

Introduction to Machine Learning

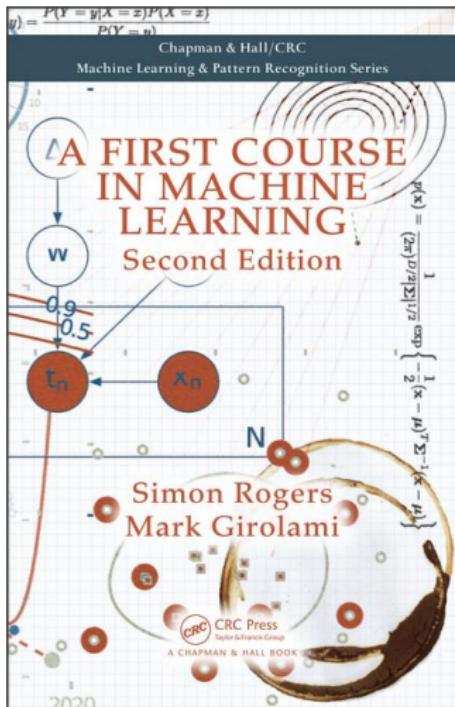
Mauricio A. Álvarez

Machine Learning and Adaptive Intelligence
The University of Sheffield

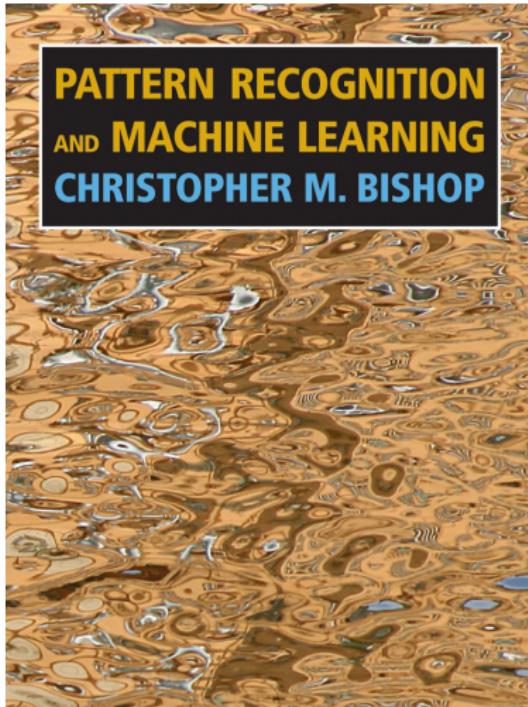


The
University
Of
Sheffield.

Textbooks

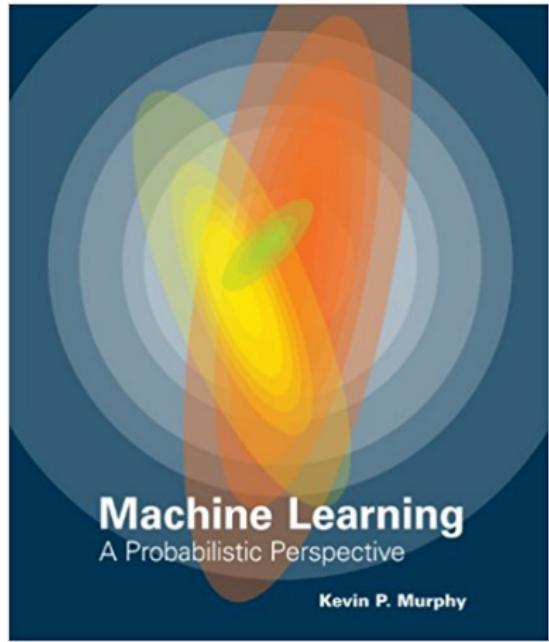


Rogers and Girolami, *A First Course in Machine Learning*, Chapman and Hall/CRC Press, 2nd Edition, 2016.

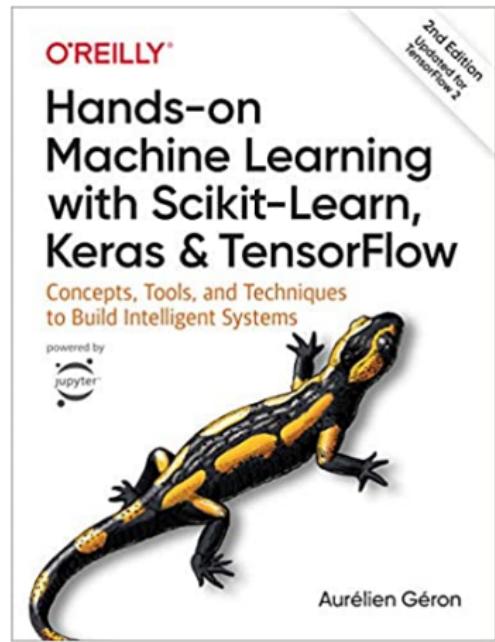


Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, 2006.

Textbooks



Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.



Géron, *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow*, O'Reilly, 2nd Edition, 2019.

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

Machine learning or Statistical Learning

- We would like to design an algorithm that help us to solve different prediction problems.
- The algorithm is designed based on a mathematical model or function, and a dataset.
- Extract knowlegde from data.

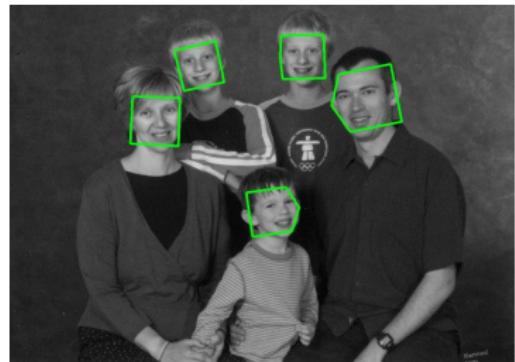
Examples of ML problems

Handwritten digit recognition



Examples of ML problems

Face detection and face recognition



From Murphy (2012).

Examples of ML problems

Predicting the age of a person looking at a particular YouTube video.



Examples of ML problems

Stock market



Examples of ML problems

Clustering: segmenting customers in e-commerce



Examples of ML problems

Recommendation systems

Customers Who Bought This Item Also Bought

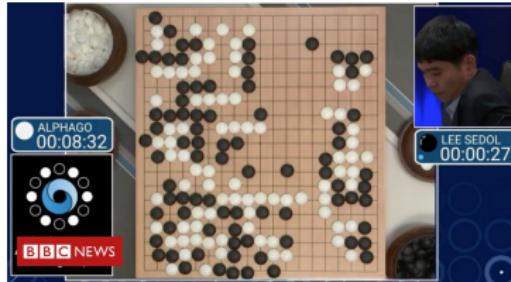
Page 1 of 17

The screenshot shows a horizontal scrollable list of recommended books. Each book entry includes the book cover, title, author, rating, best-seller status, price, and Prime delivery information. Navigation arrows are visible at the top and bottom of the list.

Book Title	Author	Rating	Best-Seller Status	Price	Delivery
Machine Learning: A Probabilistic...	Kevin P. Murphy	4.5	#1 Best Seller	\$81.71	Prime
The Elements of...	Trevor Hastie	4.5		\$84.04	Prime
Probabilistic Graphical Models: Principles and...	Daphne Koller	4.5		\$99.75	Prime
Machine Learning with R	Brett Lantz	4.5		\$49.49	Prime
An Introduction to...	Gareth James	4.5	#1 Best Seller	\$75.99	Prime
Reinforcement Learning: An Introduction...	Richard S. Sutton	4.5		\$64.60	Prime



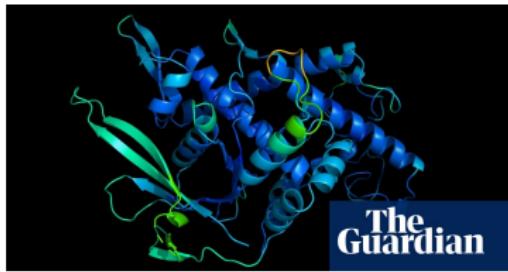
ML has contributed to advances in AI



AlphaGo



Autonomous driving



AlphaFold

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

Basic definitions

- Handwritten digit recognition



- Variability
- Each image can be transformed into a vector \mathbf{x} (feature extraction).
- An instance is made of the pair (\mathbf{x}, y) , where y is the label of the image.
- Objective: find a function $f(\mathbf{x}, \mathbf{w})$.

Basic definitions

- **Training set:** a set of N images and their labels $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, to fit the predictive model.
- **Estimation or training phase:** process of getting the values of \mathbf{w} of the function $f(\mathbf{x}, \mathbf{w})$, that best fit the data.
- **Generalisation:** ability to correctly predict the label of new images \mathbf{x}_* .

Supervised and unsupervised learning

- **Supervised learning:**
 - Variable y is discrete: *classification*.
 - Variable y is continuous: *regression*.
- **Unsupervised learning:** from the set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, we only have access to $\mathbf{x}_1, \dots, \mathbf{x}_N$
 - Find similar groups: *clustering*.
 - Find a probability function for \mathbf{x} : *density estimation*.
 - Find a lower dimensionality representation for \mathbf{x} : *dimensionality reduction and visualisation*.
- **Other types of learning:** reinforcement learning, semi-supervised learning, active learning, multi-task learning.

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

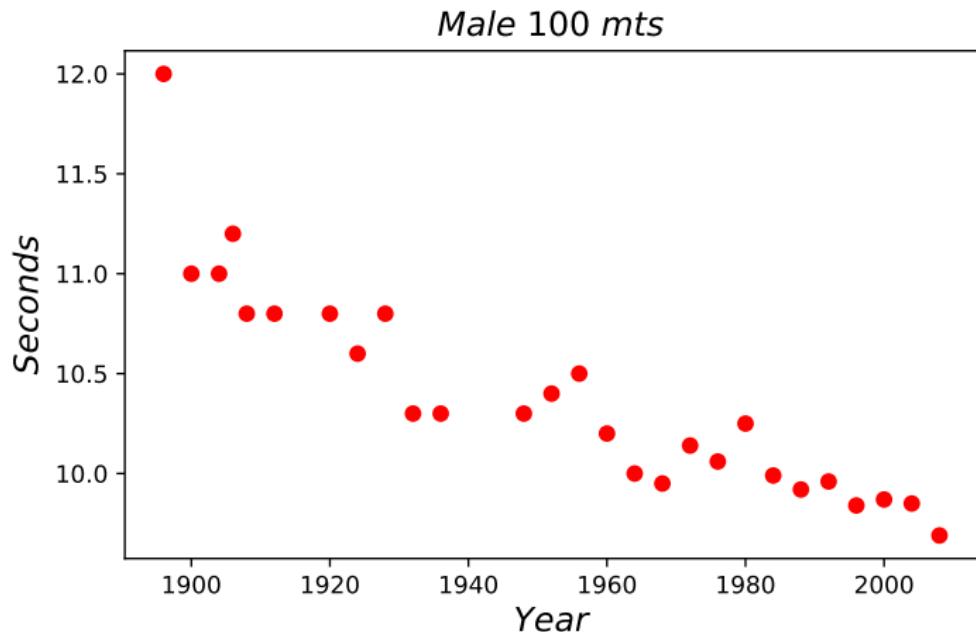
Additional comments

Olympic 100m Data



Image from Wikimedia Commons <http://bit.ly/191adDC>.

Dataset



Model

- We will use a linear model $y = f(x, \mathbf{w})$, where y is the time in seconds and x the year of the competition.
- The linear model is given as

$$y = w_1 x + w_0,$$

where w_0 is the intercept and w_1 is the slope.

- We use \mathbf{w} to refer both to w_0 and w_1 .

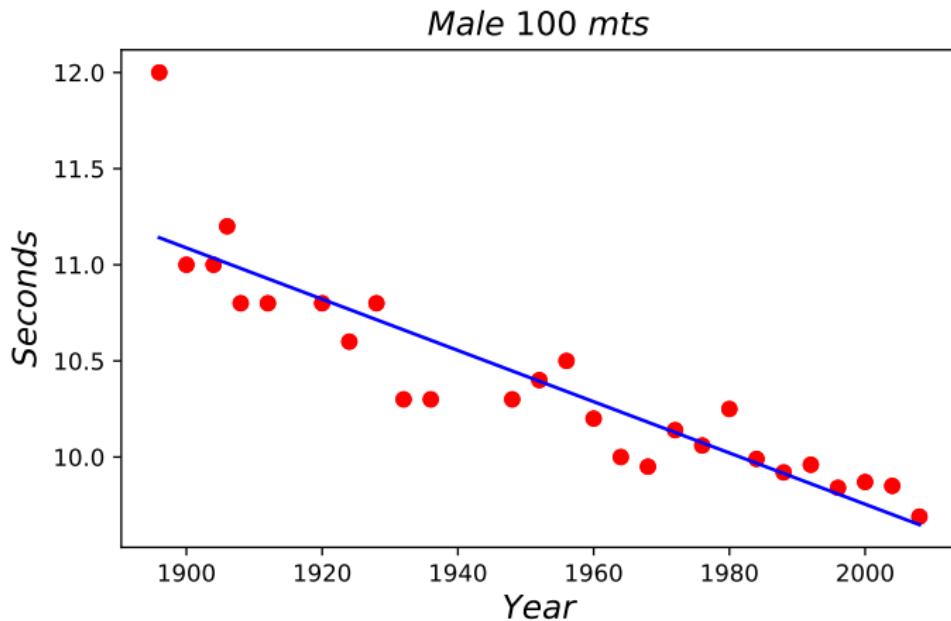
Objective function

- We use an objective function to estimate the parameters w_0 and w_1 that best fit the data.
- In this example, we use a least squares objective function

$$E(w_0, w_1) = \sum_{\forall i} (y_i - f(x_i))^2 = \sum_{\forall i} [y_i - (w_1 x_i + w_0)]^2.$$

- By minimising the error with respect to \mathbf{w} , we get the solution as $w_0 = 36.4$ and $w_1 = -1.34 \times 10^{-2}$.

Data and model



Predictions

- We can now use this model for making predictions.
- For example, what does the model predict for 2012?
- If we say $x = 2012$, then

$$\begin{aligned}y &= f(x, \mathbf{w}) = f(x = 2012, \mathbf{w}) \\&= w_1 x + w_0 = (-1.34 \times 10^{-2}) \times 2012 + 36.4 = 9.59.\end{aligned}$$

- The actual value was 9.63.

Main challenges of machine learning

- Insufficient quantity of training data.
- Nonrepresentative training data.
- Poor-quality data.
- Irrelevant features.
- Overfitting the training data.
- Underfitting the training data.

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

Definition

- A *random variable* (RV) is a *function* that assigns a number to the outcome of a random experiment.
- For example, we toss a coin (random experiment).
- We assign the number 0 to the outcome “tails” and the number “1” to the outcome “heads”.

Discrete and continuous random variables

- A random variable can either be discrete or continuous.
- A **discrete RV** can take a value only from a countable number of distinct values, like 1, 2, 3,
- For example, the number of phone calls received in a call-center from 9:00 to 10:00, the number of COVID patients in the Royal Hallamshire Hospital on May 30, 2020.
- A **continuous RV** can take any value from an infinite possible values within one or more intervals.
- Examples include the time that a cyclist takes to finish the Tour de France; the exchange rate between the british pound and the US dollar on June 30, 2020.

Notation

- We use capital letters to denote random variables, e.g. X, Y, Z, \dots
- We use lowercase letters to denote the values that the random variable takes, x, y, z, \dots

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

Probability mass function

- A discrete RV X is completely defined by a set of values it can take, x_1, x_2, \dots, x_n , and their corresponding probabilities.
- The probability that $X = x_i$ is denoted as $P(X = x_i)$ for $i = 1, \dots, n$, and it is known as the *probability mass function* (pmf).
- Properties
 1. $P(X = x_i) \geq 0, \quad i = 1, \dots, n.$
 2. $\sum_{i=1}^n P(X = x_i) = 1.$

Two discrete RVs

- In machine learning, we are usually interested in more than one random variable.
- Consider two RVs X and Y taking values x_1, x_2, \dots, x_n and y_1, y_2, \dots, y_m , respectively.
- These two random variables can be fully described with **a joint probability mass function** $P(X = x_i, Y = y_j)$ specifying the probability of $X = x_i$ and $Y = y_j$.
- Properties
 1. $P(X = x_i, Y = y_j) \geq 0, \quad i = 1, \dots, n, j = 1, \dots, m.$
 2. $\sum_{i=1}^n \sum_{j=1}^m P(X = x_i, Y = y_j) = 1.$

Rules of probability

- **Marginal**

$$P(X = x_i) = \sum_{j=1}^m P(X = x_i, Y = y_j).$$

We obtain the probability of $P(X = x_i)$ regardless of the value of Y .
This is also known as the **sum rule of probability**.

- **Conditional**

$$P(X = x_i | Y = y_j) = \frac{P(X = x_i, Y = y_j)}{P(Y = y_j)}, \quad P(Y = y_j) \neq 0.$$

- From the above expression, we can also write

$$P(X = x_i, Y = y_j) = P(X = x_i | Y = y_j)P(Y = y_j).$$

This expression is also known as the **product rule of probability**.

How do we compute $P(X = x_i)$ from data?

- A way to compute the probability $P(X = x_i)$ is to repeat an experiment several times, say N , see how many outcomes we get for which $X = x_i$, say $n_{X=x_i}$ and then approximate the probability as

$$P(X = x_i) \approx \frac{n_{X=x_i}}{N}.$$

- We expect the approximation to become an equality when $N \rightarrow \infty$,

$$P(X = x_i) = \lim_{N \rightarrow \infty} \frac{n_{X=x_i}}{N}.$$

What about $P(X = x_i, Y = y_j)$ and $P(X = x_i | Y = y_j)$?

- We can follow a similar procedure to compute $P(X = x_i, Y = y_j)$,

$$P(X = x_i, Y = y_j) = \lim_{N \rightarrow \infty} \frac{n_{X=x_i, Y=y_j}}{N},$$

where $n_{X=x_i, Y=y_j}$ is the number of times we observe a simultaneous occurrence of $X = x_i$ and $Y = y_j$.

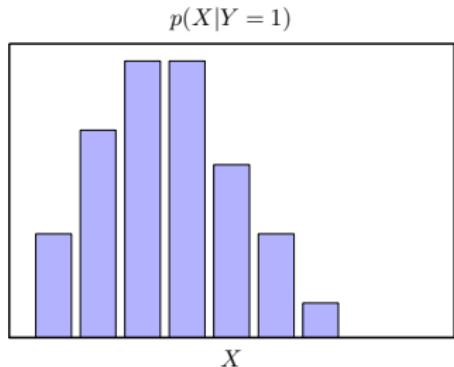
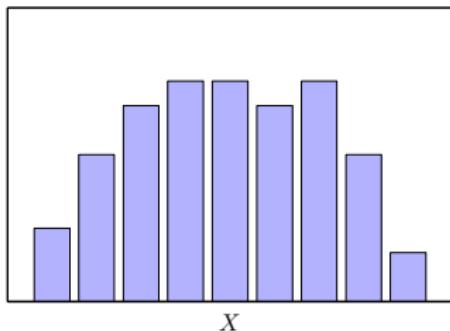
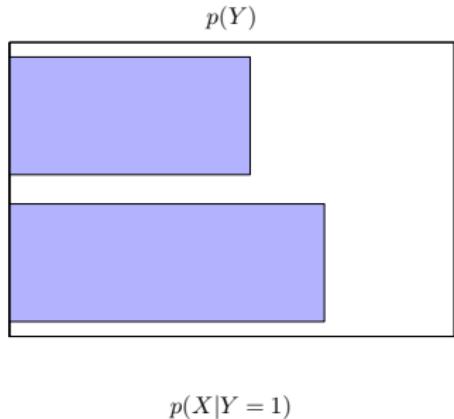
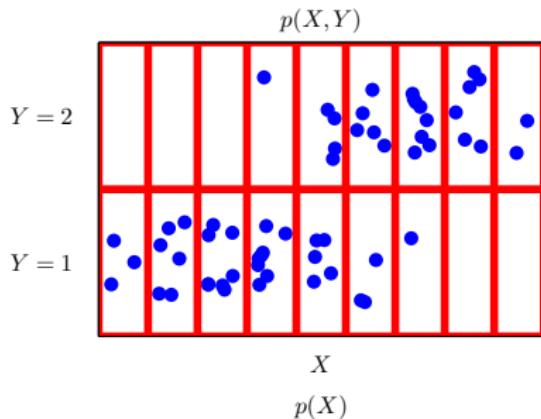
- To compute $P(X = x_i | Y = y_j)$, we can use the definition of the conditional probability

$$P(X = x_i | Y = y_j) = \frac{P(X = x_i, Y = y_j)}{P(Y = y_j)} \approx \frac{\frac{n_{X=x_i, Y=y_j}}{N}}{\frac{n_{Y=y_j}}{N}} = \frac{n_{X=x_i, Y=y_j}}{n_{Y=y_j}}.$$

- In the limit $N \rightarrow \infty$,

$$P(X = x_i | Y = y_j) = \lim_{N \rightarrow \infty} \frac{n_{X=x_i, Y=y_j}}{n_{Y=y_j}}.$$

Examples of the different pmf



From Bishop (2006).

Statistical independence

Two discrete RVs are *statistically independent* if

$$P(X = x_i, Y = y_j) = P(X = x_i)P(Y = y_j), \quad i = 1, \dots, n \quad j = 1, \dots, m.$$

Bayes theorem

- Bayes theorem allows to go from $P(X = x)$ to $P(X = x|Y = y)$.
- According to Bayes theorem

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

Example: Bayes theorem

There are two barrels in front of you. Barrel One contains 20 apples and 4 oranges. Barrel Two contains 4 apples and 8 oranges. You choose a barrel randomly and select a fruit. It is an apple. What is the probability that the barrel was Barrel One?

Answer (I)

- There are two random variables involved.
- Let B be the random variable associated to picking one of the barrels. So B can either be “One” or “Two”.
- Let F be the random variables associated to picking a fruit. So F can either be “Apple” (A) or “Orange” (O).
- The probability we want to compute is the conditional probability $P(B = \text{One}|F = A)$.

Answer (II)

- The statement says “You choose a barrel randomly” which means that $P(B = \text{One}) = P(B = \text{Two}) = \frac{1}{2}$.
- Since we want to go from $P(B = \text{One})$ to $P(B = \text{One}|F = A)$, we can use Bayes theorem,

$$P(B = \text{One}|F = A) = \frac{P(F = A|B = \text{One})P(B = \text{One})}{P(F = A)}.$$

- We need to compute $P(F = A|B = \text{One})$ and $P(F = A)$.

Answer (III)

- Using the sum rule of probability and the product rule of probability

$$\begin{aligned}P(F = A) &= \sum P(F = A, B) = \sum P(F = A|B)P(B) \\&= P(F = A|B = \text{One})P(B = \text{One}) \\&\quad + P(F = A|B = \text{Two})P(B = \text{Two}).\end{aligned}$$

- From the statement,

$$P(F = A|B = \text{One}) = \frac{20}{24}$$

$$P(F = A|B = \text{Two}) = \frac{4}{12}$$

- We then have $P(F = A) = \frac{20}{24} \frac{1}{2} + \frac{4}{12} \frac{1}{2}$.

