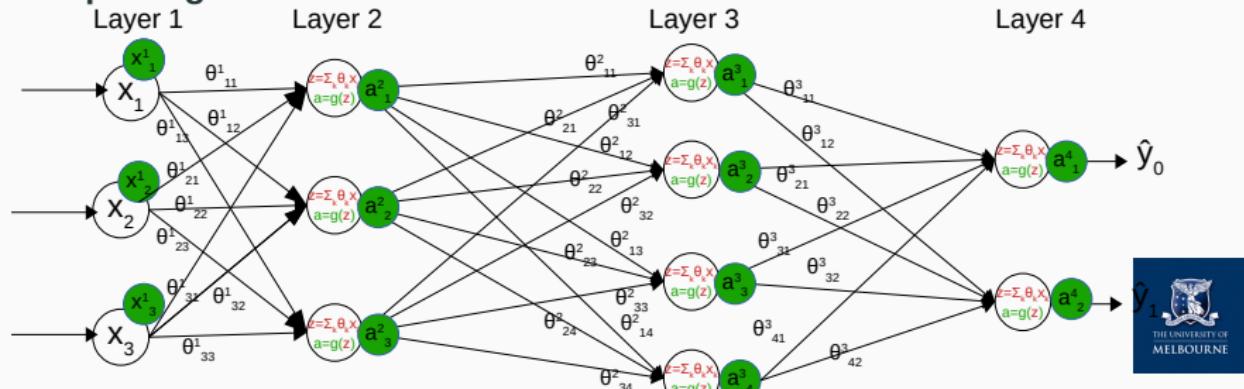
# Towards Backpropagation

Recall Perceptron learning:

- Pass an input through and compute  $\hat{y}$
- Compare  $\hat{y}$  against  $y$
- Weight update  $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$



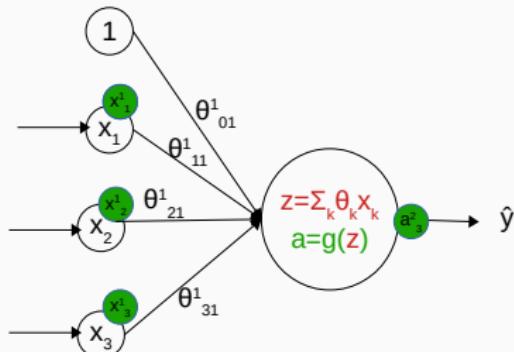
Compare against the MLP:



# Towards Backpropagation

## Recall Perceptron learning:

- Pass an input through and compute  $\hat{y}$
- Compare  $\hat{y}$  against  $y$
- Weight update  $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$

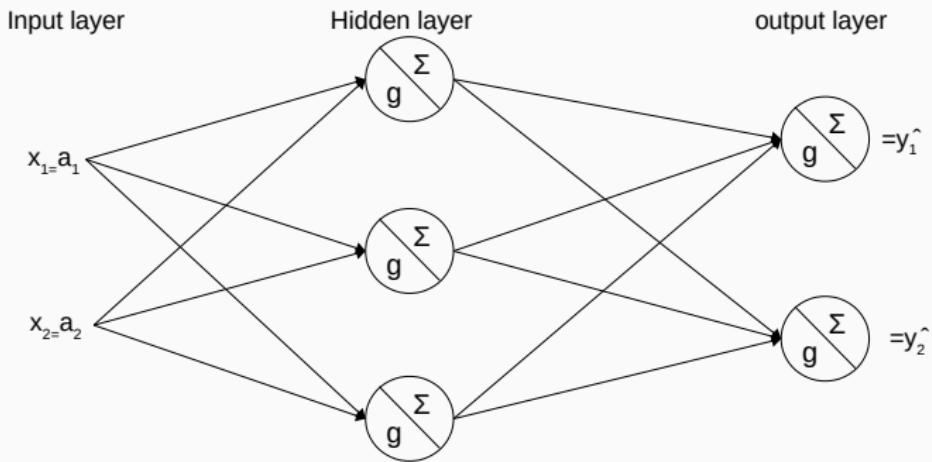


## Problems

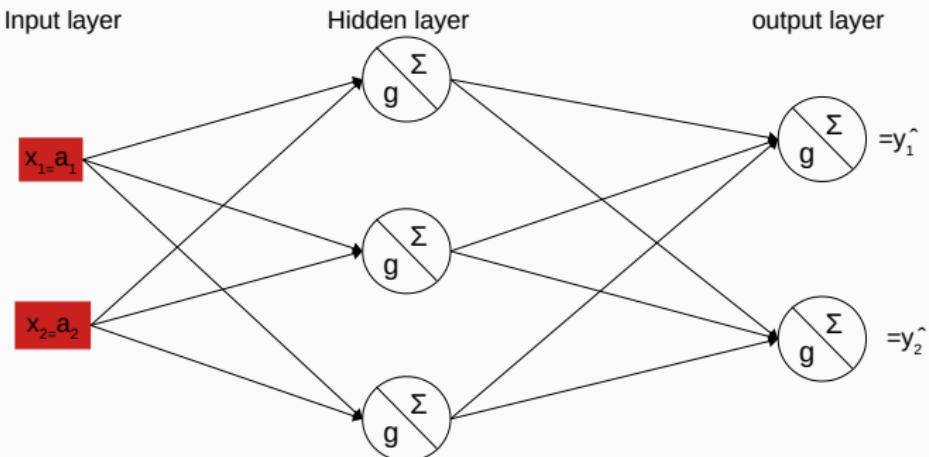
- This update rule depends on **true target outputs  $y$**
- We only have access to true outputs for the **final layer**
- We do not know the **true activations** for the **hidden layers**. Can we **generalize** the above rule to also update the hidden layers?

Backpropagation provides us with an efficient way of computing **partial derivatives** of the **error** of an MLP wrt. **each individual weight**.

# Backpropagation: Demo

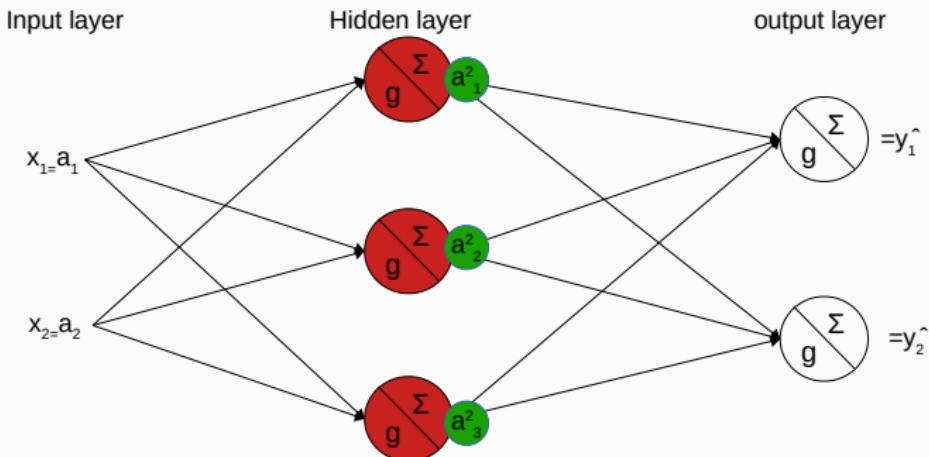


# Backpropagation: Demo



- Receive input

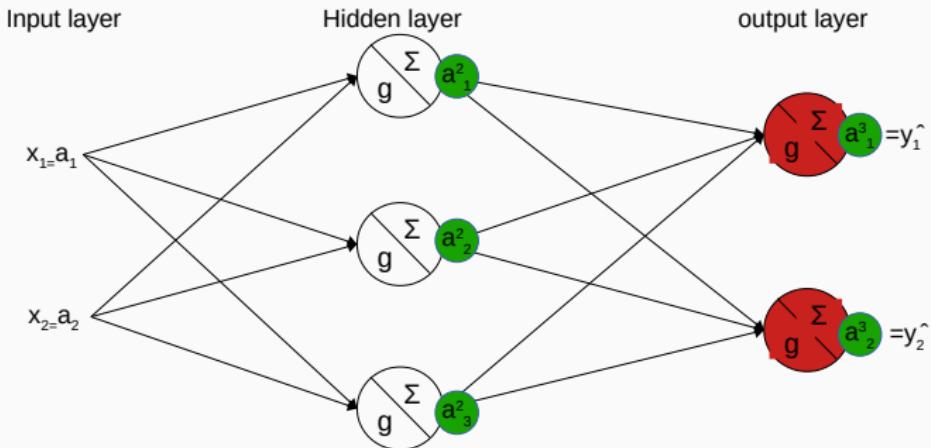
# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network

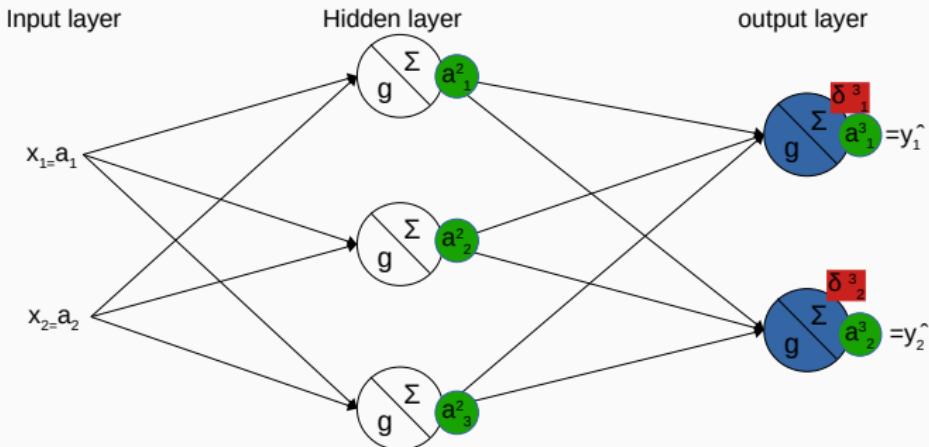


# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network

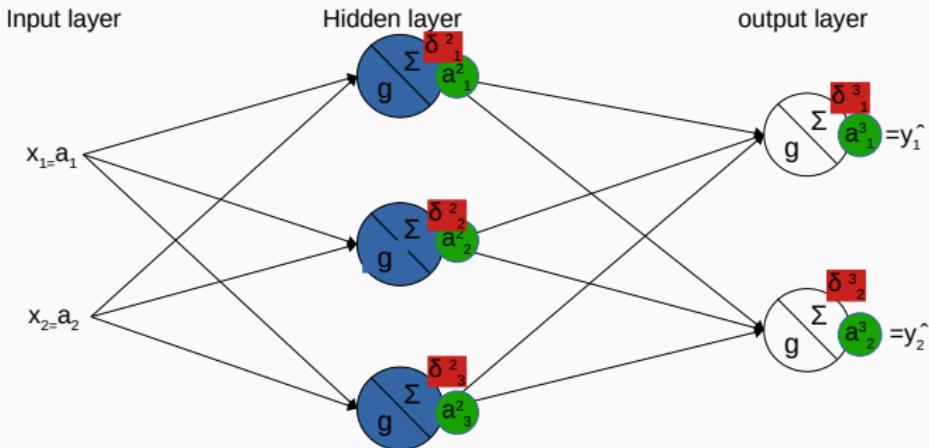
# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$



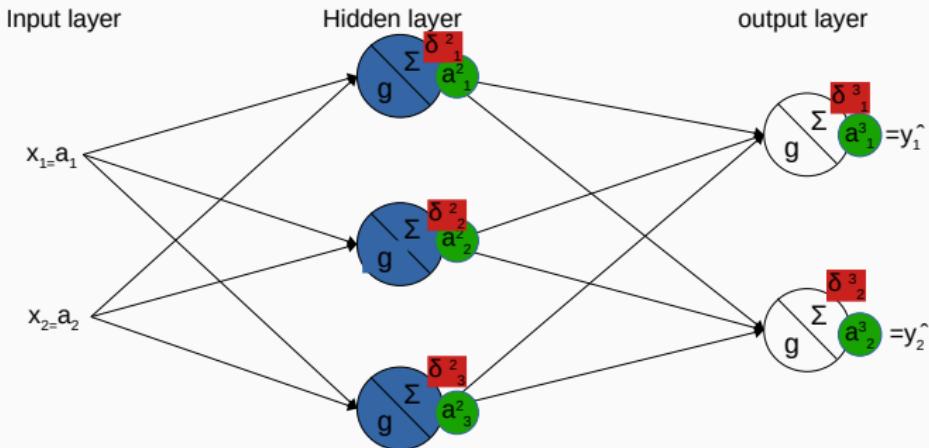
# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate **error terms** through the network



# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate **error terms** through the network
- Calculate  $\Delta\theta_{ij}^l$  for all  $\theta_{ij}^l$
- Update weights  $\theta_{ij}^l \leftarrow \theta_{ij}^l + \Delta\theta_{ij}^l$

## Interim Summary

- We recall what a MLP is
- We recall that we want to learn its parameters such that our prediction error is minimized
- We recall that gradient descent gives us a rule for updating the weights

$$\theta_i \leftarrow \theta_i + \Delta\theta_i \text{ with } \Delta\theta_i = -\eta \frac{\partial E}{\partial \theta_i}$$

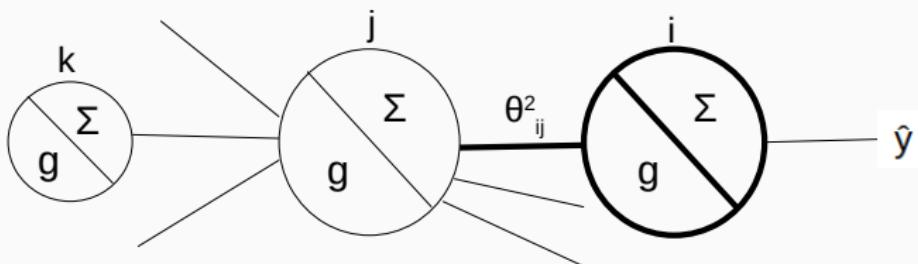
- But how do we compute  $\frac{\partial E}{\partial \theta_i}$ ?
- **Backpropagation** provides us with an **efficient way** of computing partial derivatives of the error of an MLP wrt. each individual weight.



## The (Generalized) Delta Rule

---

## Backpropagation 1: Model definition



- Assuming a sigmoid activation function, the output of neuron  $i$  (or its activation  $a_i$ ) is

$$a_i = g(z_i) = \frac{1}{1 + e^{-z_i}}$$

- And  $z_i$  is the input of all incoming activations into neuron  $i$

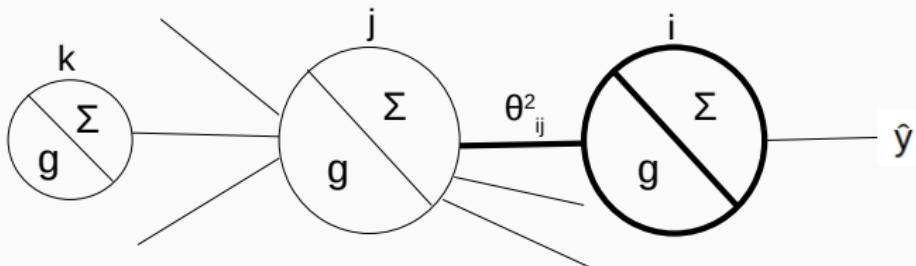
$$z_i = \sum_j \theta_{ij} a_j$$

- And Mean Squared Error (MSE) as **error function  $E$**

$$E = \frac{1}{2N} \sum_{i=1}^N (y^i - \hat{y}^i)^2$$



## Backpropagation 2: Error of the final layer

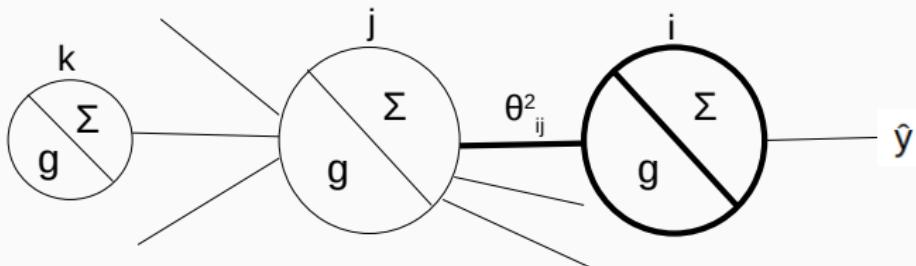


- Apply gradient descend for input  $p$  and weight  $\theta^2_{ij}$  connecting node  $j$  with node  $i$

$$\Delta\theta^2_{ij} = -\eta \frac{\partial E}{\partial \theta^2_{ij}} = \eta(y^p - \hat{y}^p)g'(z_i)a_j$$



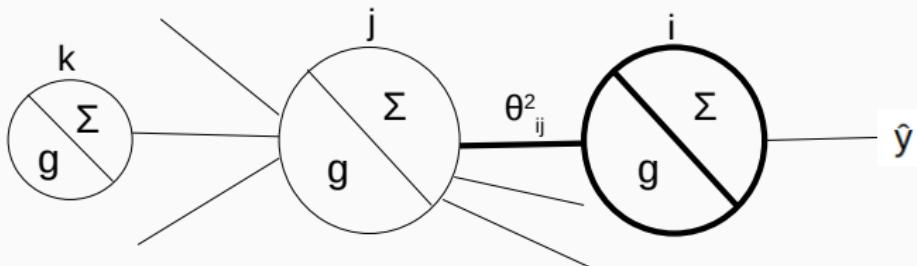
## Backpropagation 2: Error of the final layer



- Apply gradient descend for input  $p$  and weight  $\theta_{ij}^2$  connecting node  $j$  with node  $i$

$$\Delta \theta_{ij}^2 = -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta (\mathbf{y}^p - \hat{\mathbf{y}}^p) g'(z_i) a_j$$

## Backpropagation 2: Error of the final layer

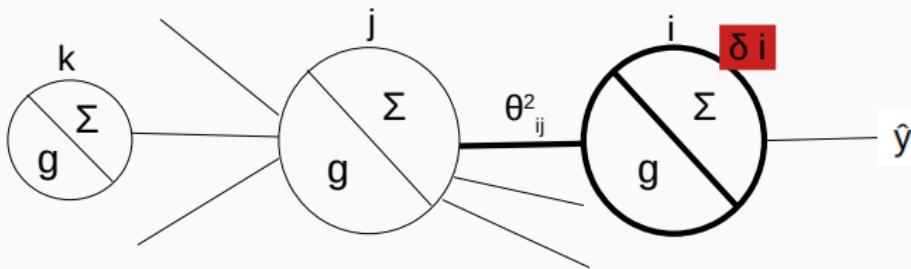


- Apply gradient descend for input  $p$  and weight  $\theta_{ij}^2$  connecting node  $j$  with node  $i$

$$\begin{aligned}\triangle \theta_{ij}^2 &= -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta (\mathbf{y}^p - \hat{\mathbf{y}}^p) g'(z_i) a_j \\ &= \eta \delta_i a_j\end{aligned}$$

- The weight update corresponds to an error term ( $\delta_i$ ) scaled by the incoming activation
- We attach a  $\delta$  to **node  $i$**

## Backpropagation: The Generalized Delta Rule



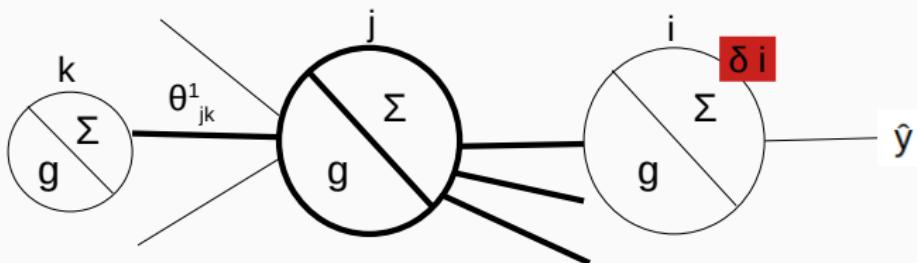
- The **Generalized Delta Rule**

$$\Delta \theta_{ij}^2 = -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta(y^p - \hat{y}^p)g'(z_i)a_j = \eta \delta_i a_j$$

$$\delta_i = (y^p - \hat{y}^p)g'(z_i)$$

- The above  $\delta_i$  can only be applied to output units, because it relies on the **target outputs**  $y^p$ .
- We do not have target outputs  $y$  for the intermediate layers

## Backpropagation: The Generalized Delta Rule

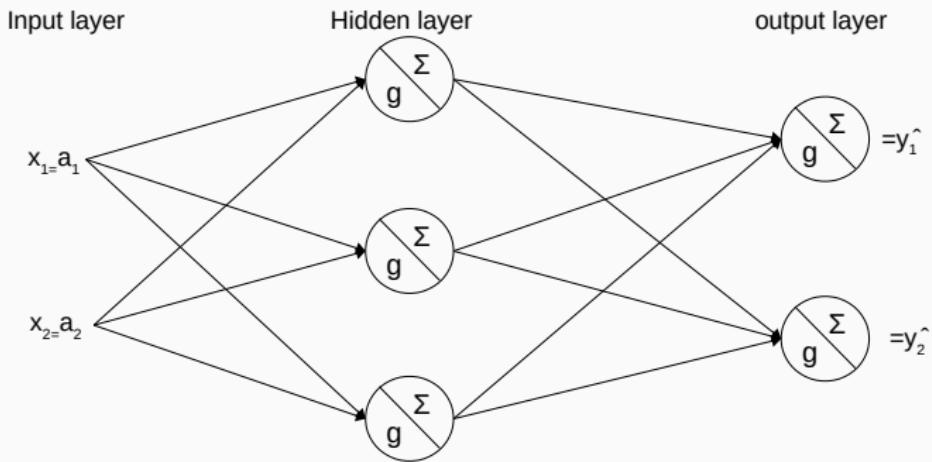


- Instead, we **backpropagate** the errors ( $\delta$ s) from right to left through the network

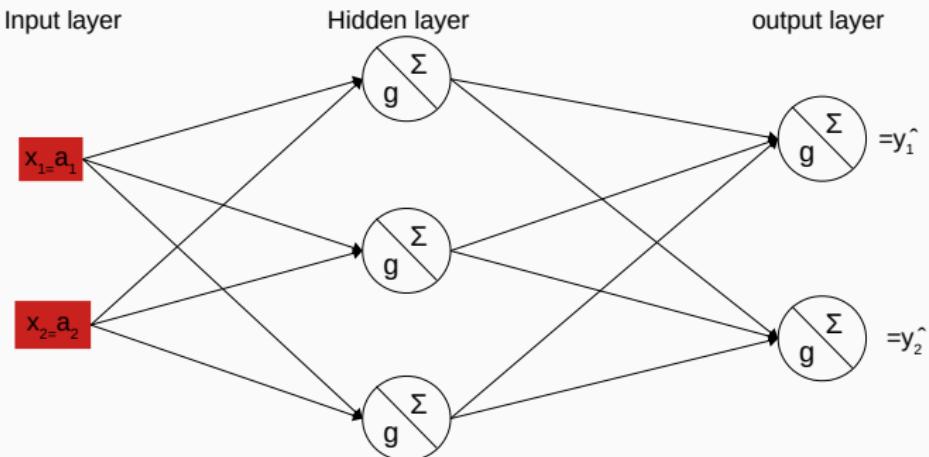
$$\Delta\theta_{jk}^1 = \eta \delta_j a_k$$

$$\delta_j = \sum_i \theta_{ij}^1 \delta_i g'(z_j)$$

# Backpropagation: Demo

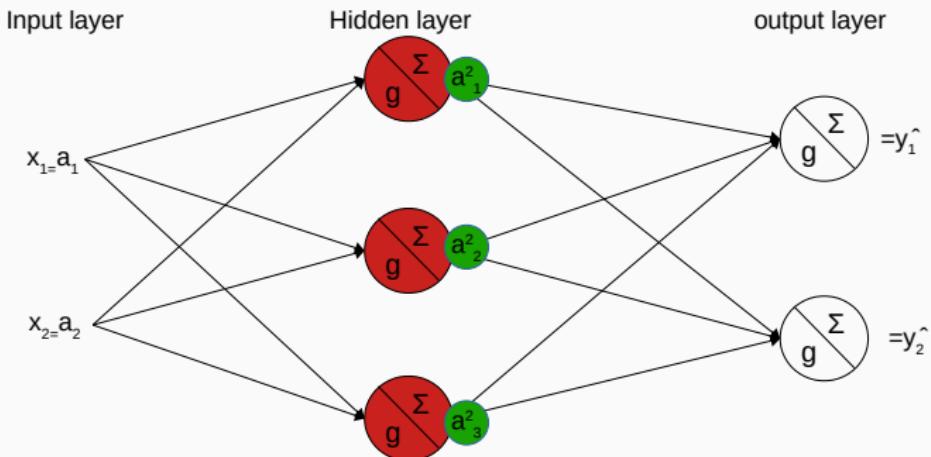


# Backpropagation: Demo



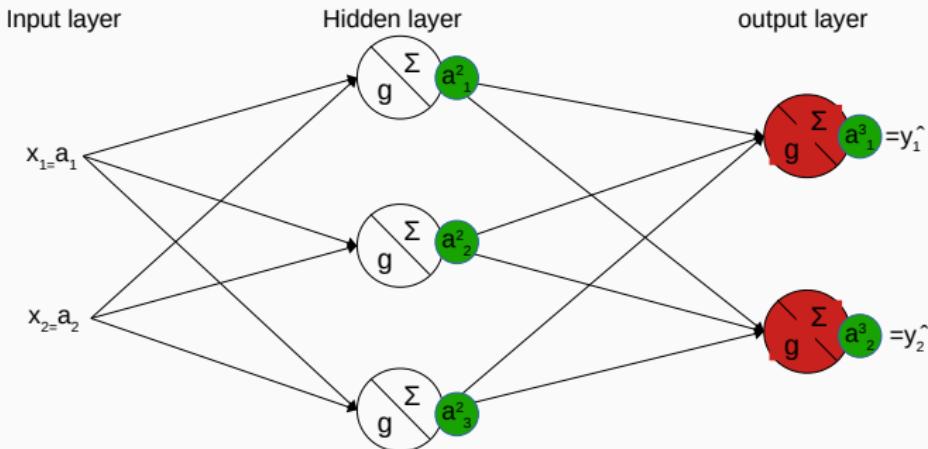
- Receive input

# Backpropagation: Demo



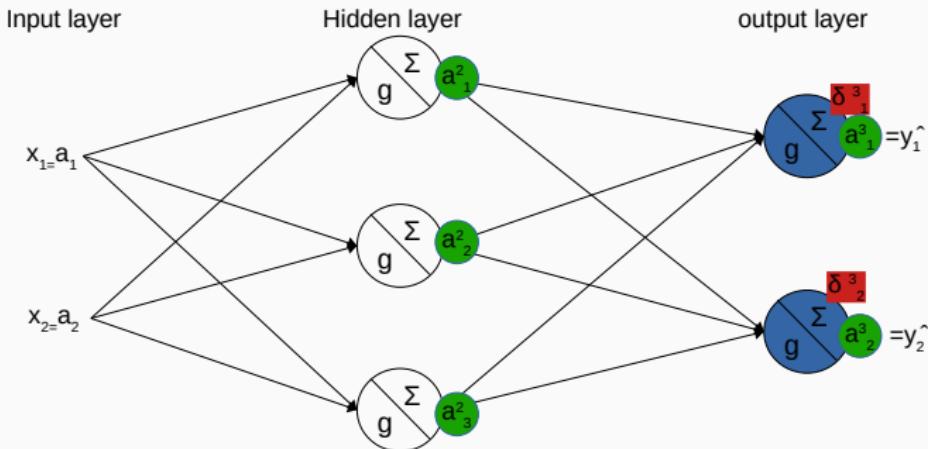
- Receive input
- Forward pass: propagate activations through the network

# Backpropagation: Demo



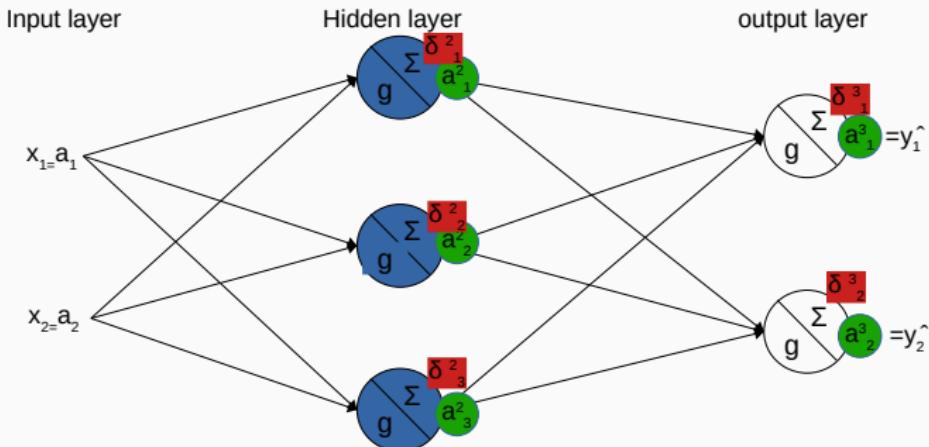
- Receive input
- Forward pass: propagate activations through the network

# Backpropagation: Demo



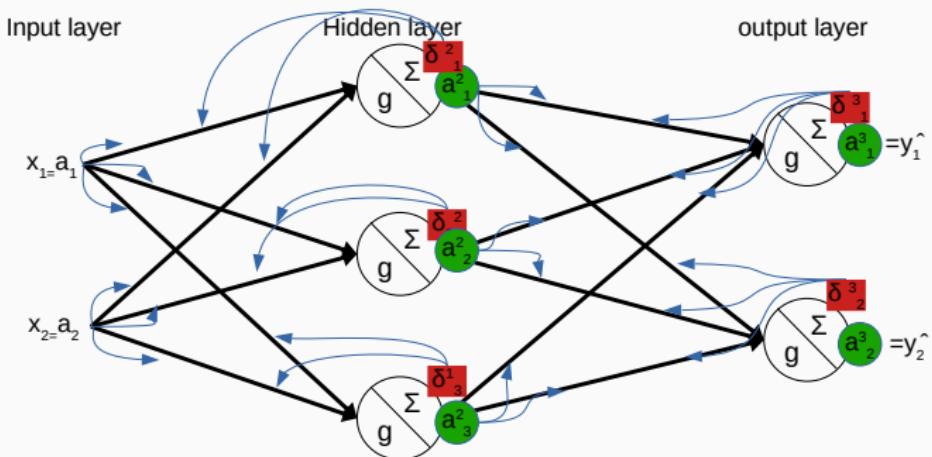
- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate error terms through the network

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate error terms through the network
- Calculate  $\frac{\partial E}{\partial \theta_{ij}^l}$  for all  $\theta_{ij}^l$
- Update weights  $\theta_{ij}^l \leftarrow \theta_{ij}^l + \Delta \theta_{ij}^l$

# 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**

3. Propagate error **back** through the network, and compute for all weights between nodes  $ij$  in all layers  $l$

$$\Delta\theta_{ij}^l = -\eta \frac{\partial E}{\partial \theta_{ij}^l} = \eta \delta_i a_j$$

4. Update **all** parameters **at once**

$$\theta_{ij}^l \leftarrow \theta_{ij}^l + \Delta\theta_{ij}^l$$

**until** stopping criteria reached.



## Derivation of the update rules

... optional slides after the next (summary) slide, for those who are interested!