Answer (IV)

We can finally compute $P(B = \text{One}|F = A)$

$$\begin{aligned} P(B = \text{One}|F = A) &= \frac{P(F = A|B = \text{One})P(B = \text{One})}{P(F = A)} \\ &= \frac{\frac{20}{24} \frac{1}{2}}{\frac{20}{24} \frac{1}{2} + \frac{4}{12} \frac{1}{2}} \\ &= \frac{\frac{20}{24}}{\frac{20}{24} + \frac{4}{12}} = \frac{5}{7} \approx 0.71 \end{aligned}$$

Expected value and statistical moments

- The expected value of a function of a discrete RV, $g(X)$ is defined as

$$E\{g(X)\} = \sum_{i=1}^n g(x_i)P(X = x_i).$$

- Two expected values or *statistical moments* of the discrete RV X , used frequently are the *mean* μ_X and the *variance* σ_X^2 ,

$$\mu_X = E\{X\} = \sum_{i=1}^n x_i P(X = x_i),$$

$$\begin{aligned}\sigma_X^2 &= \text{var}\{X\} = E\{(X - \mu_X)^2\} = \sum_{i=1}^n (x_i - \mu_X)^2 P(X = x_i) \\ &= E\{X^2\} - \mu_X^2\end{aligned}$$

- The squared root of the variance, σ_X , is known as the *standard deviation*.

Example: Expected values

- Consider the following discrete RV X and its pmf. Compute μ_X and σ_X^2 .

X	1	2	3	4
$P(X)$	0.3	0.2	0.1	0.4

- For the mean μ_X , we have

$$\mu_X = \sum_{i=1}^n x_i P(X = x_i) = (1)(0.3) + (2)(0.2) + (3)(0.1) + (4)(0.4) = 2.6.$$

- For the variance, we first compute $E\{X^2\}$ and then use $\sigma_X^2 = E\{X^2\} - \mu_X^2$.
- To compute $E\{X^2\}$, we can use $E\{g(X)\}$, where $g(X) : X \rightarrow X^2$,

$$E\{X^2\} = \sum_{i=1}^n x_i^2 P(X = x_i) = (1)^2(0.3) + (2)^2(0.2) + (3)^2(0.1) + (4)^2(0.4) = 8.4.$$

- We finally get $\sigma_X^2 = E\{X^2\} - \mu_X^2 = 8.4 - (2.6)^2 \approx 1.64$.

Notation

- When referring to the probability $P(X = x)$, we usually simply write $P(x)$.
- Likewise, instead of writing $P(X = x, Y = y)$, we simply write $P(x, y)$.

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

Probability density function

- A continuous RV X takes values within one or more intervals of the real line.
- We use **probability density functions** (pdf), $p_X(x)$, to describe a continuous RV X .
- Properties of a pdf
 1. $p_X(x) \geq 0$.
 2. $\int_{-\infty}^{\infty} p_X(x)dx = 1$.
 3. $P(X \leq a) = \int_{-\infty}^a p_X(x)dx$.
 4. $P(a \leq X \leq b) = \int_a^b p_X(x)dx$.

Two continuous RVs

- As it was the case for discrete RVs, in ML, we are interested in analysing more than one continuous RV.
- We can use a **joint probability density function**, $p_{X,Y}(x,y)$ to fully characterise two continuous random variables X and Y .
- Properties of a joint pdf
 1. $p_{X,Y}(x,y) \geq 0$.
 2. $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_{X,Y}(x,y) dx dy = 1$.
 3. $P(X \leq a, Y \leq c) = \int_{-\infty}^a \int_{-\infty}^c p_{X,Y}(x,y) dx dy$.
 4. $P(a \leq X \leq b, c \leq Y \leq d) = \int_a^b \int_c^d p_{X,Y}(x,y) dx dy$.

Rules of probability (continuous RVs)

- **Sum rule of probability.** In the case of continuous RVs, we replace the sums we had before with an integral

$$p_X(x) = \int_{-\infty}^{\infty} p_{X,Y}(x,y)dy,$$

where $p_X(x)$ is known as the **marginal pdf**.

- **Product rule of probability.** The conditional pdf can be obtained as

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)},$$

which can also be written as

$$p_{X,Y}(x,y) = p_{X|Y}(x|y)p_Y(y).$$

The conditional pdf in this last form is known as the product rule of probability for two continuous RVs.

Bayes theorem and statistical independence

- For the case of continuous RVs, Bayes theorem follows as

$$p_{Y|X}(y|x) = \frac{p_{X|Y}(x|y)p_Y(y)}{p_X(x)}.$$

- We say that two continuous RVs X and Y are statistically independent if

$$p_{X,Y}(x,y) = p_X(x)p_Y(y).$$

Expected values and statistical moments

For continuous RVs, expected values are computed as

$$E\{g(X)\} = \int_{-\infty}^{\infty} g(x)p_X(x)dx,$$

$$\mu_X = E\{X\} = \int_{-\infty}^{\infty} xp_X(x)dx,$$

$$\begin{aligned}\sigma_X^2 = \text{var}\{X\} &= E\{(X - \mu_X)^2\} = \int_{-\infty}^{\infty} (x - \mu_X)^2 p_X(x)dx. \\ &= E\{X^2\} - \mu_X^2.\end{aligned}$$

Notation

We have been using $p_X(x)$ or $p_{X,Y}(x,y)$ to refer to pdfs. We will normally drop the subindex for the RVs and simply use $p(x)$ or $p(x,y)$ to refer to the pdfs.

Contents

Machine learning

Definitions

An example of a predictive model

Review of probability

Random variables

Discrete random variables

Continuous random variables

Additional comments

What if we don't have the pmf or pdf?

- ❑ Discrete RVs. In practice, we can use data to compute the probabilities $P(X = x_i)$ or $P(X = x_i, Y = y_j)$ by applying the definitions we saw before.
- ❑ Notice that those definitions are valid in the limit $N \rightarrow \infty$.
- ❑ Continuous RVs. In practice, we assume a particular model for the pdf, eg. a Gaussian pdf, and estimate the parameters of that pdf, e.g. the mean and variance for the Gaussian pdf.
- ❑ There are advanced methods to model both pmf and pdfs but we will not study those in this module.

What about moments?

- We can estimate μ_X and σ_X^2 when we have access to observations of the random variable X , x_1, x_2, \dots, x_N , but no access to the pmf or the pdf.
- In statistics, these are called “estimators” for μ_X and σ_X^2 , denoted as $\hat{\mu}_X$ and $\hat{\sigma}^2_X$.
- An estimator for μ_X is given as

$$\hat{\mu}_X = \frac{1}{N} \sum_{k=1}^N x_k.$$

- An estimator for σ_X^2 is given as

$$\hat{\sigma}^2_X = \frac{1}{N-1} \sum_{k=1}^N (x_k - \hat{\mu}_X)^2.$$

What if we have more than two RVs?

- ❑ In ML, we are usually faced with problems where we have more than two RVs.
- ❑ In fact, there are applications of ML in Natural Language Processing, speech processing, computer vision, computational biology, etc. where we can have hundreds of thousands or even millions of RVs.
- ❑ The ideas that we saw before can be extended to these cases and we will see some examples in the following lectures.

Take precautions on campus

- ★ **Wear a face covering** in crowded places and when sitting close to others
- **Get vaccinated if you haven't already**
- ▲ **Wash your hands regularly** - hand sanitiser is available across campus
- **Follow one-way systems** - make space for each other when moving around
- **Take two Covid-19 tests each week**
- ◆ **Stay at home if you feel unwell**, even if it's just a cold



Enjoy **campus** **safely**

An end-to-end ML project

Mauricio A. Álvarez

Machine Learning and Adaptive Intelligence
The University of Sheffield



The
University
Of
Sheffield.

Overview

- This week's session will show an example of an end-to-end project in ML. The focus is on supervised learning.

- The lecture provides a general overview and the Jupyter Notebook will provide a hands-on exercise.

- We will describe the typical process involved in providing an ML solution.

- We will use *scikit-learn* (<https://scikit-learn.org/stable/>) to illustrate several of the steps involved.

Why *scikit-learn*?

- scikit-learn is a machine learning library for Python.
- It supports supervised and unsupervised learning.
- It provides several utilities for data preprocessing and model selection and evaluation.
- More importantly: it is open source (BSD licence).

Machine learning project checklist

1. Frame the problem and look at the big picture.
2. Get the data.
3. Explore the data to get insights.
4. Prepare the data to better expose the underlying data patterns.
5. Explore many different models and shortlist the best ones.
6. Fine-tune your models and combine them into a solution.
7. Present your solution.

Contents

Frame the problem and look at the big picture

Get the data

Explore the data

Prepare the data

Shortlist models and then fine-tune them

Present your solution

Set the problem in context (I)

- How will your solution be used?
- What are the current solutions to the problem (if any)?
- Is it a regression or a classification problem?
- How is performance measured?

Set the problem in context (II)

- What would be the minimum performance necessary?
- Are there comparable problems?
- Is human expertise available?

Model performance assessment

- In ML, we use several metrics to assess the performance of a model.
- Performance measures commonly used in regression are
 - root mean squared error (RMSE).
 - mean absolute error (MAE).
- For classification, metrics of performance include:
 - confusion matrix.
 - precision/recall.
 - accuracy.

Metrics in regression

- We have a predictive model $f(\mathbf{x}, \mathbf{w})$ and we want to assess its performance over a dataset of instances $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$.
- The root mean squared error is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N [y_i - f(\mathbf{x}_i, \mathbf{w})]^2}.$$

- The mean absolute error is defined as

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - f(\mathbf{x}_i, \mathbf{w})|.$$

Metrics in classification: confusion matrix

- We have a predictive model $f(\mathbf{x}, \mathbf{w})$ and we want to assess its performance over a dataset of instances $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$.
- Binary classification problem: spam detection. There are two classes, spam and not-spam.
- The confusion matrix is a table that summarizes how successful the classification model is at predicting examples belonging to various classes

	spam (predicted)	not-spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not-spam (actual)	12 (FP)	556 (TN)

where TP stands for **true positive**, TN stands for **true negative**, FP stands for **false positive** and FN stands for **false negative**.

Metrics in classification: precision/recall (I)

- **Precision** is the ratio of correct positive predictions to the overall number of positive predictions

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

- **Recall** is the ratio of correct positive predictions to the overall number of positive examples in the dataset

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Metrics in classification: precision/recall (II)

- ❑ Say you look for the terms “pattern recognition and machine learning” in the StarPlus University’s Library website.
- ❑ Precision is the fraction of relevant documents in the list of returned documents (“how valid the results are?”)
- ❑ Recall is the fraction of relevant documents returned to the total number of relevant documents that could have been returned (“how complete the results are?”)
- ❑ In the spam example, we would like high precision, this is we would like the FP to be low meaning having few genuine messages predicted as spam.
- ❑ We are probably Ok with a low recall, some spam messages sent to Inbox because they are predicted as not-spam.

Metrics in classification: accuracy

- ❑ **Accuracy** is the ratio of examples correctly classified over the total number of examples classified

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

- ❑ Accuracy is a useful metric when errors predicting all classes are equally important.
- ❑ In the spam example, FP are worse than FN.
- ❑ Accuracy can be misleading in imbalance classification problems, e.g. you can have a high a TP value and a low TN value, and your accuracy could still be high.

Contents

Frame the problem and look at the big picture

Get the data

Explore the data

Prepare the data

Shortlist models and then fine-tune them

Present your solution

Be aware of

- what data do you need? is it enough?
- legal obligations regarding the data. Who owns the data? Do you have to sign an NDA to work with the data? Do you require a legal agreement?
- For example, since 2018 the UK abides to the General Data Protection Regulation (GDPR) under EU law.
- removing sensitive information (e.g. anonymisation)
- institutions usually have in place a data ethics advisory committees.
- check the size and type of the data (e.g time series? images? spatio-temporal?)
- sample a test set and do not look at it (e.g. avoid data snooping).

Where do we get data from?

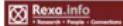
UC Irvine Machine Learning Repository


Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact
Repository Web Google Search
View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 557 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. For a general overview of the Repository, please visit our [About](#) page. For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#).

Supported By:  In Collaboration With: 

Latest News:	Newest Data Sets:	Most Popular Data Sets (hits since 2007):
<p>09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou!</p> <p>04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!</p> <p>03-01-2010: Note from donor regarding Netflix data</p> <p>10-16-2009: Two new data sets have been added.</p> <p>09-14-2009: Several data sets have been added.</p> <p>03-24-2008: New data sets have been added!</p> <p>06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope</p>	<p>07-22-2020:  Facebook Large Page-Page Network</p> <p>07-17-2020:  Amphibians</p> <p>07-12-2020:  Early stage diabetes risk prediction dataset</p> <p>06-28-2020:  Taiwanese Bankruptcy Prediction</p> <p>06-20-2020:  South German Credit (UPDATE)</p>	<p>3551844:  Iris</p> <p>1931560:  Adult</p> <p>1490003:  Wine</p> <p>1331678:  Breast Cancer Wisconsin (Diagnostic)</p> <p>1315682:  Heart Disease</p>

Featured Data Set: Forest Fires
 Task: Regression
Data Type: Multivariate
Attributes: 13

<https://archive.ics.uci.edu/ml/index.php>

Where do we get data from?

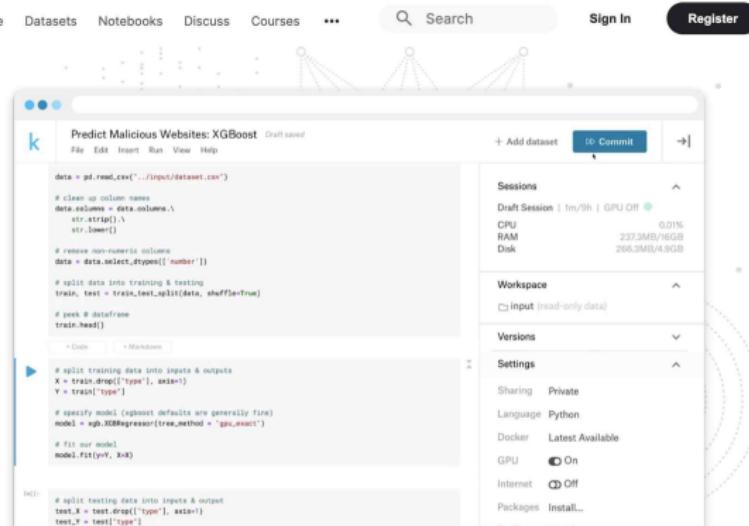
Kaggle

Start with more than a blinking cursor

Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data & code.

 REGISTER WITH GOOGLE

Register with Email



The screenshot shows a Kaggle Jupyter Notebook titled "Predict Malicious Websites: XGBoost". The notebook contains Python code for data cleaning, splitting, and training an XGBoost model. The right sidebar displays session details, workspace, versions, and settings, including GPU usage information.

```
data = pd.read_csv("../input/dataset.csv")
# Clean up column names
data.columns = [col.lower().strip() for col in data.columns]
# remove non-numeric columns
data = data.select_dtypes(['number'])

# split data into training & testing
train, test = train_test_split(data, shuffle=True)

# peek @ dataframe
train.head()

# split training data into inputs & outputs
X = train.drop(['type'], axis=1)
y = train['type']

# specify model (xgb defaults are generally fine)
model = xgb.XGBRegressor(tree_method = 'gpu_hist')

# fit our model
model.fit(y=y, X=X)

# split testing data into inputs & output
test_X = test.drop(['type'], axis=1)
test_y = test['type']
```

Sessions

- Draft Session | 1m/9h | GPU Off
- CPU 0.01%
- RAM 237.0MB/16GB
- Disk 296.3MB/4.9GB

Workspace

- Input (read-only data)

Versions

Settings

- Sharing Private
- Language Python
- Docker Latest Available
- GPU On
- Internet Off
- Packages Install...

<https://www.kaggle.com/>

Where do we get data from?

Amazon's AWS datasets

Registry of Open Data on AWS



About

This registry exists to help people discover and share datasets that are available via AWS resources. Learn more about sharing data on AWS.

See all usage examples for datasets listed in this registry.

See datasets from Facebook Data for Good, NASA Space Act Agreement, NIH STRIDES, NOAA Big Data Project, Space Telescope Science Institute, and Amazon Sustainability Data Initiative.

Search datasets (currently 188 matching datasets)

Search datasets

Add to this registry

If you want to add a dataset or example of how to use a dataset to this registry, please follow the instructions on the [Registry of Open Data on AWS GitHub repository](#).

Unless specifically stated in the applicable dataset documentation, datasets available through the Registry of Open Data on AWS are not provided and maintained by AWS. Datasets are provided and maintained by a variety of third parties under a variety of licenses. Please check dataset licenses and related documentation to determine if a dataset may be used for your application.

The Cancer Genome Atlas

[cancer](#) [genomic](#) [life sciences](#) [STRIDES](#) [whole genome sequencing](#)

The Cancer Genome Atlas (TCGA), a collaboration between the National Cancer Institute (NCI) and National Human Genome Research Institute (NHGRI), aims to generate comprehensive, multi-dimensional maps of the key genomic changes in major types and subtypes of cancer. TCGA has analyzed matched tumor and normal tissues from 11,000 patients, allowing for the comprehensive characterization of 33 cancer types and subtypes, including 10 rare cancers. The dataset contains open Clinical Supplement, Biospecimen Supplement, RNA-Seq Gene Expression Quantification, miRNA-Seq Isoform Expression Quantificati...

[Details →](#)

Usage examples

- [GDC Legacy Archive by National Cancer Institute](#)
- [Comprehensive Characterization of Cancer Driver Genes and Mutations by Matthew H. Bailey, Collin Tokheim, et al.](#)
- [Genomic and Functional Approaches to Understanding Cancer Aneuploidy by Alison M. Taylor, Juliann Shih, et al.](#)
- ["Before and After: A Comparison of Legacy and Harmonized TCGA Data at the Genomic Data Commons" by Galen F. Gao, Joel S. Parker, et al.](#)
- [Using TCGA Data, Resources, and Materials by National Cancer Institute](#)

[See 29 usage examples →](#)

<https://registry.opendata.aws/>

Where do we get data from?

- Meta portals listing open data repositories.
- Data portals (<http://dataportals.org/>)
- Open data monitor (<https://www.opendatemonitor.eu/>)
- Quandl (<https://www.quandl.com/>)

Three sets: training, validation and test sets

- ❑ In the previous session, we mentioned that we use a dataset to train an ML model.
- ❑ Actually, when we get the data, we should partition the data into three sets
 - training set.
 - validation set.
 - test set.
- ❑ The partition of the dataset into these three sets is done randomly.
- ❑ The training set is usually the biggest one.
- ❑ The validation and test sets have roughly the same amount of samples.

Three sets: what for?

- The training set is used to fit the model using the objective function.
- The validation set is used to choose the best predictive model among a set of candidates.
- The test set is used to assess the final performance of the model before shipping the model to production.
- You can use the training and validation sets as you prefer.
- As we said before, never look at your test set when designing your model, only until you have decided what model to use based on the validation set.

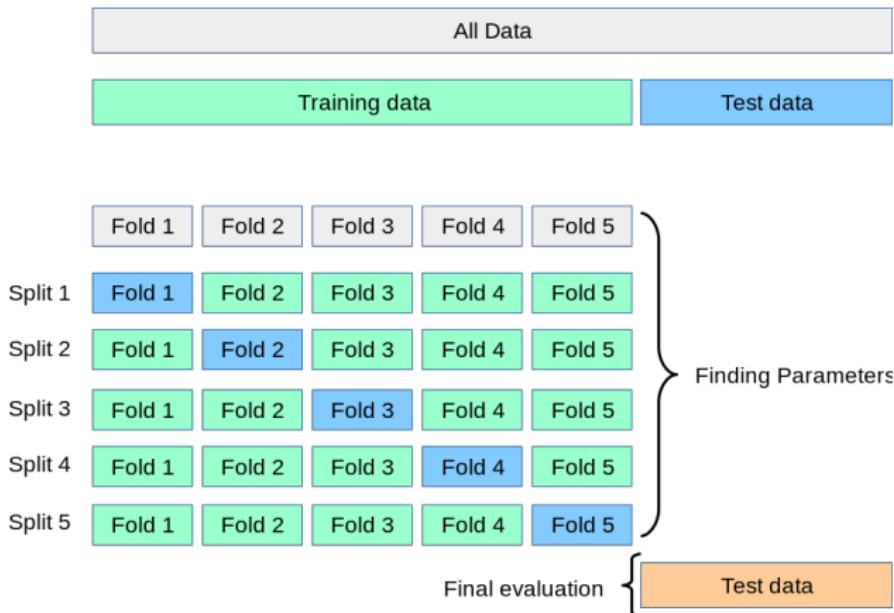
Three sets: large datasets

- When you have a large dataset, you can have 70% data for training, 15% for validation and 15% for testing.
- Nowdays, you can have a dataset with millions of data instances. In those cases, you might want to try 95% for training, 2.5% for validation and 2.5% for testing.

Three sets: smaller datasets

- ❑ If your data is small, say in the hundreds or just a few thousands, you want to use as many training instances as possible.
- ❑ We can use a strategy known as *cross-validation* to make better use of the training/validation sets.

k -fold cross-validation



The extreme case is when $k = N$, the number of folds is equal to the number of instances in the training/validation set: leave-one-out (LOO) cross-validation.

Contents

Frame the problem and look at the big picture

Get the data

Explore the data

Prepare the data

Shortlist models and then fine-tune them

Present your solution

To take into account (I)

- Create a copy of the data for exploration.
- If you have a large dataset, you can sample a fraction of the data for exploring it.
- Create a Jupyter Notebook so that you keep track of your data exploration process.
- Study each feature (inputs and outputs) and its characteristics including
 - name
 - type (continuous, bounded/unbounded, categorical, etc.)
 - percentage of missing values
 - noisiness and type of noise (e.g. outliers)
 - Is this feature useful for the task?
 - type of distribution (Gaussian, Gamma, logarithmic, etc.)

To take into account (II)

- Identify the target attribute (for supervised learning).
- Visualise the data using histograms, scatterplots, maps, etc.
- Study correlations between attributes.
- Identify what transformations you might be able to apply.
- Is there any additional data that you might find useful? Can you get it?

Predicting bike rentals

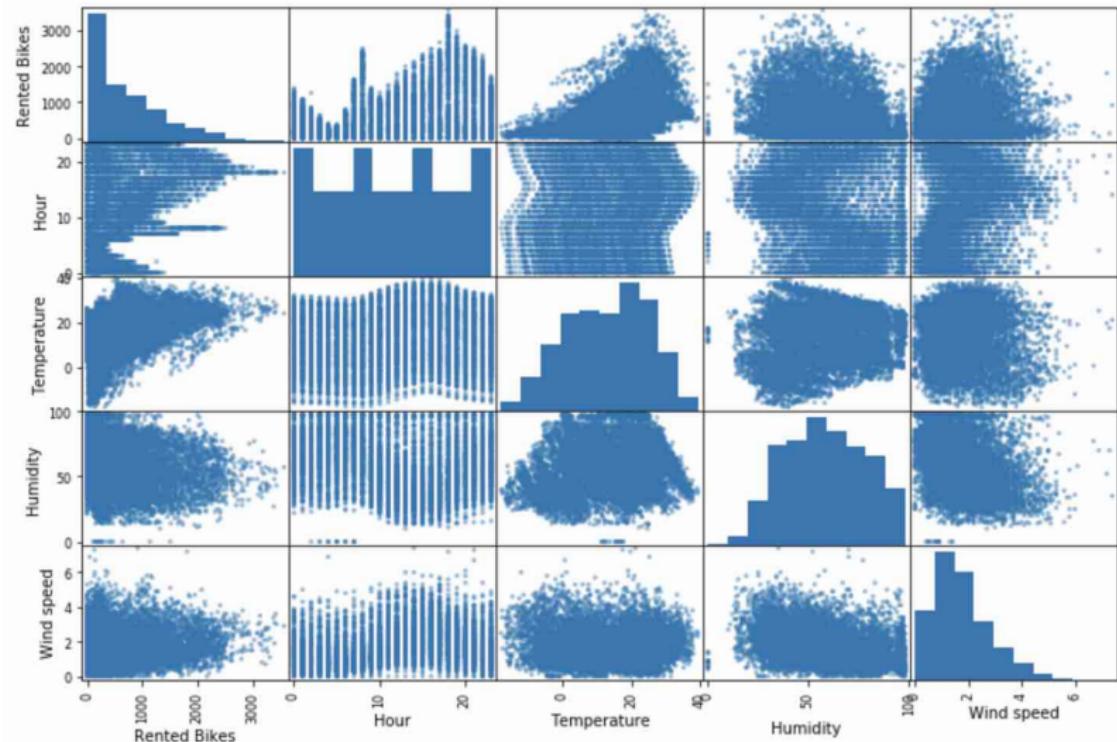


By Sarah Stierch - Own work, CC BY 4.0, <https://commons.wikimedia.org/w/index.php?curid=16075615>

Dataset

- ❑ The feature vector \mathbf{x} includes the following features: hour, temperature, humidity, wind speed, visibility, Dew point temperature, solar radiation, rainfall, snowfall, seasons, holiday, functioning day.
- ❑ The variables hour, temperature, humidity, wind speed, visibility, Dew point temperature, solar radiation, rainfall, snowfall can be considered as continuous.
- ❑ The variables seasons, holiday, and functioning day are categorical variables.
- ❑ The output variable y is the number of bikes rented.

Scatter plot



Study correlations between attributes

The correlation coefficient between two RVs X and Y is given as

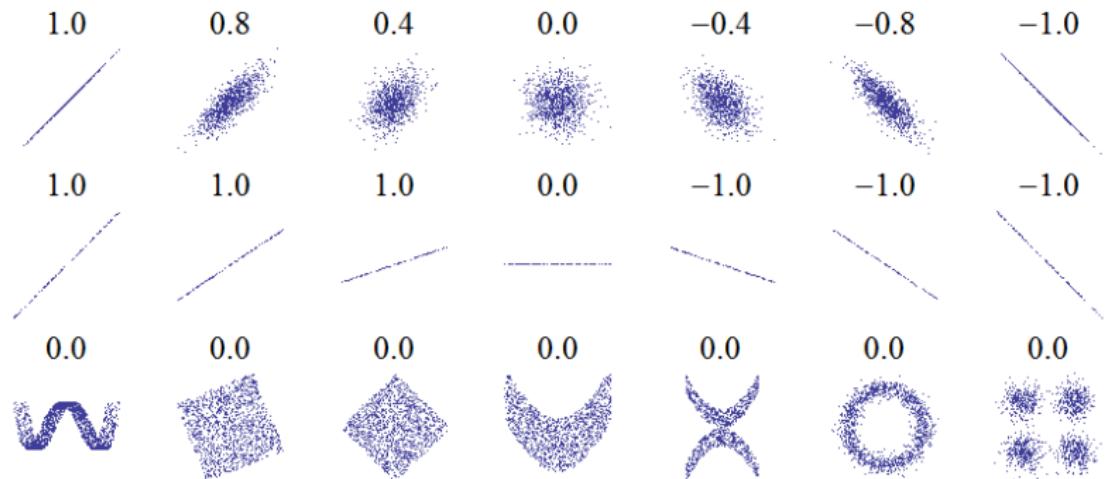
$$\rho_{X,Y} = \frac{E\{(X - \mu_X)(Y - \mu_Y)\}}{\sigma_X \sigma_Y} = \frac{\sigma_{X,Y}}{\sigma_X \sigma_Y},$$

where $-1 < \rho_{X,Y} < 1$ and $\sigma_{X,Y}$ is known as the covariance between X and Y .

```
In [18]: corr_matrix["Rented Bikes"].sort_values(ascending=False)
```

```
Out[18]: Rented Bikes      1.000000
Temperature        0.538558
Hour              0.410257
Dew point temperature  0.379788
Solar Radiation    0.261837
Visibility         0.199280
Wind speed         0.121108
Rainfall           -0.123074
Snowfall           -0.141804
Humidity          -0.199780
Name: Rented Bikes, dtype: float64
```

Correlation and dependence



By DenisBoigelot, original uploader was Imagecreator - Own work, original uploader was Imagecreator, CC0,

<https://commons.wikimedia.org/w/index.php?curid=15165296>

Contents

Frame the problem and look at the big picture

Get the data

Explore the data

Prepare the data

Shortlist models and then fine-tune them

Present your solution

Data cleaning (I)

- ❑ Remove the outliers in your data (optional).
- ❑ Handle the missing values
 - filling them in using the mean, the median, or any other value.
 - drop the feature if most of the instances have a missing value.
 - drop the instance if you have several instances with missing values.
 - you can add an additional feature indicating whether the instance has a missing feature or not, and then use a value of 0 for the missing feature.

Data cleaning (II)

- Most ML methods require features that are numbers rather than categories (usually appearing as text).
- Also, if the feature is categorical, it is useful to use a different representation.
- In the previous bike rentals example, there were three categorical features
 - season that can take four categories.
 - holiday and functioning day, each taking two categories.

Data cleaning (III)

- For example, the feature season takes values autumn, winter, spring and summer.
- The way to handle this feature is to use a representation known as *one-hot encoding* to obtain a higher-dimensional binary representation for each value,

autumn = [1, 0, 0, 0]

winter = [0, 1, 0, 0]

spring = [0, 0, 1, 0]

summer = [0, 0, 0, 1]

- The values of the feature season do not have a natural order, therefore one should not map these values to numbers like 1 for autumn, 2 for winter, 3 for spring and 4 for summer.
- The ML method will try to find regularities within these ordered values even though they do not exist.

Feature selection and feature engineering

- You have the option to remove features that are uninformative.
- You have the option to discretise a continuous feature (e.g. binning).
- You can also create new features from the ones you have available.
- For example, instead of using feature x , you can use $\log(x)$, \sqrt{x} , x^2 , etc.

Feature scaling (I)

- Several ML methods do not perform well when the input features have very different scales.
- In the rental bikes example, the variable humidity is in the range 0 to 100, whereas wind speed is in the range 0 to 8.
- Two ways to get all features to have the same scale are *normalisation* (or min-max scaling) and *standardisation* (or z-score normalisation).

Feature scaling (II)

- In normalisation, we map the range of values that a feature takes to the range $[-1, 1]$ or $[0, 1]$.
- The normalisation formula is given as

$$\bar{x}_j = \frac{x_j - \min x_j}{\max x_j - \min x_j},$$

where $\min x_j$ and $\max x_j$ are the minimum and maximum values for the feature in the training set.

- In standardisation, the features are scaled so that they have mean zero and standard deviation equal to one,

$$\hat{x}_j = \frac{x_j - \mu_j}{\sigma_j},$$

where μ_j and σ_j are the mean and standard deviation of the feature x_j .

Feature scaling (III)

- Which scaling to use?
- If the dataset is not big and there is time, one can try both and see which one performs better in the validation set.
- If there is no time, as a rule of thumb:
 - unsupervised learning algorithms usually benefit more from standardisation than normalisation.
 - standardisation is preferred if the feature has already a distribution close to a Gaussian.
 - if the feature has outliers standardisation is preferred since normalisation will squeeze all the other values into a small range.
 - normalisation is preferred in all the other cases.

Contents

Frame the problem and look at the big picture

Get the data

Explore the data

Prepare the data

Shortlist models and then fine-tune them

Present your solution

Shortlist promising models

- If the data is big, use a smaller subset of the data to try different models in a reasonable time.
- Try different models from several categories (e.g. linear, non-linear, probabilistic, non-probabilistic).
- This can be easily done with scikit-learn.
- Measure and compare the performance of the ML models you used.
 - make sure you are using the exact same data in the training and the validation sets.
 - when using cross-validation, compute the mean and standard deviation of the performance measure over the k -folds for each model.
- shortlist two to three models that look promising.

Fine-tune the system (I)

- ❑ It is a good idea to use as much data as you can at this stage.
- ❑ Fine-tune the hyperparameters of your model using cross-validation.
 - data transformations are hyperparameters, e.g. should I input missing data with zero or with the mean?
 - use random search or grid search to explore hyperparameters
 - if the training takes long, use Auto-ML to fine-tune the hyperparameters.

Fine-tune the system (II)

- ❑ Once you are confident about your final model, usually the one that performs better on average on the validation data:
 - merge the training and the validation datasets
 - fit the model again using the set of hyperparameters you found before.
- ❑ Finally, use the test data to assess the generalisation error of your ML system.
 - Do not try to change your model based on the performance on the test data, you might start overfitting your model to the data.

Contents

Frame the problem and look at the big picture

Get the data

Explore the data

Prepare the data

Shortlist models and then fine-tune them

Present your solution

Present your solution

- Document what you have done.
- Create a nice presentation.
- Explain why your solution achieves the business objective.
- Bring up interesting points you noticed in the process.

Decision trees and ensemble methods

Mauricio A. Álvarez

Machine Learning and Adaptive Intelligence
The University of Sheffield



The
University
Of
Sheffield.

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Nodes in a decision tree

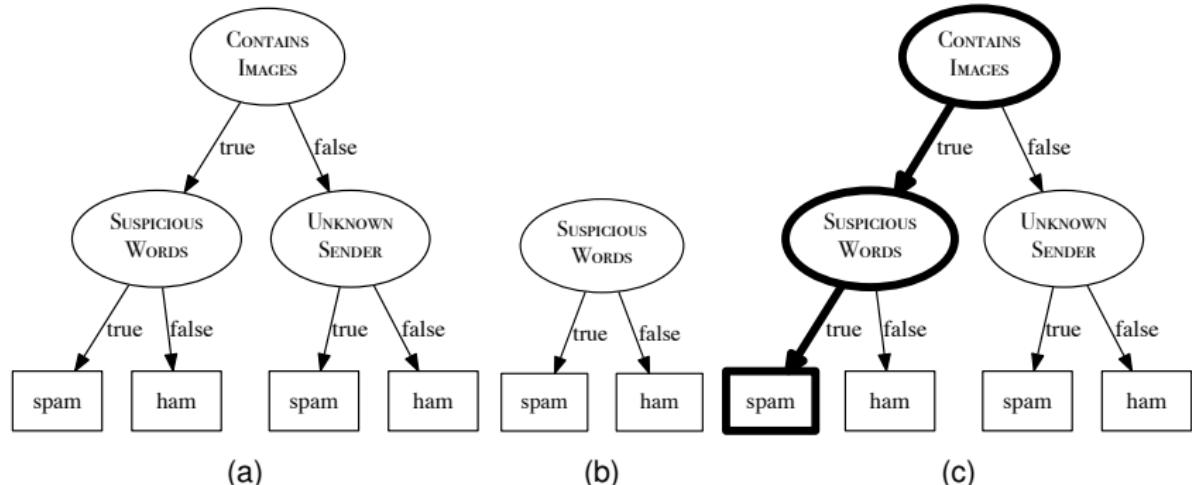
- A decision tree consists of:
 1. a **root node** (or starting node),
 2. **interior nodes**
 3. and **leaf nodes** (or terminating nodes).
- Each of the non-leaf nodes (root and interior) in the tree specifies a test to be carried out on one of the query's descriptive features.
- Each of the leaf nodes specifies a predicted classification for the query.

An email spam prediction dataset

An email spam prediction dataset.

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Two decision trees and a query instance



(a) and (b) show two decision trees that are consistent with the instances in the spam dataset. (c) shows the path taken through the tree shown in (a) to make a prediction for the query instance: SUSPICIOUS WORDS = 'true', UNKNOWN SENDER = 'true', CONTAINS IMAGES = 'true'.

Which tree to use?

- ❑ Both of these trees will return identical predictions for all the examples in the dataset.
- ❑ So, which tree should we use?
- ❑ Occam's Razor: “*Among competing hypotheses, the one with the fewest assumptions should be selected*”.

Informative features

- ❑ A decision tree is a machine learning algorithm that tries to build predictive models **using the most informative features**.
- ❑ An informative feature is a feature whose values split the instances in the dataset into **homogeneous sets** with respect to the target value.
- ❑ So the problem is to figure out which features are the most informative ones to ask questions about.

How do we create shallow trees?

- ❑ The tree that tests SUSPICIOUS WORDS at the root is very shallow because the SUSPICIOUS WORDS feature perfectly splits the data into pure groups of '*spam*' and '*ham*'.
- ❑ Descriptive features that split the dataset into pure sets with respect to the target feature provide information about the target feature.
- ❑ So we can make shallow trees by testing the informative features early on in the tree.
- ❑ All we need to do that is a computational metric of the purity of a set: **entropy**

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Entropy

- ❑ Claude Shannon's entropy model defines a computational measure of the impurity of the elements of a set.
- ❑ An easy way to understand the entropy of a set is to think in terms of the uncertainty associated with guessing the result if you were to make a random selection from the set.

Probability and entropy

- Entropy is related to the probability of an outcome.
 - High probability → Low entropy
 - Low probability → High entropy
- If we take the **log** of a probability and multiply it by -1 we get this mapping!

Mathematical definition of entropy

- Shannon's model of entropy is a weighted sum of the logs of the probabilities of each of the possible outcomes when we make a random selection from a set.

$$H(t) = - \sum_{i=1}^l (P(t = i) \times \log_s(P(t = i)))$$

Entropy in poker cards

- What is the entropy of a set of 52 different playing cards?

$$\begin{aligned} H(card) &= - \sum_{i=1}^{52} P(card = i) \times \log_2(P(card = i)) \\ &= - \sum_{i=1}^{52} 0.019 \times \log_2(0.019) = - \sum_{i=1}^{52} -0.1096 \\ &= 5.700 \text{ bits} \end{aligned}$$

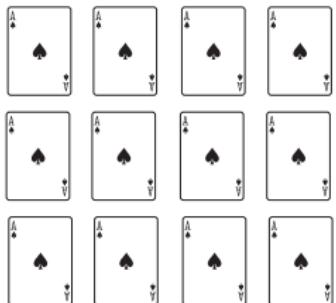
Entropy in poker cards

- What is the entropy of a set of 52 playing cards if we only distinguish between the cards based on their suit {♥, ♣, ♦, ♠}?

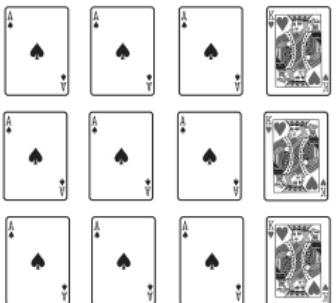
Entropy in poker cards

$$\begin{aligned} H(\text{suit}) &= - \sum_{I \in \{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}} P(\text{suit} = I) \times \log_2(P(\text{suit} = I)) \\ &= -((P(\heartsuit) \times \log_2(P(\heartsuit))) + (P(\clubsuit) \times \log_2(P(\clubsuit)))) \\ &\quad + (P(\diamondsuit) \times \log_2(P(\diamondsuit))) + (P(\spadesuit) \times \log_2(P(\spadesuit))) \\ &= -((13/52 \times \log_2(13/52)) + (13/52 \times \log_2(13/52)) \\ &\quad + (13/52 \times \log_2(13/52)) + (13/52 \times \log_2(13/52))) \\ &= -((0.25 \times -2) + (0.25 \times -2) \\ &\quad + (0.25 \times -2) + (0.25 \times -2)) \\ &= 2 \text{ bits} \end{aligned}$$

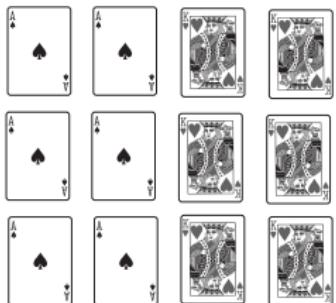
Different entropies



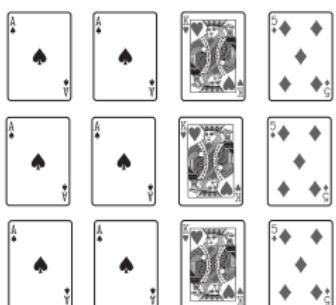
$$(a) H(card) = 0.00$$



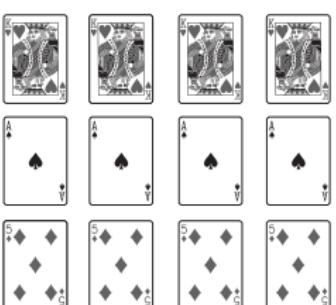
$$(b) H(card) = 0.81$$



$$(c) H(card) = 1.00$$



$$(d) H(card) = 1.50$$



$$(e) H(card) = 1.58$$



$$(f) H(card) = 3.58$$

The entropy of different sets of playing cards measured in bits.

Entropy of a message

The relationship between the entropy of a message and the set it was selected from.

Entropy of a Message	Properties of the Message Set
High	A large set of equally likely messages.
Medium	A large set of messages, some more likely than others.
Medium	A small set of equally likely messages.
Low	A small set of messages with one very likely message.

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

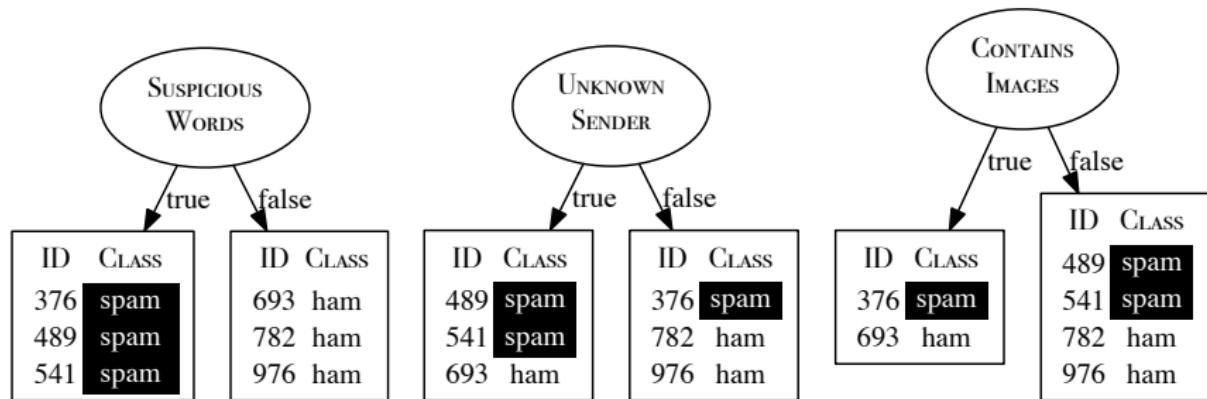
Model Ensembles

The spam dataset again

An email spam prediction dataset.

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Partitions in the spam dataset



How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset table.

Pure subsets

- Our intuition is that the ideal discriminatory feature will partition the data into **pure** subsets where all the instances in each subset have the same classification.
 - SUSPICIOUS WORDS perfect split.
 - UNKNOWN SENDER mixture but some information (when '*true*' most instances are '*spam*').
 - CONTAINS IMAGES no information.
- One way to implement this idea is to use a metric called **information gain**.

Information Gain

- The information gain of a descriptive feature can be understood as a measure of the reduction in the overall entropy of a prediction task by testing on that feature.

Computing the information gain

1. Compute the entropy of the original dataset with respect to the target feature. This gives us a measure of how much information is required in order to organize the dataset into pure sets.
2. For each descriptive feature, create the sets that result by partitioning the instances in the dataset using their feature values, and then sum the entropy scores of each of these sets. This gives a measure of the information that remains required to organize the instances into pure sets after we have split them using the descriptive feature.
3. Subtract the remaining entropy value (computed in step 2) from the original entropy value (computed in step 1) to give the information gain.

Computing the information gain

Computing information gain involves the following three equations:

$$H(t, \mathcal{D}) = - \sum_{l \in levels(t)} (P(t = l) \times \log_2(P(t = l)))$$

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D})$$

Example with the spam dataset

- As an example we will calculate the information gain for each of the descriptive features in the spam email dataset.

Step 1: Entropy for the target feature

- Calculate the **entropy** for the target feature in the dataset.

$$H(t, \mathcal{D}) = - \sum_{I \in levels(t)} (P(t = I) \times \log_2(P(t = I)))$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 1: Entropy for the target feature

$$\begin{aligned} H(t, \mathcal{D}) &= - \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} (P(t = l) \times \log_2(P(t = l))) \\ &= - ((P(t = \text{'spam'}) \times \log_2(P(t = \text{'spam'}))) \\ &\quad + (P(t = \text{'ham'}) \times \log_2(P(t = \text{'ham'})))) \\ &= - \left(\left(\frac{3}{6} \times \log_2(\frac{3}{6}) \right) + \left(\frac{3}{6} \times \log_2(\frac{3}{6}) \right) \right) \\ &= 1 \text{ bit} \end{aligned}$$

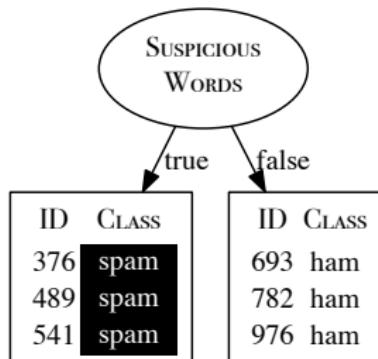
Step 2: Remainder for the SUSPICIOUS W. feature

- Calculate the **remainder** for the SUSPICIOUS WORDS feature in the dataset.

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 2: Partitions for the SUSPICIOUS W. feature



Partitions for the SUSPICIOUS W. feature

Step 2: Remainder for the SUSPICIOUS W. feature

$rem(\text{WORDS}, \mathcal{D})$

$$\begin{aligned} &= \left(\frac{|\mathcal{D}_{\text{WORDS}=T}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{WORDS}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{WORDS}=F}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{WORDS}=F}) \right) \\ &= \left(\frac{3}{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t = l) \times \log_2(P(t = l)) \right) \right) \\ &\quad + \left(\frac{3}{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t = l) \times \log_2(P(t = l)) \right) \right) \\ &= \left(\frac{3}{6} \times \left(- \left(\left(\frac{3}{3} \times \log_2(\frac{3}{3}) \right) + \left(\frac{0}{3} \times \log_2(\frac{0}{3}) \right) \right) \right) \right) \\ &\quad + \left(\frac{3}{6} \times \left(- \left(\left(\frac{0}{3} \times \log_2(\frac{0}{3}) \right) + \left(\frac{3}{3} \times \log_2(\frac{3}{3}) \right) \right) \right) \right) = 0 \text{ bits} \end{aligned}$$

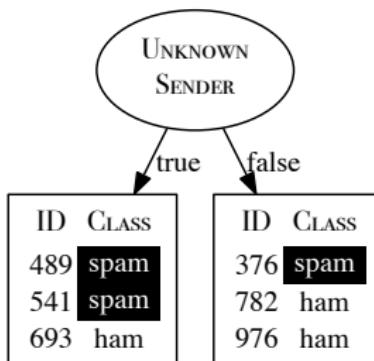
Step 2: Remainder for the UNKNOWN SENDER feature

- Calculate the **remainder** for the UNKNOWN SENDER feature in the dataset.