## After this lecture, you be able to understand

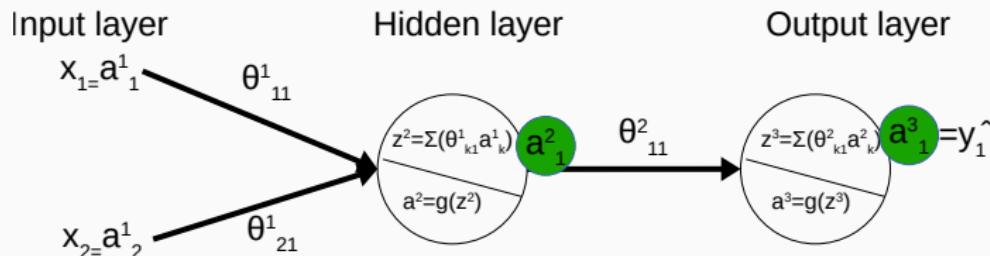
- Why estimation of the MLP parameters is difficult
- How and why we use Gradient Descent to optimize the parameters
- How Backpropagation is a special instance of gradient descent, which allows us to efficiently compute the gradients of all weights wrt. the error
- The mechanism behind gradient descent
- The mathematical justification of gradient descent

## Good job, everyone!

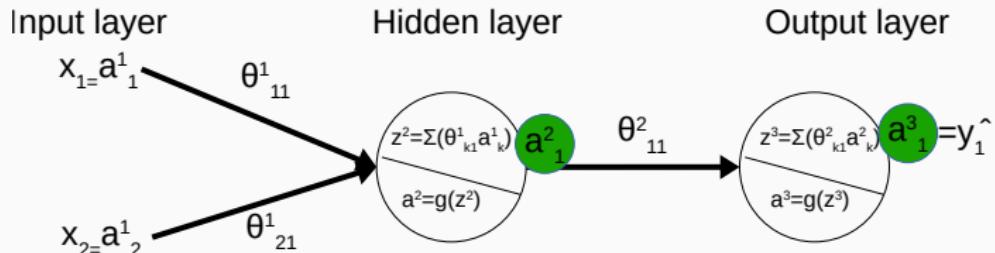
- You now know what (feed forward) neural networks are
- You now know what to consider when designing neural networks
- You now know how to estimate their parameters
- That's more than the average 'data scientist' out there!



# Backpropagation: Derivation



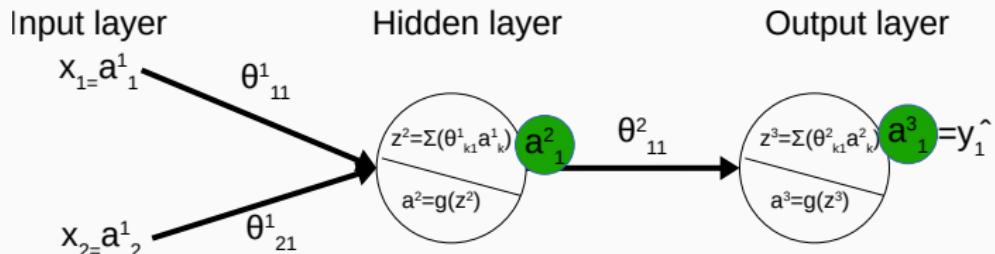
# Backpropagation: Derivation



**Chain of reactions** in the forward pass, focussing on the **output layer**

- varying  $a^2$  causes a change in  $z^3$
- varying  $z^3$  causes a change in  $a^3_1 = g(z^3)$
- varying  $a^3_1 = \hat{y}$  causes a change in  $E(y, \hat{y})$

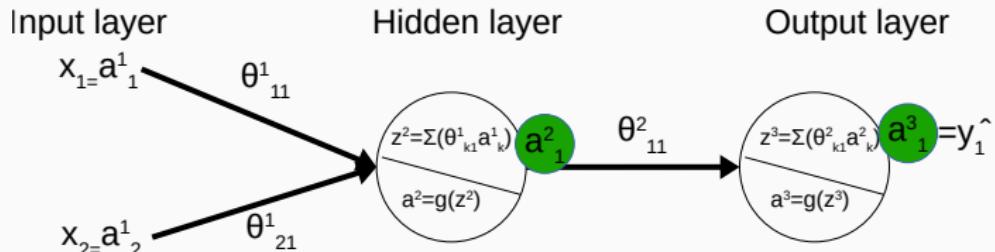
# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta \theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

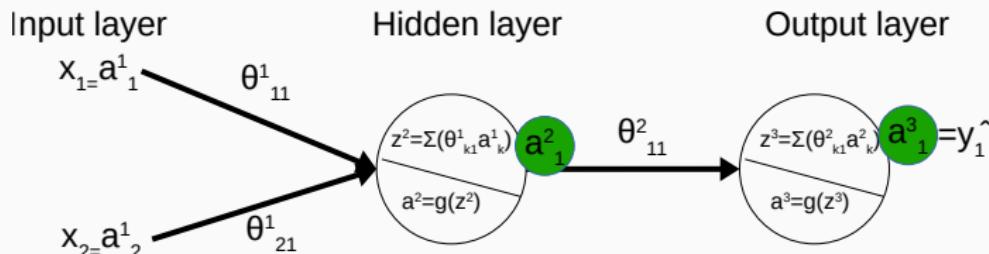
$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

Let's look at each term individually

$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$



# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

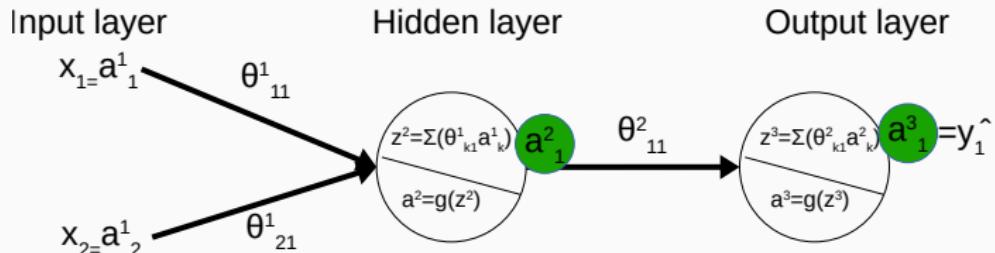
Let's look at each term individually

$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$

$$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial z} = g'(z)$$



# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta_{11}^2$  wrt  $E$

$$\Delta \theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a_1^3} \right) \left( \frac{\partial a_1^3}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

Let's look at each term individually

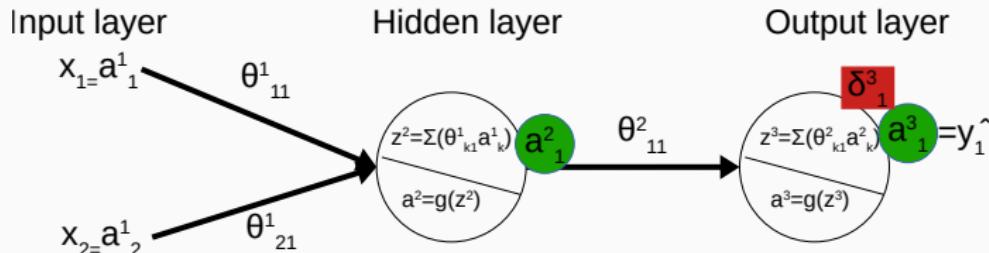
$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$

$$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial z} = g'(z)$$

$$\frac{\partial z}{\partial \theta_{ij}} = \frac{\partial}{\partial \theta_{ij}} \sum_{i'} \theta_{i'j} a_{i'} = \sum_{i'} \frac{\partial}{\partial \theta_{ij}} \theta_{i'j} a_{i'} = a_i$$



# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a_1^3} \right) \left( \frac{\partial a_1^3}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) = \underbrace{\eta \left( y - a_1^3 \right) \left( g'(z^3) \right)}_{= \delta_1^3} \left( a_1^2 \right) = \eta \delta_1^3 a_1^2$$

Let's look at each term individually

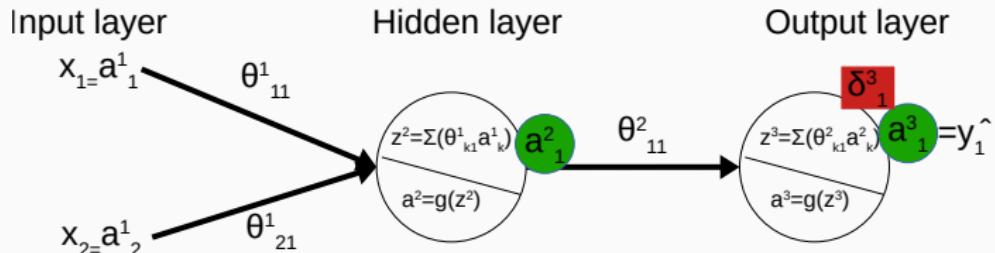
$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$

$$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial z} = g'(z)$$

$$\frac{\partial z}{\partial \theta_{ij}} = \frac{\partial}{\partial \theta_{ij}} \sum_{i'} \theta_{i'j} a_{i'} = \sum_{i'} \frac{\partial}{\partial \theta_{ij}} \theta_{i'j} a_{i'} = a_i$$



# Backpropagation: Derivation

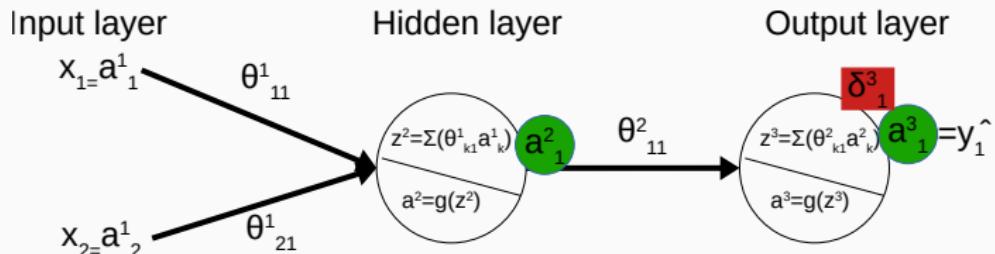


We have another **chain reaction**. Let's consider **layer 2**

- varying any  $\theta^1_{k1}$  causes a change in  $z^2$
- varying  $z^2$  causes a change in  $a^2_1 = g(z^2)$
- varying  $a^2_1$  causes a change in  $z^3$  (we consider  $\theta^2$  fixed for the moment)
- varying  $z^3$  causes a change in  $a^3_1 = g(z^3)$
- varying  $a^3_1 = \hat{y}$  causes a change in  $E(y, \hat{y})$



# Backpropagation: Derivation

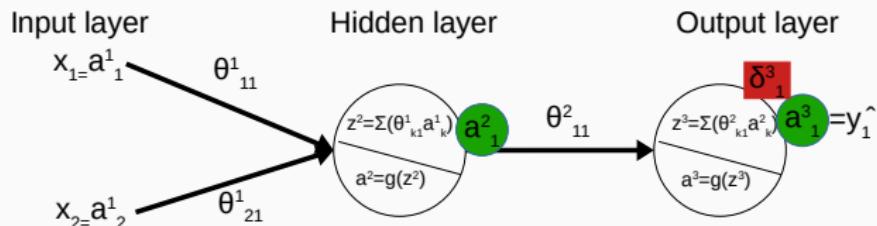


Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right) \right)$$



# Backpropagation: Derivation



Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right) \right)$$

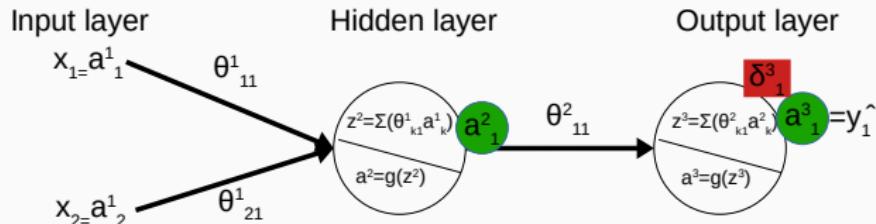
We already know that

$$\frac{\partial E}{\partial a^3_1} = -(y - a^3_1)$$

$$\frac{\partial a^3_1}{\partial z^3} = g'(z^3)$$



# Backpropagation: Derivation



Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right) \right)$$

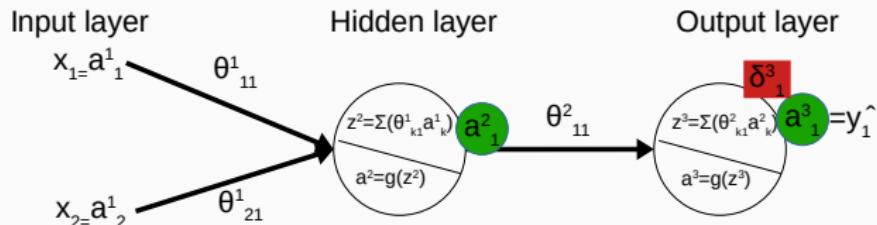
And following the previous logic, we can calculate that

$$\frac{\partial z^3}{\partial a^2_1} = \frac{\partial \theta^2 a^2_1}{\partial a^2_1} = \theta^2$$

$$\frac{\partial a^2_1}{\partial z^2} = \frac{\partial g(z^2)}{\partial z^2} = g'(z^2) \qquad \qquad \frac{\partial z^2}{\partial \theta^1_{k1}} = a_k$$



# Backpropagation: Derivation



Formulating this again as the chain rule

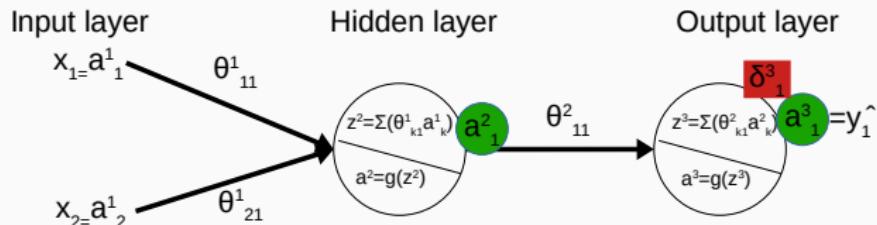
$$\Delta\theta_{k1}^1 = -\eta \frac{\partial E}{\partial \theta_{k1}^1} = -\eta \left( \left( \frac{\partial E}{\partial a_1^3} \right) \left( \frac{\partial a_1^3}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a_1^2} \right) \right) \left( \frac{\partial a_1^2}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta_{k1}^1} \right)$$

Plugging these into the above we get

$$-\eta \frac{\partial E}{\partial \theta_{k1}^1} = -\eta \left( -(y - a_1^3)g'(z^3)\theta^2 \right) g'(z^2)a_k$$



# Backpropagation: Derivation



Formulating this again as the chain rule

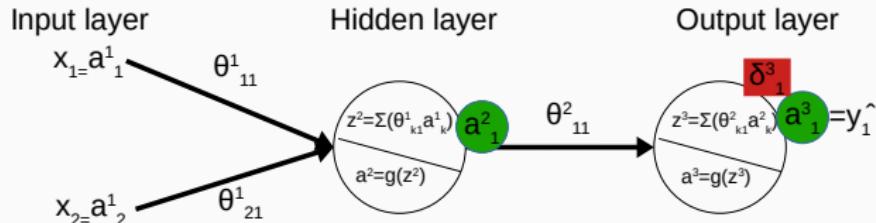
$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right) \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( -(y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \left( (y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

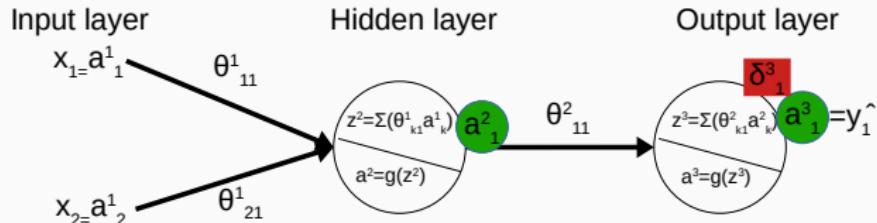
$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( -(y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \underbrace{\left( (y - a^3_1) g'(z^3) \theta^2 \right)}_{= \delta^3_1} g'(z^2) a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

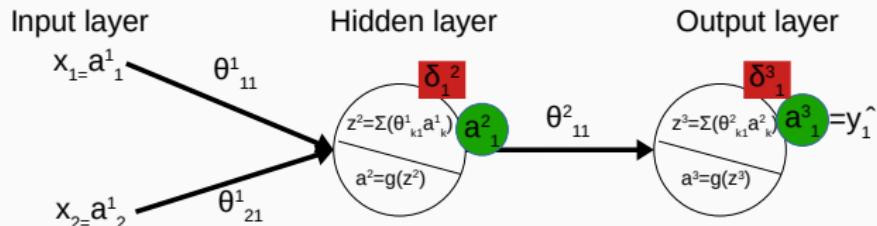
$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( -(y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \underbrace{\left( (y - a^3_1) g'(z^3) \theta^2 \right)}_{= \delta^3_1} g'(z^2) a_k = \eta (\delta^3_1 \theta^2) g'(z^2) a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

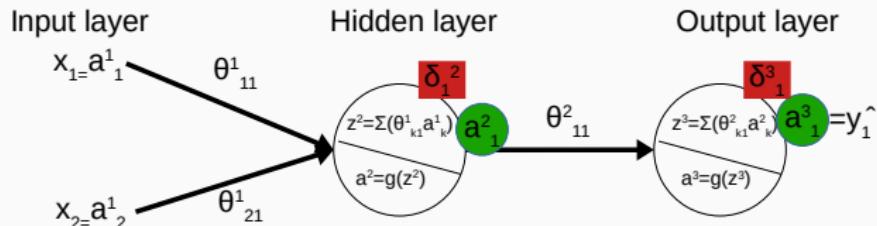
$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( -(y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \underbrace{\left( (y - a^3_1) g'(z^3) \theta^2 \right)}_{= \delta^3_1} g'(z^2) a_k = \eta \underbrace{\left( \delta^3_1 \theta^2 \right)}_{= \delta^2_1} g'(z^2) a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

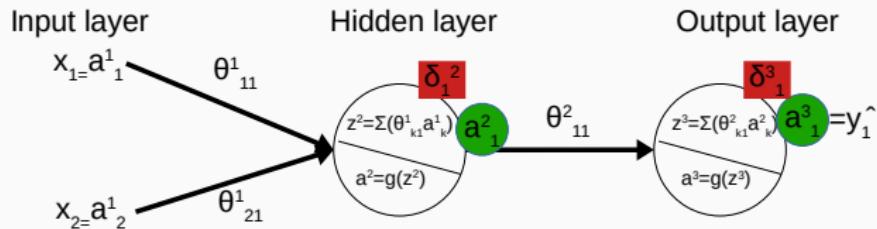
$$\Delta \theta_{k1}^1 = -\eta \frac{\partial E}{\partial \theta_{k1}^1} = -\eta \left( \left( \frac{\partial E}{\partial a_1^3} \right) \left( \frac{\partial a_1^3}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a_1^2} \right) \right) \left( \frac{\partial a_1^2}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta_{k1}^1} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta_{k1}^1} &= -\eta \left( -(y - a_1^3) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \underbrace{\left( (y - a_1^3) g'(z^3) \theta^2 \right)}_{= \delta_1^3} g'(z^2) a_k = \eta \underbrace{\left( \delta_1^3 \theta^2 \right)}_{= \delta_1^2} g'(z^2) a_k = \eta \delta_1^2 a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

$$-\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right) \right)$$

If we had more than one weight  $\theta^2$

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= \eta \left( \sum_j \underbrace{(y_j - a^3_1) g'(z^3_j) \theta^2_{1j}}_{= \delta^3_j} \right) g'(z^2) a_k \\ &= \eta \left( \underbrace{\sum_j \delta^3_j \theta^2_{1j}}_{= \delta^2_1} \right) g'(z^2) a_k = \eta \delta^2_1 a_k \end{aligned}$$



# Lecture 13: Evaluation Part 2

---

**COMP90049**

Semester 1, 2021

Lea Frermann, 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



## So far ...

- Intuition, maths, and application of different classification models of varying complexity
- Feature selection
- Evaluation: How well are we doing?

## Today... Evaluation part II

- How do we know whether the model performs ‘good enough’? When to stop/continue model training, parameter tuning or model selection?
- Types of poor model performance
- Diagnosing poor model performance



## Evaluation

---

# Evaluation I

Given a dataset of instances comprising of attributes and labels:

- We use a learner and the dataset to build a classifier
- We assess the effectiveness of the classifier
  - Generally, by comparing its predictions with the actual labels on some unseen instances
  - Metrics: accuracy, precision, recall, Error rate, F-score, etc.



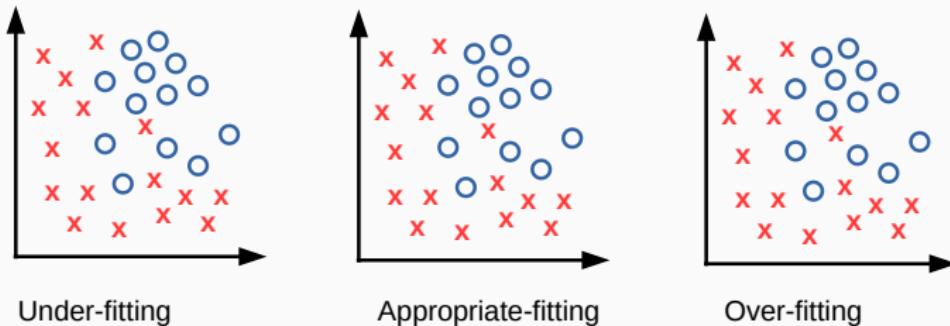
## Tensions in Classification

- **Generalisation:** how well does the classifier generalise from the specifics of the training examples to predict the target function?
- **Overfitting:** has the classifier tuned itself to the idiosyncrasies of the training data rather than learning its generalisable properties?
- **Consistency:** is the classifier able to flawlessly predict the class of all training instances?



# Generalisation Problem in Classification

- **Under-fitting:** model not expressive enough to capture patterns in the data.
- **Over-fitting:** model too complicated; capture noise in the data.
- **Appropriate-fitting** model captures essential patterns in the data.



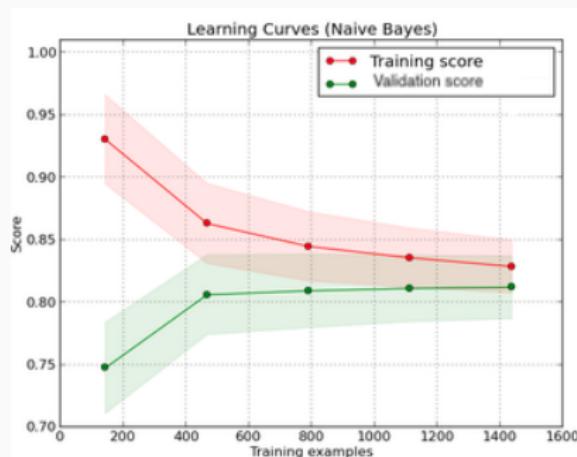
## **Learning Curve**

---

# Learning Curve I

Learning curve is a plot of learning performance over experience or time

- y-axis: performance measured by an evaluation metric (F-score, precision, ...)
- x-axis: different conditions, e.g. sizes of training dataset, model complexity, number of iterations etc.



Plot on the left

- Learner: Naive Bayes
- What can we say about the difficulty of the problem?



## Learning Curve II

- Holdout (and cross-validation, to a lesser extent), is based on dividing the data into two (three?) parts:
  - Training set, which we use to build a model
  - Evaluation set (“validation data”, “test data”), which we use to assess the effectiveness of that model
- More training instances → (usually) better model
- More evaluation instances → more reliable estimate of effectiveness



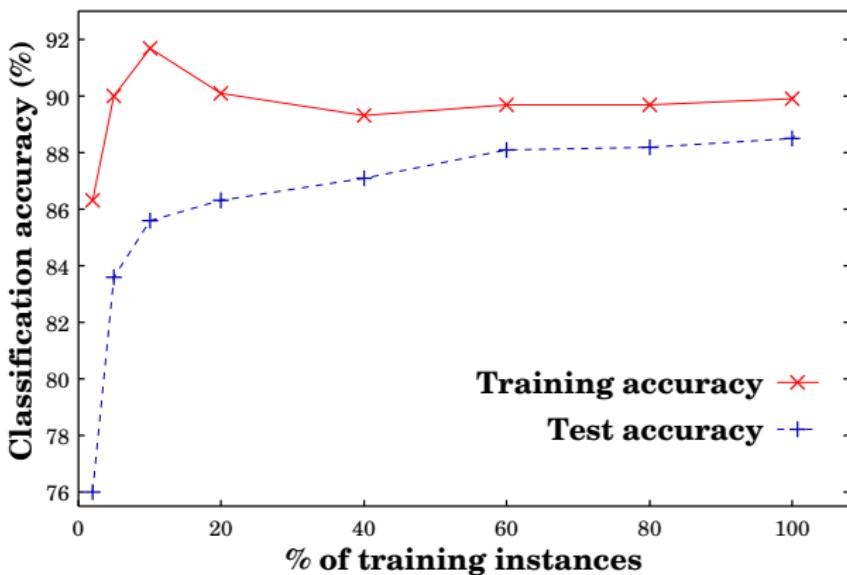
## Learning Curve III

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)
- This allows us to visualise the data trade-off

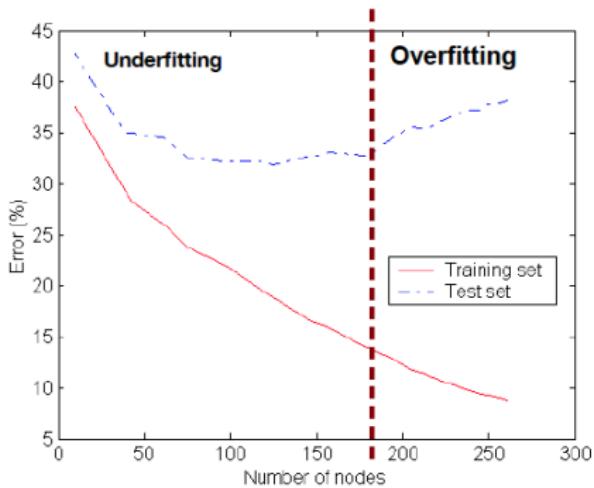


## Learning Curve IV



# Overfitting and Underfitting

- **Underfitting:** when model is too simple → both training and test errors are large
- **Overfitting:** when model is too complex → training error is small and test error is large

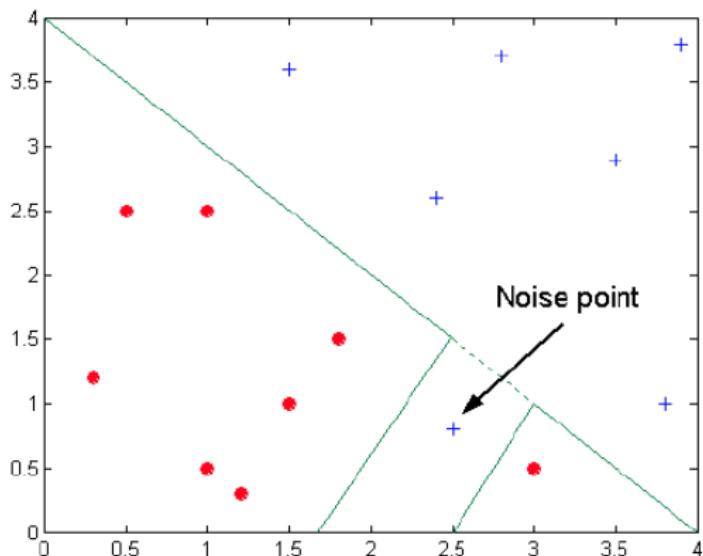


- What would be a good model complexity?

# Overfitting

Overfitting due to noise:

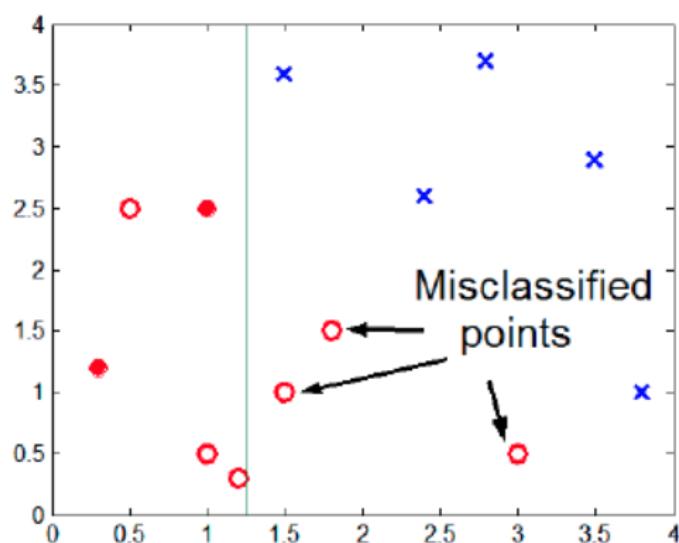
- The decision boundary is distorted by noise



# Overfitting

Overfitting due to insufficient training instances

- The data points do not fully represent the patterns in the dataset



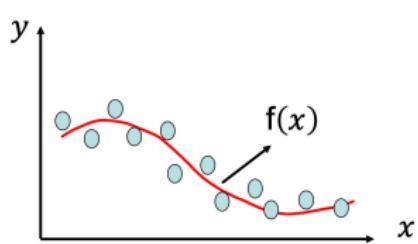
## **Generalization**

---

# Generalization

- A good model generalizes well to unseen data!
- How do we measure the generalizability of a model ?
- Given a training dataset  $D = \{x_i, y_i\}, i = 1 \dots n$  and  $y \in \mathbb{R}$ :
  - Assume the data points are generated with a function  $f(\cdot)$  plus a noise  $\epsilon \in \mathcal{N}(0, \sigma)$ . This noise comes from an unknown and unmeasurable source, e.g., annotation error, measure error:

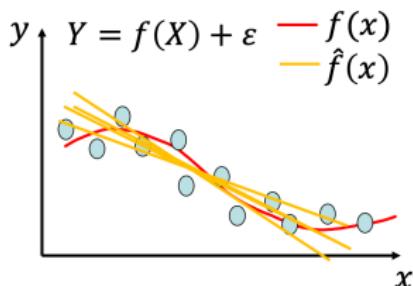
$$Y = f(X) + \epsilon$$



## Generalization Error I

- We may estimate a model  $\hat{f}(X)$  of  $f(X)$  using linear regression or another modelling technique
- **But** different training sets  $\rightarrow$  different model weights and outputs
- To remove the dependency  $\rightarrow$  repeat modelling many times (on different training sets)
- In this case, the expected squared prediction error at a point  $x$  is:

$$Err(x) = E \left[ (Y - \hat{f}(x))^2 \right]$$



## Generalization Error II

- The generalization error can be decomposed to:

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E \left[ \left( \hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$

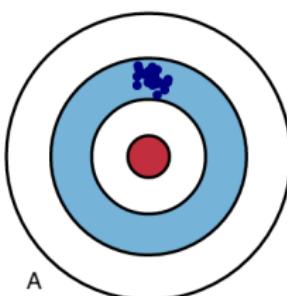
- Or simply written as:

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

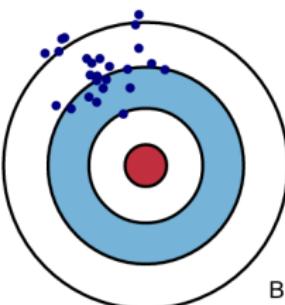
- Variance:** Captures how much your model changes if you train on a different training set. How "over-specialized" is your classifier to a particular training set?
- Bias:** What is the inherent error that you obtain from your model even with infinite training data? This is due to your model being "biased" to a particular kind of solution. In other words, bias is inherent to your model.
- Noise:** This error measures ambiguity due to your data distribution and feature representation. You can never beat this, it is an aspect of the data.