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 2: Partitions for the UNKNOWN SENDER feature



Partitions for the UNKNOWN SENDER feature

Step 2: Remainder for the UNKNOWN SENDER feature

$rem(\text{SENDER}, \mathcal{D})$

$$\begin{aligned} &= \left(\frac{|\mathcal{D}_{\text{SENDER}=T}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{SENDER}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{SENDER}=F}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{SENDER}=F}) \right) \\ &= \left(\frac{3}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\ &\quad + \left(\frac{3}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\ &= \left(\frac{3}{6} \times \left(- \left(\left(\frac{2}{3} \times \log_2(\frac{2}{3}) \right) + \left(\frac{1}{3} \times \log_2(\frac{1}{3}) \right) \right) \right) \right) \\ &\quad + \left(\frac{3}{6} \times \left(- \left(\left(\frac{1}{3} \times \log_2(\frac{1}{3}) \right) + \left(\frac{2}{3} \times \log_2(\frac{2}{3}) \right) \right) \right) \right) = 0.9183 \text{ bits} \end{aligned}$$

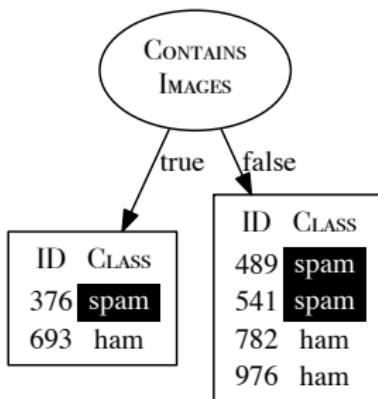
Step 2: Remainder for the CONTAINS IMAGES feature

- Calculate the **remainder** for the CONTAINS IMAGES feature in the dataset.

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 2: Partitions for the CONTAINS IMAGES feature



Partitions for the CONTAINS IMAGES feature

Step 2: Remainder for the CONTAINS IMAGES feature

$rem(\text{IMAGES}, \mathcal{D})$

$$\begin{aligned} &= \left(\frac{|\mathcal{D}_{\text{IMAGES}=T}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{IMAGES}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{IMAGES}=F}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{IMAGES}=F}) \right) \\ &= \left(\frac{2}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\ &\quad + \left(\frac{4}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\ &= \left(\frac{2}{6} \times \left(- \left(\left(\frac{1}{2} \times \log_2(\frac{1}{2}) \right) + \left(\frac{1}{2} \times \log_2(\frac{1}{2}) \right) \right) \right) \right) \\ &\quad + \left(\frac{4}{6} \times \left(- \left(\left(\frac{2}{4} \times \log_2(\frac{2}{4}) \right) + \left(\frac{2}{4} \times \log_2(\frac{2}{4}) \right) \right) \right) \right) = 1 \text{ bit} \end{aligned}$$

Step 3: Compute the Information Gain

- Calculate the **information gain** for the three descriptive feature in the dataset.

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D})$$

Step 3: Compute the Information Gain

$$\begin{aligned}IG(\text{SUSPICIOUS WORDS}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - rem(\text{SUSPICIOUS WORDS}, \mathcal{D}) \\&= 1 - 0 = 1 \text{ bit}\end{aligned}$$

$$\begin{aligned}IG(\text{UNKNOWN SENDER}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - rem(\text{UNKNOWN SENDER}, \mathcal{D}) \\&= 1 - 0.9183 = 0.0817 \text{ bits}\end{aligned}$$

$$\begin{aligned}IG(\text{CONTAINS IMAGES}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - rem(\text{CONTAINS IMAGES}, \mathcal{D}) \\&= 1 - 1 = 0 \text{ bits}\end{aligned}$$

- The results of these calculations match our intuitions.

Contents

Decision trees

- Definitions

- Shannon's Entropy

- Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

- Alternative impurity measures

- Overfitting and Tree Pruning

- Model Ensembles

Turn continuous features into boolean features

- The easiest way to handle continuous valued descriptive features is to turn them into boolean features by defining a threshold.
- The threshold is used to partition the instances based on their value of the continuous descriptive feature.
- How do we set the threshold?

Sorting the instances

1. The instances in the dataset are sorted according to the continuous feature values.
2. The adjacent instances in the ordering that have different classifications are then selected as possible threshold points.
3. The optimal threshold is found by computing the information gain for each of these classification transition boundaries and selecting the boundary with the highest information gain as the threshold.

Treat the feature as a categorial feature

- Once a threshold has been set the dynamically created new boolean feature can compete with the other categorical features for selection as the splitting feature at that node.
- This process can be repeated at each node as the tree grows.

Vegetation classification dataset



(a) chaparral veg.



(b) riparian veg.



(c) conifer veg.

Dataset for predicting the vegetation in an area with a continuous ELEVATION feature (measured in feet).

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	3 900	chaparral
2	true	moderate	300	riparian
3	true	steep	1 500	riparian
4	false	steep	1 200	chaparral
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer
7	true	steep	3 000	chaparral

Sorted instances

Dataset for predicting the vegetation in an area sorted by the continuous ELEVATION feature.

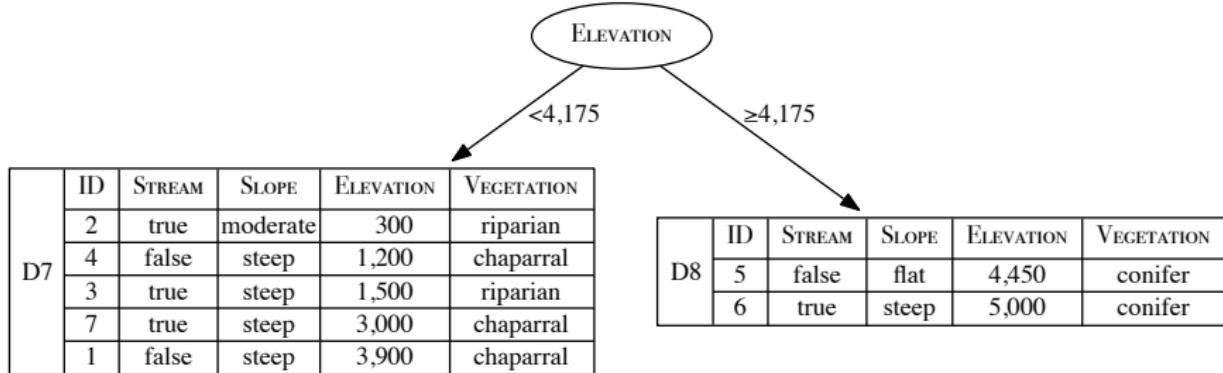
ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1 200	chapparal
3	true	steep	1 500	riparian
7	true	steep	3 000	chapparal
1	false	steep	3 900	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer

Thresholds and partitions

Partition sets (Part.), entropy, remainder (Rem.), and information gain (Info. Gain) for the candidate ELEVATION thresholds: ≥ 750 , ≥ 1350 , ≥ 2250 and ≥ 4175 .

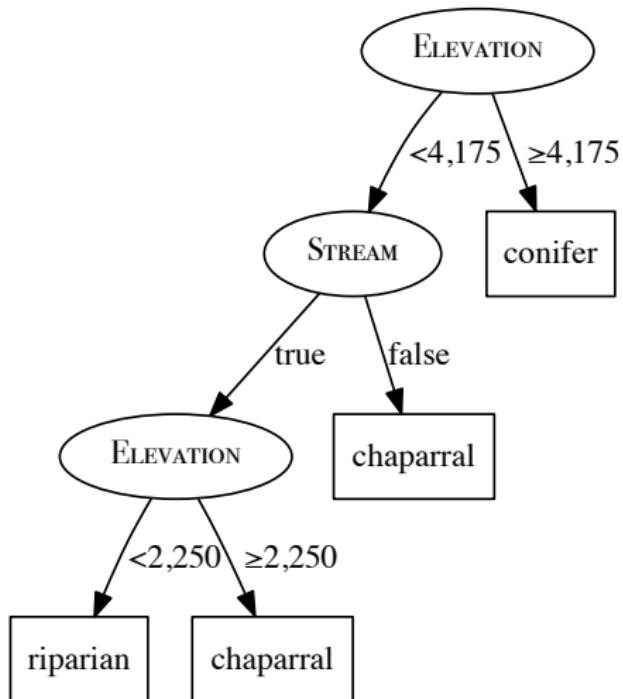
Split by Threshold	Part.	Instances	Partition Entropy	Rem.	Info. Gain
≥ 750	\mathcal{D}_1	d_2	0.0		
	\mathcal{D}_2	$d_4, d_3, d_7, d_1, d_5, d_6$	1.4591	1.2507	0.3060
≥ 1350	\mathcal{D}_3	d_2, d_4	1.0		
	\mathcal{D}_4	d_3, d_7, d_1, d_5, d_6	1.5219	1.3728	0.1839
≥ 2250	\mathcal{D}_5	d_2, d_4, d_3	0.9183		
	\mathcal{D}_6	d_7, d_1, d_5, d_6	1.0	0.9650	0.5917
≥ 4175	\mathcal{D}_7	d_2, d_4, d_3, d_7, d_1	0.9710		
	\mathcal{D}_8	d_5, d_6	0.0	0.6935	0.8631

First split



The vegetation classification decision tree after the dataset has been split using $\text{ELEVATION} \geq 4175$.

Second split



The decision tree that would be generated for the vegetation classification dataset using information gain.

Contents

Decision trees

- Definitions

- Shannon's Entropy

- Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

- Alternative impurity measures

- Overfitting and Tree Pruning

- Model Ensembles

The ID3 Algorithm

- ❑ ID3 Algorithm (Iterative Dichotomizer 3)
- ❑ Attempts to create the shallowest tree that is consistent with the data that it is given.
- ❑ The ID3 algorithm builds the tree in a recursive, depth-first manner, beginning at the root node and working down to the leaf nodes.

How the different nodes are treated?

1. The algorithm begins by choosing the best descriptive feature to test (i.e., the best question to ask first) using **information gain**.
2. A root node is then added to the tree and labelled with the selected test feature.
3. The training dataset is then partitioned using the test.
4. For each partition a branch is grown from the node.
5. The process is then repeated for each of these branches using the relevant partition of the training set in place of the full training set and with the selected test feature excluded from further testing.

The CART Algorithm

- ❑ CART (Classification and Regression Trees).
- ❑ Constructs binary trees.
- ❑ At each node it looks for the feature and threshold in that feature that provides the largest information gain at that node.
- ❑ This is the one implemented in scikit-learn.

Greedy algorithms

- ❑ Both ID3 and CART are examples of *greedy* algorithms.
- ❑ They find the “best tree” by greedily finding the best partitions at each level of the tree.
- ❑ They don’t check whether the best splits done at the higher levels of the tree will lead to the lowest possible impurity several levels down.

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Regression trees

- ❑ Regression trees are constructed so as to reduce the **variance** in the set of training examples at each of the leaf nodes in the tree
- ❑ We can do this by adapting the ID3 algorithm to use a measure of variance rather than a measure of classification impurity (entropy) when selecting the best attribute

Impurity for regression

- The impurity (variance) at a node can be calculated using the following equation:

$$var(t, \mathcal{D}) = \frac{\sum_{i=1}^n (t_i - \bar{t})^2}{n - 1}$$

- We select the feature that minimizes the weighted variance across the resulting partitions:

$$\mathbf{d}[best] = \operatorname{argmin}_{d \in \mathbf{d}} \sum_{l \in levels(d)} \frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|} \times var(t, \mathcal{D}_{d=l})$$

A dataset listing the number of bike rentals per day

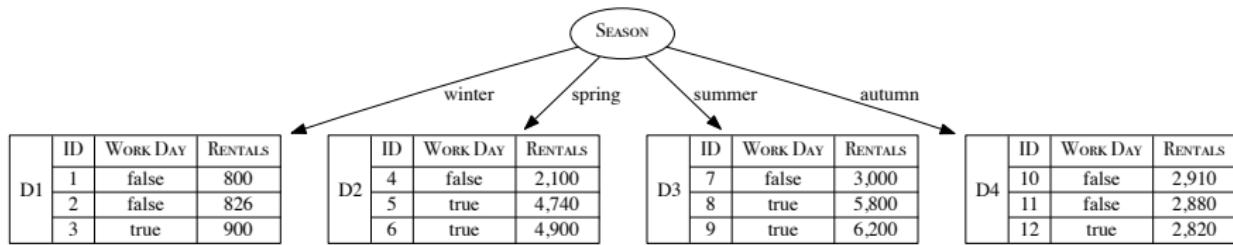
ID	SEASON	WORK DAY	RENTALS
1	winter	false	800
2	winter	false	826
3	winter	true	900
4	spring	false	2 100
5	spring	true	4 740
6	spring	true	4 900
7	summer	false	3 000
8	summer	true	5 800
9	summer	true	6 200
10	autumn	false	2 910
11	autumn	false	2 880
12	autumn	true	2 820

Weighted variance according to the partitions

Partitioning of the dataset for bike rentals based on SEASON and WORK DAY features and the computation of the weighted variance for each partitioning.

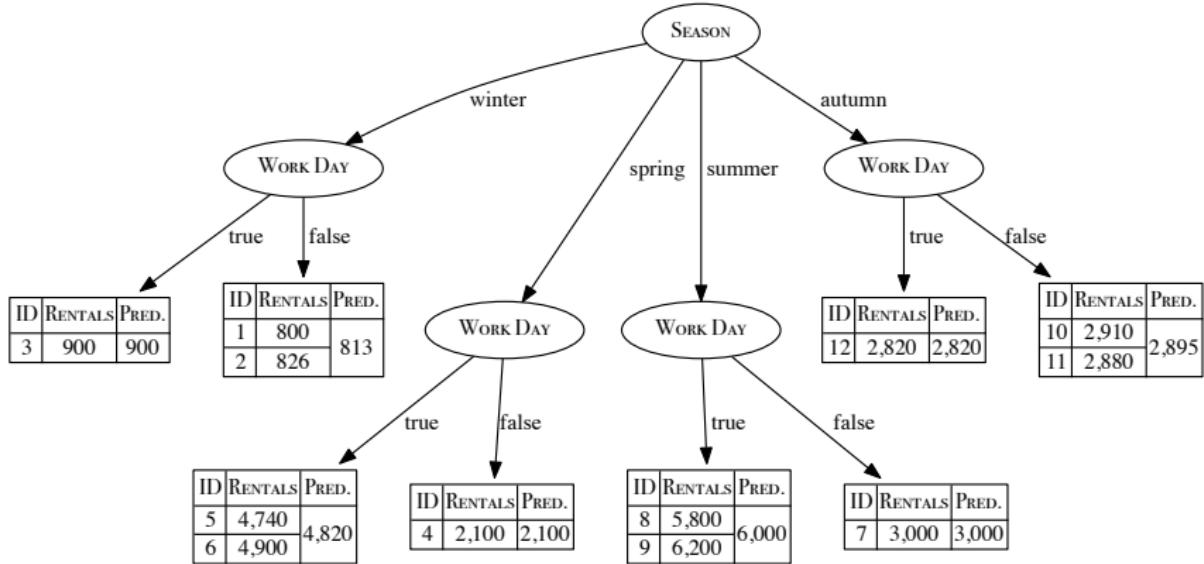
Split by Feature	Level	Part.	Instances	$\frac{ \mathcal{D}_{d=I} }{ \mathcal{D} }$	$\text{var}(t, \mathcal{D})$	Weighted Variance
SEASON	'winter'	\mathcal{D}_1	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0.25	2 692	
	'spring'	\mathcal{D}_2	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.25	2 472 533 $\frac{1}{3}$	
	'summer'	\mathcal{D}_3	$\mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_9$	0.25	3 040 000	
	'autumn'	\mathcal{D}_4	$\mathbf{d}_{10}, \mathbf{d}_{11}, \mathbf{d}_{12}$	0.25	2 100	
WORK DAY	'true'	\mathcal{D}_5	$\mathbf{d}_3, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_8, \mathbf{d}_9, \mathbf{d}_{12}$	0.50	4 026 346 $\frac{1}{3}$	
	'false'	\mathcal{D}_6	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_7, \mathbf{d}_{10}, \mathbf{d}_{11}$	0.50	1 077 280	2 551 813 $\frac{1}{3}$

Resulting decision tree



The decision tree resulting from splitting the data using the feature **SEASON**.

Final decision tree



The final decision tree induced from the dataset. To illustrate how the tree generates predictions, this tree lists the instances that ended up at each leaf node and the prediction (PRED.) made by each leaf node.

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Gini index

- Another commonly used measure of impurity is the **Gini index**:

$$Gini(t, \mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t = l)^2$$

- The Gini index can be thought of as calculating how often you would misclassify an instance in the dataset if you classified it based on the probability distribution of the target feature in the dataset.
- Information gain can be calculated using the Gini index by replacing the entropy measure with the Gini index.

Information gain ratio

- ❑ Entropy based information gain has a preference for features with many values.
- ❑ One way of addressing this issue is to use **information gain ratio** which is computed by dividing the information gain of a feature by the amount of information used to determine the value of the feature:

$$GR(d, \mathcal{D}) = \frac{IG(d, \mathcal{D})}{-\sum_{l \in levels(d)} (P(d = l) \times \log_2(P(d = l)))}$$

Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Why overfitting?

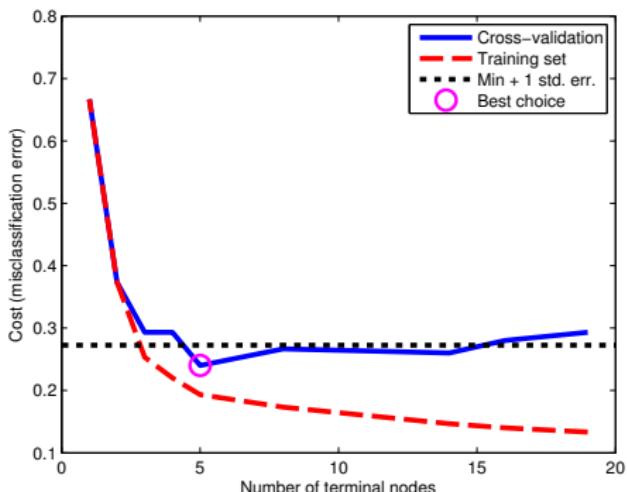
The likelihood of over-fitting occurring increases as a tree gets deeper because the resulting classifications are based on smaller and smaller subsets as the dataset is partitioned after each feature test in the path.

Pruning strategies

- **Pre-pruning:** stop the recursive partitioning early. Pre-pruning is also known as **forward pruning**.
- We can stop growing the tree if the decrease in the error is not sufficient to justify the extra complexity of adding an extra subtree.
- **Post-pruning:** allow the algorithm to grow the tree as much as it likes and then prune the tree of the branches that cause over-fitting.

Common Post-pruning Approach

- Using the validation set evaluate the prediction accuracy achieved by both the fully grown tree and the pruned copy of the tree.
- If the pruned copy of the tree performs no worse than the fully grown tree the node is a candidate for pruning.



Contents

Decision trees

Definitions

Shannon's Entropy

Information Gain

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Alternative impurity measures

Overfitting and Tree Pruning

Model Ensembles

Definition

- ❑ Rather than creating a single model, we can generate a set of models and then make predictions by aggregating the outputs of these models.
- ❑ A prediction model that is composed of a set of models is called a **model ensemble**.
- ❑ In order for this approach to work the models that are in the ensemble must be different from each other.

Approaches

There are two standard approaches to creating ensembles:

1. **boosting**
2. **bagging.**

Boosting: How does it work?

- ❑ Boosting works by iteratively creating models and adding them to the ensemble.
- ❑ The iteration stops when a predefined number of models have been added.
- ❑ When we use **boosting** each new model added to the ensemble is biased to pay more attention to instances that previous models miss-classified.
- ❑ This is done by incrementally adapting the dataset used to train the models. To do this we use a **weighted dataset**

Boosting: Weighted Dataset

- Each instance has an associated weight $\mathbf{w}_i \geq 0$,
- Initially set to $\frac{1}{n}$ where n is the number of instances in the dataset.
- After each model is added to the ensemble it is tested on the **training data** and the weights of the instances the model gets correct are decreased and the weights of the instances the model gets incorrect are increased.
- These weights are used as a distribution over which the dataset is sampled to create a **replicated training set**, where the replication of an instance is proportional to its weight.

Predictions

- Once the set of models have been created the ensemble makes **predictions** using a weighted aggregate of the predictions made by the individual models.

- The weights used in this aggregation are simply the confidence factors associated with each model.

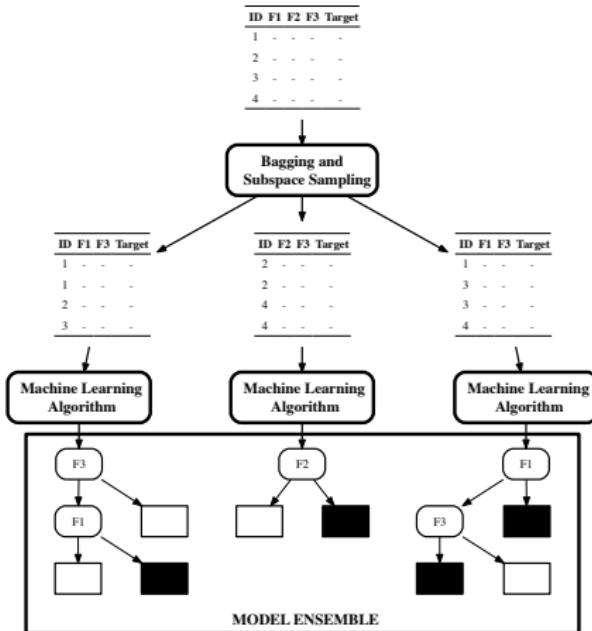
Bagging: Definition

- When we use **bagging** (or **bootstrap aggregating**) each model in the ensemble is trained on a random sample of the dataset known as **bootstrap samples**.
- Each random sample is the same size as the dataset and **sampling with replacement** is used.
- Consequently, every bootstrap sample will be missing some of the instances from the dataset so each bootstrap sample will be different and this means that models trained on different bootstrap samples will also be different

Bagging: Random forest

- When bagging is used with decision trees each bootstrap sample only uses a randomly selected subset of the descriptive features in the dataset. This is known as **subspace sampling**.
- The combination of bagging, subspace sampling, and decision trees is known as a **random forest** model.

Example of using bagging



The process of creating a model ensemble using bagging and subspace sampling.

Linear regression

Mauricio A. Álvarez

Machine Learning and Adaptive Intelligence
The University of Sheffield



The
University
Of
Sheffield.

Contents

Review of vector/matrix notation and linear algebra

A regression model

Linear regression

Gradient descent

Stochastic Gradient Descent

Regularisation

Scalar and vectors

- A **scalar** is just a numeric value like 0.9 or -18.7 .
- Scalars are usually denoted as lower case letters like x or a .
- A **vector** is an ordered list of scalar values. Sometimes we refer to these scalar values of the vector as *attributes* or *entries* of the vector.
- Vectors are usually denoted by bold lowercase letters like \mathbf{x} or \mathbf{y} .

Vectors

- A vector can appear sometimes written as a row vector, e.g.

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$$

Or as a column vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

- In this module, ALL vectors will be column vectors by default. So, when you see a vector, e.g. $\mathbf{x}, \mathbf{y}, \mathbf{z}$ always think this vector has a column-wise shape.

Matrices

- A **matrix** is a rectangular array of scalars arranged in rows and columns.
- Matrices are usually denoted by bold uppercase letters, e.g. **X** or **Y**.
- The following matrix has three rows and two columns

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

- The entries in the matrix above are of the form x_{ij} , where the first subindex i indicates the row of the element and the second subindex j indicates the column.

Matrix transpose

- Let \mathbf{X} be a matrix with elements x_{ij} .
- The transpose of a matrix \mathbf{X} is a new matrix \mathbf{X}^\top with elements x_{ji} .

$$\mathbf{X} = \begin{bmatrix} 4.1 & -5.6 \\ -2.6 & 7.9 \\ 3.5 & 1.8 \end{bmatrix}, \quad \mathbf{X}^\top = \begin{bmatrix} 4.1 & -2.6 & 3.5 \\ -5.6 & 7.9 & 1.8 \end{bmatrix}$$

Matrix multiplication

- Let **A** be a matrix with entries a_{ik} of dimensions $p \times q$.
- Let **B** be a matrix with entries b_{kj} of dimensions $t \times s$.
- Matrix multiplication of the form **AB** is only possible if $q = t$.
- If this is the case, the matrix **C** = **AB** has dimensions $p \times s$ with entries

$$c_{ij} = \sum_k a_{ik} b_{kj}.$$

Transpose of a product

- Let \mathbf{w} be a vector of dimensions $d \times 1$. Let \mathbf{X} be a matrix with dimensions $n \times d$.
- The transpose of the product \mathbf{Xw} , $(\mathbf{Xw})^\top$ is

$$(\mathbf{Xw})^\top = \mathbf{w}^\top \mathbf{X}^\top.$$

- We can apply this result to a product of several matrices

$$\begin{aligned}(\mathbf{ABCD})^\top &= ((\mathbf{AB})(\mathbf{CD}))^\top \\&= (\mathbf{CD})^\top (\mathbf{AB})^\top \\&= \mathbf{D}^\top \mathbf{C}^\top \mathbf{B}^\top \mathbf{A}^\top.\end{aligned}$$

From a scalar operation to a vector operation

- ❑ It is usually desirable to transform a scalar operation into a vector operation.
- ❑ When coding scalar operations, we require making use of loops, which can be expensive.
- ❑ In contrast, vector operations are handled efficiently by low-level routines already included in modules like numpy.

Example

Write the following scalar operation into a vector/matrix form

$$\sum_{i=1}^n \left(y_i - \sum_{j=1}^d x_{ij} w_j \right)^2.$$

Answer (I)

- The sum above can be written as

$$\begin{aligned}\sum_{i=1}^n (y_i - \sum_{j=1}^d x_{ij} w_j)^2 &= (y_1 - \sum_{j=1}^d x_{1j} w_j)(y_1 - \sum_{j=1}^d x_{1j} w_j) \\ &\quad + \cdots \\ &\quad + (y_n - \sum_{j=1}^d x_{nj} w_j)(y_n - \sum_{j=1}^d x_{nj} w_j).\end{aligned}$$

- Let us define a vector \mathbf{v} of dimensions $n \times 1$ with entries given as

$$(y_i - \sum_{j=1}^d x_{ij} w_j).$$

Answer (II)

- The product of vectors $\mathbf{v}^\top \mathbf{v}$ gives the same result than the required sum,

$$\begin{aligned}\mathbf{v}^\top \mathbf{v} &= \left[(y_1 - \sum_{j=1}^d x_{1j} w_j) \quad \cdots \quad (y_n - \sum_{j=1}^d x_{nj} w_j) \right] \begin{bmatrix} (y_1 - \sum_{j=1}^d x_{1j} w_j) \\ \vdots \\ (y_n - \sum_{j=1}^d x_{nj} w_j) \end{bmatrix} \\ &= \sum_{i=1}^n (y_i - \sum_{j=1}^d x_{ij} w_j)^2.\end{aligned}$$

- How do we express the elements in \mathbf{v} with vectors and matrices?

Answer (III)

- For a fixed i , x_{i1}, \dots, x_{id} can be grouped into a vector \mathbf{x}_i^\top .
- The internal sums in the entries of \mathbf{v} can then be written as

$$\sum_{j=1}^d x_{ij} w_j = \mathbf{x}_i^\top \mathbf{w} = [x_{i1} \quad x_{i2} \quad \cdots \quad x_{id}] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

- We can now write \mathbf{v} as

$$\mathbf{v} = \begin{bmatrix} y_1 - \mathbf{x}_1^\top \mathbf{w} \\ \vdots \\ y_n - \mathbf{x}_n^\top \mathbf{w} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^\top \mathbf{w} \\ \vdots \\ \mathbf{x}_n^\top \mathbf{w} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \mathbf{w}$$

- We can group the scalars y_1, \dots, y_n into a vector \mathbf{y} .
- We can group the row vectors $\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top$ into a matrix \mathbf{X} .

Answer (IV)

- It means that $\mathbf{v} = \mathbf{y} - \mathbf{Xw}$.
- Finally

$$\sum_{i=1}^n (y_i - \sum_{j=1}^d x_{ij} w_j)^2 = \mathbf{v}^\top \mathbf{v} = (\mathbf{y} - \mathbf{Xw})^\top (\mathbf{y} - \mathbf{Xw}).$$

Two common types of products

- **Inner product.** The inner product between two vectors results in a scalar.
- Let \mathbf{x} and \mathbf{y} be vectors of dimension $m \times 1$. The inner product is given as

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^m x_i y_i,$$

- **Outer product.** The outer product between two vectors results in a matrix.
- Let \mathbf{x} be a vector of dimension $m \times 1$ and \mathbf{y} a vector of dimension $p \times 1$. The outer product is given as

$$\mathbf{x}\mathbf{y}^\top = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_p \\ x_2 y_1 & \cdots & x_2 y_p \\ \vdots & \vdots & \vdots \\ x_m y_1 & \cdots & x_m y_p \end{bmatrix}$$

Differentiating a function in a vector/matrix form (I)

- We will see cases in which a function $f(\mathbf{w})$ depends on some parameters grouped in a vector \mathbf{w} .
- We would like to find the vector of parameters \mathbf{w} that maximise $f(\mathbf{w})$.
- For example, suppose $f(\mathbf{w})$ is defined as

$$f(\mathbf{w}) = \sum_{i=1}^d w_i x_i.$$

- We can group the scalars x_1, \dots, x_d into \mathbf{x} . Likewise for \mathbf{w} .
- According to what we saw before, we can write $f(\mathbf{w})$ as $f(\mathbf{w}) = \mathbf{x}^\top \mathbf{w}$.

Differentiating a function in a vector/matrix form (II)

- For a fixed \mathbf{x} , we are interested in computing the gradient of $f(\mathbf{w})$ with respect to \mathbf{w}

$$\frac{df(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial f(\mathbf{w})}{\partial w_d} \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_d. \end{bmatrix} = \mathbf{x}.$$

- Some useful identities when differentiating with respect to a vector

$f(\mathbf{w})$	$\frac{df(\mathbf{w})}{d\mathbf{w}}$
$\mathbf{w}^\top \mathbf{x}$	\mathbf{x}
$\mathbf{x}^\top \mathbf{w}$	\mathbf{x}
$\mathbf{w}^\top \mathbf{w}$	$2\mathbf{w}$
$\mathbf{w}^\top \mathbf{Cw}$	$2\mathbf{Cw}$.

Identity matrix and the inverse of a matrix

- The identity matrix of size N is a square matrix with ones on the main diagonal and zeros elsewhere, e.g.,

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The inverse matrix of a matrix \mathbf{A} of dimensions $d \times d$, denoted as \mathbf{A}^{-1} , satisfies

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_d$$

Contents

Review of vector/matrix notation and linear algebra

A regression model

Linear regression

Gradient descent

Stochastic Gradient Descent

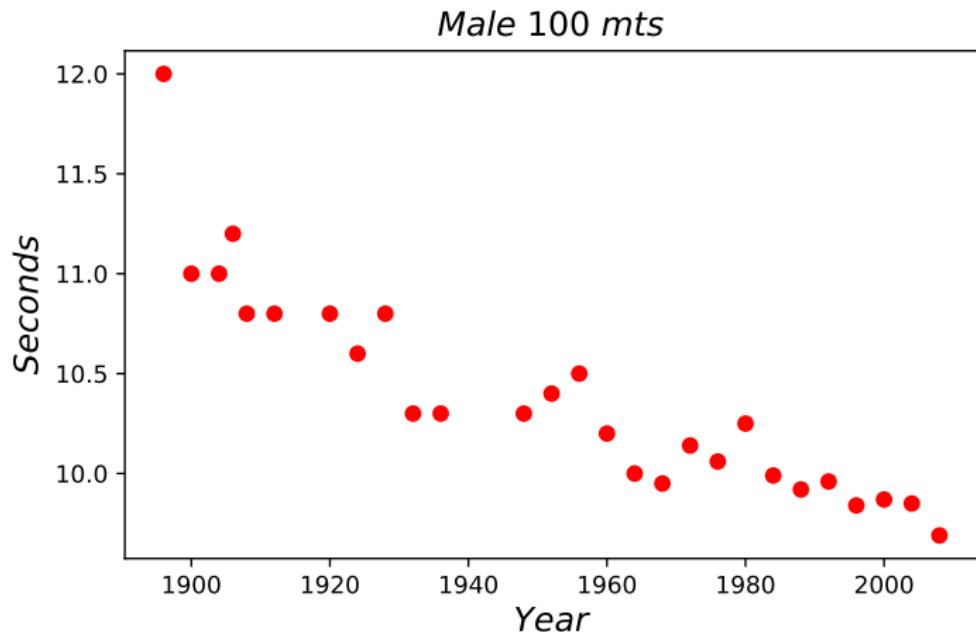
Regularisation

Olympic 100m Data



Image from Wikimedia Commons <http://bit.ly/191adDC>.

Dataset



Model

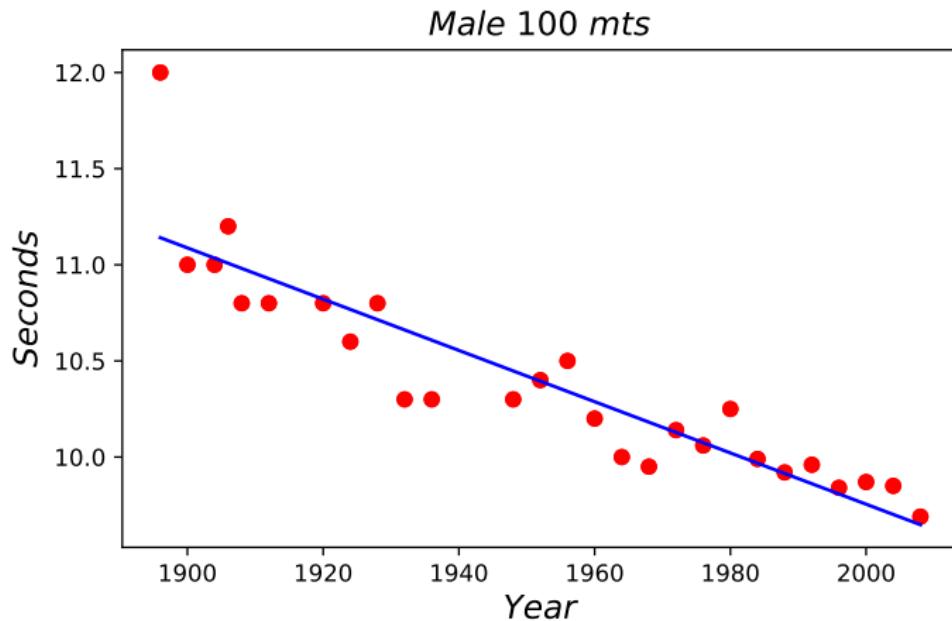
- We will use a linear model $f(x, \mathbf{w})$ to predict y , where y is the time in seconds and x the year of the competition.
- The linear model is given as

$$f(x, \mathbf{w}) = w_0 + w_1 x,$$

where w_0 is the intercept and w_1 is the slope.

- We use \mathbf{w} to refer both to w_0 and w_1 .

Data and model



Contents

Review of vector/matrix notation and linear algebra

A regression model

Linear regression

Gradient descent

Stochastic Gradient Descent

Regularisation

Linear model

- A simple model for regression consists in using a linear combination of the attributes to predict the output

$$f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D,$$

where w_0, w_1, \dots, w_D are the parameters of the regression model.

- The term w_0 is the bias term or intercept, e.g. $f(\mathbf{0}, \mathbf{w}) = w_0$.
- The expression above can be written in a vectorial form

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}.$$

where we have defined $\mathbf{w} = [w_0, w_1, \dots, w_D]^\top$ and $\mathbf{x} = [1, x_1, \dots, x_D]^\top$.

- Notice that $x_0 = 1$.

Parenthesis: Gaussian pdf

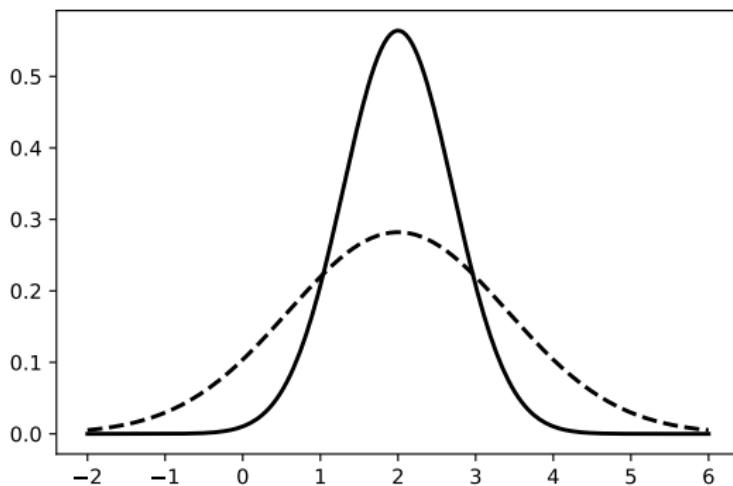
- The Gaussian pdf has the form

$$p(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y-\mu)^2}{2\sigma^2}\right\}.$$

- A Gaussian pdf requires two parameters μ and σ^2 , the mean and the variance of the RV Y .
- We denote the Gaussian pdf as $p(y|\mu, \sigma^2) = \mathcal{N}(y|\mu, \sigma^2)$ or $y \sim \mathcal{N}(\mu, \sigma^2)$.

Parenthesis: Gaussian pdf

The mean of the three Gaussians is $\mu = 2$ and the variances are $\sigma^2 = 0.5$ (solid), and $\sigma^2 = 2$ (dashed).



Gaussian regression model (I)

- We use a Gaussian regression model to relate the inputs and outputs

$$y = f(\mathbf{x}, \mathbf{w}) + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

- It assumes that each output y_i that we observe can be explained as the prediction of an underlying model, $f(\mathbf{x}_i, \mathbf{w})$ plus a noise term ϵ_i .
- For a fixed \mathbf{x} and a fixed \mathbf{w} , $f(\mathbf{x}, \mathbf{w})$ is a constant, then

$$y = \text{constant} + \epsilon,$$

where ϵ is a continuous RV.

- What is the pdf for y ? (we are adding a constant to a Gaussian RV)
 - $E\{y\} = E\{\text{constant} + \epsilon\} = \text{constant}$
 - $\text{var}\{y\} = \text{var}\{\text{constant}\} + \text{var}\{\epsilon\} = \sigma^2$.

Gaussian regression model (II)

- This means that

$$y \sim \mathcal{N}(\text{constant}, \sigma^2),$$

where we said constant was $f(\mathbf{x}, \mathbf{w})$, this is,

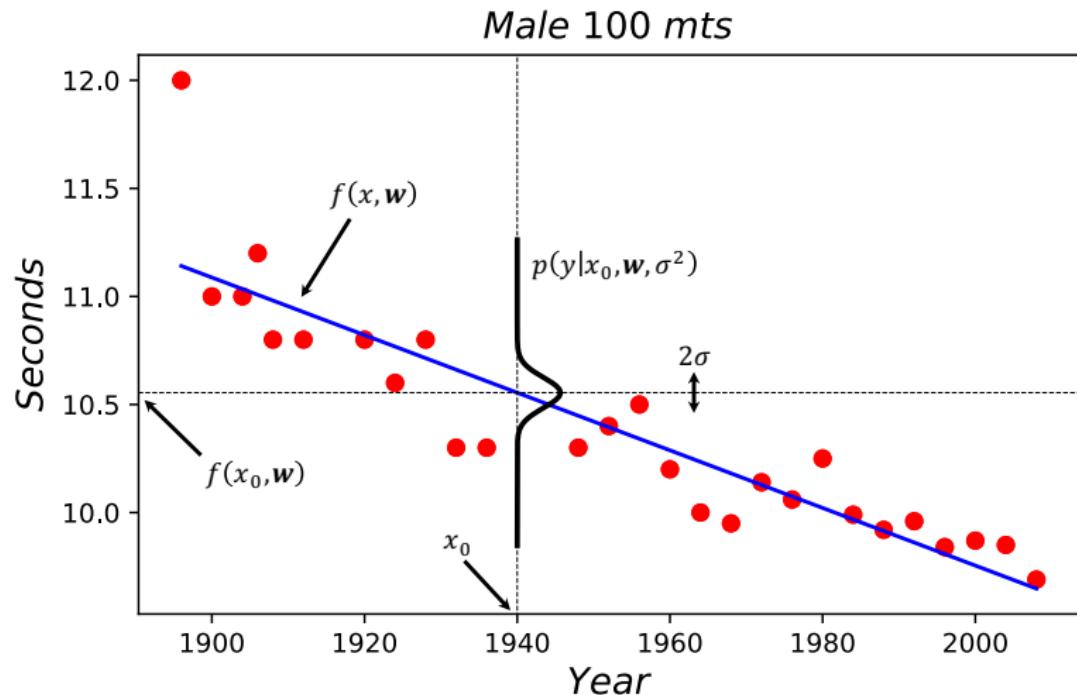
$$y \sim \mathcal{N}(f(\mathbf{x}, \mathbf{w}), \sigma^2).$$

- Because we assumed that \mathbf{x} and \mathbf{w} are given, we can also write

$$p(y|\mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(y|f(\mathbf{x}, \mathbf{w}), \sigma^2).$$

- If we knew the value for \mathbf{w} , once we have a new \mathbf{x}_* , we can predict the output as $f(\mathbf{x}_*, \mathbf{w})$.
- σ^2 tells us the noise variance.

Gaussian regression model (III)



How do we estimate \mathbf{w} ? (I)

- We start with a training dataset $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$.
- We assume that the random variables Y_1, \dots, Y_N are *independent*,

$$p(y_1, \dots, y_N | \mathbf{x}_1, \dots, \mathbf{x}_N) = p(y_1 | \mathbf{x}_1) \cdots p(y_N | \mathbf{x}_N) = \prod_{n=1}^N p(y_n | \mathbf{x}_n).$$

- We also assume that the RVs Y_1, \dots, Y_N follow an *identical* distribution, Gaussian in this case

$$p(y_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(y_n | f(\mathbf{x}_n, \mathbf{w}), \sigma^2) = \mathcal{N}(y_n | \mathbf{w}^\top \mathbf{x}_n, \sigma^2).$$

- Both assumptions go by the name of the *iid* assumption, *independent and identically distributed*.

How do we estimate \mathbf{w} ? (II)

- Putting both assumptions together, we get

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(y_n|\mathbf{w}^\top \mathbf{x}_n, \sigma^2),$$

where $\mathbf{y} = [y_1, \dots, y_N]^\top \in \mathbb{R}^{N \times 1}$ and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times (D+1)}$.

- The expression above can then be written as

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) &= \prod_{n=1}^N \mathcal{N}(y_n|\mathbf{w}^\top \mathbf{x}_n, \sigma^2), \\ &= \prod_{n=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(y_n - \mathbf{w}^\top \mathbf{x}_n)^2}{2\sigma^2} \right\}. \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 \right\}. \end{aligned}$$

How do we estimate \mathbf{w} ? (III)

- When we look at a Gaussian pdf, like

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

we assume that both μ and σ^2 are given. In this case, the pdf follows all the properties we reviewed before.

- The same is true for

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(y_n|\mathbf{w}^\top \mathbf{x}_n, \sigma^2).$$

- Given $\mathbf{w}^\top \mathbf{x}_n$ and σ^2 , then each $p(y_n|\mathbf{x}_n, \mathbf{w}, \sigma^2)$ is a pdf.
- A different approach would be to say: I have some data for $\{y_n\}_{n=1}^N$ and $\{\mathbf{x}_n\}_{n=1}^N$ but
 - “I don’t know what is \mathbf{w}^\top (therefore I don’t know what is $\mathbf{w}^\top \mathbf{x}_n$)”
 - “I don’t know what is σ^2 ”.

How do we estimate \mathbf{w} ? (IV)

- With y_n and \mathbf{x}_n given but with unknown values for \mathbf{w} and σ^2 , each $p(y_n|\mathbf{x}_n, \mathbf{w}, \sigma^2)$ is not a pdf anymore.
- In that case, the function

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^\top \mathbf{x}_n, \sigma^2),$$

receives the name of a *likelihood function*.

- We can think of a likelihood function as a function of the parameters \mathbf{w} and σ^2 ,

$$g(\mathbf{w}, \sigma^2) = p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2),$$

- And subsequently, we can use *multivariate calculus* to find the values of \mathbf{w}, σ^2 that maximise $g(\mathbf{w}, \sigma^2)$.
- In statistics, this is known as the *maximum-likelihood* (ML) criterion to estimate parameters.

How do we estimate \mathbf{w} ? (V)

- Given \mathbf{y}, \mathbf{X} , we use the ML criterion to find the parameters \mathbf{w} and σ^2 that maximise

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 \right\}.$$

- In practice, we prefer to maximise the log of the likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2)$,

$$\begin{aligned} LL(\mathbf{w}, \sigma^2) &= \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) \\ &= -\frac{N}{2} \log (2\pi) - \frac{N}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2. \end{aligned}$$

- Consistency of the ML criterion** If data was really generated according to the probability we specified, the correct parameters will be recovered in the limit as $N \rightarrow \infty$.

Connection with the sum of squared errors

- If we multiply $LL(\mathbf{w}, \sigma^2)$ by minus one, we get

$$E(\mathbf{w}, \sigma^2) = -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) \propto \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2.$$

- The ML criterion for this model has a close connection with the sum-of-squared errors used in non-probabilistic formulations of linear regression.
- Maximising the log-likelihood function is equivalent to minimising the sum-of-squares errors.
- Notice that the log is a monotonic function, meaning that if we find \mathbf{w}, σ^2 that maximise $g(\mathbf{w}, \sigma^2)$, those will also maximise $\log(g(\mathbf{w}, \sigma^2))$.

Normal equation (I)

- Let us find an estimate for \mathbf{w} .
- From what we saw before,

$$LL(\mathbf{w}, \sigma^2) = -\frac{N}{2} \log(2\pi) - \frac{N}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2.$$

- Using what we reviewed in the section on vector/matrix notation, it can be shown that this expression can be written in a vectorial form as

$$LL(\mathbf{w}, \sigma^2) = -\frac{N}{2} \log(2\pi) - \frac{N}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})$$

- Let us focus on the term $(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})$,

$$(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{y}^\top \mathbf{y} - \mathbf{w}^\top \mathbf{X}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{w} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

Normal equation (II)

- We can find the \mathbf{w} that maximises $LL(\mathbf{w}, \sigma^2)$ by taking the gradient $\frac{dLL(\mathbf{w}, \sigma^2)}{d\mathbf{w}}$, equating to zero and solving for \mathbf{w} .
- Taking the gradient of each term in $LL(\mathbf{w}, \sigma^2)$ wrt \mathbf{w} , we get

$$\frac{d}{d\mathbf{w}} \left[-\frac{N}{2} \log(2\pi) \right] = 0, \quad \frac{d}{d\mathbf{w}} \left[-\frac{N}{2} \log \sigma^2 \right] = 0, \quad \frac{d}{d\mathbf{w}} \left[-\frac{1}{2\sigma^2} \mathbf{y}^\top \mathbf{y} \right] = 0,$$
$$\frac{d}{d\mathbf{w}} \left[\frac{1}{2\sigma^2} \mathbf{w}^\top \mathbf{X}^\top \mathbf{y} \right] = \frac{1}{2\sigma^2} \mathbf{X}^\top \mathbf{y},$$
$$\frac{d}{d\mathbf{w}} \left[\frac{1}{2\sigma^2} \mathbf{y}^\top \mathbf{X} \mathbf{w} \right] = \frac{1}{2\sigma^2} \mathbf{X}^\top \mathbf{y}$$
$$\frac{d}{d\mathbf{w}} \left[-\frac{1}{2\sigma^2} \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} \right] = -\frac{1}{2\sigma^2} 2 \mathbf{X}^\top \mathbf{X} \mathbf{w}$$

Normal equation (III)

- Putting these terms together, we get

$$\begin{aligned}\frac{d}{d\mathbf{w}} LL(\mathbf{w}, \sigma^2) &= \frac{1}{2\sigma^2} \mathbf{X}^\top \mathbf{y} + \frac{1}{2\sigma^2} \mathbf{X}^\top \mathbf{y} - \frac{1}{2\sigma^2} 2\mathbf{X}^\top \mathbf{X}\mathbf{w} \\ &= \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{y} - \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X}\mathbf{w}\end{aligned}$$

- Now, equating to zero and solving for \mathbf{w} , we get

$$\begin{aligned}\frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{y} - \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X}\mathbf{w} &= \mathbf{0} \\ \mathbf{X}^\top \mathbf{X}\mathbf{w} &= \mathbf{X}^\top \mathbf{y} \\ \mathbf{w}_* &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.\end{aligned}$$

- The expression for \mathbf{w}_* is known as the *normal equation*.
- The solution for \mathbf{w}^* exists if we can compute $(\mathbf{X}^\top \mathbf{X})^{-1}$.
- The inverse can be computed as long as $\mathbf{X}^\top \mathbf{X}$ is non-singular (e.g. determinant different from zero, or has full-rank).