## Generalization Error III

- Which one has lower variance:

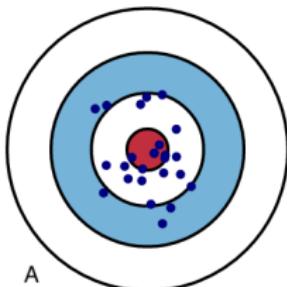


A

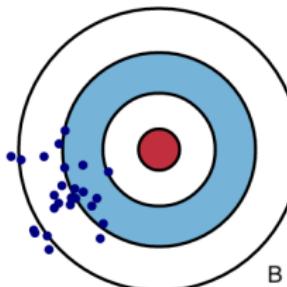


B

- Which one has lower bias:



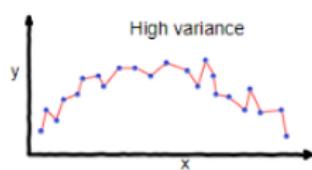
A



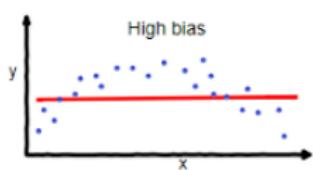
B

## Generalization Error VI

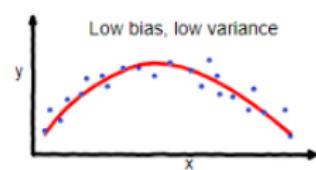
- Causes of Poor Generalization:
  - Underfitting: Variance is zero and bias is large
  - Overfitting: bias is zero and variance is substantial
- A Good model
  - Lower bias and lower variance → better generalisation



overfitting



underfitting



Good balance



## Quiz (via Zoom)

**which baseline has lower variance?**

1. weighted random classifier
2. 0-R (majority voting)

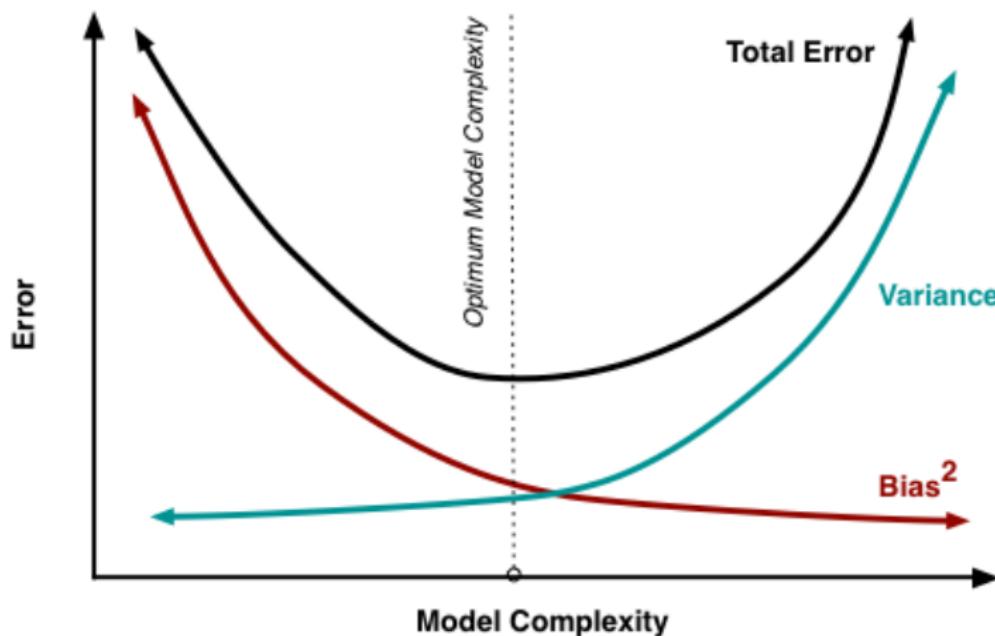


## **Diagnosing High Bias and Variance**

---

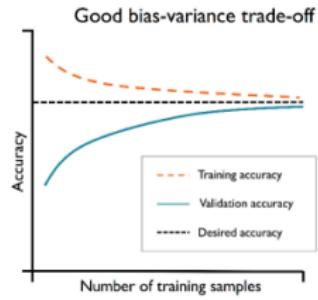
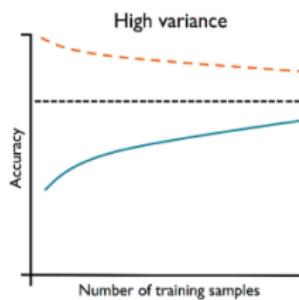
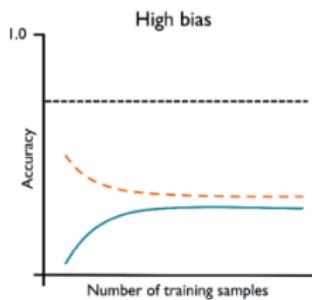
## Bias-Variance Tradeoff

At its root, dealing with bias and variance is really about dealing with overfitting and underfitting. Bias is reduced and variance is increased in relation to model complexity.



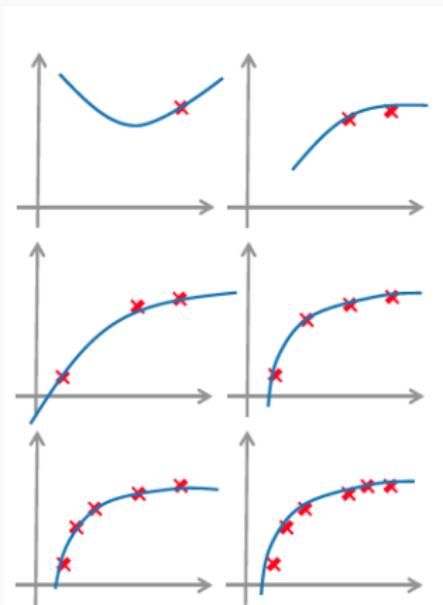
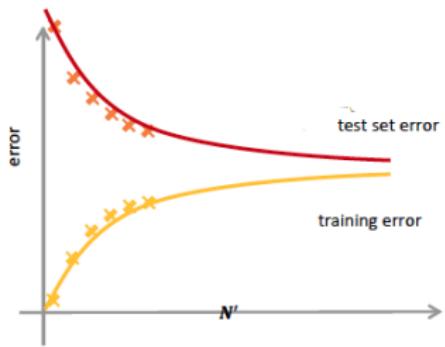
# Diagnose Overfitting and Underfitting I

- Plot Training and Test Error as function of data size
- The following situations may occur:



## Diagnose Overfitting and Underfitting II

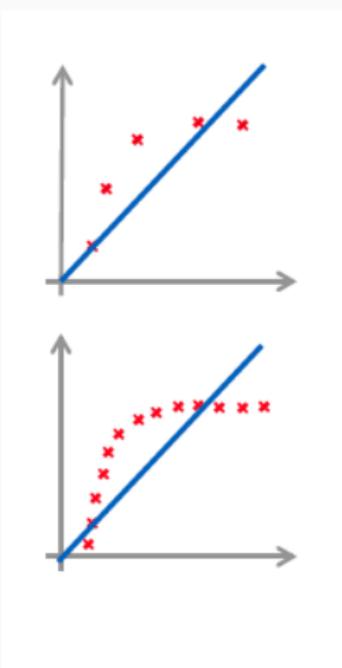
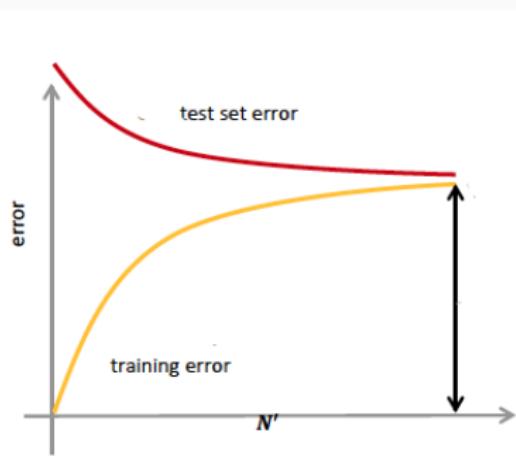
- Fitting a quadratic regression function to data:  
$$h(x : \theta) = \theta_0 + \theta_1 x + \theta_2 x^2$$
- Plot training and test errors vs. training set size  $N' = 1, 2, 3 \dots n$



# Diagnose Overfitting and Underfitting III

## High Bias

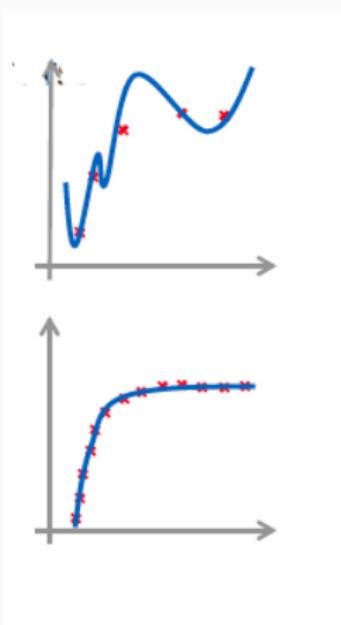
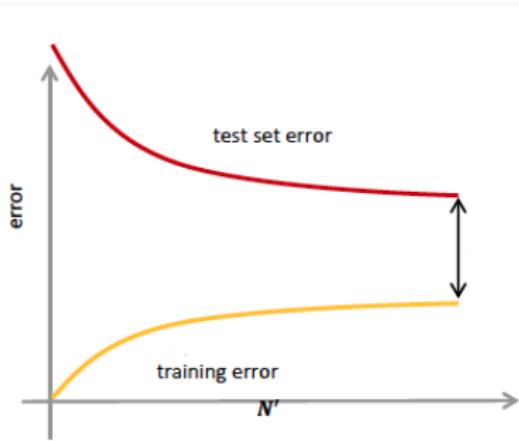
- Getting more training data will not (by itself) help much
- Learning curve is characterized by high training and test errors



# Diagnose Overfitting and Underfitting VI

## High Variance

- Getting more training data is likely to help
- Learning curve is characterized by gap between the two errors



## **Remedy for High Bias and Variance**

---

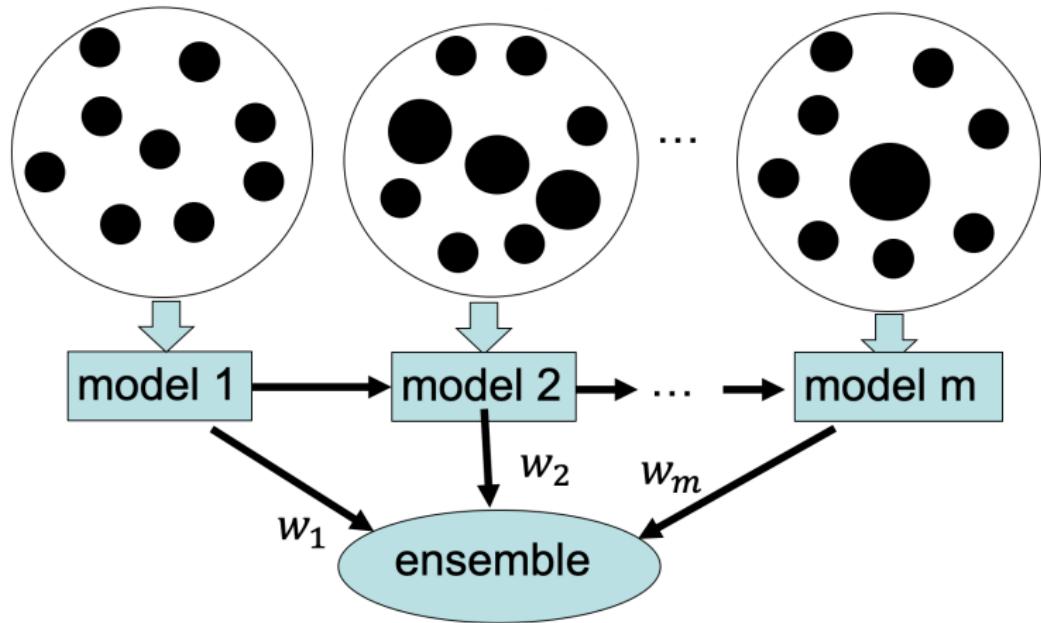
# High Bias Remedy

- 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)



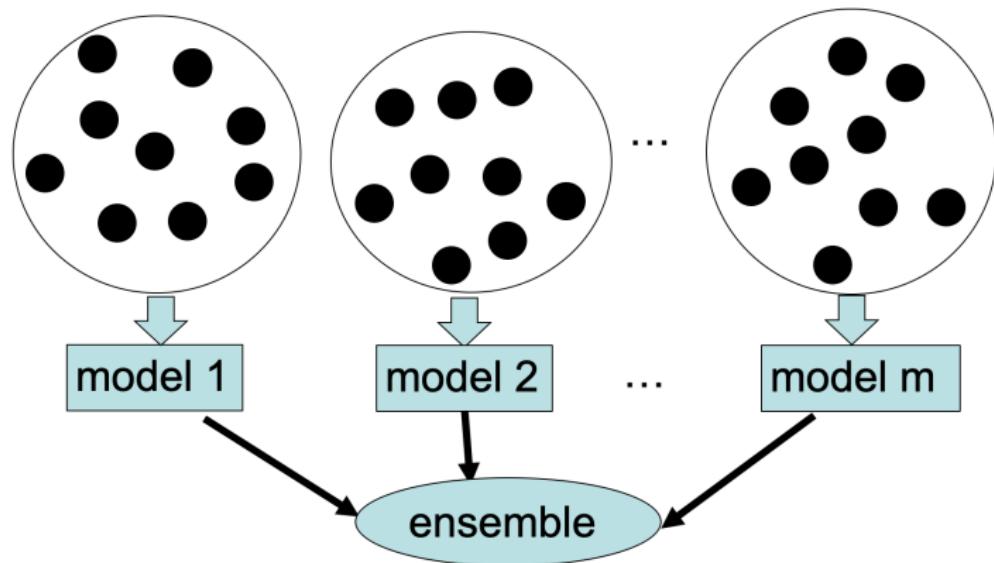
## High Variance Remedy

- 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**

---

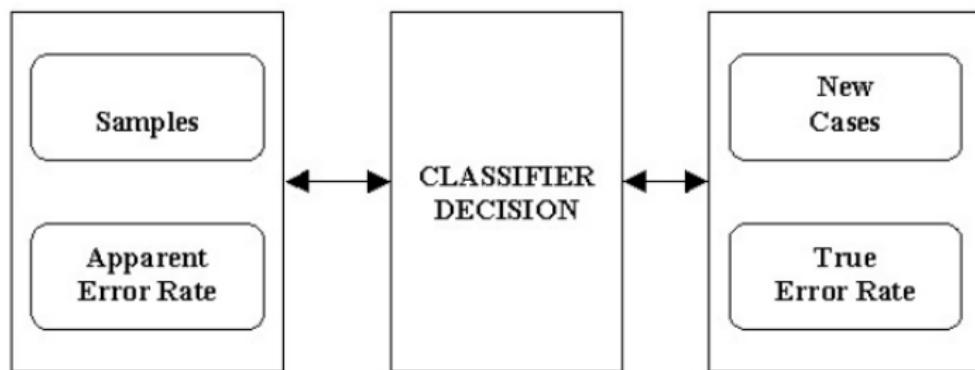
## Evaluation Bias and Variance I

- Our evaluation metric is also an estimator
- Desire to know the ‘true’ error rate of a classifier, but only have an estimate of the error rate, subject to some particular set of evaluation instances
- The quality of the estimation is independent of the trained model



## Evaluation Bias and Variance II

- We extrapolate performance from a finite sample of cases.
- Training error is one starting point in estimating the performance of a classifier on new cases.
- With unlimited samples, apparent error rate will become the true error rate eventually.



## Evaluation Bias and Variance III

- What are the potential problems with our estimated error rate?
  - We have good accuracy with respect to some specific evaluation set, but poor accuracy with respect to other unseen evaluation sets
  - It's also possible to overfit the validation data, with respect to our evaluation function



## Evaluation Bias and Variance VI

- 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
  - **Evaluation Bias:** Our estimate of the effectiveness of a model is systematically too high/low
  - **Evaluation Variance:** Our estimate of the effectiveness of a model changes a lot, as we alter the instances in the evaluation set (very hard to distinguish from model variance)



## Evaluation Bias and Variance V

How do we control bias and variance in evaluation?

- Holdout partition size
  - More training data: less model variance, more evaluation variance
  - Less training (more test) data: more model variance, less evaluation variance
- Repeated random subsampling and K-fold Cross-Validation
  - Less variance than Holdout for model and evaluation
- Stratification
  - less model and evaluation bias
- Leave-one-out Cross-Validation
  - No sampling bias, lowest bias/variance in general



## **Summary**

---

## Today... Evaluation part II

- What is generalization?
- How are underfitting and overfitting different?
- How are bias and variance different?
- What is a learning curve, and why is it useful?
- 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?

## Next week

- Guest lecture on academic writing (Assignment 3!)
- Decision Trees & Ensembling



## 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



# **COMP90049 Introduction to Machine Learning Report**



**academic-skills@unimelb.edu.au**

**Academic Skills**

**What are your  
academic  
writing pain  
points?**

**Task requires  
you to 'critically  
assess'?**

**What does it  
mean to be  
'critical'?**



THE UNIVERSITY OF  
**MELBOURNE**

INTELLIGENTIAE

## TIP: Use a process to produce work

### 1. Determine the task type

What *type* of writing is it?

Essay

Report

What is the structure?

Lit Review

What am I expected to do?

### 2. Carefully analyse the task

Highlight key information

TIP: Make sure this highlighted language appears in your response

# The task

**The goal:** critically assess the effectiveness of various Machine Learning classification algorithms on the problem of determining a tweeter's location, and to express the knowledge that you have gained in a technical report.

Anonymised report of 2000 words in length (+/-10%), including in-text references

**Introduction:** context and position

**Literature review:** a short summary of some *related* (relevant) literature, including the data set reference and at least two additional relevant research papers of your choice

**Method:** Identify the newly engineered feature(s), and the rationale behind including them

**Results:** in terms of evaluation metric(s) and examples

**Discussion / Critical Analysis:** 2 areas – *Contextualise* the system's behavior, i.e. reasons for the relative performance of different methods & *Discuss* any ethical issues

**Conclusion:** demonstrate identified knowledge

**Bibliography:** incl. Eisenstein et al. (2010) + other related work (min 2) – APA recommended

Task type

2: Topic / function

What do others say?

How did I do this?

What did I find?

What does it mean?

What do I now know?

Evidence

INTELLIGENTIAE

## 3. Plan / organise ideas

Based on analysis generate a ‘sectioned plan’

On your computer

Allocate word counts

**TIP: method, results, discussion substantive**

- Definition of blah (Williams p44)

-

- Good idea (Long p15)

-

- Great thought (Ng p1)

-

- Possible idea (Ng p3)

-

## 4. Research

Find info and read it

*As you read, put bullet points in your plan*

**TIP:** Research using

<https://unimelb.libguides.com/>

## 5. Draft – start writing

Start anywhere in the body

Start where you feel confident

**TIP:** write Intro / Conclusion last (5% of words)

Intro

- Definition of Blah (Williams p44)
- 
- 

- Good idea (Long p15)
- 
- 

- Great thought (Ng p1)
- 
- 

- Possible idea (Ng p3)
- 
- 

Conclusion

## 6. Finalise *then* submit

*Do final check on hard copy*

Read out loud

Know by when & how to submit

**Task requires you to ‘critically assess’?  
What does it mean to be ‘critical’ in an  
academic context?**

# Two key elements to critical literacy

## 1. A reporting / describing element:      The ‘catalogue’

You find information about the topic and **report, describe** what it says, what they did, found, claim etc.

This frames the second part: important but don’t stop here

## 2. An interpretive / response element:      The ‘dialogue’

**You interpret and respond to** what you have read –  
***critically engage with*** the readings and the topic



INTELIGENTIAE

## What are some reflective questions?

What happened? What do I/we know? (describing, assessing, context)

What has changed? (gap btw expectation and actual)

What worked? **Why?** (strengths)

What didn't? **Why?** (limitations)

What have I learned? (understandings)

What might we do differently? (learning)

How can we get there? (development)

What do I know now? (conclusions)

# Key points

Think about how information is ordered within a paragraph

How does each sentence link to the one before it *and* after it

– **how** are you showing this?

Is the sequence logical?

Same for paragraphs – how does the paragraph relate to the next one?



# Cohesion - linking

2 types of language in writing:

**Content** – from reading and learning, e.g. *tweet, machine, baseline,*

**Functional** – cohesive-linking-highlighting

*Therefore, however, first, next, for example, though, and, which*

*This is important because ... This shows that... This tells us ...*

<https://m.eliteediting.com.au/50-linking-words-to-use-in-academic-writing/>

<http://www.phrasebank.manchester.ac.uk/>

# Interpretive language

This shows\* that ... (\*suggests / implies / gives the impression that ...)

This is important / significant because...

This is worth noting as / because it ...

This calls attention to ...

This can be illustrated by ...

What this means\* is ... (\*shows / tells us / reveals / highlights / points to / implies)

... tells us that ...

... importantly\* suggests that ... (\*crucially, significantly)

... which points to / suggests the need for ...

... which is vital / crucial as it ...

... which shows / illustrates that ...

... which is significant as it ...

... is illustrative because it ...

... meaning that ...

... illustrating / pointing to the need for ...

In doing so, it points to ... / In so doing, tells us that ...

Use this language!  
It moves your  
thinking from the  
descriptive to the  
interpretive

# Report writing Tips

Be on task – show you are doing so with **key language**

*This paper explores / examines / identifies... (intent / position) - TIP  
frame this in present simple*

*This suggested that... (discussion / analysis)*

*Having carried out ... we conclude / find that ... (concluding)*

Use numbered **headings**

Be aware of the **functionality** of the report sections: make relationships clear to current work

Be **interpretive** and **analytical**, not just descriptive, of both text *and* data

**Clear graphics** – simple is best; refer to the graphics

# Report writing Tips

## Edit *before* submitting – looking for

**Formal language:** full forms (*didn't* vs *did not*); avoid emotive language  
'frustrating', 'disappointing', 'obvious', 'good', 'bad'

Be aware of **tense** : time ... a second experiment *is* was performed.

## **Tense use TIPs:**

**Past tense** for finished action; *did, found, discovered, proved, showed*

**Present simple** for fact, current observation or current feeling; '... it refers to ... it signals that ... we can see that ... it shows...

Lee (2010) **proposes** that this is ...

# Cohesion – consider link & transition between ideas

## Short, join:

*A majority class baseline was used for this experiment. It is based on the ‘Zero Rule’. This rule classifies all tweets according to labels with the greatest training set ratio.*

*A majority class baseline was used for this experiment **which** is based on the ‘Zero Rule’ **classifying** all tweets according to labels with the greatest training set ratio. (28w) OR*

*The majority class baseline used for this experiment was based on the ‘Zero Rule’ **classifying** all tweets according to labels with the greatest training set ratio. (26W)*

## Long; cut:

*By analysing the training set, it was somewhat surprising to find a number of feature values equal to 0, especially as many samples have 0-value for all their attributes, which means that none of the feature terms ever occurs in them. (41w)*

*Analysing the training set, we found a somewhat surprising number of feature values equal to 0, especially as many samples have 0-value for all their attributes. This means that none of the feature terms ever occurs in them.*

# The impact of sentence length

If the sentences are **short and related**, then join them

Less than 8 words in the sentence <b>(1 line in a Word doc)</b>	Very short sentence	OK, but don't use too many of these; writing can appear short and choppy, hard to read.
8-15 words <b>(1 - 1.5 lines)</b>	Short	✓ OK combined with 15-25.
15-25 words <b>(2 - 3 lines)</b>	Average number per sentence.	✓✓ This length will form the majority of your sentences.
25-35 words <b>(3 - 4 lines)</b>	OK, but becoming long.	✓ Can be effective if the point is worth making in a single sentence; make sure you have control over the idea(s) though and use appropriate connecting / linking words.
35-45 words <b>(4 - 5 lines)</b>	Long	Consider breaking up the idea(s) into two or more shorter sentences.
More than 45 words <b>(+ 5 lines in a Word doc)</b>	Too long	✗ Avoid this; the point gets lost, control over the language is lessened and chances for errors in form and logic increase.

If the sentence is +40 words (not a list), consider breaking it up

# Make time to edit

**Editing attitude: task focus - Do I/we *need* it? Is it relevant? No? get rid of it**

**Big edits:** Possible removal of whole sections, paragraphs or sentences

**Small edits:** word, sentence, text level

Correctness, flow-link of ideas, expression

Trim or compression editing:

... which enabled us to make clear recommendations (7 words) vs

... **which enabled clear recommendations ... (4 words)** OR

... **enabling clear recommendations (3 words)**

Reflexive repetition: removal of unnecessary repetition

This **was evidence of** the program's **three features. These features** address ... vs

This **evidenced** the program's three features **which** address ...

## TIPS:

Edit when fresh – get distance from paper

Final edit on hard copy

Read aloud – you or MS Word Read Aloud function

**“Direct quotations”** – Author’s ***exact*** words – *do not overuse*

Cheng et al. (2010) proposed a user location framework, “based purely on the content of the user's tweets, even in the absence of any other geospatial cues” (p. 759).

**Paraphrasing** (Indirect quoting) or summarising  
– present the idea *in your words* – still need to cite



A framework was proposed whereby location of the user was estimated solely on tweet content even where geospatial data is absent (Cheng et al., 2010).

## NOTES:

Author vs idea focus in citations

Citation is a writing skill; making a Reference list is a technical skill

# APA tips

Always need the year with the author(s) in text, not the initials (that's for the Ref List)

Avoid starting or finishing a paragraph with a citation – **try to start and finish on your words**

Et al. (three or more authors, shortened from the first citation)

- always takes a full stop in *or* out of the brackets
- always a plural; it means ‘and others’, e.g. Letts et al. (2015) argue that ... (not ‘~~argues~~’)
- Don’t possess et al.

~~Cheng et al.’s~~ (2010) paper proposed ... VS ... the paper by Cheng et al. (2010) proposed ...

**Always comment** on direct quotes, e.g.

... part of the data set” (Lee, 2017, p. 5). **The implications of this are ...**

## Stage II task

Write 200-400 words total per review, responding to three 'questions':

- Briefly summarise what the author has done in one paragraph (50-100 words)
- Indicate what you think that the author has done well, and why in one paragraph (100-200 words)

*The strengths of the writing to me are ... What is clear about the paper is ...*

*You have ... and this is evident in the way you ... because ...*

- Indicate what you think could have been improved, and why in one paragraph (50-100 words)

*The writing could improve in the following areas ... You could ... It needs more ...*

*You could try to ... Think about having more of ... / less of...*

### What could we look for?

Message clarity / language / content / citation & Ref List / link / flow / accuracy / headings / relevance / critique / interpretation / use of data / figures / charts



## ► Semester Planner

**SEMESTER 1 2021**

**Semester planners:** Any library or Stop 1 or  
<https://students.unimelb.edu.au/academic-skills/explore-our-resources/time-management/semester-planner>

Use this space to write the weeks of the semester and add in key dates > **WEEK 1** **WEEK 2** **WEEK 3** **WEEK 4**

**January**

**February**

**March**

**April**

**May**

**June**

**Plan**

**Start!**

**Read**

**Write**

**Submit!**

**Finalise\***

# Writing resources

Developing a research question

<https://www.youtube.com/watch?v=mrWeLJZydUU&t=1s>

Cohesive, Critical and Interpretive Writing

<https://www.youtube.com/watch?v=AMeyQPYHbGg&list=PLJSPTc0K-PITaEBbDi5e15O6a4AMQlqwg&index=19>

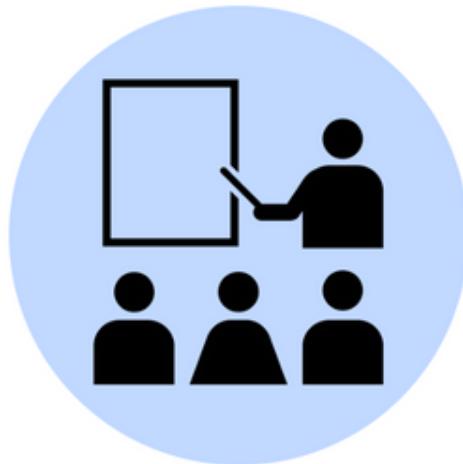
APA 7

<https://students.unimelb.edu.au/academic-skills/explore-our-resources/referencing/using-apa-7th-style>

# Academic Skills Support



Online  
resources



Workshops &  
courses



Individual  
appointments

<http://services.unimelb.edu.au/academicskills>



<https://www.youtube.com/user/UoMAcademicSkills>

# Lecture 15: Decision Trees

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, 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.



## So far ... Classification and Evaluation

- KNN, Naive Bayes, Logistic Regression, Perceptron
- Probabilistic models
- Loss functions, and estimation
- Evaluation

## Today... Decision Trees

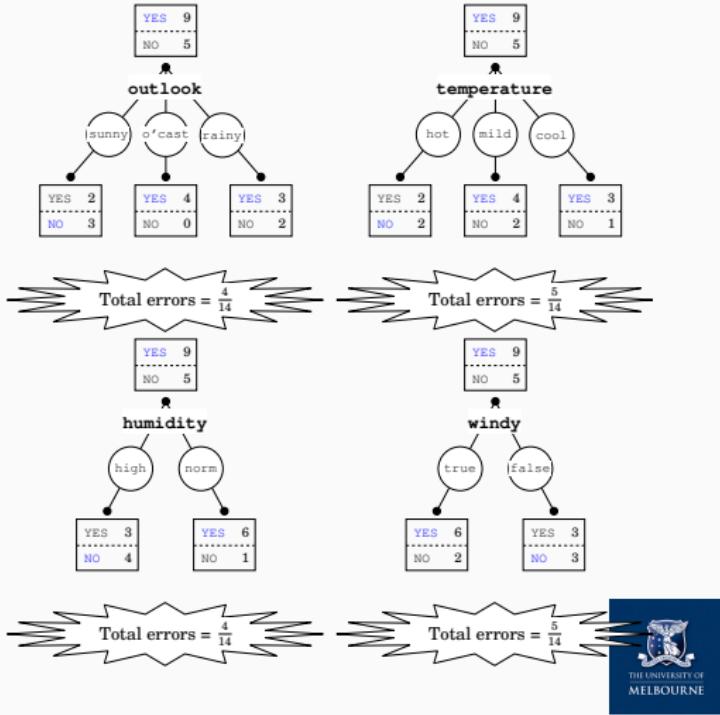
- Definition and motivation
- Estimation (ID3 Algorithm)
- Discussion



# From Decision Stumps to Decision Trees

We have seen decision stumps in action in the context of 1-R

Given the obvious myopia of decision stumps, how can we construct **decision trees** (of arbitrary depth) which have the ability to capture complex feature interaction?