Solving for σ_*^2

- Following a similar procedure, it can be shown that the ML solution for σ_*^2 is given as

$$\sigma_*^2 = \frac{1}{N}(\mathbf{y} - \mathbf{X}\mathbf{w}_*)^\top(\mathbf{y} - \mathbf{X}\mathbf{w}_*).$$

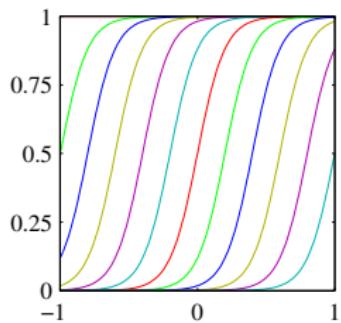
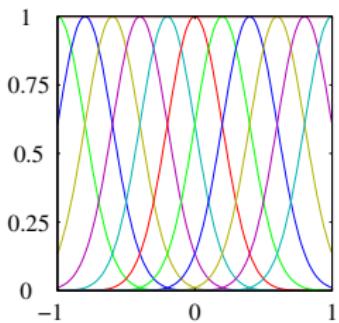
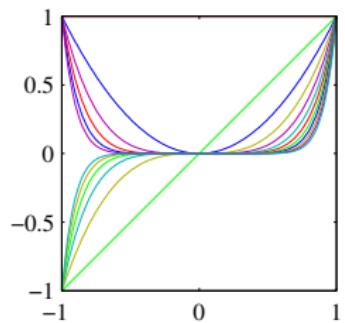
Basis functions

- The model that is linear in \mathbf{x} only allows linear relationships between \mathbf{x} and y .
- We can extend the model to describe non-linear relationships between the inputs and the output by using basis functions, non-linear mappings from inputs to outputs.
- However, we keep the linear relationship of y wrt \mathbf{w} for tractability.
- The predictive model follows as $f(\mathbf{x}, \mathbf{w})$

$$f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^M w_i \phi_i(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}),$$

where $\phi_i(\mathbf{x})$ are basis functions and we have $M + 1$ parameters for the vector \mathbf{w} and $\phi(\mathbf{x}) = [1, \phi_1(\mathbf{x}), \dots, \phi_M(\mathbf{x})]^\top$.

Examples of basis functions



Polynomial: $\phi_i(x) = x^i$.

Exponential: $\phi_i(x) = \exp\left\{-\frac{(x-\mu_i)^2}{2s^2}\right\}$

Sigmoidal: $\phi_i(x) = \sigma\left(\frac{x-\mu_i}{s}\right)$, $\sigma(a) = 1/(1 + \exp(-a))$.

Transforming the input using the basis functions

- As an example, let us use polynomial basis functions to predict y , the time in seconds in the 100 mt Olympics competition.
- For each x (year of the competition), we now compute the vector of polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^M \end{bmatrix}$$

- We have converted the unidimensional input feature x into a higher dimensional feature representation $\phi(x) \in R^{M+1}$.

Normal equations with a design matrix

- Given \mathbf{X} , we first compute a new design matrix Φ ,

$$\Phi = \begin{bmatrix} \phi(\mathbf{x}_1)^\top \\ \phi(\mathbf{x}_2)^\top \\ \vdots \\ \phi(\mathbf{x}_N)^\top \end{bmatrix} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_M(\mathbf{x}_N) \end{bmatrix}$$

- We now can use (\mathbf{y}, Φ) and write the Gaussian linear regression problem

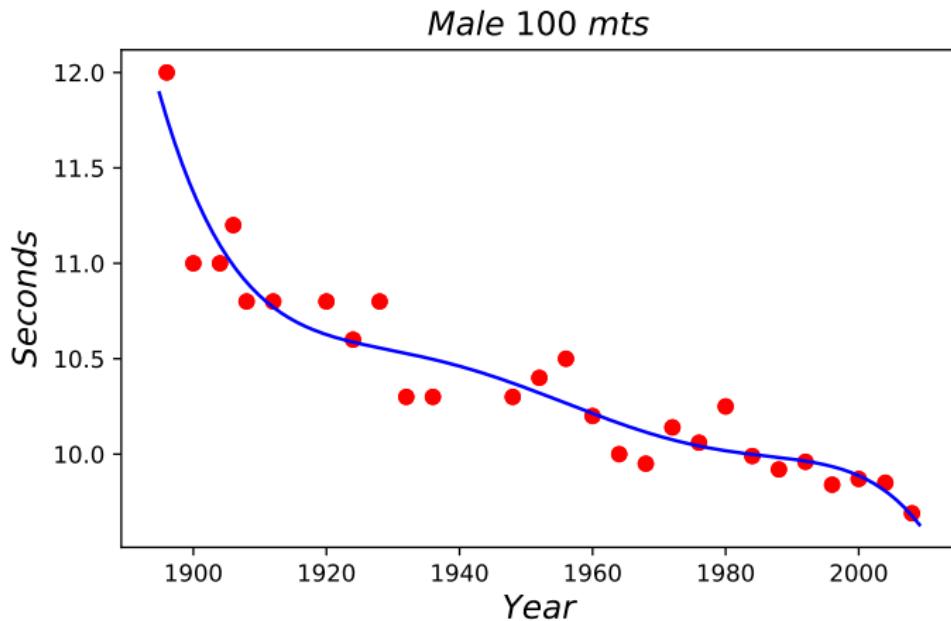
$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^\top \phi_n, \sigma^2),$$

where $\phi_n = \phi(\mathbf{x}_n)$.

- Using the ML criterion, we arrive to the following normal equation

$$\mathbf{w}_* = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}.$$

Olympic 100-mt data with $M = 5$



Alternative to find \mathbf{w}

- For solving the normal equation, we need to invert $\mathbf{X}^\top \mathbf{X}$.
- This inversion has a computational complexity between $\mathcal{O}((D + 1)^{2.4})$ to $\mathcal{O}((D + 1)^3)$ (depending on the implementation).
- The normal equation is linear regarding the number of instances in the training data, $\mathcal{O}(N)$.
- It can handle a large training set as long as it fits in memory.
- Alternatively, we can use iterative optimisation in cases with a large number of features and too many instances to fit in memory.

Contents

Review of vector/matrix notation and linear algebra

A regression model

Linear regression

Gradient descent

Stochastic Gradient Descent

Regularisation

General problem

- We are given a function $h(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^p$.
- Aim: to find a value for \mathbf{w} that minimises $h(\mathbf{w})$.
- Use an iterative procedure

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \mathbf{d}_k,$$

where \mathbf{d}_k is known as the search direction and it is such that

$$h(\mathbf{w}_{k+1}) < h(\mathbf{w}_k).$$

- The parameter η is known as the **step size** or **learning rate**.

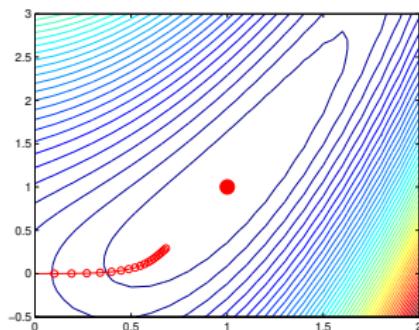
Gradient descent

- Perhaps, the simplest algorithm for unconstrained optimisation.
- It assumes that $\mathbf{d}_k = -\mathbf{g}_k$, where $\mathbf{g}_k = \mathbf{g}(\mathbf{w}_k)$.
- Also known as **steepest descent**.
- It can be written like

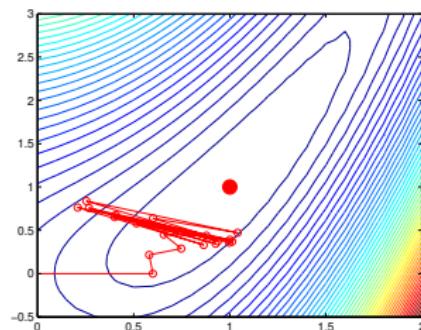
$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \mathbf{g}_k.$$

Step size

- ❑ The main issue in gradient descent is how to set the step size.
- ❑ If it is too small, convergence will be very slow. If it is too large, the method can fail to converge at all.



(a)



(b)

Figure: The function to optimise is $h(w_1, w_2) = 0.5(w_1^2 - w_2)^2 + 0.5(w_1 - 1)^2$. The minimum is at $(1, 1)$. In (a) $\eta = 0.1$. In (b) $\eta = 0.6$.

Alternatives to choose the step size η

- Line search methods (there are different alternatives).
- Line search methods may use search directions other than the steepest descent direction.
- Conjugate gradient (method of choice for quadratic objectives $g(\mathbf{w}) = \mathbf{w}^\top \mathbf{A}\mathbf{w}$).
- Use a *Newton* search direction.

Gradient descent for linear regression (I)

- For simplicity, let us assume that the objective function $h(\mathbf{w})$ corresponds to the mean squared error

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

- We could also minimise the negative $LL(\mathbf{w})$ instead.
- We write the update equation as

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \frac{d}{d\mathbf{w}} E(\mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}_k}.$$

Gradient descent for linear regression (II)

- Computing the gradient for $E(\mathbf{w})$, we get

$$\frac{d}{d\mathbf{w}} E(\mathbf{w}) = \frac{2}{N} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{x}_n - y_n) \mathbf{x}_n = \frac{2}{N} \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}).$$

- The update equation follows as

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \frac{2}{N} \mathbf{X}^\top (\mathbf{X}\mathbf{w}_k - \mathbf{y}).$$

- The computation of the gradient involves using the whole dataset (\mathbf{X}, \mathbf{y}) at every step.
- For this reason, this algorithm is known as *batch gradient descent*.

Gradient descent and feature scaling

- ❑ Always normalise the features if using gradient descent.
- ❑ Gradient descent converges faster if all features have a similar scale.
- ❑ If the attributes are in very different scales, it may take a long time to converge.

Contents

Review of vector/matrix notation and linear algebra

A regression model

Linear regression

Gradient descent

Stochastic Gradient Descent

Regularisation

Online learning and large datasets

- ❑ Traditionally in machine learning, the gradient \mathbf{g}_k is computed using the whole dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$.
- ❑ There are settings, though, where only a subset of the data can be used.
- ❑ **Online learning:** the instances (\mathbf{x}_n, y_n) appear one at a time.
- ❑ **Large datasets:** computing the exact value for \mathbf{g}_k would be expensive, if not impossible.

Stochastic gradient descent (I)

- ❑ In stochastic gradient descent (SGD), the gradient \mathbf{g}_k is computed using a subset of the instances available.
- ❑ The word stochastic refers to the fact that the value for \mathbf{g}_k will depend on the subset of the instances chosen for computation.

Stochastic gradient descent (II)

- In the stochastic setting, a better estimate can be found if the gradient is computed using

$$\mathbf{g}_k = \frac{1}{|S|} \sum_{i \in S} \mathbf{g}_{k,i},$$

where $S \in \mathcal{D}$, $|S|$ is the cardinality of S , and $\mathbf{g}_{k,i}$ is the gradient at iteration k computed using the instance (\mathbf{x}_i, y_i) .

- This setting is called *mini-batch gradient descent*.

Step size in SGD

- Choosing the value of η is particularly important in SGD since there is no easy way to compute it.
- Usually the value of η will depend on the iteration k , η_k .
- It should follow the **Robbins-Monro** conditions

$$\sum_{k=1}^{\infty} \eta_k = \infty, \quad \sum_{k=1}^{\infty} \eta_k^2 < \infty.$$

- Various formulas for η_k can be used

$$\eta_k = \frac{1}{k}, \quad \eta_k = \frac{1}{(\tau_0 + k)^\kappa},$$

where τ_0 slows down early interations and $\kappa \in (0.5, 1]$.

Contents

Review of vector/matrix notation and linear algebra

A regression model

Linear regression

Gradient descent

Stochastic Gradient Descent

Regularisation

What is regularisation?

- ❑ It refers to a technique used for preventing overfitting in a predictive model.
- ❑ It consists in adding a term (a regulariser) to the objective function that encourages simpler solutions.
- ❑ With regularisation, the objective function for linear regression would be

$$h(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w}),$$

where $R(\mathbf{w})$ is the regularisation term and λ the regularisation parameter.

- ❑ In the expression for $h(\mathbf{w})$, we can use the negative $LL(\mathbf{w})$ instead of $E(\mathbf{w})$.
- ❑ If $\lambda = 0$, we get $h(\mathbf{w}) = E(\mathbf{w})$.

Different types of regularisation

- The objective function for linear regression would be

$$h(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w}),$$

where $R(\mathbf{w})$ follows as

$$R(\mathbf{w}) = \alpha \|\mathbf{w}\|_1 + (1 - \alpha) \frac{1}{2} \|\mathbf{w}\|_2^2,$$

where $\|\mathbf{w}\|_1 = \sum_{m=1}^p |w_m|$, and $\|\mathbf{w}\|_2^2 = \sum_{m=1}^p w_m^2$.

- If $\alpha = 1$, we get ℓ_1 regularisation.
- If $\alpha = 0$, we get ℓ_2 regularisation.
- If $0 < \alpha < 1$, we get the elastic net regularisation.

Ridge regression or ℓ_2 regularisation

- In ridge regression, $\alpha = 0$,

$$h(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w},$$

- It can be shown that an optimal solution for \mathbf{w}_* is given as

$$\mathbf{w}_* = \left(\mathbf{X}^\top \mathbf{X} + \frac{\lambda N}{2} \mathbf{I} \right)^{-1} \mathbf{X}^\top \mathbf{y}.$$

- Notice that we can also use iterative procedure for optimising $h(\mathbf{w})$ either through batch gradient decent, SGD or mini-batch SGD.

Automatic differentiation

Mauricio A. Álvarez

Machine Learning and Adaptive Intelligence
The University of Sheffield



The
University
Of
Sheffield.

Contents

Derivatives and ways to compute them

AD modes

- Forward mode
- Reverse mode

Implementations

Derivatives

- Derivatives are required to perform optimisation in several ML algorithms.
- For example, computing the gradient is necessary for batch gradient descent and SGD.
- Derivatives are also necessary for computing Hessians which are used in second-order optimisation methods.

Methods to compute derivatives in computer programs

- Manually working out derivatives and coding them.
- Numeric differentiation using finite difference approximations.
- Symbolic differentiation.
- Automatic differentiation (or algorithmic differentiation).

Example

- Suppose we have the following function

$$f(x, y) = x^2y + y + 2.$$

- We require to compute the gradient of this function, for example, because we want to use it in gradient descent,

$$\frac{df(x, y)}{d\mathbf{z}} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix},$$

where $\mathbf{z} = [x \ y]^\top$.

Manual differentiation (I)

- We use our calculus knowledge to derive the proper equation.
- For the function we saw before, we need to apply the following rules of calculus
 - The derivative of a constant is 0.
 - The derivative of ax with respect to x is a , where a is a constant.
 - The derivative of x^a is ax^{a-1} .
 - The derivative is a linear operation so, the derivative of the sum of two functions is the sum of the derivatives.
 - The derivative of a constant times a function, is equal to the constant times the derivative of that function.
- Using these rules we get the following partial derivatives,

Manual differentiation (II)

- Partial derivative of $f(x, y)$ with respect to x

$$\frac{\partial}{\partial x} f(x, y) = \frac{\partial}{\partial x} (x^2 y + y + 2) = 2xy.$$

- Partial derivative of $f(x, y)$ with respect to y

$$\frac{\partial}{\partial y} f(x, y) = \frac{\partial}{\partial y} (x^2 y + y + 2) = x^2 + 1.$$

- We can then write

$$\frac{df(x, y)}{dz} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2xy \\ x^2 + 1 \end{bmatrix}$$

Problems with manual differentiation

When a function $f(\cdot)$ depends on many variables or is a rather complicated expression, manual differentiation is tedious and prone to mistakes.

Finite difference approximations (I)

- Remember the definition of a derivative of a function $h(x)$ at a point x_0 ,

$$\begin{aligned}\frac{dh(x_0)}{dx} &= \lim_{x \rightarrow x_0} \frac{h(x) - h(x_0)}{x - x_0} \\ &= \lim_{\epsilon \rightarrow 0} \frac{h(x_0 + \epsilon) - h(x_0)}{\epsilon}\end{aligned}$$

- The partial derivative of $h(x, y)$ at point (x_0, y_0) is defined as

$$\begin{aligned}\frac{\partial h(x_0, y_0)}{\partial x} &= \lim_{\epsilon \rightarrow 0} \frac{h(x_0 + \epsilon, y_0) - h(x_0, y_0)}{\epsilon} \\ \frac{\partial h(x_0, y_0)}{\partial y} &= \lim_{\epsilon \rightarrow 0} \frac{h(x_0, y_0 + \epsilon) - h(x_0, y_0)}{\epsilon}\end{aligned}$$

Finite difference approximations (II)

```
def f(x, y):
    return x**2*y + y + 2

x_0 = 3
y_0 = 2
epsilon = 1e-6
dfdx_numerical = (f(x_0+epsilon, y_0) - f(x_0, y_0))/epsilon
dfdy_numerical = (f(x_0, y_0+epsilon) - f(x_0, y_0))/epsilon

dfdx_analytical = 2*x_0*y_0
dfdy_analytical = x_0**2 + 1|
```

Script in python for the finite differences

In [22]: dfdx_numerical
Out[22]: 12.000002001855137

In [23]: dfdx_analytical
Out[23]: 12

$$\frac{\partial f(x,y)}{\partial dx}$$

In [24]: dfdy_numerical
Out[24]: 10.000000003174137

In [25]: dfdy_analytical
Out[25]: 10

$$\frac{\partial f(x,y)}{\partial dy}$$

Problems with finite difference approximation

- ❑ The result is imprecise and gets worse with more complicated functions.
- ❑ We need to call the function at least twice. For big parametric models, we'd need to call the function several times becoming very inefficient.
- ❑ The method is easy to implement, so one can use it to test whether the manual implementation is correct.

Symbolic differentiation (I)

- Symbolic differentiation performs an automatic manipulation of expressions to obtain the corresponding derivative expressions.
- The mathematical expression is represented using data structures (e.g. trees, lists, etc.).
- It is then possible to follow a mechanistic process to obtain the derivatives.

Symbolic differentiation (II)



Maxima



Maple

Symbolic differentiation with Mathematica

In[33]:= D[x, x]

Out[33]:= 1

In[34]:= D[4 x (1 - x), x]

Out[34]:= 4 (1 - x) - 4 x

In[35]:= D[16 x (1 - x) ((1 - 2 x)^2), x]

Out[35]:= 16 (1 - 2 x)^2 (1 - x) - 16 (1 - 2 x)^2 x - 64 (1 - 2 x) (1 - x) x

In[36]:= D[64 x (1 - x) ((1 - 2 x)^2) ((1 - 8 x + 8 x^2)^2), x]

Out[36]:= 128 (1 - 2 x)^2 (1 - x) x (-8 + 16 x) (1 - 8 x + 8 x^2) +
64 (1 - 2 x)^2 (1 - x) (1 - 8 x + 8 x^2)^2 - 64 (1 - 2 x)^2 x (1 - 8 x + 8 x^2)^2 -
256 (1 - 2 x) (1 - x) x (1 - 8 x + 8 x^2)^2

Problems with symbolic differentiation

- Due to the mechanistic approach, there is usually a lot of redundancy in the expressions generated.
- If not handled properly, it produces unnecessary long expressions difficult to make sense of and to evaluate.
- Such behavior is known as *expression swell*.

Example of expression swell

n	I_n	$\frac{dI_n}{dx}$
1	x	1
2	$4x(1-x)$	$4(1-x) - 4x$
3	$16x(1-x)(1-2x)^2$	$16(1-2x)^2(1-x) - 16(1-2x)^2x - 64(1-2x)(1-x)x$
4	$64x(1-x)(1-2x)^2(8x^2 - 8x + 1)^2$	$128(1-2x)^2(1-x)x(-8 + 16x)(1 - 8x + 8x^2) + 64(1-2x)^2(1-x)(1-8x + 8x^2)^2 - 64(1-2x)^2x(1-8x + 8x^2)^2 - 256(1-2x)(1-x)x(1-8x + 8x^2)^2$

Logistic map $I_n = 4I_n(1 - I_n)$, $I_1 = x$.

Simplify with Mathematica

```
In[40]:= D[16 x (1 - x) ((1 - 2 x)^2), x]
```

```
Out[40]= 16 (1 - 2 x)^2 (1 - x) - 16 (1 - 2 x)^2 x - 64 (1 - 2 x) (1 - x) x
```

```
In[39]:= Simplify[16 (1 - 2 x)^2 (1 - x) - 16 (1 - 2 x)^2 x - 64 (1 - 2 x) (1 - x) x]
```

```
Out[39]= -16 (-1 + 10 x - 24 x^2 + 16 x^3)
```

```
In[41]:= D[64 x (1 - x) ((1 - 2 x)^2) ((1 - 8 x + 8 x^2)^2), x]
```

```
Out[41]= 128 (1 - 2 x)^2 (1 - x) x (-8 + 16 x) (1 - 8 x + 8 x^2) +  
64 (1 - 2 x)^2 (1 - x) (1 - 8 x + 8 x^2)^2 - 64 (1 - 2 x)^2 x (1 - 8 x + 8 x^2)^2 -  
256 (1 - 2 x) (1 - x) x (1 - 8 x + 8 x^2)^2
```

```
In[42]:= Simplify[128 (1 - 2 x)^2 (1 - x) x (-8 + 16 x) (1 - 8 x + 8 x^2) +  
64 (1 - 2 x)^2 (1 - x) (1 - 8 x + 8 x^2)^2 - 64 (1 - 2 x)^2 x (1 - 8 x + 8 x^2)^2 -  
256 (1 - 2 x) (1 - x) x (1 - 8 x + 8 x^2)^2]
```

```
Out[42]= -64 (-1 + 42 x - 504 x^2 + 2640 x^3 - 7040 x^4 + 9984 x^5 - 7168 x^6 + 2048 x^7)
```

Automatic differentiation

- AD is concerned about exact numerical computation of the derivatives, rather than their actual symbolic form.
- It computes the derivative by only storing the values of intermediate sub-expressions.
- It uses a combination of: symbolic differentiation at the elementary operation level and keeping intermediate numerical results.

Contents

Derivatives and ways to compute them

AD modes

- Forward mode
- Reverse mode

Implementations

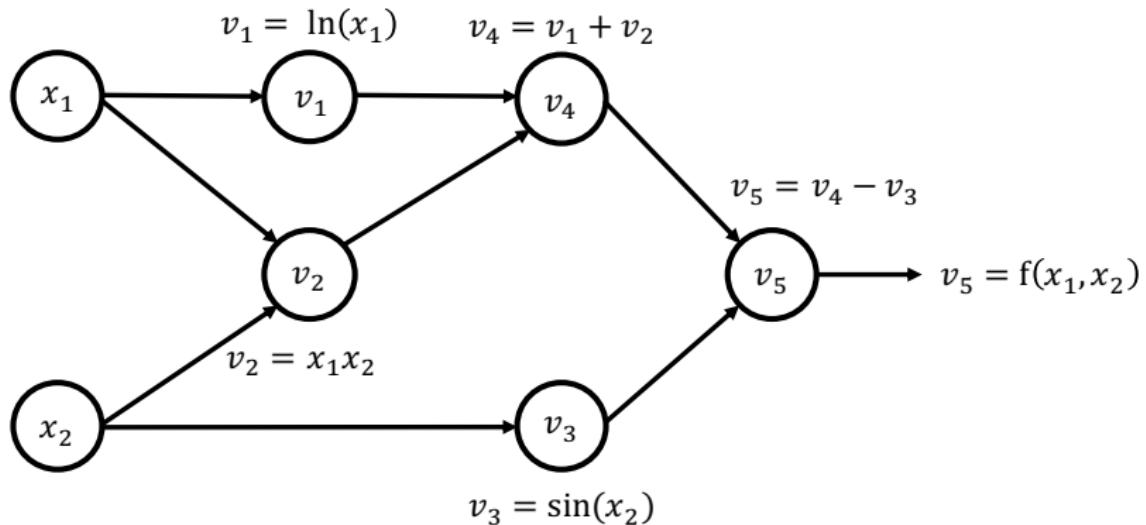
Evaluation trace

- *Evaluation trace*: composition of elementary operations that lead to a full expression.
- As an example, let us consider the function

$$f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2).$$

- The inputs are x_1 and x_2 .
- The elementary operations include
 - $v_1 = \ln(x_1)$
 - $v_2 = x_1 x_2$
 - $v_3 = \sin(x_2)$
 - $v_4 = v_1 + v_2$
 - $v_5 = v_4 - v_3$
 - $f(x_1, x_2) = v_5.$

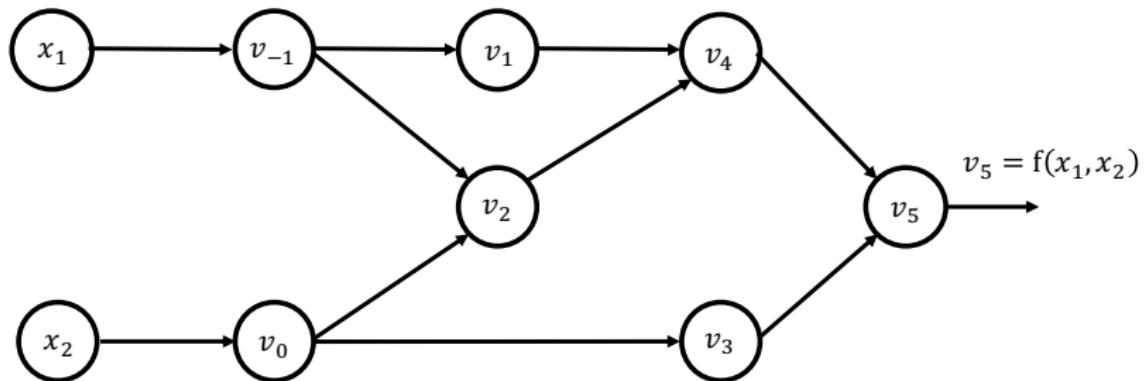
Computational graph



General notation

- Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$.
- Variables $v_{i-n} = x_i$, where $i = 1, \dots, n$ are the input variables.
- Variables v_i , with $i = 1, \dots, l$ are the intermediate variables.
- Variables $y_{m-i} = v_{l-i}$, with $i = m - 1, \dots, 0$ are the output variables.

New computational graph



Jacobian

- Say that we have several functions $y_i = f_i(\cdot)$ for $i = 1, \dots, m$ that depend on several input variables x_1, x_2, \dots, x_n ,

$$\begin{aligned}y_1 &= f_1(x_1, \dots, x_n) \\y_2 &= f_2(x_1, \dots, x_n) \\&\vdots &&\vdots \\y_m &= f_m(x_1, \dots, x_n)\end{aligned}$$

- The Jacobian \mathbf{J} of dimensions $m \times n$ is a matrix with entries $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$ given as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Contents

Derivatives and ways to compute them

AD modes

Forward mode

Reverse mode

Implementations

Forward accumulation mode

- ❑ Forward accumulation mode or tangent linear mode.
- ❑ To compute the derivative of f with respect to x_1 , each intermediate variable v_i has a derivative

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1}$$

- ❑ For each evaluation (or forward primal) trace, it builds a forward derivative (or tangent) trace.
- ❑ Essentially, this forward derivative trace is just implementing the *chain rule of differentiation*

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dz} \frac{dz}{dx}.$$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (I)

- Let us compute the forward tangent trace $\frac{\partial y}{\partial x_1}$ for the function we had before

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2).$$

- The following table shows both the forward primal trace and the forward tangent trace

Forward primal trace	Forward tangent trace
$v_{-1} = x_1$	$\dot{v}_{-1} = \dot{x}_1$
$v_0 = x_2$	$\dot{v}_0 = \dot{x}_2$
$v_1 = \ln v_{-1}$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2$
$v_1 = \ln v_{-1}$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1}$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1}$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_{-1} \times \dot{v}_0 = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_{-1} \times \dot{v}_0 = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$
$v_5 = v_4 - v_3$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace		Forward tangent trace	
$v_{-1} = x_1$	$= 2$	$\dot{v}_{-1} = \dot{x}_1$	$= 1$
$v_0 = x_2$	$= 5$	$\dot{v}_0 = \dot{x}_2$	$= 0$
$v_1 = \ln v_{-1}$	$= \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$	$= \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0$	$= 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0)$	$= \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$	$= 0 \times \cos(5)$
$v_4 = v_1 + v_2$	$= 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$v_5 = v_4 - v_3$		$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$	
$y = v_5$		$\dot{y} = \dot{v}_5$	

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2 = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5$
$v_5 = v_4 - v_3 = 10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 5.5 - 0$
$y = v_5$	$\dot{y} = \dot{v}_5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace	Forward tangent trace
$v_{-1} = x_1 = 2$	$\dot{v}_{-1} = \dot{x}_1 = 1$
$v_0 = x_2 = 5$	$\dot{v}_0 = \dot{x}_2 = 0$
$v_1 = \ln v_{-1} = \ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1} = \frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0 = 2 \times 5$	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1} = 1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0) = \sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0) = 0 \times \cos(5)$
$v_4 = v_1 + v_2 = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 = 0.5 + 5$
$v_5 = v_4 - v_3 = 10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 = 5.5 - 0$
$y = v_5 = 11.652$	$\dot{y} = \dot{v}_5 = 5.5$

Forward primal trace and tangent trace for $\frac{\partial y}{\partial x_1}$ (II)

We now compute the derivative $\frac{\partial y}{\partial x_1}$ at $x_1 = 2, x_2 = 5$.

Forward primal trace		Forward tangent trace	
$v_{-1} = x_1$	= 2	$\dot{v}_{-1} = \dot{x}_1$	= 1
$v_0 = x_2$	= 5	$\dot{v}_0 = \dot{x}_2$	= 0
$v_1 = \ln v_{-1}$	= $\ln 2$	$\dot{v}_1 = \frac{1}{v_{-1}} \dot{v}_{-1}$	= $\frac{1}{2}(1)$
$v_2 = v_{-1} \times v_0$	= 2×5	$\dot{v}_2 = \dot{v}_{-1} \times v_0 + v_0 \times \dot{v}_{-1}$	= $1 \times 5 + 0 \times 2$
$v_3 = \sin(v_0)$	= $\sin(5)$	$\dot{v}_3 = \dot{v}_0 \times \cos(v_0)$	= $0 \times \cos(5)$
$v_4 = v_1 + v_2$	= $0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$	= $0.5 + 5$
$v_5 = v_4 - v_3$	= $10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$	= $5.5 - 0$
$y = v_5$	= 11.652	$\dot{y} = \dot{v}_5$	= 5.5

If we want to compute $\frac{\partial y}{\partial x_2}$ instead, we set $\dot{v}_{-1} = 0$ and $\dot{v}_0 = 1$.

Generalisation to the Jacobian of a function

- Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function with n independent variables x_i and m dependent variables y_j .
- The derivatives in the Jacobian, $\frac{\partial y_j}{\partial x_i}$, are computed by making $\dot{x}_i = 1$ initially in the forward pass and all the other derivatives $\dot{x}_k = 0$ for $k \neq i$.
- The values of the derivatives at $\mathbf{x} = \mathbf{a}$,

$$\dot{y}_j = \left. \frac{\partial y_j}{\partial x_i} \right|_{\mathbf{x}=\mathbf{a}}.$$

are obtained by a forward pass of AD.

- Notice that for a specific x_i , we can compute all the derivatives $\frac{\partial y_j}{\partial x_i}$ for $j = 1, \dots, m$, which corresponds to the column i in the Jacobian.
- To compute the whole Jacobian, we need n forward passes, one per input variable.

Complexity

- AD with forward mode is efficient for functions like $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}^m$.
- The reason, as we saw before, is because we can compute all the derivatives $\frac{\partial y_j}{\partial x}$ for $j = 1, \dots, m$ in one pass.
- In the other extreme, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, it needs n forward passes and it can become computationally expensive when n is large.
- In general, when $n \gg m$, the reverse mode of AD is preferred.

Contents

Derivatives and ways to compute them

AD modes

- Forward mode
- Reverse mode

Implementations

Backpropagate

- AD in reverse mode propagates derivatives backwards from a given output.
- It is done by computing intermediate variables for v_i known as *adjoints*,

$$\bar{v}_i = \frac{\partial y_j}{\partial v_i},$$

representing the sensitivity of output y_j to input v_i .

- AD in reverse mode uses two-phases
 - a *forward* step to compute the variables v_i and to book-keep dependencies in the computational graph.
 - a *backward* or *reverse* step, in which the adjoints are used to compute the derivatives, starting from the outputs and going back to the inputs.

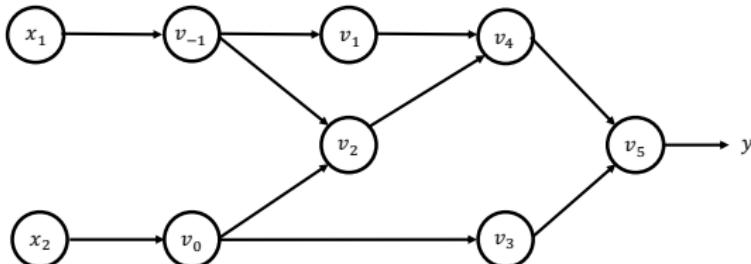
Example (I)

- Let us go back to the example we saw before

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2),$$

and focus on v_0 ($v_0 = x_2$).

- We want to compute the adjoint $\bar{v}_0 = \frac{\partial y}{\partial v_0}$, this is, how the change in v_0 affects the output y .
- From the computational graph, we see that v_0 affects y through v_2 and v_3 ,



Example (II)

- So the contribution of v_0 to y is given as

$$\frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3} \frac{\partial v_3}{\partial v_0}.$$

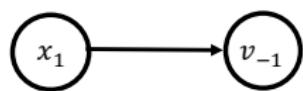
- By definition $\frac{\partial y}{\partial v_2} = \bar{v}_2$ and $\frac{\partial y}{\partial v_3} = \bar{v}_3$, so we can write the expression above as

$$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0}.$$

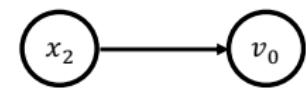
- After the forward pass to compute v_i , the reverse pass computes the adjoints, starting with $\bar{v}_5 = \bar{y} = \frac{\partial y}{\partial y} = 1$, and computing $\frac{\partial y}{\partial x_1} = \bar{x}_1$ and $\frac{\partial y}{\partial x_2} = \bar{x}_2$ at the end.

Forward primal trace (forward pass)

$$x_1 = 2$$



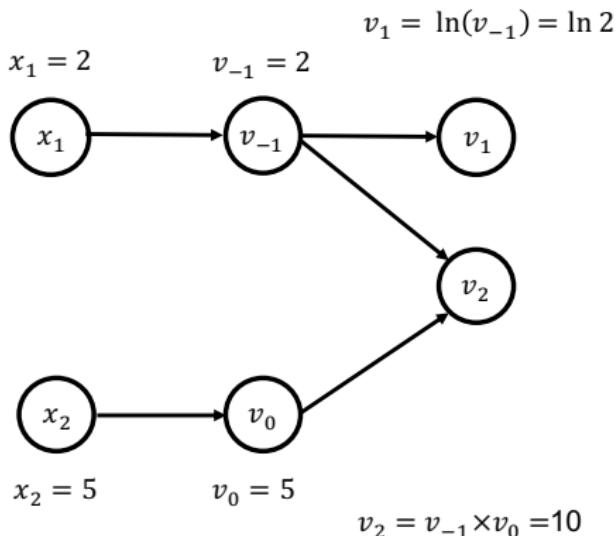
$$v_{-1} = 2$$



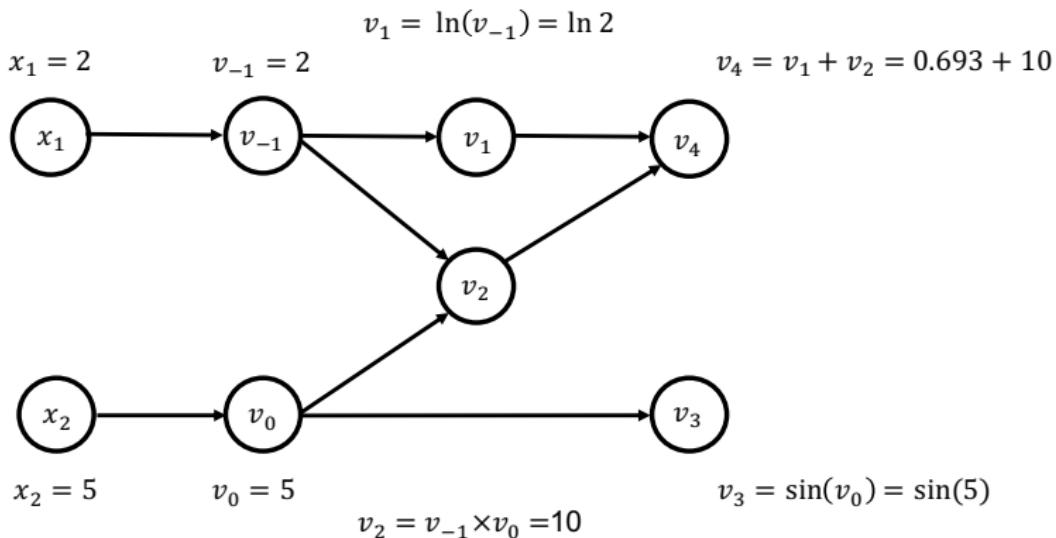
$$x_2 = 5$$

$$v_0 = 5$$

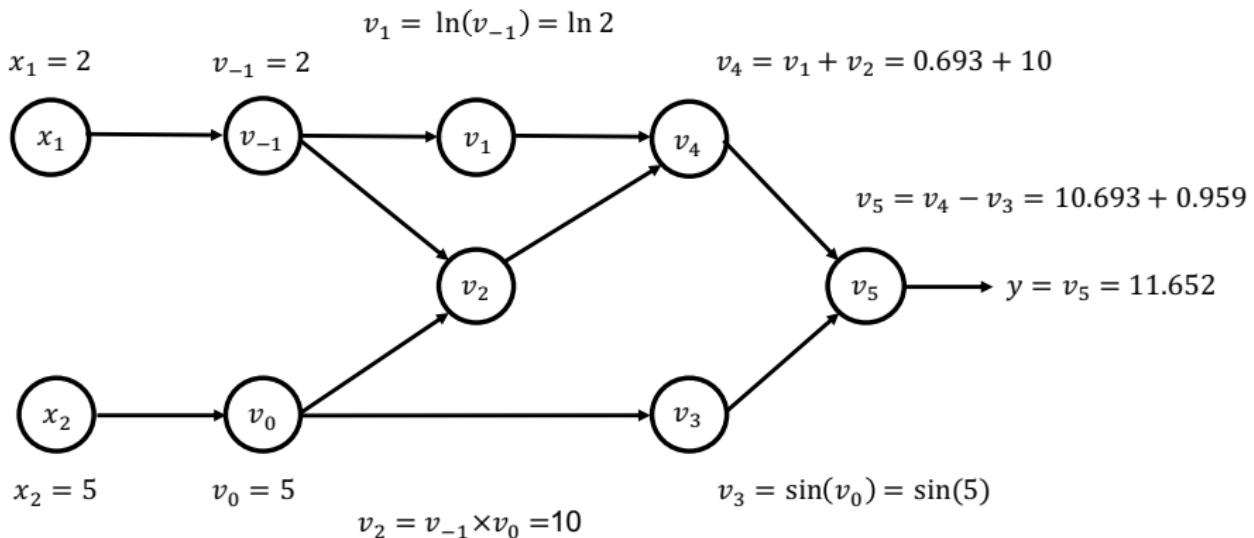
Forward primal trace (forward pass)



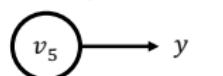
Forward primal trace (forward pass)



Forward primal trace (forward pass)

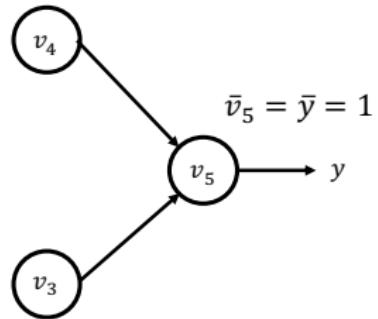


Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_5 = \bar{y} = 1$$


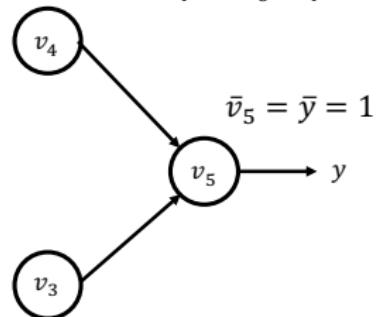
A circular node with the label v_5 inside. An arrow originates from the right side of the circle and points to the right, ending with the label y .

Reverse adjoint (derivative) trace (reverse pass)



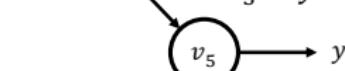
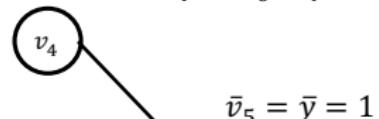
Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$

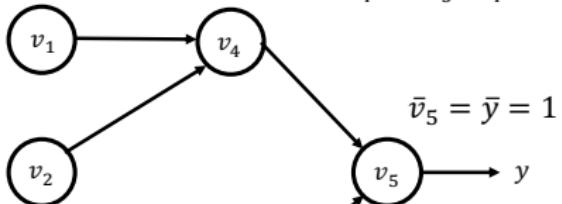


$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$

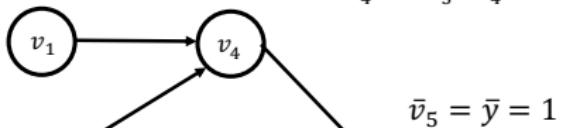


$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



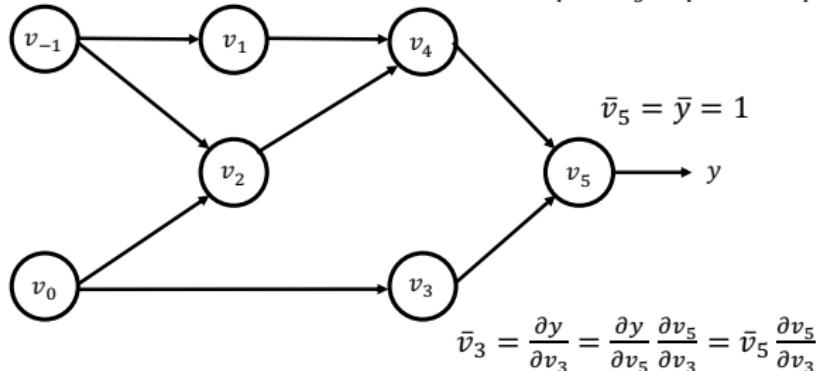
$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_{-1} = \frac{\partial y}{\partial v_{-1}} = \frac{\partial y}{\partial v_1} \frac{\partial v_1}{\partial v_{-1}} + \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} \quad \bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$

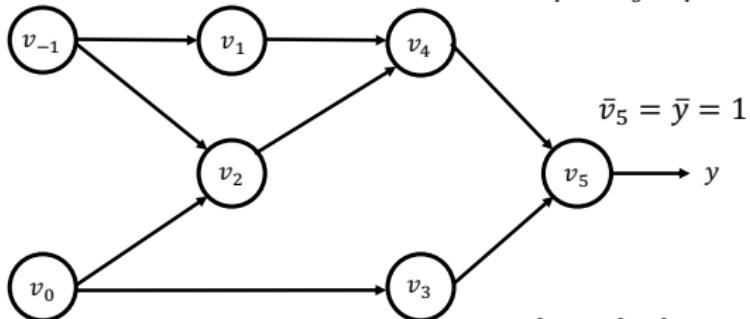


$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

Reverse adjoint (derivative) trace (reverse pass)

$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_{-1} = \frac{\partial y}{\partial v_{-1}} = \frac{\partial y}{\partial v_1} \frac{\partial v_1}{\partial v_{-1}} + \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} \quad \bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



$$\bar{v}_0 = \frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3} \frac{\partial v_3}{\partial v_0} = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0}$$

$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

Reverse adjoint (derivative) trace (reverse pass)

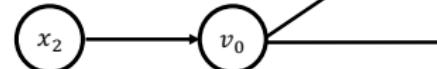
$$\bar{v}_1 = \frac{\partial y}{\partial v_1} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$$

$$\bar{v}_{-1} = \frac{\partial y}{\partial v_{-1}} = \frac{\partial y}{\partial v_1} \frac{\partial v_1}{\partial v_{-1}} + \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} \quad \bar{v}_4 = \frac{\partial y}{\partial v_4} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$$



$$\bar{x}_1 = \frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial v_{-1}} \frac{\partial v_{-1}}{\partial x_1} = \bar{v}_{-1} \frac{\partial v_{-1}}{\partial x_1}$$

$$\bar{x}_2 = \frac{\partial y}{\partial x_2} = \frac{\partial y}{\partial v_0} \frac{\partial v_0}{\partial x_2} = \bar{v}_0 \frac{\partial v_0}{\partial x_2}$$

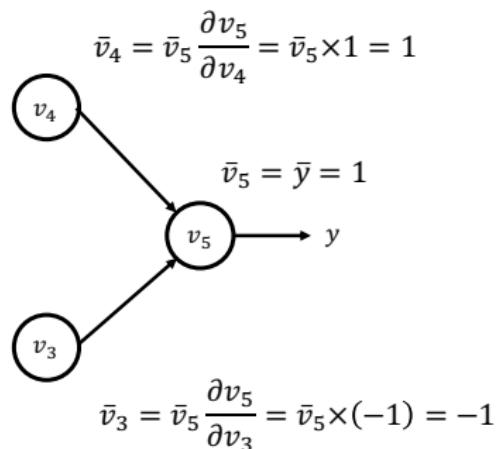


$$\bar{v}_0 = \frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2} \frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3} \frac{\partial v_3}{\partial v_0} = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0}$$

$$\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$$

$$\bar{v}_2 = \frac{\partial y}{\partial v_2} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$$

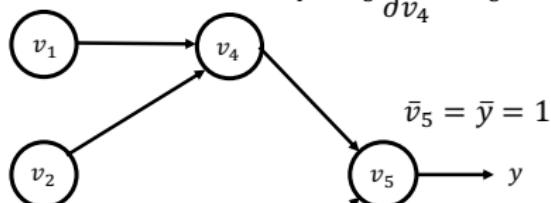
Numerical evaluation of the reverse adjoint trace



Numerical evaluation of the reverse adjoint trace

$$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \times 1 = 1$$

$$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \times 1 = 1$$



$$\bar{v}_5 = \bar{y} = 1$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \times (-1) = -1$$

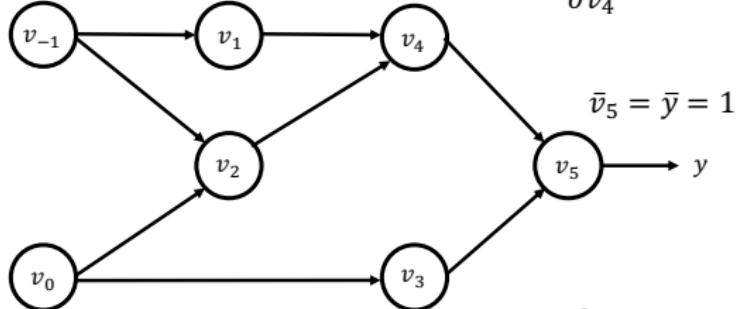
$$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \times 1 = 1$$

Numerical evaluation of the reverse adjoint trace

$$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \times 1 = 1$$

$$\bar{v}_{-1} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \times \frac{1}{v_{-1}} + \bar{v}_2 \times v_0 = 5.5$$

$$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \times 1 = 1$$



$$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0} = \bar{v}_2 \times v_{-1} + \bar{v}_3 \cos(v_0) = 1.716$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \times (-1) = -1$$

$$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \times 1 = 1$$

Numerical evaluation of the reverse adjoint trace

$$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \times 1 = 1$$

$$\bar{v}_{-1} = \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_1 \times \frac{1}{v_{-1}} + \bar{v}_2 \times v_0 = 5.5$$

$$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \times 1 = 1$$

x_1

v_{-1}

v_1

v_4

v_2

v_5

$$\bar{v}_5 = \bar{y} = 1$$

y

$$\bar{x}_1 = \bar{v}_{-1} \frac{\partial v_{-1}}{\partial x_1} = 5.5 \times 1 = 5.5$$

$$\bar{x}_2 = \bar{v}_0 \frac{\partial v_0}{\partial x_2} = 1.716 \times 1 = 1.716$$

x_2

v_0

v_3

v_2

v_5

$$\bar{v}_0 = \bar{v}_2 \frac{\partial v_2}{\partial v_0} + \bar{v}_3 \frac{\partial v_3}{\partial v_0} = \bar{v}_2 \times v_{-1} + \bar{v}_3 \cos(v_0) = 1.716$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \times (-1) = -1$$

$$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \times 1 = 1$$

Complexity

- Reverse mode AD performs better when $n \gg m$.
- The downside is the cost of increased storage, since we need to save intermediate values for v_i in the evaluation trace.