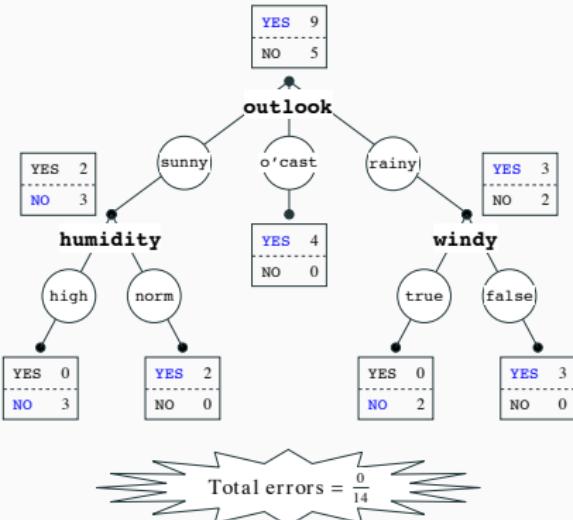


## The Weather Dataset (again!)

	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

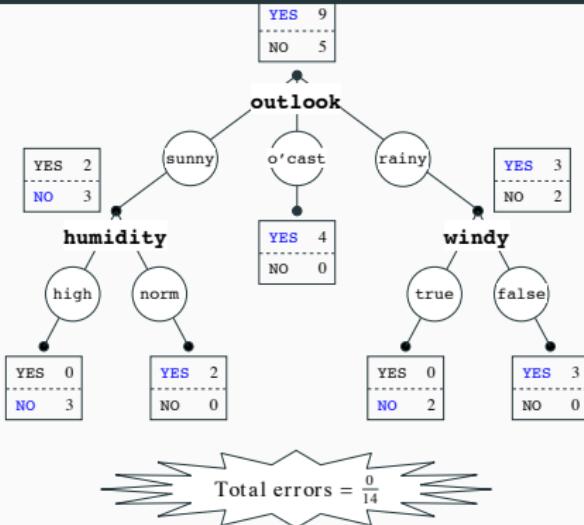


# Rule-based classification



- Construct the tree
- Extract one rule per leaf node
  1. if (outlook == o'cast) → yes
  2. if (outlook == sunny & humidity == normal) → yes
  3. if (outlook == rainy) & windy == false) → yes
  4. ...

# Disjunctive descriptions



Decision Trees can be read as a disjunction; for example, Yes:

$$(outlook = \text{sunny} \wedge \text{humidity} = \text{normal})$$

$$\vee (outlook = \text{overcast})$$

$$\vee (outlook = \text{rainy} \wedge \text{windy} = \text{false})$$



# Decision Trees: Classifying Novel Instances

## At test time...

- Assume we have constructed a decision tree
- Now, classify novel instances by traversing down the tree and predict the class according to the label of the deepest reachable point in the tree structure (leaf)

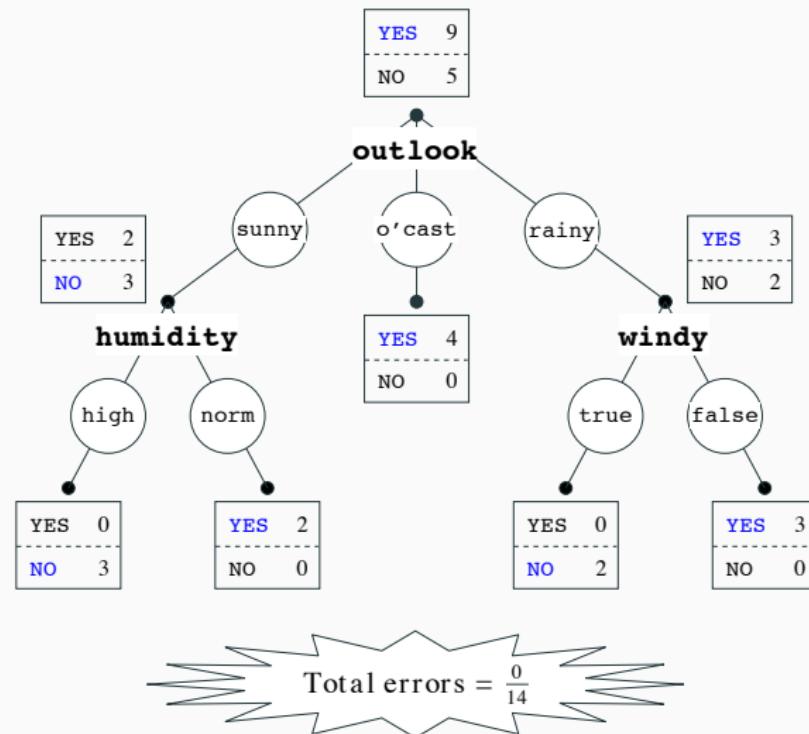
## Complications

- unobserved attribute–value pairs
- missing values



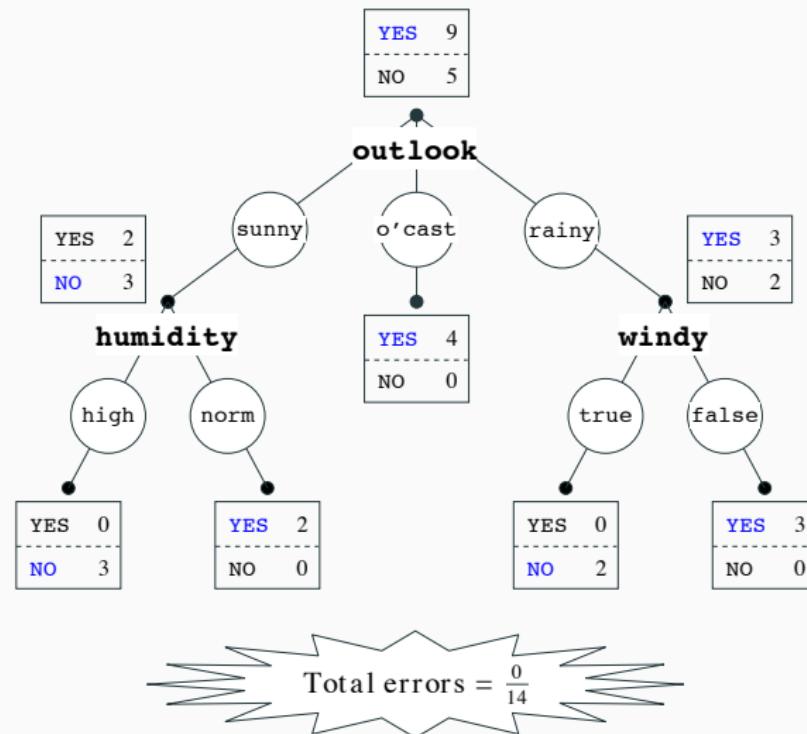
# Classification Example

Classify test instance: (**sunny, hot, normal, False**)



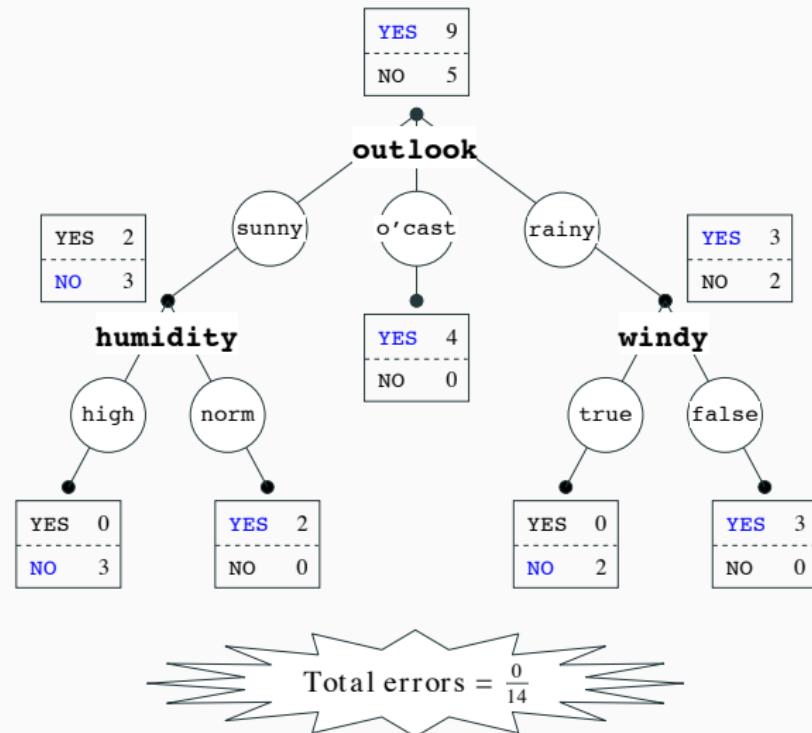
# Classification Example

Classify test instance: (**rainy, hot, low, False**)



# Classification Example

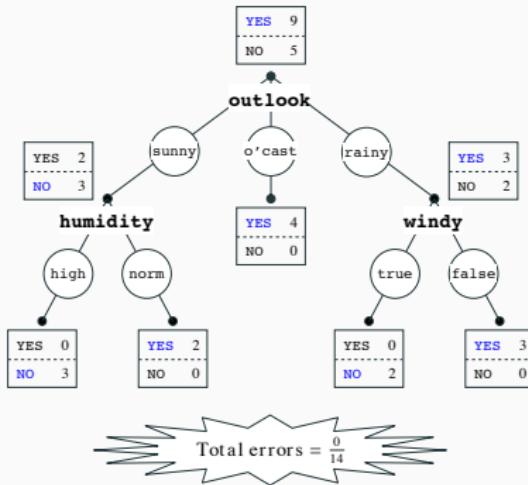
Classify test instance: **(?,cool, high, True)**



# Decision Trees: Issues

## Issues

- How to build an optimal tree?
- What does 'optimal' mean?
- How to choose attributes for decision points?
- When to stop growing the tree?



## ID3 Algorithm

---

## ID3: Overview

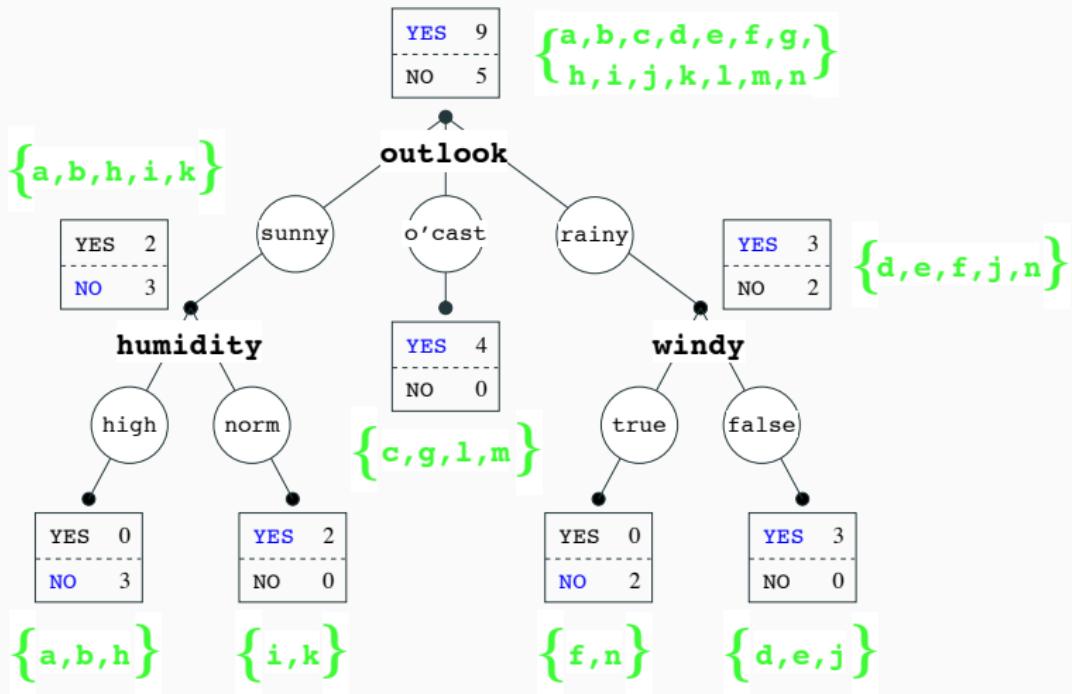
**Optimal** construction of a Decision Tree is **NP hard** (non-deterministic polynomial).

### So we use heuristics:

- Choose an attribute to partition the data at the node such that each partition is as **pure** (homogeneous) as possible.
- In each partition most of the instances should belong to as few classes as possible
- Each partition should be as large as possible.

We can stop the growth of the tree if all the leaf nodes are (largely) dominated by a single class (that is the leaf nodes are nearly pure).





# Constructing Decision Trees: ID3

**Basic method:** recursive divide-and-conquer

FUNCTION ID3 (Root)

IF all instances at root have same class<sup>\*\*</sup>

THEN stop

ELSE

1. Select a new attribute to use in partitioning root node instances
2. Create a branch for each attribute value and partition up root node instances according to each value
3. Call ID3(LEAF<sub>i</sub>) for each leaf node LEAF<sub>i</sub>

<sup>\*\*</sup>This is overly simplified, as we will discuss momentarily



# Criterion for Attribute Selection

**How do we choose the attribute to partition the instances at a given node?**

We want to get the smallest tree (Occam's Razor; generalisability). Prefer the shortest hypothesis that fits the data.

In favor:

- Fewer short hypotheses than long hypotheses
  - a short hyp. that fits the data unlikely to be a coincidence
  - a long hyp. that fits data might be a coincidence

Against:

- Many ways to define small sets of hypotheses



## Entropy and Information Gain (Intuition)

**Information Gain:** ‘Reduction of entropy before and after the data is partitioned using the attribute A’.

**Entropy:** The expected (average) level of surprise or uncertainty.

*Given a random variable (e.g., a coinflip), how surprised am I when seeing a certain outcome?*



## Entropy and Information Gain (Intuition)

**Information Gain:** ‘Reduction of entropy before and after the data is partitioned using the attribute A’.

**Entropy:** The expected (average) level of surprise or uncertainty.

*Given a random variable (e.g., a coinflip), how surprised am I when seeing a certain outcome?*

- **Low probability** event: if it happens, it’s big news! High surprise! **High information!**
- **High probability** event: it was likely to happen anyway. Not very surprising. **Low information!**



## Entropy (Definition)

- A measure of **unpredictability**
- Level of unpredictability (surprise) for a single event  $i$ : **self-information**

$$\text{self-info}(i) = \frac{1}{P(i)} = -\log_2 P(i)$$

- Given a probability distribution, the information (in bits) required to predict an event is the distribution's **entropy** or **information value**
- The entropy of a discrete random event  $x$  with possible outcomes  $1, \dots, n$  is:

$$\begin{aligned} H(x) &= \sum_{i=1}^n P(i) \text{self-info}(i) \\ &= - \sum_{i=1}^n P(i) \log_2 P(i) \end{aligned}$$

where  $0 \log_2 0 =^{\text{def}} 0$



## Entropy (Definition)

### Example 1 Coin flips.

- Biased coin. 55 flips: 50x *head*, 5x *tail*:

$$\approx 0.44 \text{ bits}$$

- Fair coin. 55 flips: 30x *head*, 25x *tail*:

$$\approx 0.99 \text{ bits}$$

The more uncertainty, the higher the entropy.



## Entropy (Definition)

**Example 2** In the context of Decision Trees, we are looking at the class distribution at a node:

- 50 Y instances, 5 N instances:

$$\approx 0.44 \text{ bits}$$

- 30 Y instances, 25 N instances:

$$\approx 0.99 \text{ bits}$$

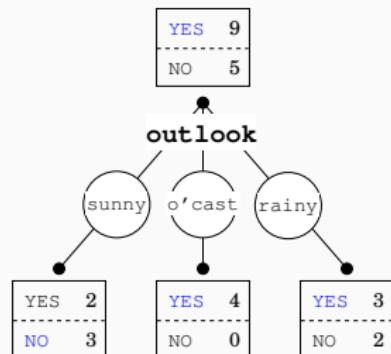
We want to classify with high certainty. We want leaves with **low entropy!**



# Entropy: Summary

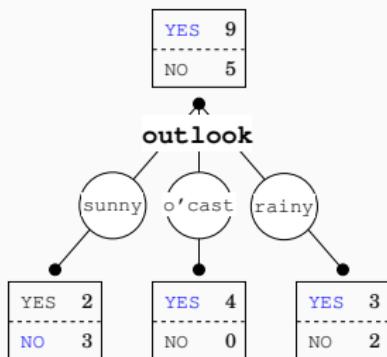
Entropy is a measure of **unpredictability**

- If the probability of a single class is **high**
  - Probability mass is centered
  - Entropy is **low**
  - The event is **predictable**
- If the probability is **evenly divided** between multiple classes
  - Probability mass is spread out
  - Entropy is **high**
  - The event is **unpredictable**



# From Entropy to Information Gain

- Decision tree with **low** entropy: class is more predictable.
- Information Gain** (reduction of entropy): measures how much **uncertainty** was **reduced**.
- Select the **attribute** that has **largest information gain**: the most entropy (uncertainty) is reduced, class is **most predictable**.



# Information Gain

The **expected reduction in entropy** caused by knowing the value of an attribute.

## Compare

- the **entropy before splitting** the tree using the attribute's values
- the **weighted average of the entropy over the children** after the split. This is called the (**Mean Information**)

If the entropy **decreases**, then we have a better tree (more predictable)



## Mean Information Associated with a Decision Stump

- We calculate the mean information for a tree stump with  $m$  attribute values as:

$$\text{Mean Info}(x_1, \dots, x_m) = \sum_{i=1}^m P(x_i)H(x_i)$$

where  $H(x_i)$  is the entropy of the class distribution for the instances at node  $x_i$

and  $P(x_i)$  is the proportion of instances at sub-node  $x_i$

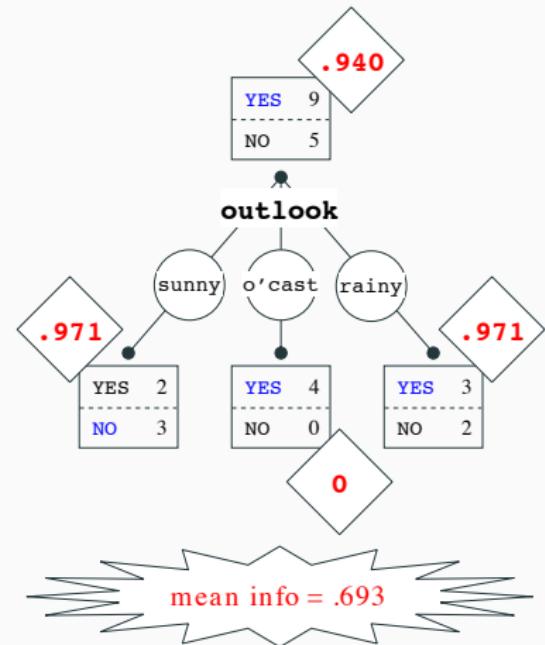


## Mean Information (outlook)

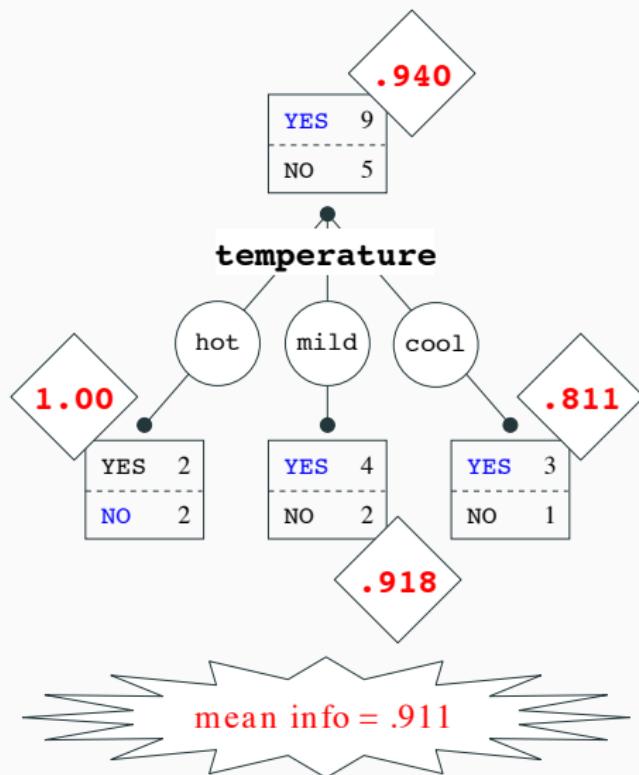
$$H(x) = - \sum_i P(x_i) \log_2 P(x_i)$$

$$H(\text{rainy}) =$$

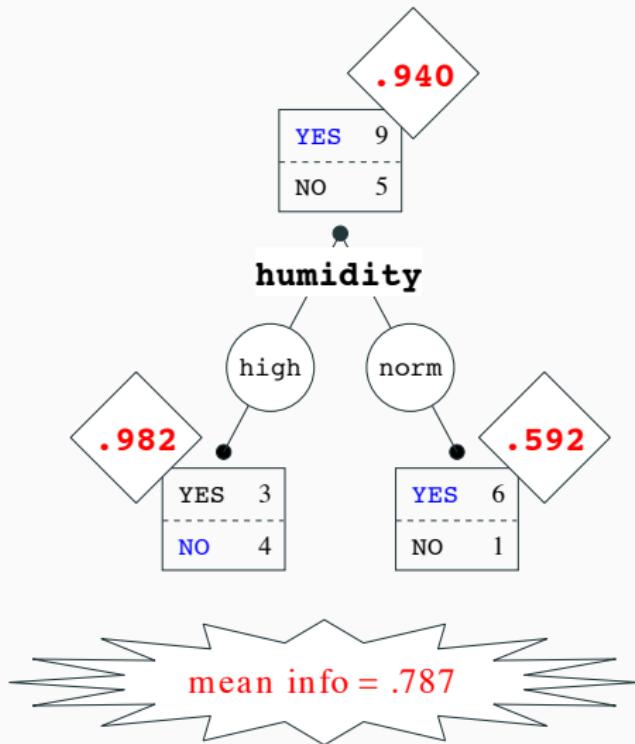
$$= 0.971$$



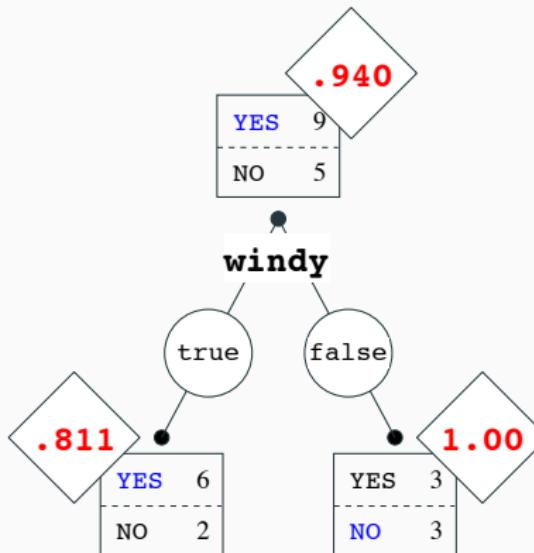
## Mean Information (temperature)



## Mean Information (humidity)



## Mean Information (windy)



mean info = .892

## Attribute Selection: Information Gain

- We determine which attribute  $R_A$  (with values  $x_1, \dots, x_m$ ) best partitions the instances at a given root node  $R$  according to **information gain** (IG):

$$\begin{aligned} IG(R_A|R) &= H(R) - \text{mean-info}(R_A) \\ &= H(R) - \sum_{i=1}^m P(x_i)H(x_i) \end{aligned}$$

$$IG(\text{outlook}|R) = 0.247$$

$$IG(\text{temperature}|R) = 0.029$$

$$IG(\text{humidity}|R) = 0.152$$

$$IG(\text{windy}|R) = 0.048$$

$$H(R) = 0.94$$

$$\text{Mean.info}(\text{outlook}) = 0.693$$

$$\text{Mean.info}(\text{temperature}) = 0.911$$

$$\text{Mean.info}(\text{humidity}) = 0.787$$

$$\text{Mean.info}(\text{windy}) = 0.892$$



## Attribute Selection: Information Gain

- We determine which attribute  $R_A$  (with values  $x_1, \dots, x_m$ ) best partitions the instances at a given root node  $R$  according to **information gain**:

$$\begin{aligned} IG(R_A|R) &= H(R) - \text{mean-info}(R_A) \\ &= H(R) - \sum_{i=1}^m P(x_i)H(x_i) \end{aligned}$$

$$IG(\text{outlook}|R) = 0.247$$

$$IG(\text{temperature}|R) = 0.029$$

$$IG(\text{humidity}|R) = 0.152$$

$$IG(\text{windy}|R) = 0.048$$

$$H(R) = 0.94$$

$$\text{Mean.info(outlook)} = 0.693$$

$$\text{Mean.info(temperature)} = 0.911$$

$$\text{Mean.info(humidity)} = 0.787$$

$$\text{Mean.info(windy)} = 0.892$$



## Shortcomings of Information Gain

Information gain tends to **prefer highly-branching attributes**:

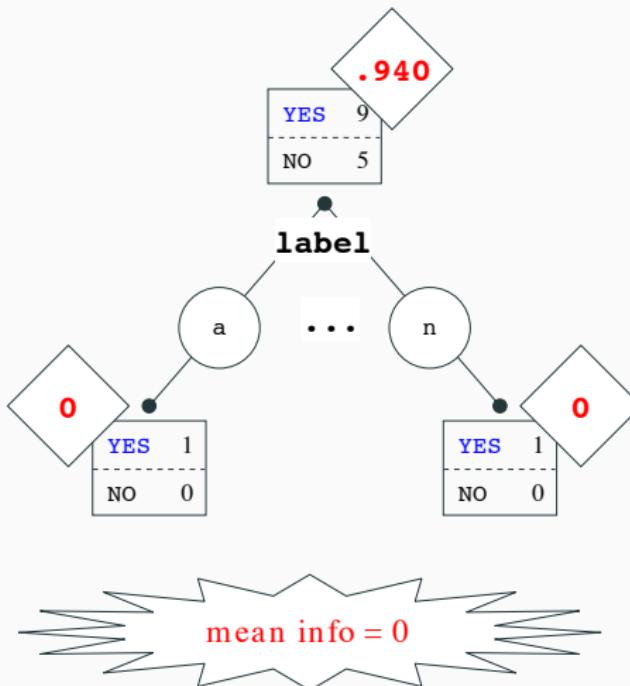
- A subset of instances is more likely to be homogeneous (pure) if there are only a few instances
- Attribute with many values will have fewer instances at each child node

This may result in **overfitting** / fragmentation



## Mean Information (label)

Information gain tends to **prefer highly-branching attributes**:



## Solution: Gain Ratio

- **Gain ratio (GR)** reduces the bias for information gain towards highly-branching attributes by normalising relative to the **split information**
- **Split info (SI)** is the entropy of a given split (evenness of the distribution of instances to attribute values)

$$\begin{aligned} GR(R_A|R) &= \frac{IG(R_A|R)}{SI(R_A|R)} = \frac{IG(R_A|R)}{H(R_A)} \\ &= \frac{H(R) - \sum_{i=1}^m P(x_i)H(x_i)}{-\sum_{i=1}^m P(x_i)\log_2 P(x_i)} \end{aligned}$$

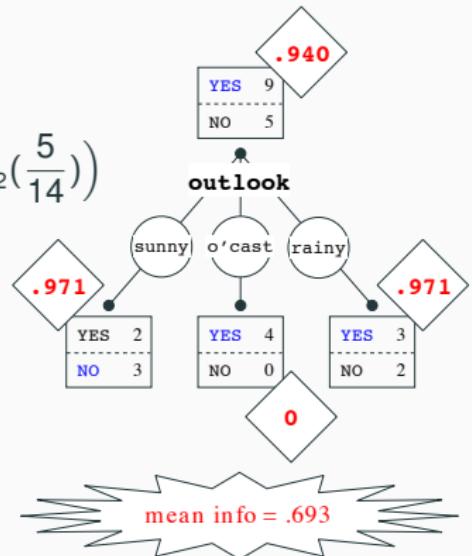
- Split Info sometimes called *Intrinsic Value*
- Discourages the selection of attributes with many uniformly distributed values



## Split Info

$$SI(\text{outlook}|R)$$

$$\begin{aligned} &= - \left( \left( \frac{5}{14} \right) \log_2 \left( \frac{5}{14} \right) + \left( \frac{4}{14} \right) \log_2 \left( \frac{4}{14} \right) + \left( \frac{5}{14} \right) \log_2 \left( \frac{5}{14} \right) \right) \\ &= 1.577 \end{aligned}$$



NB: Entropy of distribution of instances to attribute *values* (disregarding classes, unlike Mean Info)

## Gain Ratio: Example

$$IG(\text{outlook}|R) = 0.247$$

$$SI(\text{outlook}|R) = 1.577$$

$$GR(\text{outlook}|R) = 0.156$$

$$IG(\text{humidity}|R) = 0.152$$

$$SI(\text{humidity}|R) = 1.000$$

$$GR(\text{humidity}|R) = 0.152$$

$$IG(\text{label}|R) = 0.940$$

$$SI(\text{label}|R) = 3.807$$

$$GR(\text{label}|R) = 0.247$$

$$IG(\text{temperature}|R) = 0.029$$

$$SI(\text{temperature}|R) = 1.557$$

$$GR(\text{temperature}|R) = 0.019$$

$$IG(\text{windy}|R) = 0.048$$

$$SI(\text{windy}|R) = 0.985$$

$$GR(\text{windy}|R) = 0.049$$



## Stopping criteria i

The definition of ID3 above suggests that:

- We recurse until the instances at a node are of the same class
- This is consistent with our usage of entropy: if all of the instances are of a single class, the entropy of the distribution is 0
- Considering other attributes cannot “improve” an entropy of 0 — the Info Gain is 0 by definition

This helps to ensure that the tree remains compact (Occam’s Razor)



## Stopping criteria ii

The definition of ID3 above suggests that:

- The Info Gain/Gain Ratio allows us to choose the (seemingly) best attribute at a given node
- However, it is also an approximate indication of how much absolute improvement we expect from partitioning the data according to the values of a given attribute
- An Info Gain of 0 means that there is no improvement; a very small improvement is often unjustifiable
- Typical modification of ID3: choose best attribute only if  $IG/GR$  is greater than some **threshold**  $\tau$
- Other similar approaches use **pruning** — post-process the tree to remove undesirable branches (with few instances, or small  $IG/GR$  improvements)



## Stopping criteria iii

The definition of ID3 above suggests that:

- We might observe improvement through every layer of the tree
- We then run out of attributes, even though one or more leaves could be improved further
- Fall back to majority class label for instances at a leaf with a mixed distribution — unclear what to do with ties
- Possibly can be taken as evidence that the given attributes are insufficient for solving the problem



## **Discussion**

---

## Hypothesis Space Search in ID3

- ID3 can be characterized as searching a space of hypotheses for one that fits the training examples.
- The hypothesis space searched by ID3 is the set of possible decision trees.
- ID3 performs a simple-to-complex, hill-climbing search through this hypothesis space (with no backtracking),
  - beginning with the empty tree
  - considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data



# Pros / Cons of Decision Trees

## Pros

- Highly regarded among basic supervised learners
- Fast to train, even faster to classify
- Very transparent (probably the most interpretable of all classification algorithms!)

## Cons

- Prone to Overfitting
- Loss of information for continuous variables
- Complex calculation if there are many classes
- No guarantee to return the globally optimal decision
- Information gain: Bias for attributes with greater no. of values.



## Variants of Decision Trees

ID3 is not the only (nor most popular) Decision Tree learner:

- **Oblivious Decision Trees** require the same attribute at every node in a layer
- **Random Tree** only uses a sample of the possible attributes at a given node
  - Helps to account for irrelevant attributes
  - Basis for a better Decision Tree variant: **Random Forest**



## Summary

- Describe the basic decision tree induction method used in ID3
- What is information gain, how is it calculated and what is its primary shortcoming?
- What is gain ratio, and how does it attempt to overcome the shortcoming of information gain?
- What are the theoretical and practical properties of ID3-style decision trees?

Mitchell, Tom (1997). Machine Learning. Chapter 3: *Decision Tree Learning*.

Tan et al (2006) Introduction to Data Mining. Section 4.3, pp 150-171.



# Lecture 16: Ensemble Learning

---

**COMP90049**

Semester 1, 2021

Lea Frermann, 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



## So far...

- individual classification algorithms in isolation
- choose the "optimal" classifier by comparing the performance of individual classifiers over a given dataset/task
- When evaluating, we only get one shot at classifying a given test instance and are stuck with the bias inherent in a given algorithm

## Today

- Wrapping up classification:
  - aside on (non)-linear classification
  - aside on (non)-parametric machine learning
- Ensembles: combining multiple (weak) models into one strong model!



## **Aside: (Non)-linear and (non)-parametric classification**

---

## Aside I: linear vs. non-linear classification

### Linear classifiers

- Naive Bayes
- Logistic Regression
- Perceptron

... because their **decision boundary** is a linear function of the input  $x$ .  
(There's still a non-linear activation function, so  $y$  is not a linear function of  $x$ ).



## Aside I: linear vs. non-linear classification

### Linear classifiers

- Naive Bayes
- Logistic Regression
- Perceptron

... because their **decision boundary** is a linear function of the input  $x$ .  
(There's still a non-linear activation function, so  $y$  is not a linear function of  $x$ ).

### Non-linear classifiers

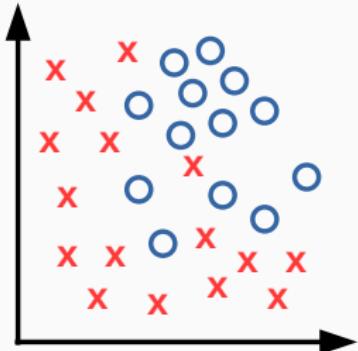
- Multi-layer perceptron (with non-linear activations)
- K-Nearest Neighbors
- Decision trees

... because their **decision boundary** is \*not\* a linear function of the input  $x$ .  
They can learn more complex decision boundaries.



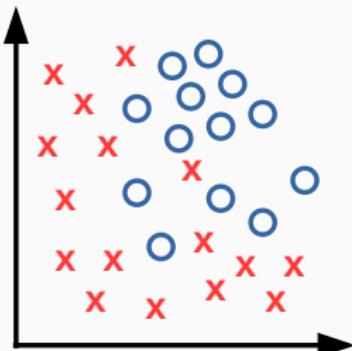
## Aside I: linear vs. non-linear classification

**Decision boundary** of a well-trained decision tree



Appropriate-fitting

**Decision boundary** of a well-trained Naive Bayes classifier



Appropriate-fitting

## Aside II: parametric vs. non-parametric models

**Warning:** these terms are ambiguous and several definitions exist. We'll adopt the following.

### Parametric Models

- Naive Bayes, Logistic Regression, Multi-layer perceptron, ...

... because they have a **constant number of parameters**, irrespective of the **amount of training data**. We can write down the model  $y = f(x; \theta)$  which holds true no matter what  $x$ . We fit parameters to a **given model**.



## Aside II: parametric vs. non-parametric models

**Warning:** these terms are ambiguous and several definitions exist. We'll adopt the following.

### Parametric Models

- Naive Bayes, Logistic Regression, Multi-layer perceptron, ...

... because they have a **constant number of parameters**, irrespective of the **amount of training data**. We can write down the model  $y = f(x; \theta)$  which holds true no matter what  $x$ . We fit parameters to a **given model**.

### Non-parametric models

- K-Nearest Neighbors, Decision trees, ...

... because the parameters grow with the training data and are **possibly infinite**. We **learn our model directly from the data**.

- Discuss: what's 'non-parametric' about KNN?
- Discuss: what's 'non-parametric' about Decision Trees?



**Now, on to ensembles**

---

- **Ensemble learning (aka. Classifier combination):** constructs a set of base classifiers from a given set of training data and aggregates the outputs into a single meta-classifier
- **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



## Ensembles II

- When does ensemble learning work?
  - the base classifiers should not make the same mistakes
  - the base classifiers are reasonably accurate

	$t_1$	$t_2$	$t_3$
$C_1$	✓	✓	✗
$C_2$	✗	✓	✓
$C_3$	✓	✗	✓
$C^*$	✓	✓	✓

	$t_1$	$t_2$	$t_3$
$C_1$	✓	✓	✗
$C_2$	✓	✓	✗
$C_3$	✓	✓	✗
$C^*$	✓	✓	✗

	$t_1$	$t_2$	$t_3$
$C_1$	✓	✗	✗
$C_2$	✗	✓	✗
$C_3$	✗	✗	✓
$C^*$	✗	✗	✗

**Let's assume that:**

- We have a set of 25 binary base classifiers
- Each has an error rate of  $\epsilon = 0.35$
- The base classifiers are independent (that's usually false)
- We perform classifier combination by voting

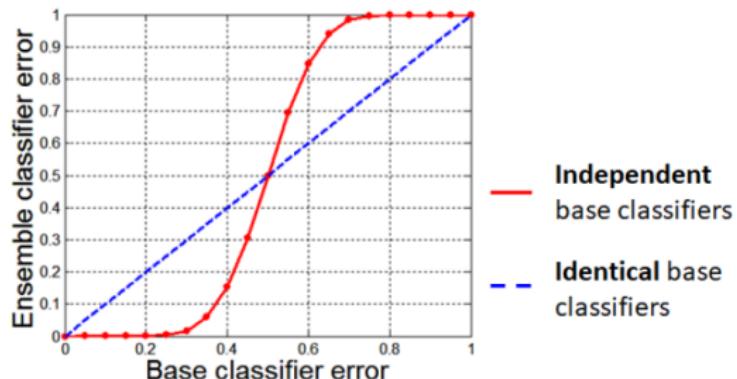
**The error rate of the combined classifier is:**

$$\sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1 - \epsilon)^{25-i} \approx 0.06$$



## Ensembles VI

- When does ensemble learning work?



# Classification with Ensemble Learning

- The simplest means of classification over multiple base classifiers is simple **voting**:
  - **Classification**: run multiple base classifiers over the test data and select the class predicted by the most base classifiers (e.g. k-NN)
  - **Regression**: average over the numeric predictions of our base classifiers



# Approaches to 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
- **Algorithm manipulation:** semi-randomly “tweak” internal parameters within a given algorithm to generate multiple base classifiers over a given dataset
- **Class label manipulation:** generate multiple training datasets by manipulating the class labels in a reversible manner



## **Stacking**

---

- **Intuition:** “smooth” errors over a range of algorithms with different biases
- **Simple Voting:** generate multiple training datasets through different feature subsets, and train a base classifier over each dataset
  - presupposes the classifiers have equal performance
- **Meta Classification:** train a classifier over the outputs of the base classifiers
  - train using nested cross validation to reduce bias



- **Level 0:** Given training dataset  $(X, y)$ :
  - Train Neural Network
  - Train Naive Bayes
  - Train Decision Tree
  - ...
- Discard (or keep)  $X$ , **add new attributes** for each instance:
  - predictions of the classifiers above
  - other data as available (NB scores etc.)
- **Level 1:** Train meta-classifier. Usually Logistic Regression or Neural Network



# Stacking III

## Nested Cross-validation



## Stacking VI

- 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
- Widely seen in applied research; less interest within theoretical circles (esp. statistical learning)



## **Bagging**

---

# Bagging I

- **Intuition:** the more data, the better the performance (lower the variance), so how can we get ever more data out of a fixed training dataset?
- **Method:** construct “novel” datasets through a combination of random sampling and replacement
  - Randomly sample the original dataset  $N$  times, **with replacement** (bootstrap)
  - Thus, we get a new dataset of the same size, where any individual instance is absent with probability  $(1 - \frac{1}{N})^N$
  - construct  $k$  random datasets for  $k$  base classifiers, and arrive at prediction via voting



## Bagging II

- 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

- The same (weak) classification algorithm is used throughout
- As bagging is aimed towards minimising variance through sampling, the algorithm should be unstable (=high-variance) ... e.g.?



- Simple method based on sampling and voting
- Possibility to parallelise computation of individual base classifiers
- Highly effective over noisy datasets (outliers may vanish)
- Performance is generally significantly better than the base classifiers and only occasionally substantially worse



## **Bagging - Random Forest**

---

# Random Forest I

A **Random Tree** is a Decision Tree where:

- At each node, only some of the possible attributes are considered
- For example, a fixed proportion of all of the attributes, except the ones used earlier in the tree
- Attempts to control for unhelpful attributes in the feature set
- Much faster to build than a “deterministic” Decision Tree, but increases model variance

This is an instance of **Feature Manipulation**.



## Random Forest II

A Random Forest is:

- An ensemble of Random Trees (many trees = forest)
- Each tree is built using a different Bagged training dataset
- As with Bagging the combined classification is via voting
- The idea behind them is to minimise overall model variance, without introducing (combined) model bias



This is an instance of **Instance Manipulation**.



# Random Forest III

## Hyperparameters:

- number of trees  $B$  (can be tuned, e.g. based on “out-of-bag” error rate)
- feature sub-sample size (e.g.  $(\log |F| + 1)$ )

## Interpretation:

- logic behind predictions on individual instances can be tediously followed through the various trees
- logic behind overall model: ???



## Practical Properties of Random Forests:

- Generally a very strong performer
- Embarrassingly parallelisable
- Surprisingly efficient
- Robust to overfitting
- Interpretability sacrificed



## **Boosting**

---

- **Intuition:** tune base classifiers to focus on the “hard to classify” instances
- **Approach:** iteratively change the distribution and weights of training instances to reflect the performance of the classifier on the previous iteration
  - start with each training instance having a probability of  $\frac{1}{N}$  being included in the sample
  - over  $T$  iterations, train a classifier and **update the weight of each instance** according to whether it is correctly classified
  - 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
---	---	---	---	---	---	---	----	---	---

⋮



# AdaBoost I

**AdaBoost** (Adaptive Boosting) is a *sequential* ensembling algorithm.

## Basic idea:

- Base classifier:  $C_0$
- Training instances  $(x_j, y_j) | j = 1, 2, \dots, N$
- Initial instance weights  $w_j^{(0)} = \frac{1}{N} | j = 1, 2, \dots, N$
- In iteration  $i$ :
  - Construct classifier  $C_i$  and compute **error rate**  $\epsilon_i$

$$\epsilon_i = \sum_{j=1}^N w_j^{(i)} \delta(C_i(x_j) \neq y_j)$$

- Use  $\epsilon_i$  to compute the **classifier weight**  $\alpha_i$  (importance of  $C_i$ )
- Use  $\alpha_i$  to update **instance weights**
- Add weighted  $C_i$  to ensemble

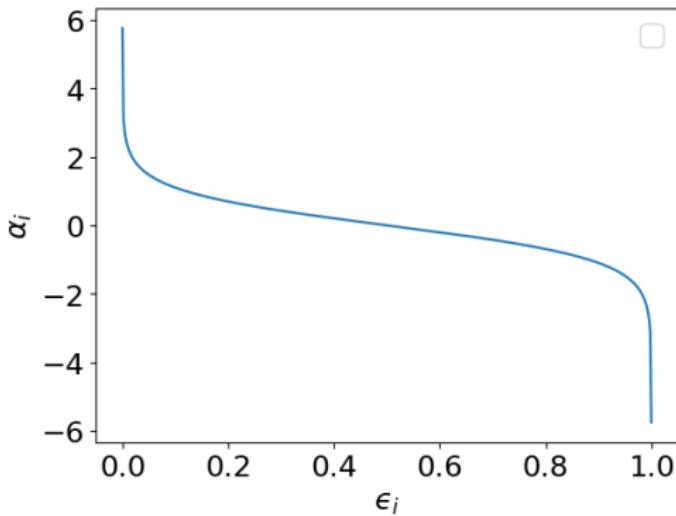
**Output:** Weighted set of base classifiers:  $\{(\alpha_1, C_1), (\alpha_2, C_2), \dots, (\alpha_T, C_T)\}$



## AdaBoost II: Computing $\alpha$

“Importance” of  $C_i$  (i.e. the weight associated with the classifiers’ votes):

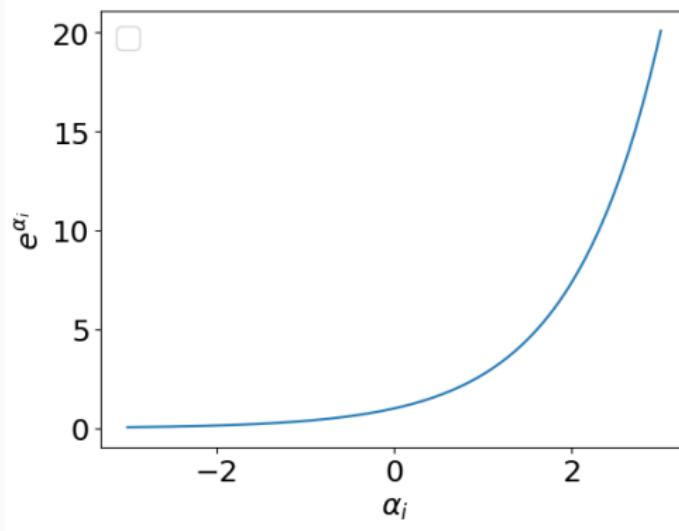
$$\alpha_i = \frac{1}{2} \log_e \frac{1 - \epsilon_i}{\epsilon_i}$$



## AdaBoost III: Updating $w$

Weights for instance  $j$  ( $i > 0$ ):

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ e^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$



- Continue iterating for  $i = 1, 2, \dots, T$ , but reinitialise the instance weights whenever  $\epsilon_i > 0.5$
- As long as each base classifier is better than random, convergence to a stronger model is **guaranteed**
- **Classification**

$$C^*(x) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

- **Typical base classifiers:** decision stumps (OneR) or decision trees.  
(Possibly the world's favorite classifier!)



## Boosting III

- Mathematically complicated but computationally cheap method based on iterative sampling and weighted voting
- More computationally expensive than bagging
- The method has guaranteed performance in the form of **error bounds** over the training data
- Interesting effect with convergence of the error rate over the training vs. test data
- In practical applications, boosting has the tendency to **overfit**



# Bagging vs. Boosting

## Bagging

- Parallel sampling
- Simple voting
- Single classification algorithm
- Minimise variance
- Not prone to overfitting

## Boosting

- Iterative sampling
- Weighted voting
- Single classification algorithm
- Minimise (instance) bias
- 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?



## 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 17: Unsupervised Learning

---

**COMP90049**

Semester 1, 2021

Lea Frermann, 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



## So far...

- Classification
- Evaluation of supervised machine learners
- Classifier combination

## Today

- Unsupervised learning: Clustering
- Concepts
- Methods
- Evaluation



## Clustering

---

## A possible clustering of the weather dataset

Outlook	Temperature	Humidity	Windy	Cluster
sunny	hot	high	FALSE	?
sunny	hot	high	TRUE	?
overcast	hot	high	FALSE	?
rainy	mild	high	FALSE	?
rainy	cool	normal	FALSE	?
rainy	cool	normal	TRUE	?
overcast	cool	normal	TRUE	?
sunny	mild	high	FALSE	?
sunny	cool	normal	FALSE	?
rainy	mild	normal	FALSE	?
sunny	mild	normal	TRUE	?
overcast	mild	high	TRUE	?
overcast	hot	normal	FALSE	?
rainy	mild	high	TRUE	?



## Clustering over the weather dataset (cf. outputs)

Outlook	Temperature	Humidity	Windy	Cluster	Play
sunny	hot	high	FALSE	0	no
sunny	hot	high	TRUE	0	no
overcast	hot	high	FALSE	0	yes
rainy	mild	high	FALSE	1	yes
rainy	cool	normal	FALSE	1	yes
rainy	cool	normal	TRUE	1	no
overcast	cool	normal	TRUE	1	yes
sunny	mild	high	FALSE	0	no
sunny	cool	normal	FALSE	1	yes
rainy	mild	normal	FALSE	1	yes
sunny	mild	normal	TRUE	1	yes
overcast	mild	high	TRUE	1	yes
overcast	hot	normal	FALSE	0	yes
rainy	mild	high	TRUE	1	no



# Clustering

- Clustering is *unsupervised*
- The class of an example is not known (or at least not used)
- Finding groups of items that are *similar*
- Success often measured subjectively
- Applications in pattern recognition, spatial data analysis, medical diagnosis, ...



# Clustering, basic contrasts

## Exclusive vs. overlapping

- Can an item be in more than one cluster?

## Deterministic vs. probabilistic (Hard vs. soft clustering)

- Can an item be partially or weakly in a cluster?

## Hierarchical vs. partitioning

- Do the clusters have subset relationships between them? e.g. nested in a tree?

## Partial vs. complete

- In some cases, we only want to cluster some of the data

## Heterogenous vs. homogenous

- Clusters of widely different sizes, shapes, and densities

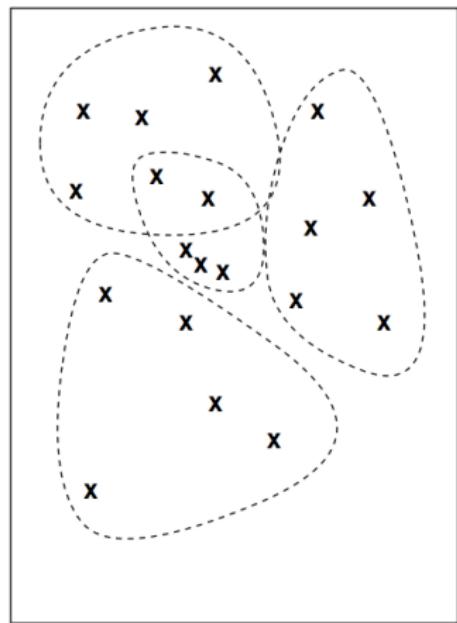
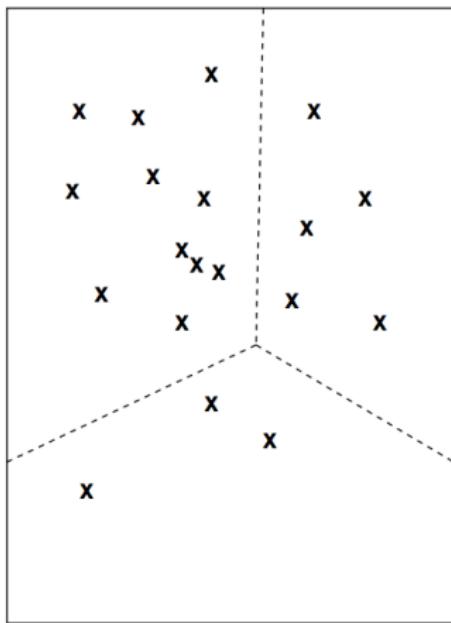
## Incremental vs. batch

- Is the whole set of items clustered in one go?



# Exclusive vs. overlapping clustering

Can an item be in more than one cluster?



# Deterministic vs. probabilistic clustering

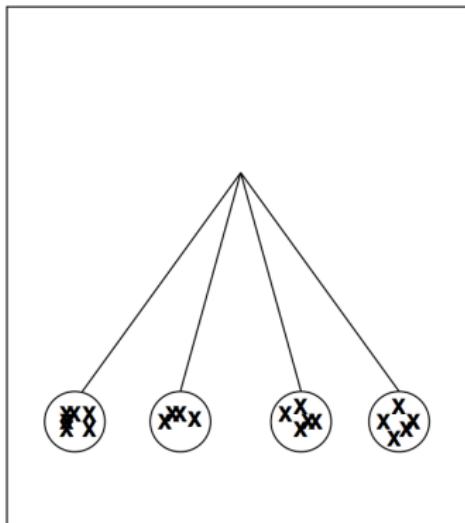
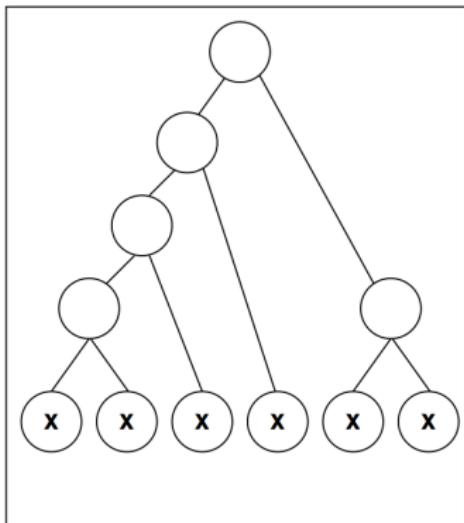
Can an item be partially or weakly in a cluster?

<i>Instance</i>	<i>Cluster</i>	<i>Instance</i>	Cluster			
			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?



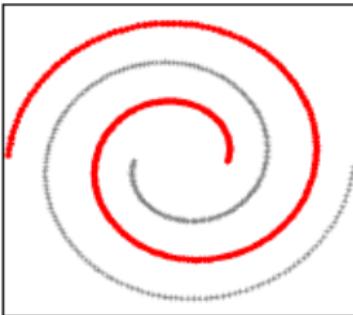
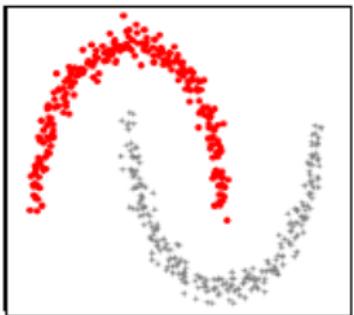
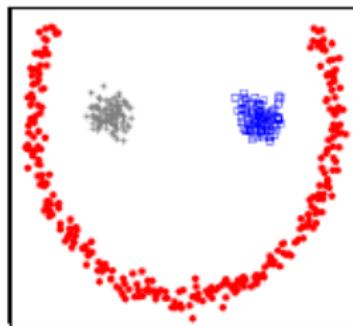
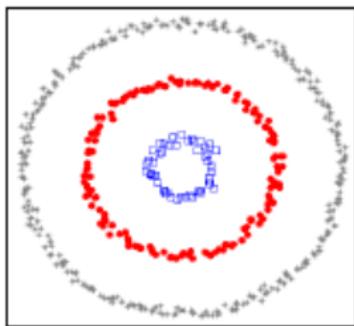
## Partial vs. complete

In some cases, we only want to cluster some of the data



# Heterogenous vs. homogenous

Clusters of widely different sizes, shapes, and densities



# Incremental vs Batch clustering

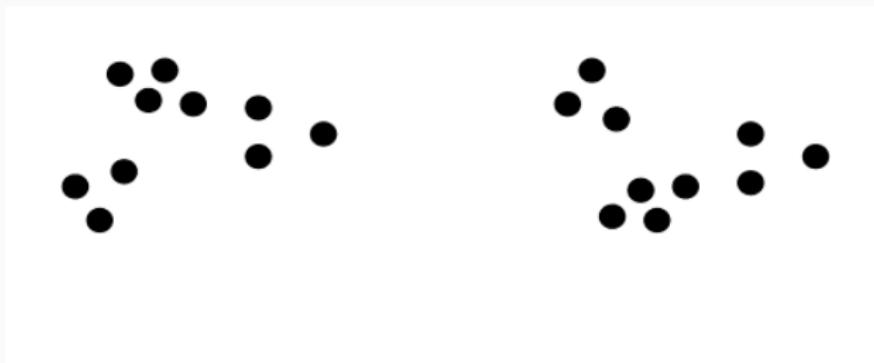


## Clustering, Desiderata

- Scalability; high dimensionality
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Able to deal with noise and outliers
- Insensitive to order of input records



# What is a good clustering?



## Two clusters?



## Four clusters?



## Six clusters?



# Types of Evaluation

## Unsupervised

- Measures the goodness of a clustering structure without respect to external information. Includes measures of **cluster cohesion** (compactness, tightness), and measures of **cluster separation** (isolation, distinctiveness).

## Supervised

- Measures the extent to which the clustering structure discovered by a clustering algorithm matches some external structure. For instance, *entropy* can measure how well cluster labels match externally supplied class labels.



# Unsupervised Evaluation I

A “good” cluster should have one or both of:

- **High cluster cohesion:** instances in a given cluster should be closely related to each other

$$cohesion(C_i) = \frac{1}{\sum_{x,y \in C_i} Distance(x,y)}$$

- **High Cluster Separation** instances in different clusters should be distinct from each other

$$separation(C_i, C_j) = \sum_{x \in C_i, y \in C_j} Distance(x,y)$$



## Unsupervised Evaluation II

Most common measure for evaluating cluster quality is **Sum of Squared Error (SSE)** or *Scatter*

$$\sum_{i=1}^k \sum_{x \in C_i} dist^2(m_i, x)$$

- $x$ : is a data point in cluster  $C_i$
- $m_i$ : is the representative point for cluster  $C_i$
- For each point, the error is the distance to the nearest cluster
- To get SSE, we square these errors and sum them.



## Unsupervised Evaluation II

Most common measure for evaluating cluster quality is **Sum of Squared Error (SSE)** or *Scatter*

$$\sum_{i=1}^k \sum_{x \in C_i} \text{dist}^2(m_i, x)$$

- Can show that the  $m_i$  that minimises SSE corresponds to the center (mean) of the cluster
- At ‘application time’: Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase  $k$ , the number of clusters
- However, a good clustering with smaller  $k$  can have a lower SSE than a poor clustering with higher  $k$



## Sum of squared errors: Example

### SSE for Categorical Attributes

Cluster 1 centroid:				Cluster 2 centroid:				
sunny	mild	high	no	overcast	cool	normal	yes	
sunny	hot	high	no	$1^2 = 1$	rainy	cool	normal	yes
sunny	hot	high	yes	$2^2 = 4$	overcast	cool	normal	yes
overcast	hot	high	no	$2^2 = 4$	sunny	cool	normal	no
rainy	mild	high	no	$1^2 = 1$	overcast	mild	normal	no
sunny	mild	high	no	0	sunny	mild	normal	yes
overcast	mild	high	yes	$2^2 = 4$	overcast	hot	normal	no
rainy	mild	high	yes	$2^2 = 4$	rainy	cool	normal	no

$SSE_1 = 18$        $SSE_2 = 20$

$$SSE = SSE_1 + SSE_2 = 38$$



## Supervised Evaluation

If labels are available, evaluate the degree to which class labels are consistent within a cluster and different across clusters

- **Purity:** Probability of the class with higher probability (representation) in each cluster. Higher is better.

$$purity = \sum_{i=1}^k \frac{|C_i|}{N} \max_j P_i(j)$$

- **Entropy:** Entropy probability of distribution over labels per cluster. Lower is better.

$$entropy = \sum_{i=1}^k \frac{|C_i|}{N} H(x_i)$$

- where  $x_i$  is the distribution of class labels in cluster  $i$



## Supervised Evaluation Example I

Ex. 1: Calculate the entropy and purity of the following cluster output

Cluster	Play = yes	Play = no
1	4	0
2	4	4

$$purity = \sum_{i=1}^k \frac{|C_i|}{N} \max_j P_i(j) \quad \quad \quad entropy = \sum_{i=1}^k \frac{|C_i|}{N} H(x_i)$$



## Supervised Evaluation Example II

Ex. 2: Calculate the entropy and purity of the following cluster output

Cluster	Play = yes	Play = no
1	2	0
2	6	4

$$entropy_1 = -1 \times \log(1) - 0 \times \log(0) = 0$$

$$entropy_2 = -0.6 \times \log(0.6) - 0.4 \times \log(0.4) = 0.97$$

$$purity_1 = \max(1, 0) = 1$$

$$purity_2 = \max(0.6, 0.4) = 0.6$$



## Supervised Evaluation Example II

Ex. 2: Calculate the entropy and purity of the following cluster output

Cluster	Play = yes	Play = no	Entropy	Purity
1	2	0	0	1
2	6	4	0.97	0.6
<b>Total:</b>			<b>0.81</b>	<b>0.67</b>

$$\text{entropy} = \frac{2}{12} \times 0 + \frac{10}{12} \times 0.97 = 0.81$$

$$\text{purity} = \frac{2}{12} \times 1 + \frac{10}{12} \times 0.6 = 0.67$$



## Methods

---

## Similarity / Proximity / Closeness

**Clustering finds groups of instances in the dataset which**

- are similar or close to each other within a group
- are being different or separated from other clusters/clusters.

A key component of any clustering algorithm is a measurement of the **distance between any points**.



# Measuring Similarity / Proximity / Closeness

## Data points in Euclidean space

- Euclidean (L2) distance
- Manhattan (L1) distance



# Measuring Similarity / Proximity / Closeness

## Discrete values

- Hamming distance (discrepancy between the bit strings)

$d$	a	b	c
a	0	1	1
b	1	0	1
c	1	1	0

For two bit strings, the number of positions at which the corresponding symbols are different



# Measuring Similarity / Proximity / Closeness

## Text documents

- Cosine similarity
- Jaccard measure

## Other measures

- Correlation
- Graph-based measures



## *k*-means Clustering

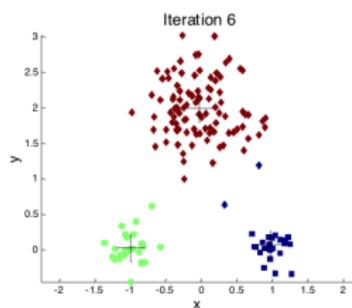
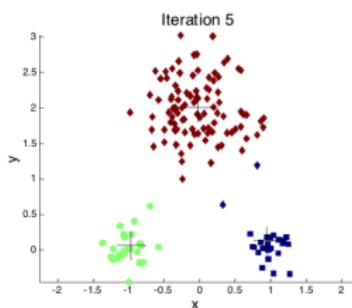
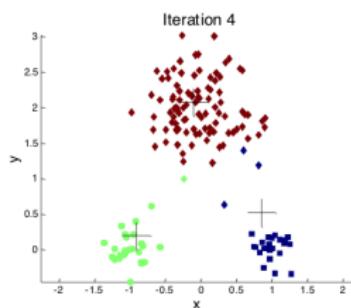
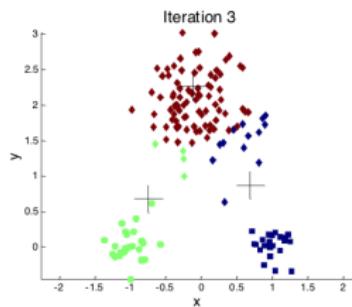
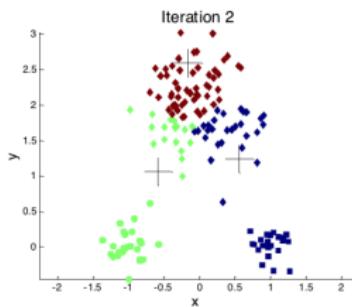
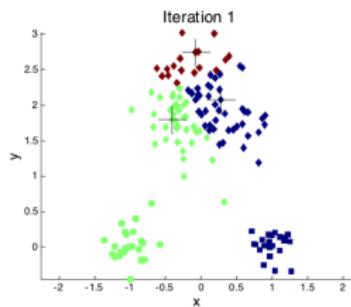
Given  $k$ , the  $k$ -means algorithm is implemented in four steps:

1. Select  $k$  points to act as seed cluster centroids
2. **repeat**
3.     Assign each instance to the cluster with the **nearest centroid**
4.     Recompute the centroid of each cluster
5. **until** the centroids don't change

It is an exclusive, deterministic, partitioning, batch clustering method



# Example, Iterations

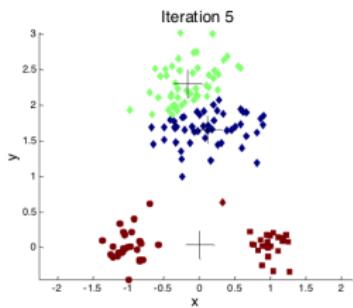
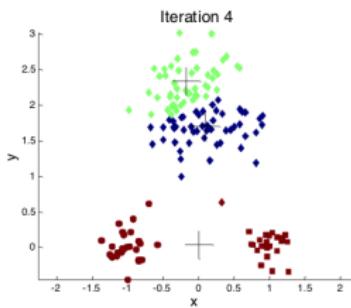
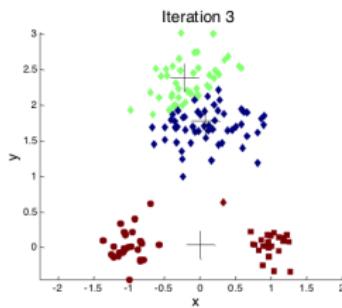
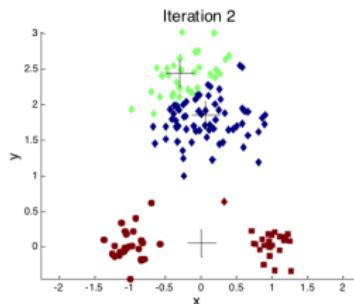
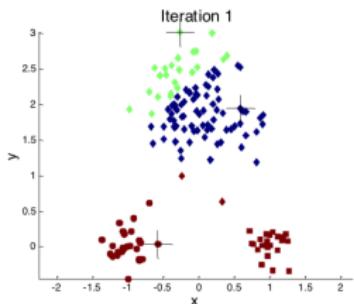


## *k*-means Clustering – Details

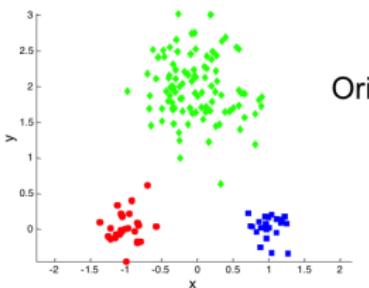
- **Initial centroids** are often chosen randomly.
  - Clusters produced vary from one run to another.
- The **centroid** is (typically) the **mean** of the points in the cluster.
- ‘Nearest’ is based on proximity/**similarity metric**.
- K-means will **converge** for common similarity measures mentioned above.
  - Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to '*Until relatively few points change clusters*'  
(this way the stopping criterion will not depend on the type of similarity or dimensionality)



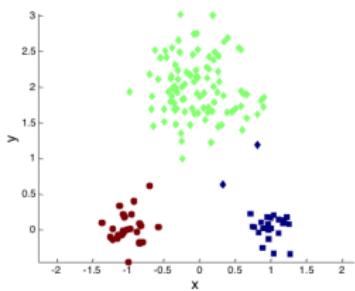
## Example, Impact of initial seeds



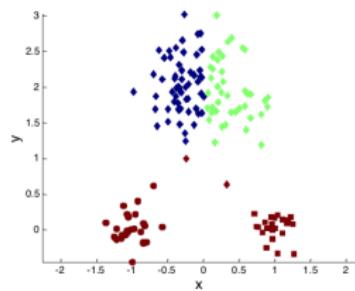
## Example, Different outcomes



Original Points



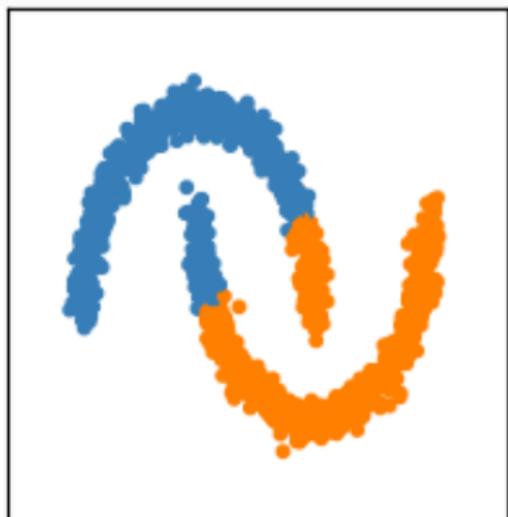
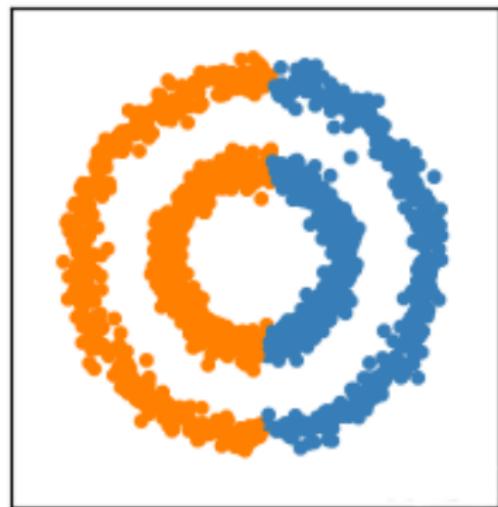
Optimal Clustering



Sub-optimal Clustering



## Shortcomings of $k$ -means



# *k*-means, Pros and Cons

## Strengths

- 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

## Weaknesses

- tends to converge to **local minimum**; sensitive to seed instances
- need to **specify  $k$  in advance**
- not able to handle non-convex clusters, or clusters of differing densities or sizes
- “mean” ill-defined for nominal or categorical attributes
- may not work well when the data contains outliers

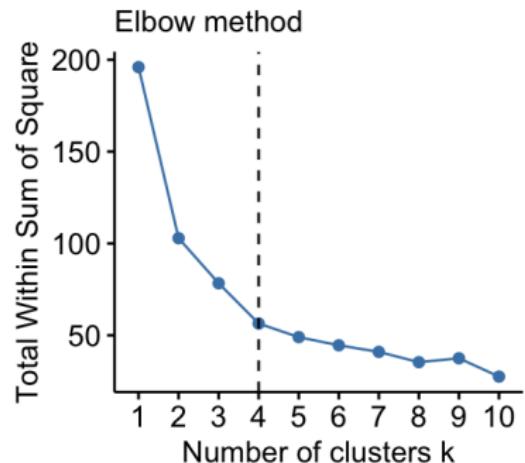


## How to choose the number of clusters?

- Calculate  $SSE$  for different number of clusters  $K$

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} (x - m_i)^2$$

- As  $K$  increases, we will have a smaller number of instances in each cluster  $\rightarrow SSE$  decreases
- **Elbow method:**  $K$  increases to  $K + 1$ , the drop of  $SSE$  starts to diminish



## Bottom-up (= agglomerative) clustering

- Start with single-instance clusters
- Iteratively **merge** clusters in a **bottom-up** fashion

## Top-down (= divisive) clustering

- Start with one universal cluster
- Iteratively **split** clusters in a **top-down** fashion

In contrast to  $k$ -means clustering, hierarchical clustering only requires a measure of similarity between *groups* of data points. No seeds, no  $k$  value.



# Agglomerative Clustering

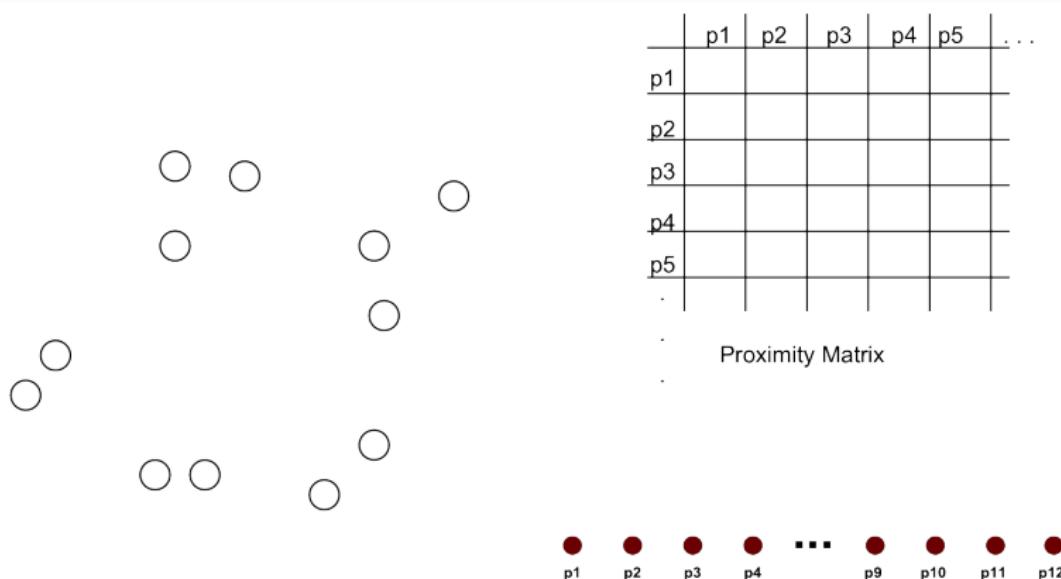
**Input:** A distance function

1. Compute the proximity matrix, if necessary.
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

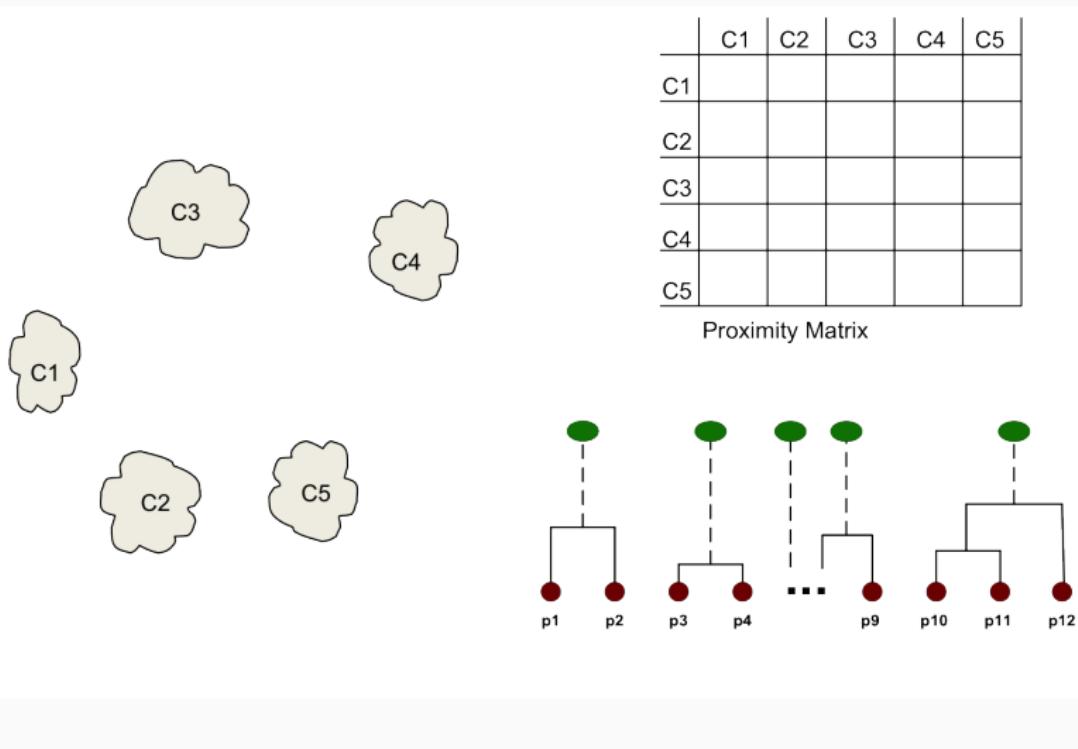
**Output:** A **dendrogram**: tree that represents the hierarchical clustering of all instances into successively smaller groups.



## Example, Step 1

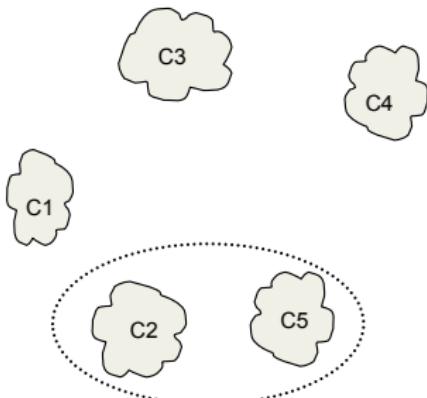


## Example, Step 2



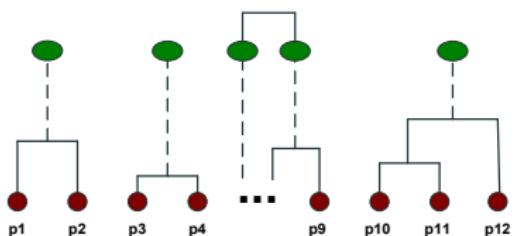
THE UNIVERSITY OF  
MELBOURNE

## Example, Step 3

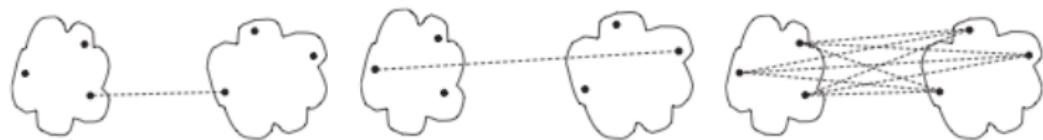


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



## 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

An initial **proximity matrix** with five clusters (aka. five points)

	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

An initial **proximity matrix** with five clusters (aka. five points)

	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: {1,2}



## Agglomerative Clustering Example

An initial **proximity matrix** with five clusters (aka. five points)

	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: {1,2}

Update (single link):

	1	2	3	4	5	{1,2}
{1,2}						



## Agglomerative Clustering Example

An initial **proximity matrix** with five clusters (aka. five points)

	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: {1,2}

Update (single link):

	1	2	3	4	5	{1,2}
{1,2}						

Update (complete link):

	1	2	3	4	5	{1,2}
{1,2}						

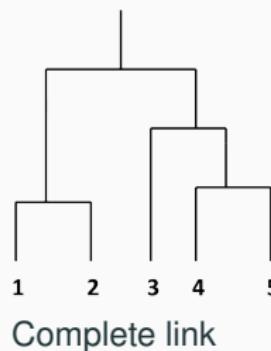
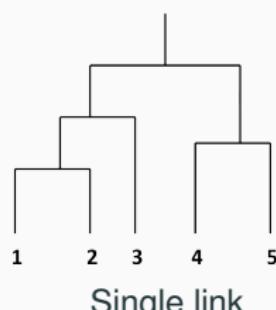


# Agglomerative Clustering Example

An initial **proximity matrix** with five clusters (aka. five points)

	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

The two resulting **dendograms**:



# Quick coding demo!

<https://docs.scipy.org/doc/scipy-1.2.1/reference/cluster.hierarchy.html>

SciPy.org    S powered by ENTHOUGHT

SciPy.org Docs SciPy v1.2.1 Reference Guide Clustering package (scipy.cluster)

This is documentation for an old release of SciPy (version 1.2.1). Read this page in the documentation of the latest stable release (version 1.6.3).

## Hierarchical clustering (scipy.cluster.hierarchy)

These functions cut hierarchical clusterings into flat clusterings or find the roots of the forest formed by a cut by providing the flat cluster ids of each observation.

`flatclust(Z, t[, criterion, depth, R, monocrit])` Form flat clusters from the hierarchical clustering defined by the given linkage matrix.

`flatclustdata(X, t[, criterion, metric, ...])` Cluster observation data using a given metric.

`leaders(Z, T)` Return the root nodes in a hierarchical clustering.

These are routines for agglomerative clustering.

`linkage(y, method='metric', optimal_ordering=False)` Perform hierarchical/agglomerative clustering.

`single(y)` Perform single/minimise/nearest linkage on the condensed distance matrix  $y$ .

`complete(y)` Perform complete/max/farthest point linkage on a condensed distance matrix.

`average(y)` Perform average/WPGMA linkage on a condensed distance matrix.

`weighted(y)` Perform weighted/WPGMA linkage on the condensed distance matrix.

`centroid(y)` Perform centroid/WPGMC linkage.

`median(y)` Perform median/WPGMC linkage.

`ward(y)` Perform Ward's linkage on a condensed distance matrix.

These routines compute statistics on hierarchies.

`cophenet(Z, Y=None)` Calculate the cophenetic distances between each observation in the hierarchical clustering defined by the linkage  $Z$ .

`from_mlab_linkage(Z)` Convert a linkage matrix generated by MATLAB(TM) to a new linkage matrix compatible with this module.

`inconsistency(Z[, d])` Calculate inconsistency statistics on a linkage matrix.

`maxinconsists(Z[, R])` Return the maximum inconsistency coefficient for each non-singleton cluster and its children.

`maxdists(Z[, R])` Return the maximum distance between any non-singleton cluster.

`maxstat(Z[, R, i])` Return the maximum statistic for each non-singleton cluster and its children.

`to_mlab_linkage(Z)` Convert a linkage matrix to a MATLAB(TM) compatible one.

Routines for visualizing flat clusters.

`dendrogram(Z[, p, truncate_mode, ...])` Plot the hierarchical clustering as a dendrogram.

These are data structures and routines for representing hierarchies as tree objects.

`ClusterNode(id[, left, right, dist, count])` A tree node class for representing a cluster.

`leaves_(init)Z` Return a list of leaf node ids.

`to_tree(Z[, rd])` Convert a linkage matrix into an easy-to-use tree object.

`cut_tree(Z[, n_clusters, height])` Given a linkage matrix  $Z$ , return the cut tree.

`optimal_leaf_ordering(Z, y[, metric])` Given a linkage matrix  $Z$  and distance, reorder the cut tree.

These are predicates for checking the validity of linkage and inconsistency matrices as well as for checking isomorphism of two flat cluster assignments.

Table Of Contents

- Hierarchical clustering (scipy.cluster.hierarchy)
  - References

Previous topic  
[scipy.cluster.vng.kmeans2](#)

Next topic  
[scipy.cluster.hierarchy.fcluster](#)

Quick search



THE UNIVERSITY OF MELBOURNE

## Top-down clustering: Bi-secting $k$ -means

Variant of K-means that can produce a partitional or a hierarchical clustering.

**Basic Idea:** to obtain  $K$  clusters, split the set of all points into two clusters, select one of these clusters to split, and so on, until  $K$  clusters have been produced.

1. Initialize the list of clusters to contain the cluster consisting of all points.
2. **repeat**
3.     Pick a cluster from the current set of clusters.
4.     {Perform several ‘trial’ bisections of the chosen cluster.}
5.     **for**  $i = 1$  to number of trials **do**
6.         Bisect the selected cluster using basic  $k$ -means.
7.     **end for.**
8.     Select the two clusters from the bisection with the lowest total SSE
9.     Add these two clusters to the set.
10. **until** We have produced  $k$  clusters.



# Final Thoughts

Clustering is in the eyes of the beholder

- “The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art [...]”<sup>1</sup> Cluster validation an open research area!

Other variants of **unsupervised learning** (beyond the scope of the subject)

- **Density Estimation:** Estimate the underlying (unobservable) probability distribution that explains the observed variables (features). E.g., Gaussian mixture models (also: ‘soft’ K-means)
- **Dimensionality Reduction:** Map original features to a smaller set of features that still explain the observed data well (either a subset, or a new set of features). E.g., Principle Component Analysis



---

<sup>1</sup>[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)

# Roadmap

## Today

- Unsupervised learning: Clustering
- Concepts
- Methods
- Evaluation

## Next up...

- Semi-supervised learning
- Anomaly detection
- Ethics in ML



## References

### Resources:

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)



# Lecture 18: Semi-Supervised and Active Learning

---

**COMP90049**

Semester 1, 2021

Lea Frermann, 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: Kris Ehinger, Tim Baldwin & Karin Verspoor



## **Semi-supervised Learning**

---

## Where we're at so far

- To date, we have talked a lot about **supervised learning** — where we have assumed (fully) labelled training data
- We also talked about **unsupervised learning** — where we have (fully) unlabelled training data
- What if we had a small amount of labelled training data, and lots of unlabelled training data?
- What if we had a small amount of labelled training data and a limited budget to label more training data?
- What if we can ‘warm-start’ our model by training it first on a (related) unsupervised task and then on the supervised target task?



## Why bother? (I)

**“Most Research in Deep Learning is a Total Waste of Time”**

Watch this short clip (in your own time) to get the gist of active learning (and some pieces of wisdom about scientific research vs. solving the world's real problems!)

<https://www.youtube.com/watch?v=Bi7f1JSSlh8>



## Why bother? (II)

- “Simple models and a lot of data trump more elaborate models based on less data!”<sup>1</sup>
- (Labelled) data is a bottleneck for machine learning
  - labels may require human experts
  - labels may require special devices
- unlabelled data is often cheap and available in large quantity



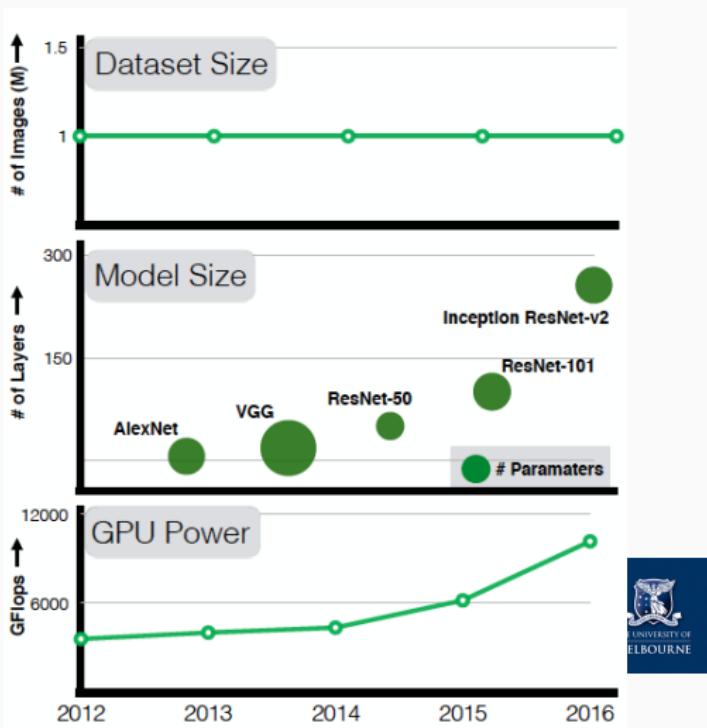
---

<sup>1</sup>Halevy, Norvig, & Pereira (2009) “The Unreasonable Effectiveness of Data”

## Example 1

### Image classification - Sun, Shrivastava, Singh, & Gupta (2017)

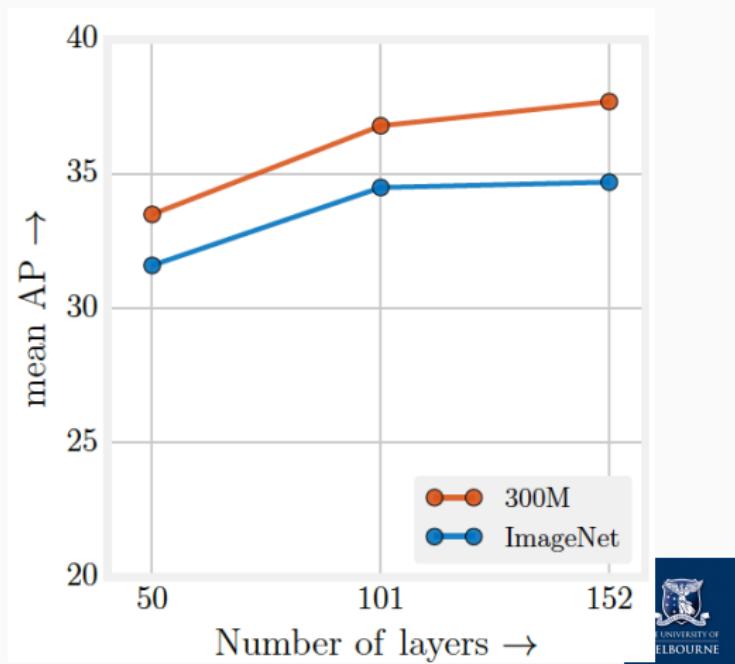
- model depth has increased dramatically
- AlexNet  $\approx$  10 layers  $\rightarrow$  ResNet  $>$  150 layers
- the size of “large scale” datasets has not kept pace
- 1 Million labelled images



## Example II

### Image classification - Sun, Shrivastava, Singh, & Gupta (2017)

- Adding data is nearly as effective as adding layers
- There is a limit to what a network can learn on a smaller dataset – more parameters are not helpful unless you have more data

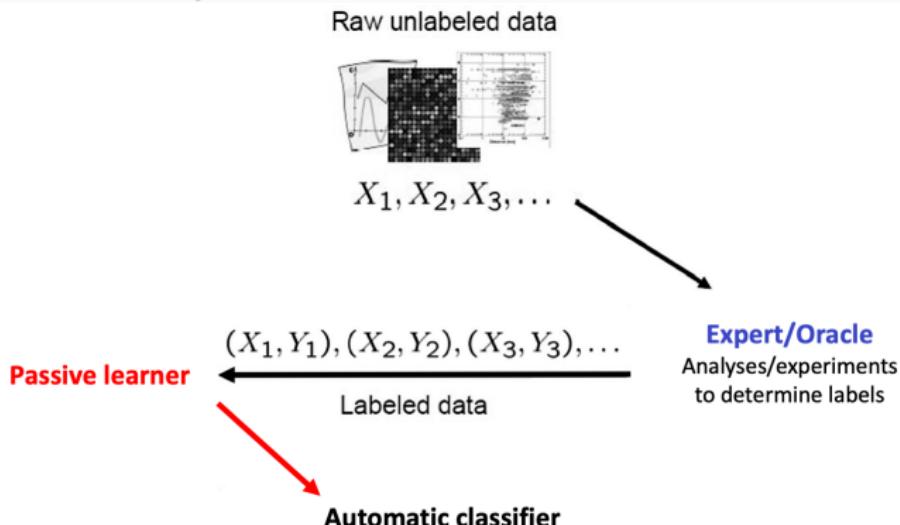


## **Semi-supervised learning**

---

# Supervised vs. Semi-supervised learning I

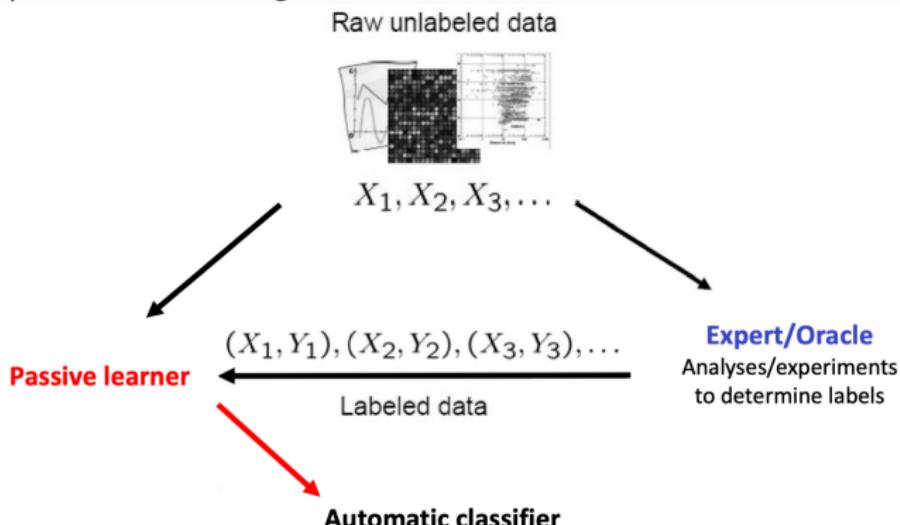
“Supervised” Learning:



THE UNIVERSITY OF  
MELBOURNE

# Supervised vs. Semi-supervised learning I

## “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



# Motivation of Semi-supervised learning

## Cognitive science

- model of how humans learn from labelled and unlabelled data
- concept learning in children:  $x = \text{animal}$ ,  $y = \text{concept}$  (e.g., dog)
- You point to a brown animal and say “dog!”
- Children also observe animals by themselves

## Hard-to-get Labels

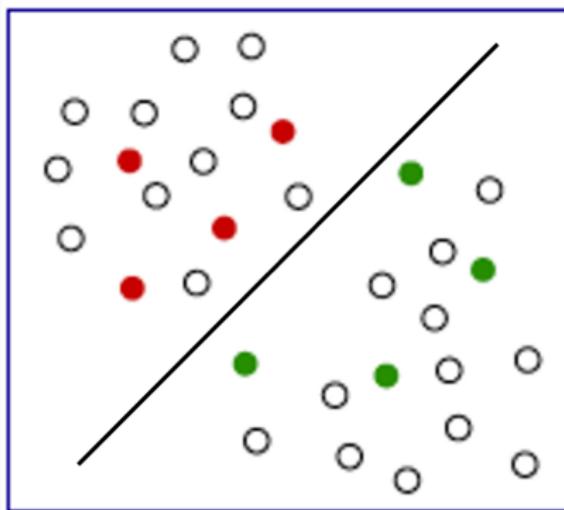
- Task: speech analysis
- Switchboard dataset and telephone conversation transcription
- 400 hours annotation time for each hour of speech



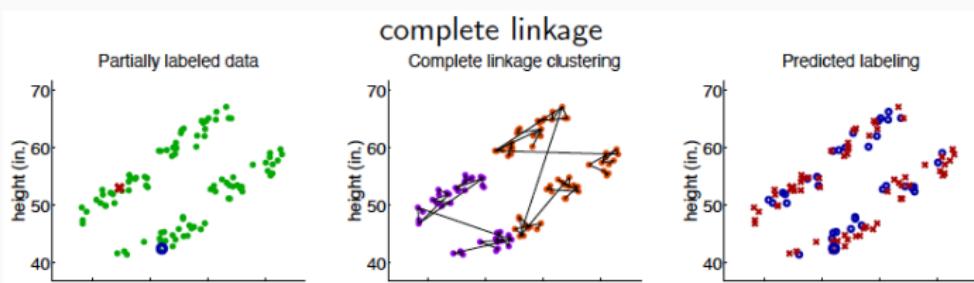
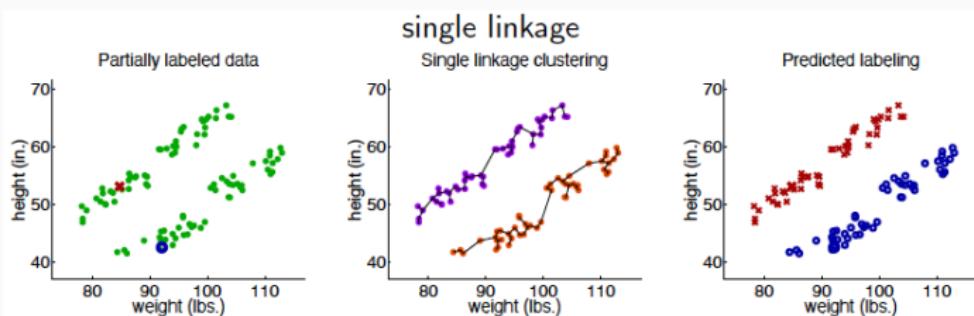
# Semi-Supervised Learning Approach I

## Clustering + Majority-voting

- A simple approach: combine a supervised and unsupervised model
- For example, Find clusters, choose a label for each (most common label?) and apply it to the unlabelled cluster members



# Semi-Supervised Learning Approach I



## 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
  - Train a model  $f$  on  $L$  using any supervised learning method
  - Apply  $f$  to predict the labels on each instance in  $U$
  - Identify a subset  $U'$  of  $U$  with “high confidence” labels
  - Remove  $U'$  from  $U$  and add it to  $L$  with the classifier predictions as the “ground-truth” labels ( $U \leftarrow U \setminus U'$  and  $L \leftarrow L \cup U'$ )
  - Until  $L$  does not change



## Self-Training Assumptions

- Propagating labels requires some assumptions about the distribution of labels over instances:
  - Points that are nearby are likely to have the same label
- Classification errors are propagated
  - One option is to move points back to the “unlabelled” pool if the classification confidence falls below a threshold
- Keep a kind of safety net...
  - Allow to move “bad” instances back into the unlabelled pool

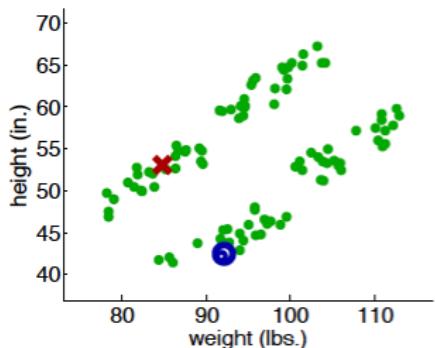


## Self-Training Example: 1-NN

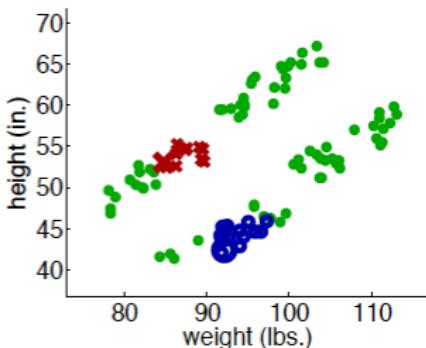
- 1-nearest neighbour with  $L = \{x_i, y_i\}_{i=1}^l$  labelled and  $U = \{x_i\}_{i=l+1}^{l+u}$  unlabelled training instances
- Repeat
  - Find neighbours for unlabelled instances in  $U$
  - For instances  $x$ , whose nearest neighbour is in  $L$ , take the labels  $y'$  from 1-NN
  - $U \leftarrow U \setminus \{x\}$
  - $L \leftarrow L \cup \{x, y'\}$
  - Until  $L$  does not change



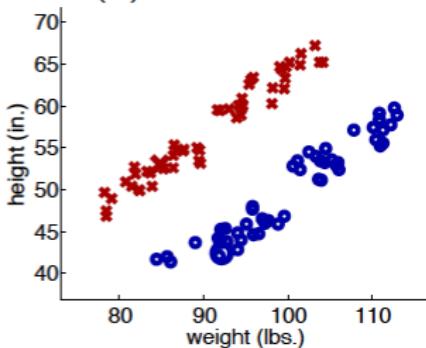
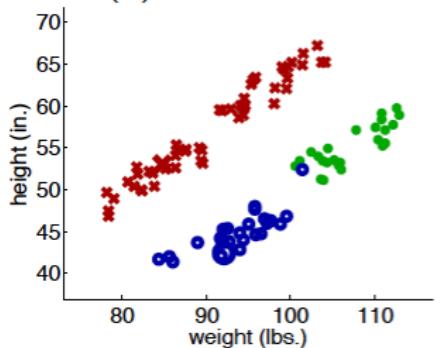
## Self-Training Example: 1-NN



(a) Iteration 1



(b) Iteration 25



## Self-Training Example: Naive Bayes

- Naive Bayes with  $L = \{x_i, y_i\}_{i=1}^l$  labelled and  $U = \{x_i\}_{i=l+1}^{l+u}$  unlabelled training instances
- Initialization: Train on  $L$  to learn  $P(X|Y)$  and  $P(Y)$  for all features  $X$  and all classes  $Y$
- Repeat (EM algorithm)
  - **Expectation:** For each unlabelled instance, compute a probability distribution over classes
  - **Maximization:** Recompute  $P(X|Y)$  and  $P(Y)$  with all data, weighting the unlabelled instances by their probability of being in each class



## Self-Training Example: Naive Bayes

- **Problem:** if the unlabelled dataset is much larger than the labelled dataset, probability estimates will be based almost entirely on unlabelled data

- **Solution:** Discuss!



## **Active Learning**

---

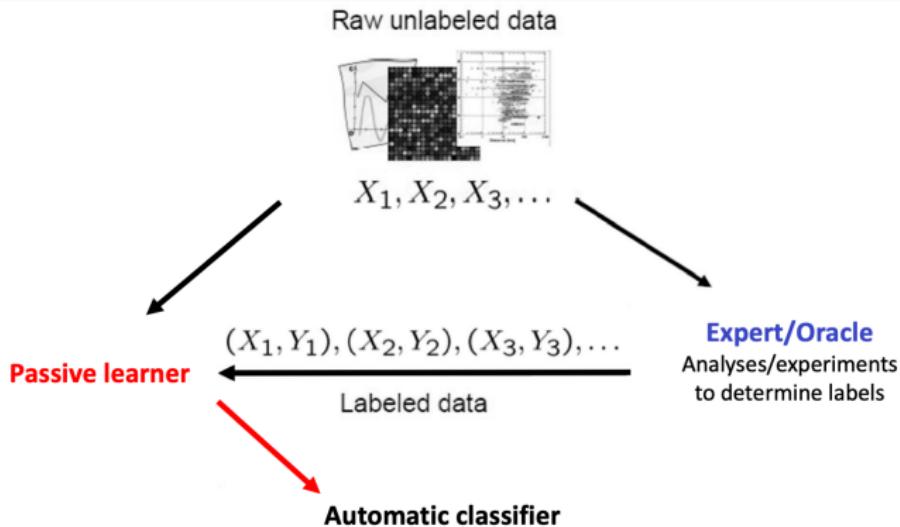
- Active learning builds off the hypothesis that a classifier can achieve higher accuracy with **fewer training instances** if it is allowed to have some say in the **selection** of the training instances
- The underlying assumption is that **labelling is a finite resource**, which should be expended in a way which optimises machine learning effectiveness
- Active learners pose **queries** (unlabelled instances) for labelling by an **oracle** (e.g. a human annotator)

Which instances are “most interesting”?



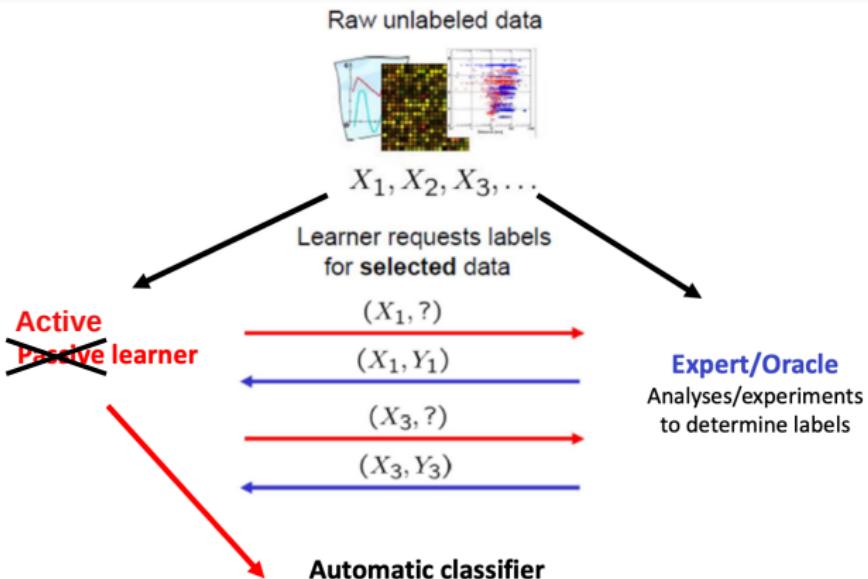
# Semi-supervised learning vs. Active Learning

## Semi-supervised Learning:



# Semi-supervised learning vs. Active Learning

## Active Learning:



- Ideally, we want to select the instances that are most effective for distinguishing between competing models
  - To do this most efficiently, we should have some sense of the likelihood of different models
  - or: knowledge of how labels are distributed over instances, which usually isn't the case
- In machine learning, querying generally focuses on instances with high uncertainty, e.g.:
  - Instances near the boundaries between classes
  - Instances in regions with few labels



# Query Strategies

Which unlabelled instances will be most useful for learning?

1. One simple strategy: query instances where the **classifier is least confident** of the classification

$$x = \underset{x}{\operatorname{argmax}}(1 - P_{\theta}(\hat{y}|x))$$

where

$$\hat{y} = \underset{y}{\operatorname{argmax}}(P_{\theta}(y|x))$$



# Query Strategies

Which unlabelled instances will be most useful for learning?

2. **Margin sampling** selects queries where the classifier is **least able to distinguish between two 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-probable labels for  $x$



## Query Strategies

Which unlabelled instances will be most useful for learning?

3. Use **entropy** as an uncertainty measure to utilize all the possible class probabilities:

$$x = \underset{x}{\operatorname{argmax}} - \sum_i P_{\theta}(\hat{y}_i|x) \log_2 P_{\theta}(\hat{y}_i|x)$$



## Which unlabelled instances will be most useful for learning?

4. A more complex strategy, if you have multiple classifiers: **query by committee (QBC)**
  - Train multiple classifiers on a labelled dataset, use each 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 entropy



## Active Learning Practicalities

Active learning is used increasingly widely, but must be handled with some care:

- empirically shown to be a robust strategy, but a theoretical justification has proven elusive
- querying is inherently biased towards a particular class set and learning approach(es), which may limit the general utility of the resulting dataset
- results to suggest that active learning is more highly reliant on “clean” labelling



## **Data Augmentation**

---

# Data Augmentation

- There are various ways to **expand** a labelled training dataset
- **General:** re-sampling methods
- **Dataset-specific:** add artificial variation to each instance, without changing ground truth label



## Bootstrap sampling

- Bootstrap sampling: create “new” datasets by resampling existing data, with or without replacement
- Common in perceptron and neural network training (“mini-batch”, “batch size”), methods that involve stochastic gradient descent
- Each “batch” has a slightly different distribution of instances, forces model to use different features and not get stuck in local minima



## Data Manipulation

- Another option: add a small amount of noise to each instance to create multiple variations:
  - Images: adjust brightness, flip left-right, shift image up /down / left / right, resize, rotate
  - Audio: adjust volume, shift in time, adjust frequencies
  - Text: synonym substitution
- These perturbations should not change the instance's label
- Generally, they should be the same kind of variations you expect in real-world data



# Data Augmentation Pros and Cons

## Advantages

- More data nearly always improves learning
- Most learning algorithms have some robustness to noise (e.g., from machine-translation errors)

## Disadvantages

- Biased training data
- May introduce features that don't exist in the real world
- May propagate errors
- Increases problems with interpretability and transparency

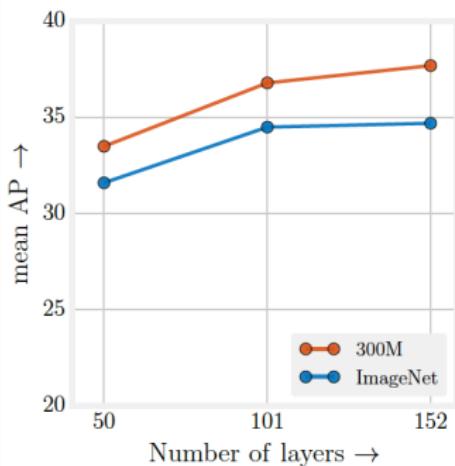


## **Unsupervised pre-Training: The secret sauce of (recent) deep learning success**

---

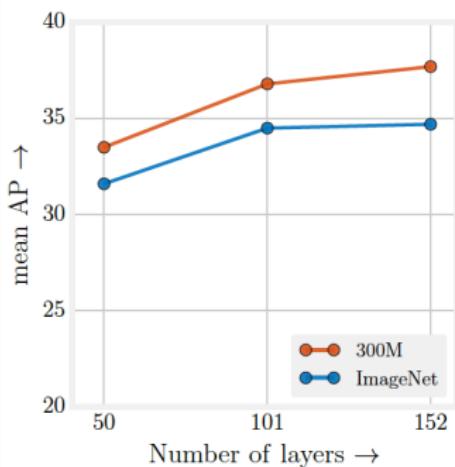
# Why is deep learning so successful?

- Better models (recurrent models, convolutional, activation functions, ...)
- Bags of tricks (dropout, mini-batching, layer normalization, ...)
- More powerful machines (GPUs)
- **More data**



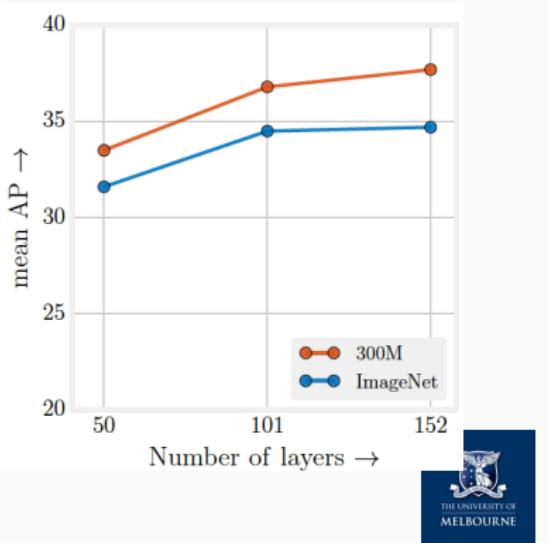
# Why is deep learning so successful?

- Better models (recurrent models, convolutional, activation functions, ...)
- Bags of tricks (dropout, mini-batching, layer normalization, ...)
- More powerful machines (GPUs)
- **More data – but we cannot label it all!**



# Why is deep learning so successful?

- Better models (recurrent models, convolutional, activation functions, ...)
- Bags of tricks (dropout, mini-batching, layer normalization, ...)
- More powerful machines (GPUs)
- **More data** – but we cannot label it all!
  - Pre-train (reuseable) parameters on some **unsupervised** task
  - Use the pre-trained weights to **initialize** your final model
  - Fine-tune the final model on a (usually) **supervised** target task.



# Unsupervised Pre-training in NLP (taster)

## Unsupervised pre-training in Natural Language Processing. Pre-training word embeddings.

Use the output of the masked word's position to predict the masked word

Possible classes:  
All English words



FFNN + Softmax

Randomly mask 15% of tokens

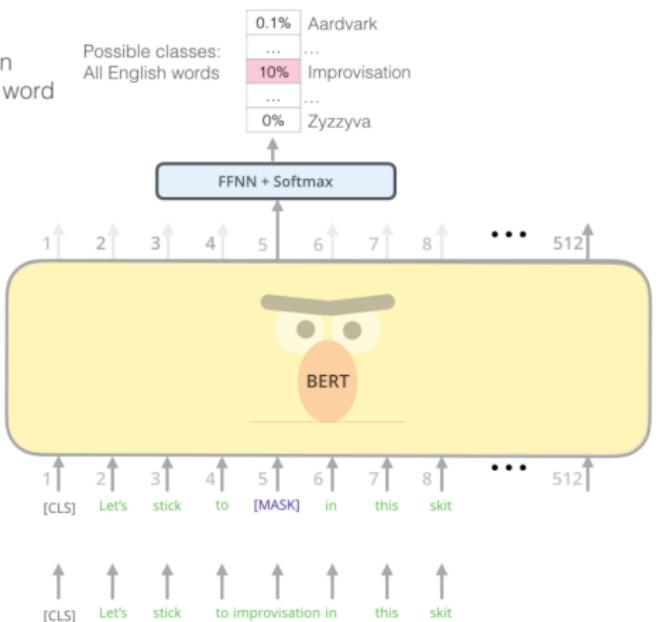


Image: <http://jalamar.github.io/illustrated-bert/>



# Unsupervised Pre-training in NLP (taster)

**Unsupervised pre-training in Natural Language Processing.** Pre-training word embeddings.

- **Input:** “The girl is coding a [MASK] network using Python.”
- **Task:** Predict the **hidden** word given its context
- **Model:** Neural networks, increasingly complex (GLOVE, BERT, GPT-2, ...)
- **Output:** A neural network which is a function:  $f(\text{word}) \rightarrow \text{feature\_vector}$

Use these feature vector to map language input → machine-readable representation. Use the representations in your **final** supervised text classification model.

...sounds familiar? :)



## **Summary**

---

# Summary

- What is semi-supervised learning?
- What is self-training, and how does it operate?
- What is active learning?
- What are the main sampling / query strategies in active learning?
- Pre-training in modern deep learning



## 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.
- Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019.





# Lecture 19: Anomaly Detection

**COMP90049**  
**Introduction to Machine Learning**  
Semester 1, 2021

Lea Frermann, CIS

© 2020 The University of Melbourne  
Acknowledgement: Lida Rashidi

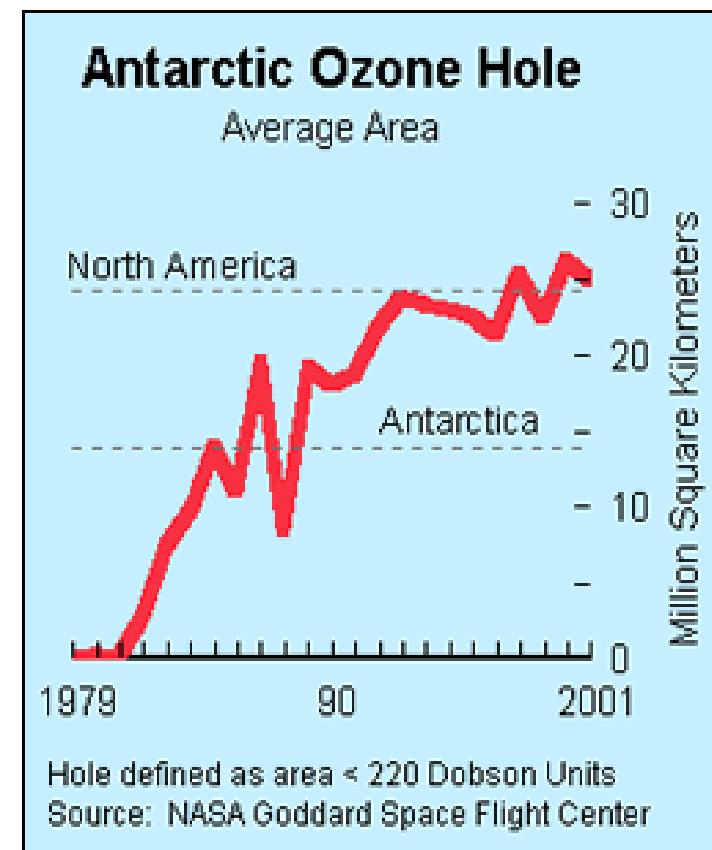
- Anomaly Detection
  - Definition
  - Importance
  - Structure
- Anomaly Detection Algorithms
  - Statistical
  - Proximity-based
  - Density-based
  - Clustering-based
- Summary

- **Anomaly:** A data object that **deviates significantly** from the normal objects as if it were **generated by a different mechanism**
  - Ex.: Unusual credit card purchase, sports: Michael Jordon, ...
- Anomalies are different from **noise**
  - Noise is random error or variance in a measured variable
  - 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
    - Cyber intrusions
    - Credit card fraud

# Importance of Anomaly Detection?

## Ozone Depletion History

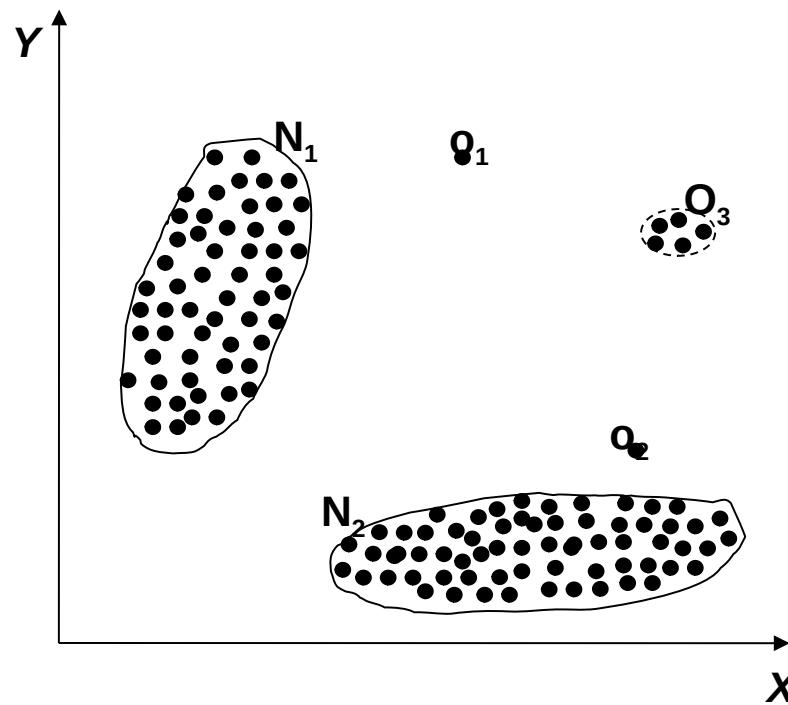
- In 1985 three researchers (Farman, Gardinar and Shanklin) were puzzled by data gathered by the British Antarctic Survey showing that ozone levels for Antarctica had dropped 10% below normal levels
- Why did the Nimbus 7 satellite, which had instruments aboard for recording ozone levels, not record similarly low ozone concentrations?
- The ozone concentrations recorded by the satellite were so low they were being treated as noise by a computer program and discarded!



- Variants of Anomaly/Outlier Detection Problems
  - Given a database  $D$ , find all the data points  $x \in D$  with anomaly scores **greater than some threshold  $t$**
  - Given a database  $D$ , find all the data points  $x \in D$  having the **top-n largest anomaly scores  $f(x)$**
  - Given a database  $D$ , containing mostly normal (but unlabeled) data points, and a **test point  $x$** , compute the **anomaly score** of  $x$  with respect to  $D$

- Global/Point anomalies
- Contextual/Conditional anomalies
- Collective anomalies

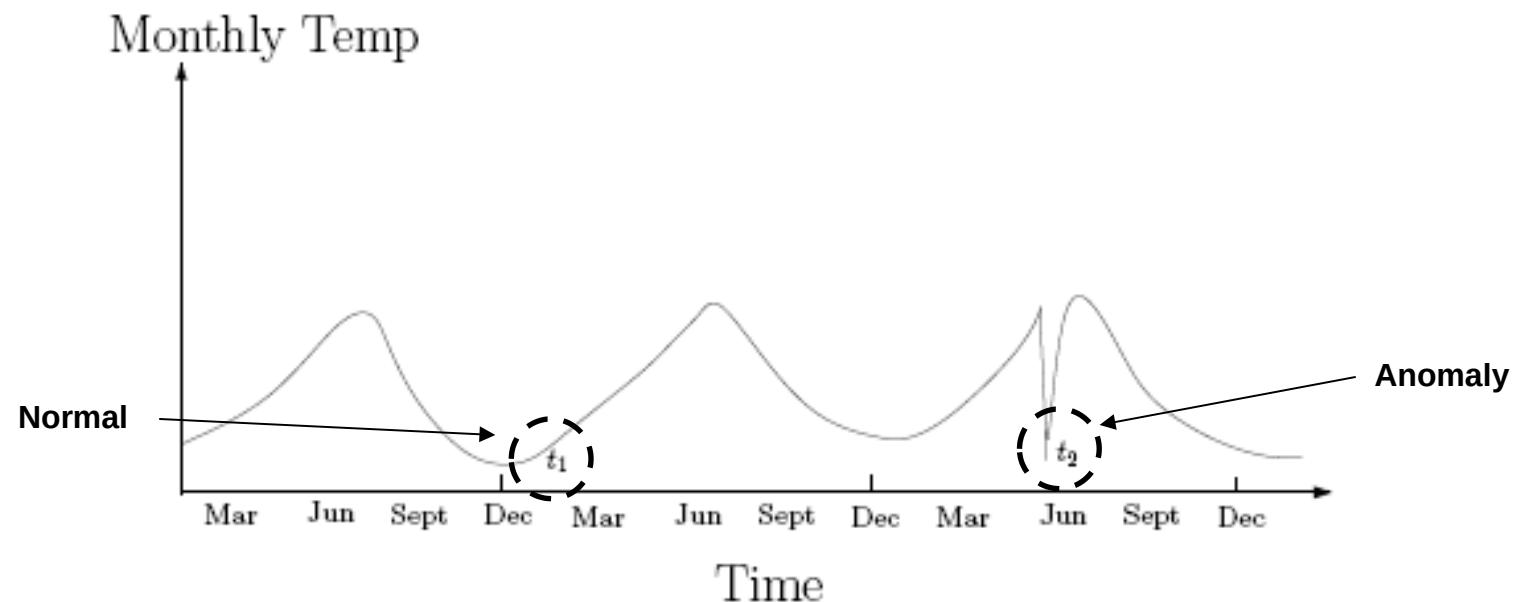
- **Global Anomaly** (or point)
  - Object is  $O_g$  if it **significantly deviates from the rest of the data** set
  - Ex. Intrusion detection in computer networks
  - Issue: Find an appropriate measurement of deviation



- **Contextual Anomaly (or *conditional*)**
  - Object is  $O_c$  if it **deviates significantly based on a selected context**
  - Attributes of data objects should be divided into two groups
    - **Contextual attributes:** defines the context, e.g., time & location
    - **Behavioral attributes:** characteristics of the object, used in anomaly evaluation, e.g., temperature
  - Can be viewed as a generalization of *local anomalies*—whose density significantly deviates from its local area
  - Issue: How to define or formulate meaningful context?

\* Song, et al, “Conditional Anomaly Detection”, IEEE Transactions on Data and Knowledge Engineering, 2006.

Ex. 10° C in Paris: Is this an anomaly?

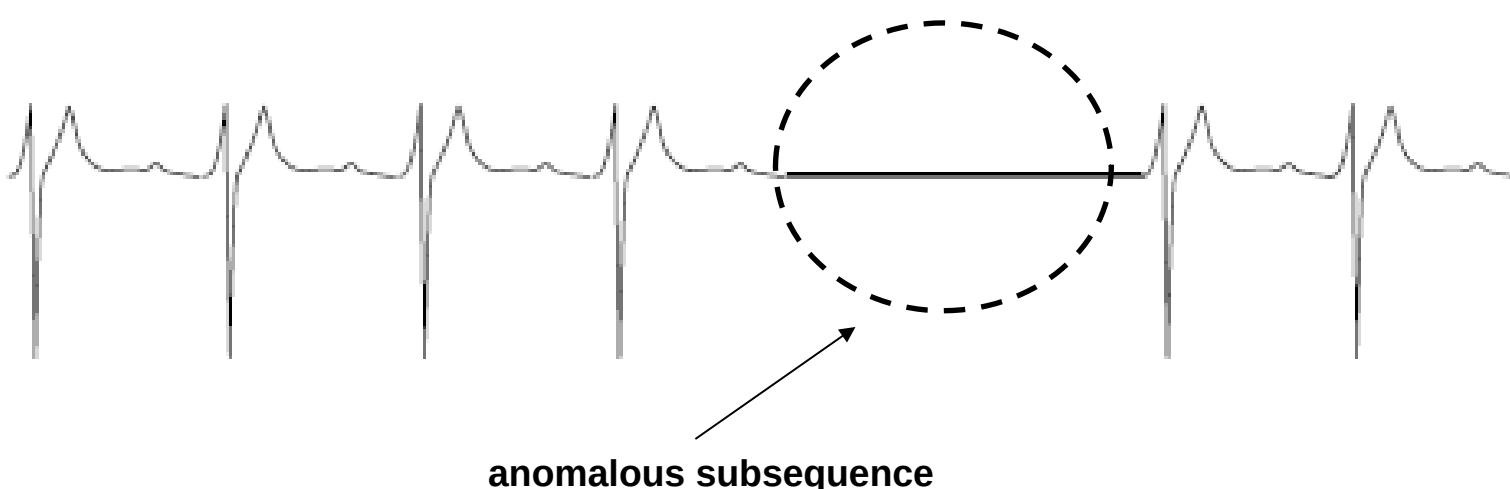


## Collective Anomalies

- A **subset** of data objects that **collectively deviate** significantly from the whole data set, even if the individual data objects may not be anomalies
- E.g., *intrusion detection*:
  - When a number of computers keep sending denial-of-service packages to each other
- Detection of collective anomalies
  - Consider not only behavior of individual objects, but also that of **groups** of objects
  - Requires background **knowledge about the relationship** among data objects, such as a distance or similarity measure on objects.

# Example of Collective anomalies

- Requires a relationship among data instances
  - Sequential data
  - Spatial data
  - Graph data
- The individual instances within a collective anomaly are not anomalous by themselves



# **Anomaly detection paradigms: supervised, semi-supervised, and unsupervised**

## Supervised anomaly detection

- Labels available for both **normal data and anomalies**
- Samples examined by domain experts used for training & testing
- Challenges
  - Require both **labels** from both normal and anomaly class
  - **Imbalanced** classes, i.e., anomalies are rare: Boost the anomaly class and make up some artificial anomalies
  - Cannot detect **unknown** and emerging anomalies
  - Catch as many outliers as possible, i.e., **recall** is more important than accuracy (i.e., not mislabeling normal objects as outliers)

## Semi-Supervised anomaly detection

- Labels available only for **normal** data
- Model normal objects & 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 anomaly detection

- Assume the **normal** objects are somewhat "clustered" into multiple **groups**, each having some **distinct features**
- An outlier is expected to be **far away from any groups** of normal objects
- **General steps**
  - Build a profile of “normal” behavior
    - summary statistics for overall population
    - model of multivariate data distribution
  - Use the “normal” profile to detect anomalies
    - anomalies are observations whose characteristics differ significantly from the normal profile

## Unsupervised anomaly detection **Challenges**

- Normal objects may not share any strong patterns, but the collective outliers may share high similarity in a small area
- *Ex.* In some intrusion or virus detection, normal activities are diverse
  - Unsupervised methods may have a high false positive rate but still miss many real outliers.

Many clustering methods can be adapted for unsupervised methods

- Find clusters, then outliers: not belonging to any cluster
- *Problem 1:* Hard to distinguish noise from outliers
- *Problem 2:* Costly since first clustering: but far less outliers than normal objects

- Statistical (or: model-based)
  - Assume that normal data follow some statistical model
- Proximity-based
  - An object is an outlier if the nearest neighbors of the object 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

Anomalies are objects that are fit poorly by a statistical model.

- **Idea:** learn a model fitting the given data set, and then identify the objects in **low probability regions** of the model as anomalies
- **Assumption:** normal **data is generated by a parametric distribution** with parameter  $\theta$ 
  - The probability density function of the parametric distribution  $f(x, \theta)$  gives the probability that object  $x$  is generated by the distribution
  - The smaller this value, the more likely  $x$  is an outlier
- **Challenges** of Statistical testing:
  - highly depends on whether the assumption of statistical model holds in the real data

## Graphical Approaches

- Boxplot (1-D), Scatter plot (2-D), Spin plot (3-D)
- Limitations: Time consuming, Subjective



Image: [https://en.wikipedia.org/wiki/Box\\_plot#/media/File:Boxplot\\_with\\_outlier.png](https://en.wikipedia.org/wiki/Box_plot#/media/File:Boxplot_with_outlier.png)

Avg. temp.:  $x = \{24.0, 28.9, 28.9, 29.0, 29.1, 29.1, 29.2, 29.2, 29.3, 29.4\}$

- Use the **maximum likelihood** method to estimate  $\mu$  and  $\sigma$

$$\hat{\mu} = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

- For the above data  $x$  with  $n = 10$ :

$$\hat{\mu} = 28.61 \quad \hat{\sigma} = \sqrt{2.29} = 1.51$$

- Decide on a confidence limits, e.g.,  $\mu \pm 3\sigma$  region contains 99.7% data

- Then 24 is an outlier since:

$$(24 - 28.61) / 1.51 = -3.04 < -3$$

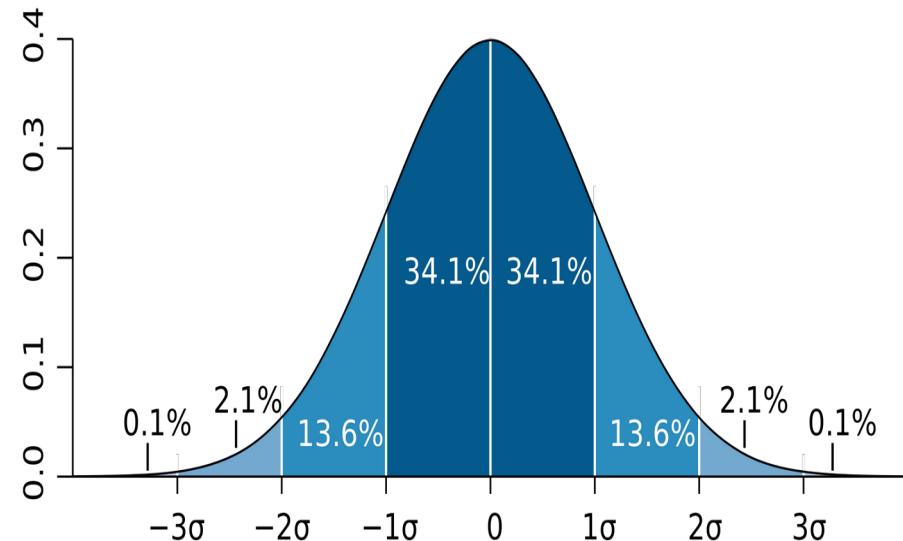
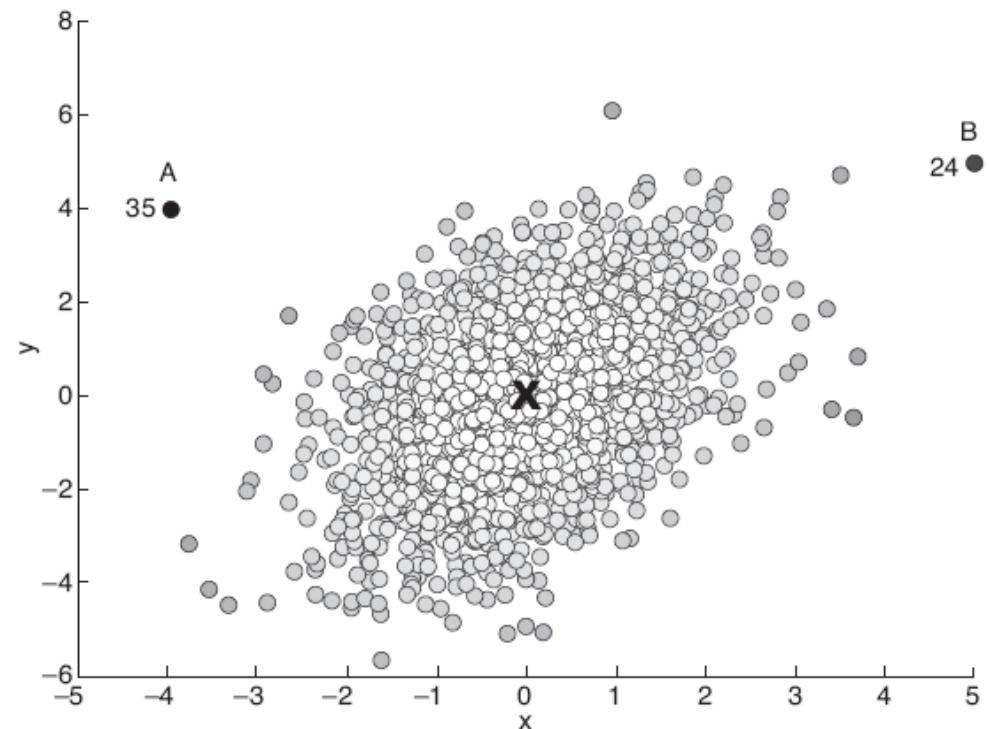


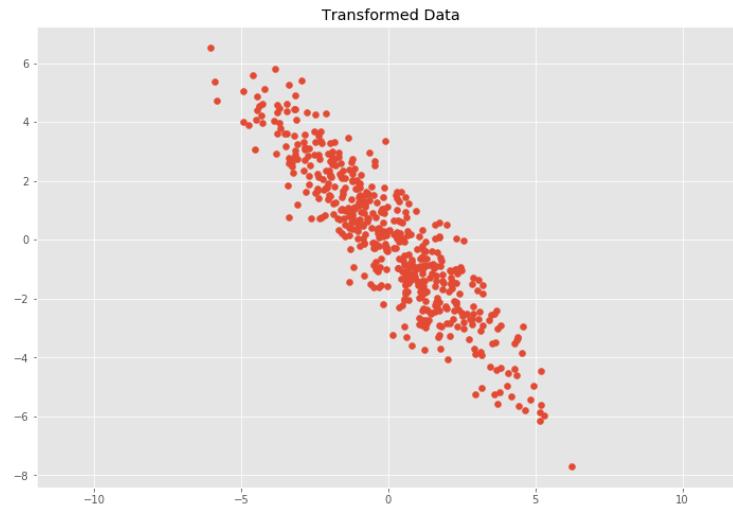
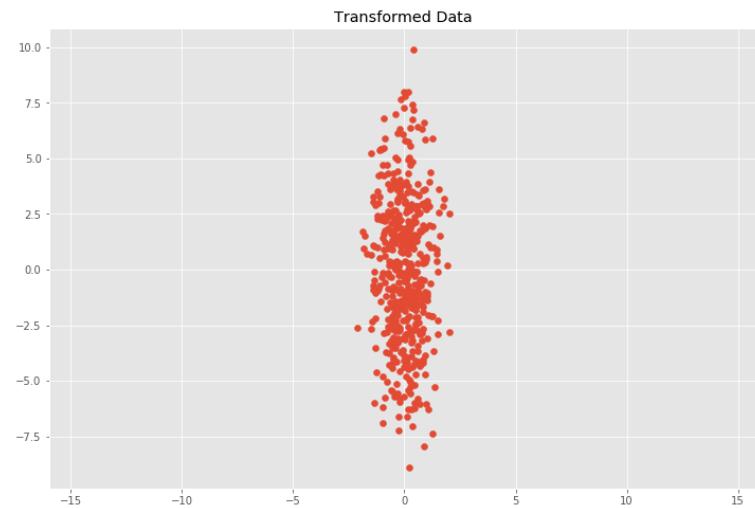
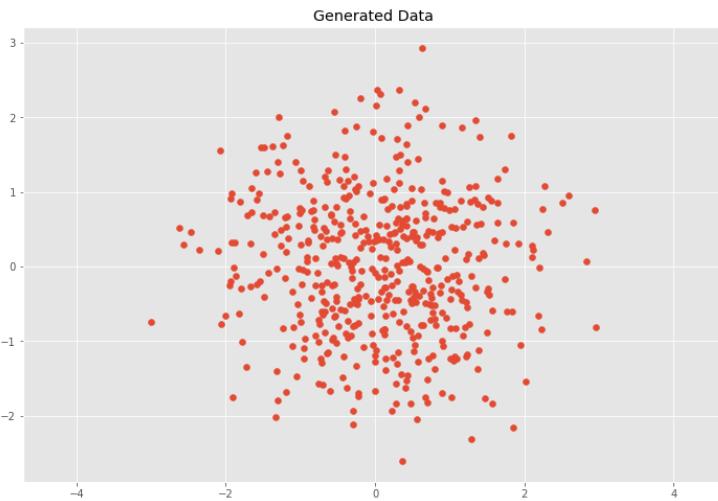
Image: [https://en.wikipedia.org/wiki/Standard\\_deviation#/media/File:Standard\\_deviations\\_diagram.svg](https://en.wikipedia.org/wiki/Standard_deviation#/media/File:Standard_deviations_diagram.svg)

- Multivariate Gaussian distribution
  - Outlier defined by **Mahalanobis distance**
  - Grubb's test on the distances

Distance		
	Euclidean	Mahalanobis
A	5.7	35
B	7.1	24



# Mahalanobis Distance



- Mahalanobis Distance

$$y^2 = (\mathbf{x} - \bar{\mathbf{x}})' S^{-1} (\mathbf{x} - \bar{\mathbf{x}})$$

- $S$  is the covariance matrix:

$$S = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})'$$

- For 2-dimensional data:

$$\begin{pmatrix} \sigma(x, x) & \sigma(x, y) \\ \sigma(y, x) & \sigma(y, y) \end{pmatrix}$$

- **Assume** the dataset D contains samples from a mixture of two probability distributions:
  - M (majority distribution)
  - A (anomalous distribution)
- **General approach:**
  - Initially, assume all the data points belong to M
  - Let  $L_t(D)$  be the log likelihood of D at time t
  - For each point  $x_t$  that belongs to M, move it to A
    - Let  $L_{t+1}(D)$  be the new log likelihood.
    - Compute the difference,  $\Delta = L_t(D) - L_{t+1}(D)$
    - If  $\Delta > c$  (some threshold), then  $x_t$  is declared as an anomaly and moved permanently from M to A

Data distribution,  $D = (1 - \lambda) M + \lambda A$

- $M$  is a probability distribution estimated from data
- $A$  is initially assumed to be uniform distribution
- Likelihood at time  $t$ :

$$L_t(D) = \prod_{i=1}^N P_D(x_i) = \left( (1 - \lambda)^{|M_t|} \prod_{x_i \in M_t} P_{M_t}(x_i) \right) \left( \lambda^{|A_t|} \prod_{x_i \in A_t} P_{A_t}(x_i) \right)$$

$$LL_t(D) = |M_t| \log(1 - \lambda) + \sum_{x_i \in M_t} \log P_{M_t}(x_i) + |A_t| \log \lambda + \sum_{x_i \in A_t} \log P_{A_t}(x_i)$$

- **Pros**

- Statistical tests are well-understood and well-validated.
- Quantitative measure of degree to which object is an outlier.

- **Cons**

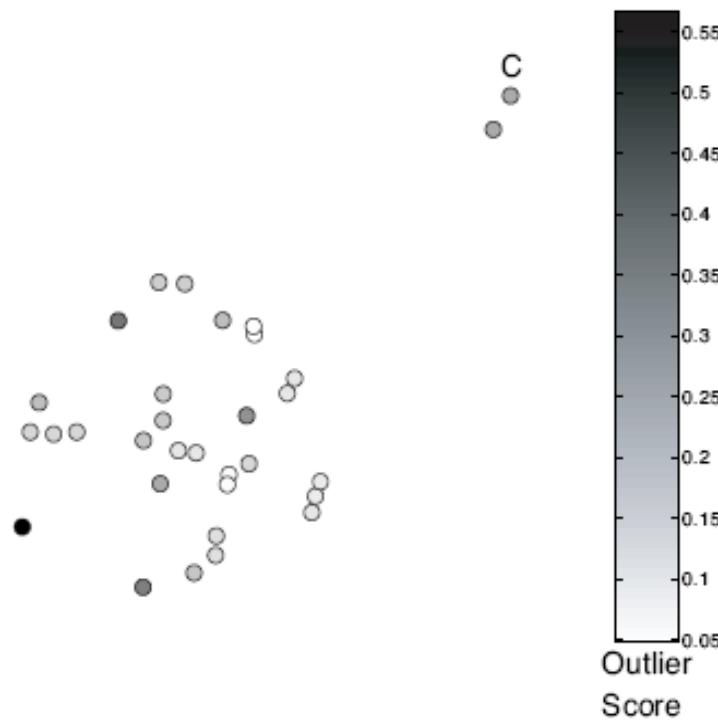
- Data may be hard to model parametrically.
  - multiple modes
  - variable density
- In high dimensions, data may be insufficient to estimate true distribution.

# Proximity-based Anomaly detection

Anomalies are objects far away from other objects.

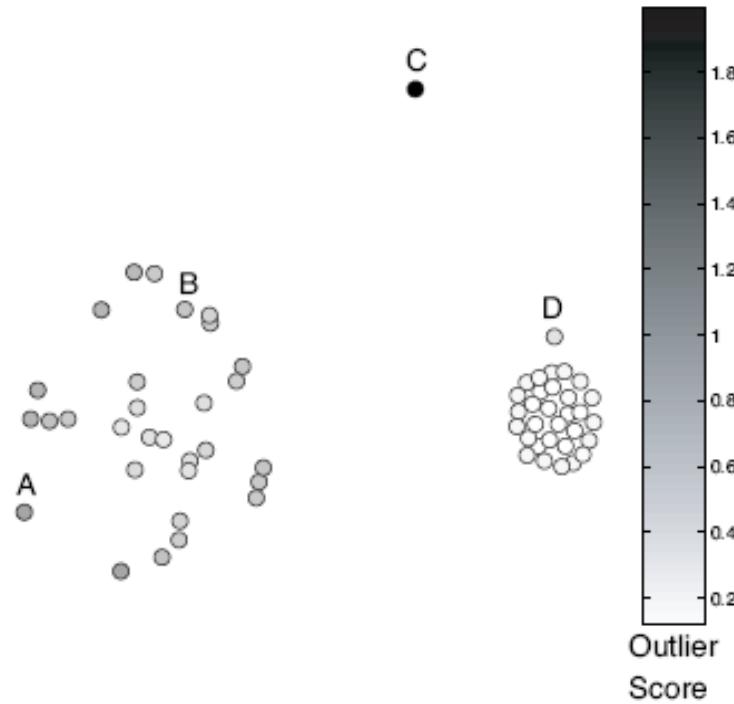
- An object is an **anomaly** if the nearest neighbors of the object are **far** away, i.e., the **proximity** of the object significantly deviates from the proximity of most of the other objects in the same data set
- Common approach:
  - Outlier score is distance to  $k^{\text{th}}$  nearest neighbor.
  - Score sensitive to choice of  $k$ .

# 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.

## 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.
- Deals naturally with multiple modes.

## Cons

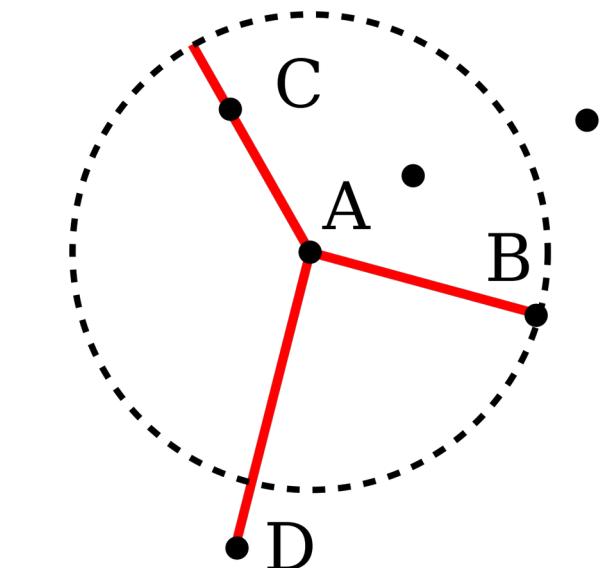
- $O(n^2)$  complexity.
- Score sensitive to choice of  $k$ .
- Does not work well if data has widely variable density.

## **Density-based Anomaly detection**

**Outliers are objects in regions of **low density**.**

- Outlier score is the **inverse of the density** around a point
- Scores usually based on **proximities**.
- Example scores:
  - # points within a fixed radius  $d$
  - Reciprocal of average distance to  $k$  nearest neighbors:

$$\text{density}(\mathbf{x}, k) = \left( \frac{1}{k} \sum_{\mathbf{y} \in N(\mathbf{x}, k)} \text{distance}(\mathbf{x}, \mathbf{y}) \right)^{-1}$$

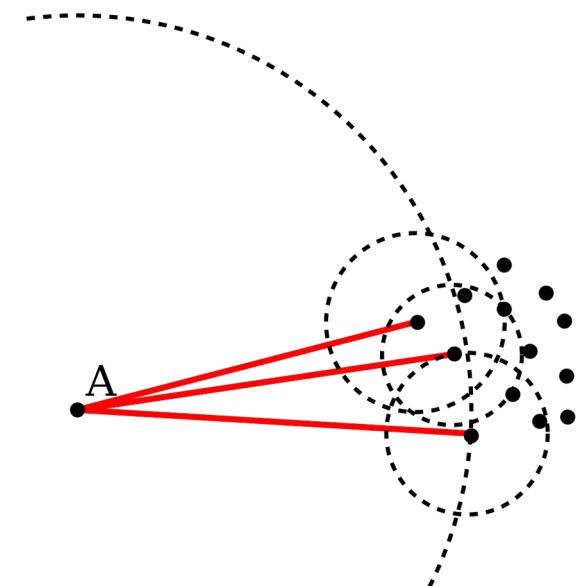


Tend to work **poorly** if data has **variable density**.

*Image: [https://en.wikipedia.org/wiki/Local\\_outlier\\_factor#/media/File:Reachability-distance.svg](https://en.wikipedia.org/wiki/Local_outlier_factor#/media/File:Reachability-distance.svg)*

## Relative density outlier score

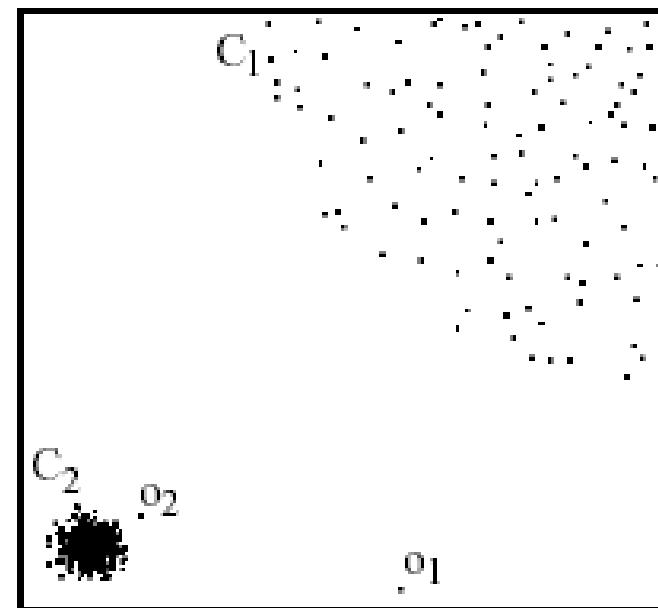
- Local Outlier Factor (LOF)
- **Reciprocal** of average distance to  $k$  nearest neighbors, relative to that of those  $k$  neighbors.



$$\text{relative density}(\mathbf{x}, k) = \frac{\text{density}(\mathbf{x}, k)}{\frac{1}{k} \sum_{\mathbf{y} \in N(\mathbf{x}, k)} \text{density}(\mathbf{y}, k)}$$

Image: <https://en.wikipedia.org/wiki/File:LOF-idea.svg>

In the NN approach,  $o_2$  is not considered as outlier, while LOF approach find both  $o_1$  and  $o_2$  as outliers!



## Pros

- Quantitative measure of degree to which object is an outlier.
- Can work well even if data has variable density.

## Cons

- $O(n^2)$  complexity
- Must choose parameters
  - $k$  for nearest neighbor
  - $d$  for distance threshold

# **Cluster-based Anomaly Detection**

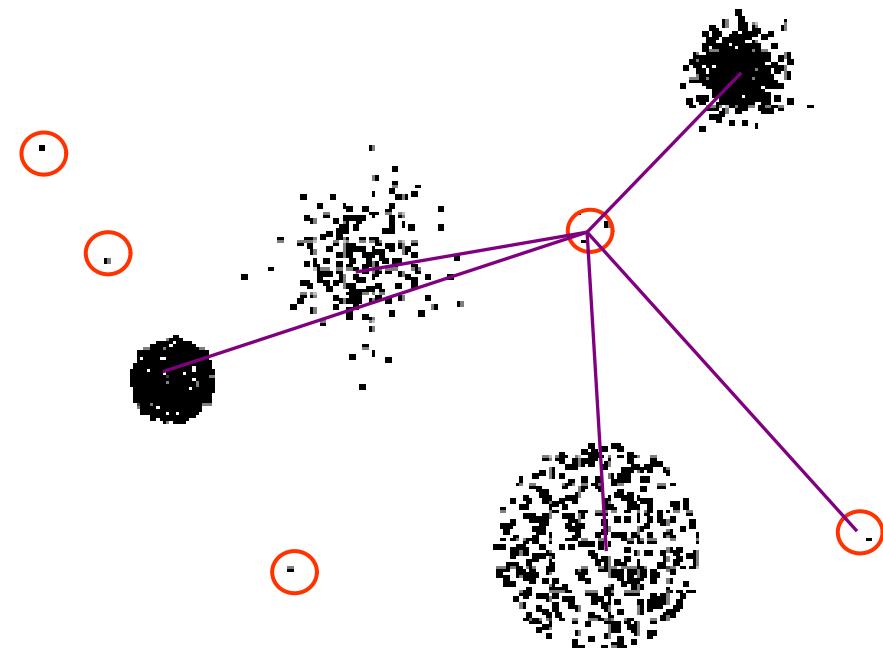
**Outliers are objects that do not belong strongly to any cluster.**

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.



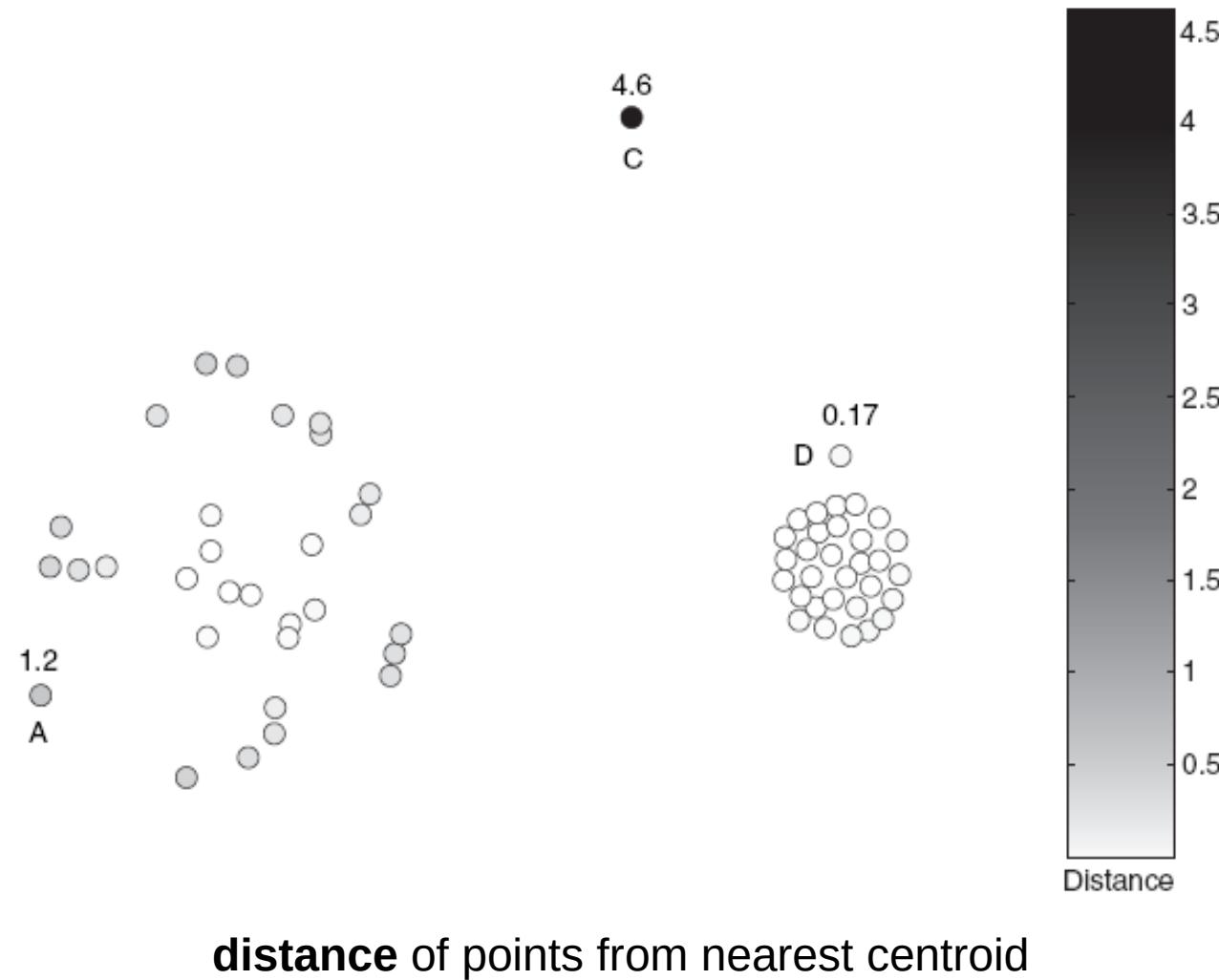
**Assess degree to which object belongs to any cluster.**

- For prototype-based clustering (e.g. k-means), use distance to cluster centers.
- To deal with variable density clusters, use relative distance:

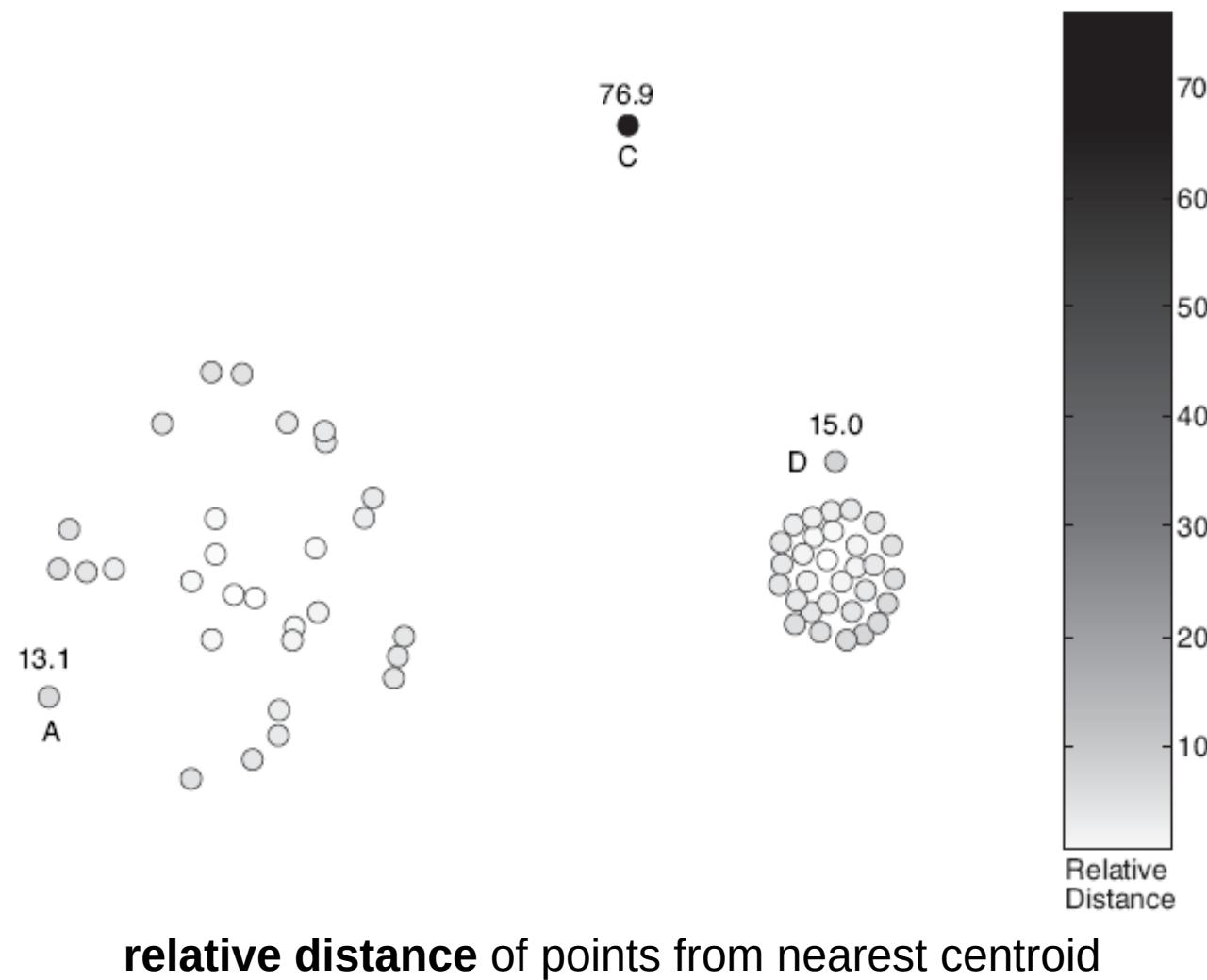
$$\frac{\text{distance}(\mathbf{x}, \text{centroid}_C)}{\text{median}(\{ \forall_{x' \in C} \text{distance}(\mathbf{x}', \text{centroid}_C) \})}$$

- Similar concepts for density-based or connectivity-based clusters.

# Cluster-based outlier detection



# Cluster-based outlier detection



**Eliminate object(s) to improve objective function.**

- 1) Form initial set of clusters.
- 2) Remove the object which most improves objective function.
- 3) Repeat step 2) until ...

Discard small clusters far from other clusters.

- Need to define thresholds for “small” and “far”.

## Pros:

- Some clustering techniques have  $O(n)$  complexity.
- Extends concept of outlier **from single objects to groups** of objects.

## Cons:

- Requires thresholds for minimum size and distance.
- Sensitive to number of clusters chosen.
- Hard to associate outlier score with objects.
- Outliers may affect initial formation of clusters.

## Today

- Types of outliers
- Supervised, semi-supervised, or unsupervised
- Statistical, proximity-based, clustering-based approaches

## Next up

- Ethics in Machine Learning
- Wednesday: Guest lecture, part I (pre-recorded)
- Friday: normal lecture
- Wednesday: Guest lecture, part II (live)

## 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)