Reverse mode AD and backpropagation

- Reverse mode AD is the algorithm used to train neural networks and deep learning models.
- To train a neural network model, we optimise an objective function, $E(\mathbf{w}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that usually depends on a high-dimensional input vector of parameters $\mathbf{w} \in \mathbb{R}^n$, with $n \gg m$.
- In the machine learning community, reverse mode AD goes by the name of *backpropagation*, which you will see again in the session on neural networks.

Contents

Derivatives and ways to compute them

AD modes

- Forward mode
- Reverse mode

Implementations

AD implementations

Table 5: Survey of AD implementations. Tools developed primarily for machine learning are highlighted in bold.

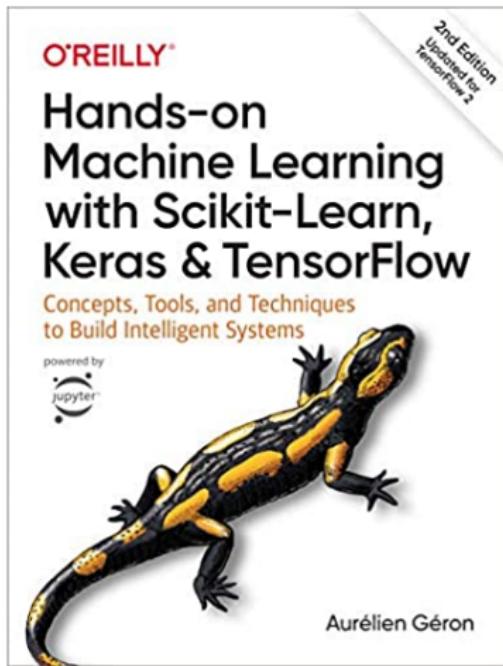
Language	Tool	Type	Mode	Institution / Project	Reference	URL
AMPL	AMPL	INT	F, R	Bell Laboratories	Fourer et al. (2002)	http://www.ampl.com/
C, C++	ADIC	ST	F, R	Argonne National Laboratory	Bischof et al. (1997)	http://www.mcs.anl.gov/research/projects/adic/
	ADOL-C	OO	F, R	Computational Infrastructure for Operations Research	Walther and Griewank (2012)	https://projects.coin-or.org/ADOL-C
C++	Ceres Solver	LIB	F	Google		http://ceres-solver.org/
	CppAD	OO	F, R	Computational Infrastructure for Operations Research	Bell and Burke (2008)	http://www.coin-or.org/CppAD/
	FABDAD++	OO	F, R	Technical University of Denmark	Bendtsen and Stauning (1996)	http://www.fabdad.com/fabdad.html
	Mxyztik	OO	F	Fermi National Accelerator Laboratory	Ostiguy and Michelotti (2007)	
C#	AutoDiff	LIB	R	George Mason Univ., Dept. of Computer Science	Shtof et al. (2013)	http://autodiff.codeplex.com/
F#, C#	DifffSharp	OO	F, R	Maynooth University, Microsoft Research Cambridge	Baydin et al. (2016a)	http://difffsharp.github.io
Fortran	ADIFOR	ST	F, R	Argonne National Laboratory	Bischof et al. (1996)	http://www.mcs.anl.gov/research/projects/adifor/
	NAGWare	COM	F, R	Numerical Algorithms Group	Naumann and Riehme (2005)	http://www.nag.co.uk/nagware/Research/ad_overview.asp
	TAMC	ST	R	Max Planck Institute for Meteorology	Giering and Kaminski (1998)	http://autodiff.com/tamc/
Fortran, C	COSY	INT	F	Michigan State Univ., Biomedical and Physical Sci.	Berz et al. (1996)	http://www.bt.pa.msu.edu/index_cosy.htm
	Tapenade	ST	F, R	INRIA Sophia-Antipolis	Hascoët and Pascual (2013)	http://www-sop.inria.fr/tropics/tapenade.html
Haskell	ad	OO	F, R	Haskell package		http://hackage.haskell.org/package/ad
Java	ADJac	ST	F, R	University Politehnica of Bucharest	Slusanschi and Dumitrel (2016)	http://adjac.cs.pub.ro
	Deriva	LIB	R	Java & Clojure library		https://github.com/lambda/Deriva
Julia	JuliaDiff	OO	F, R	Julia packages	Revels et al. (2016a)	http://www.juliadiff.org/
Lua	torch-autograd	OO	R	Twitter Cortex		https://github.com/twitter/torch-autograd
MATLAB	ADiMat	ST	F, R	Technical University of Darmstadt, Scientific Comp.	Willkomm and Vehreschild (2013)	http://adimat.sc.informatik.tu-darmstadt.de/
	INTLab	OO	F	Hamburg Univ. of Technology, Inst. for Reliable Comp.	Rump (1999)	http://www.ti3.tu-harburg.de/rump/intlab/
	TOMLAB/MAD	OO	F	Cranfield University & Tomlab Optimization Inc.	Forth (2006)	http://tomlab.biz/products/mad
Python	ad	OO	R	Python package		https://pypi.python.org/pypi/ad
	autograd	OO	F, R	Harvard Intelligent Probabilistic Systems Group	MacLaurin (2016)	https://github.com/HIPS/autograd
	Chainer	OO	R	Preferred Networks	Tokui et al. (2015)	https://chainer.org/
	PyTorch	OO	R	PyTorch core team	Paszke et al. (2017)	http://pytorch.org/
	Tangent	ST	F, R	Google Brain	van Merriënboer et al. (2017)	https://github.com/google/tangent
Scheme	R6RS-AD	OO	F, R	Purdue Univ., School of Electrical and Computer Eng.		https://github.com/qobi/R6RS-AD
	Scmutils	OO	F	MIT Computer Science and Artificial Intelligence Lab.	Sussman and Wisdom (2001)	http://groups.csail.mit.edu/mac/users/gjs/6946/refman.txt
	Stalingrad	COM	F, R	Purdue Univ., School of Electrical and Computer Eng.	Pearlmutter and Siskind (2008)	http://www.bcl.hamilton.ie/~qobi/stalingrad/

F: Forward, R: Reverse; COM: Compiler, INT: Interpreter, LIB: Library, OO: Operator overloading, ST: Source transformation

Two popular implementations in the ML community



References



Appendix D of “Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow”

References

Journal of Machine Learning Research 18 (2018) 1-43

Submitted 8/17; Published 4/18

Automatic Differentiation in Machine Learning: a Survey

Atilim Güneş Baydin

*Department of Engineering Science
University of Oxford
Oxford OX1 3PJ, United Kingdom*

GUNES@ROBOTS.OX.AC.UK

Barak A. Pearlmutter

*Department of Computer Science
National University of Ireland Maynooth
Maynooth, Co. Kildare, Ireland*

BARAK@PEARLMUTTER.NET

Alexey Andreyevich Radul

*Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139, United States*

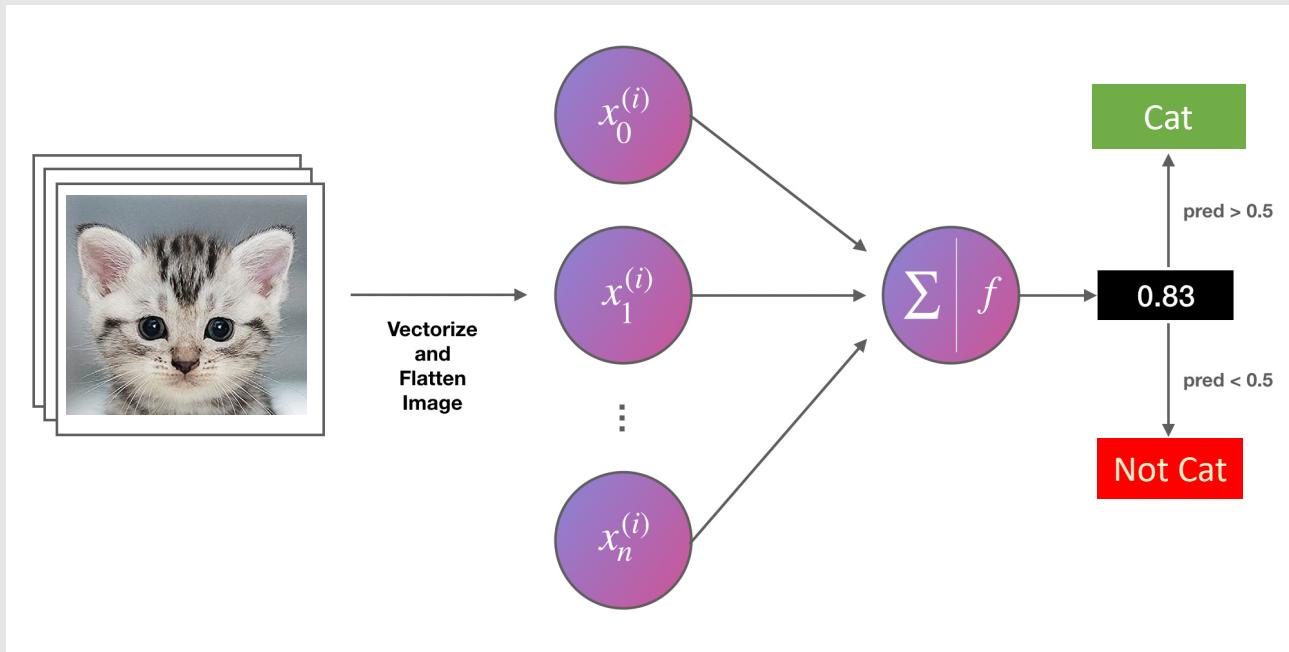
AXCH@MIT.EDU

Jeffrey Mark Siskind

*School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907, United States*

QOBI@PURDUE.EDU

Lecture 6: Logistic Regression & PyTorch for Deep Learning



Haiping Lu

YouTube Playlist: <https://www.youtube.com/c/HaipingLu/playlists>

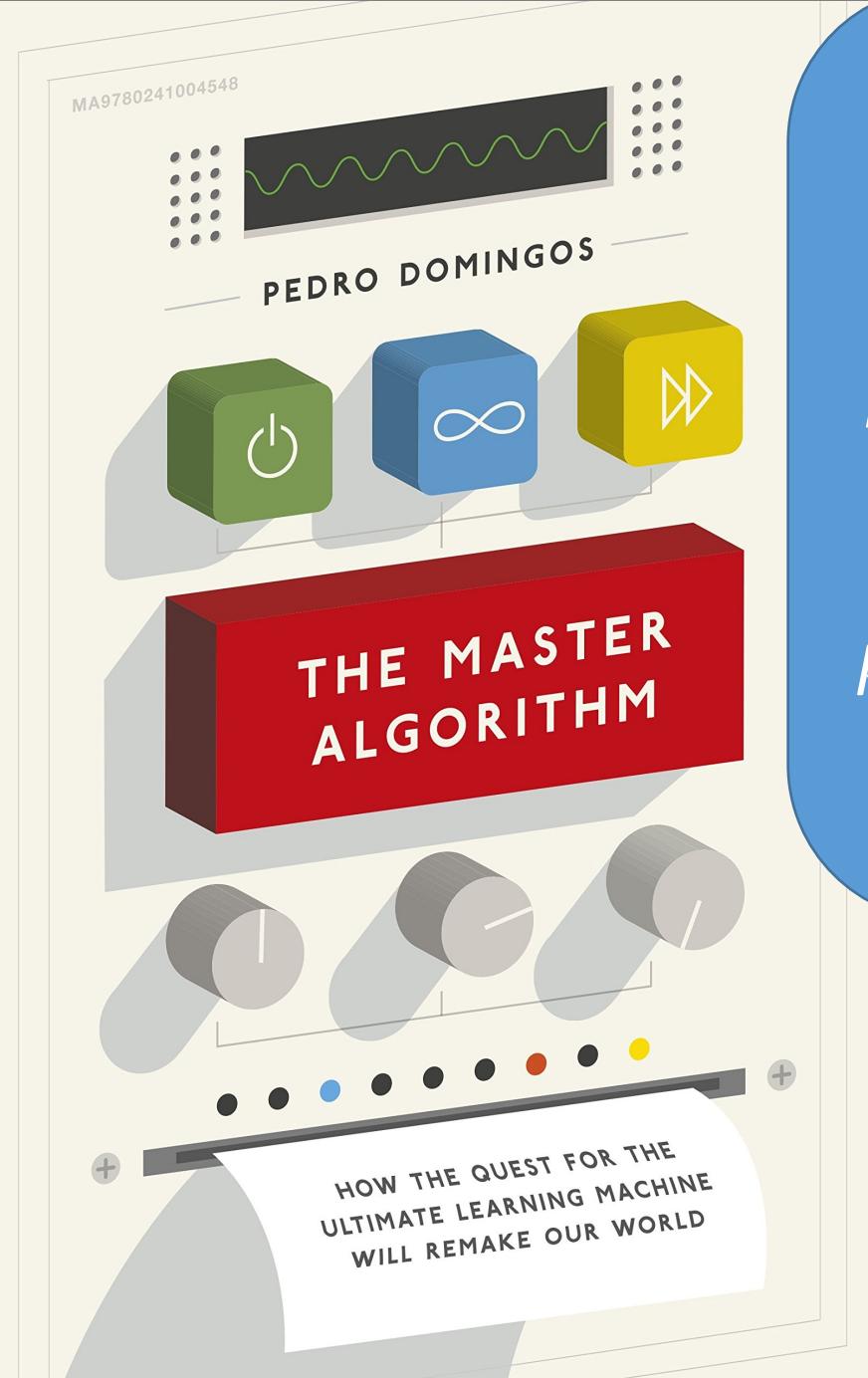
COM4059/6059: MLAI21@The University of Sheffield

Week 6 Contents / Objectives

- Machine Learning Recap
- Motivation for Logistic Regression
- Logistic Regression
- Computational Graph
- PyTorch: A Deep Learning Library

Week 6 Contents / Objectives

- **Machine Learning Recap**
- Motivation for Logistic Regression
- Logistic Regression
- Computational Graph
- PyTorch: A Deep Learning Library



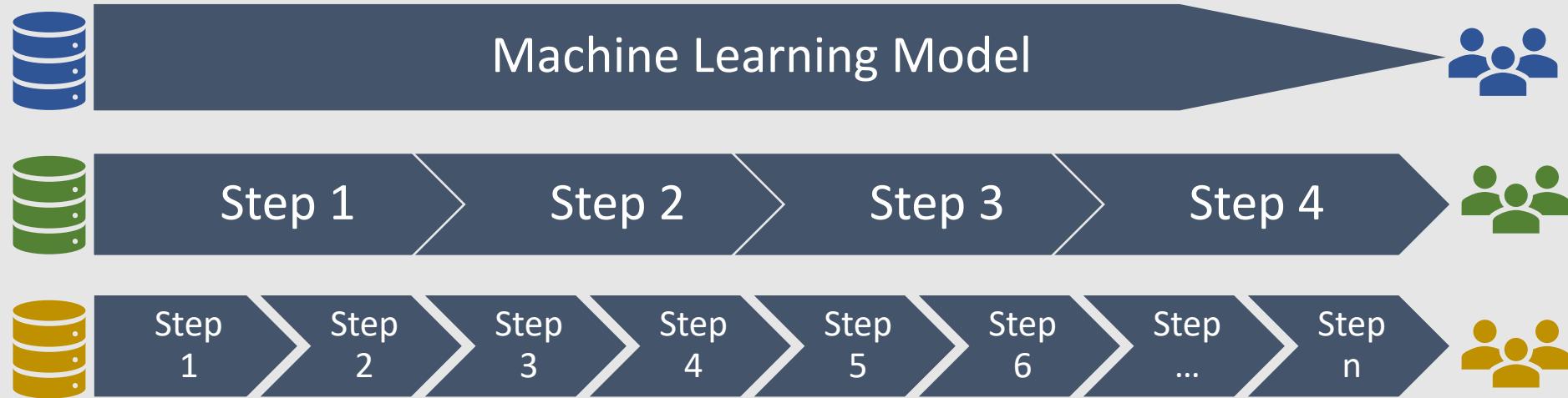
Learning algorithms are the seeds, data is the soil, and the learned programs are the grown plants. The machine-learning expert is like a farmer, sowing the seeds, irrigating and fertilizing the soil, and keeping an eye on the health of the crop but otherwise staying out of the way.



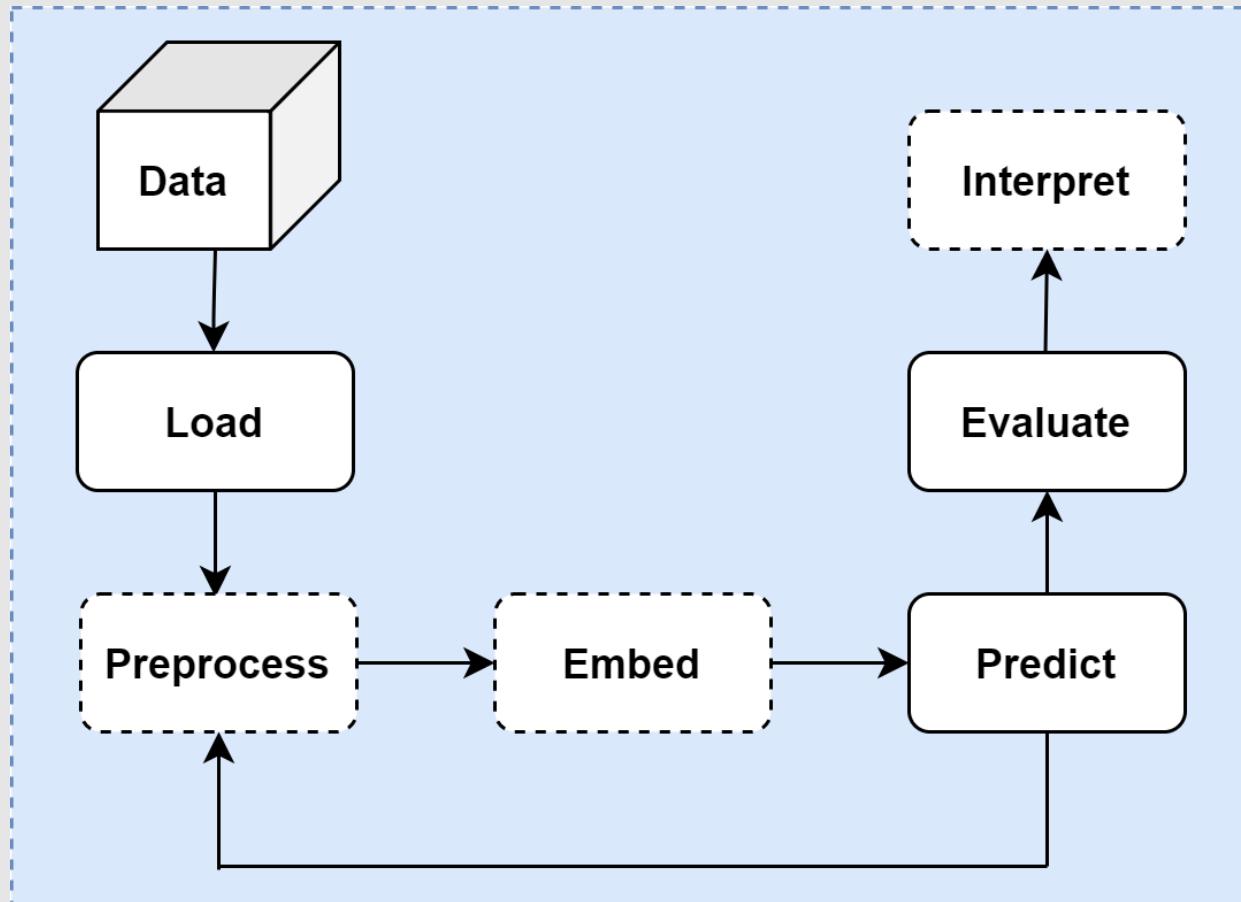
Machine Learning Ingredients

- **Data:** + pre-processing (& visualisation), e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure ~ Architecture \leftarrow expert knowledge
 - Must **specify** before ML, can optimise via cross validation (CV)
 - **Hyper-parameter**, e.g., prior, #degree, layer \leftarrow knowledge
 - Must **specify** (choices) and can optimise via CV (**tuning**)
 - Parameters (theta)
 - Compute/learn parameter, e.g., **weights**, bias \leftarrow optimisation alg.
- Evaluation metric (what's best): loss/error function
- Optimisation: (how to find the best) learnable parameters

ML API without Standardization



Machine Learning Pipeline

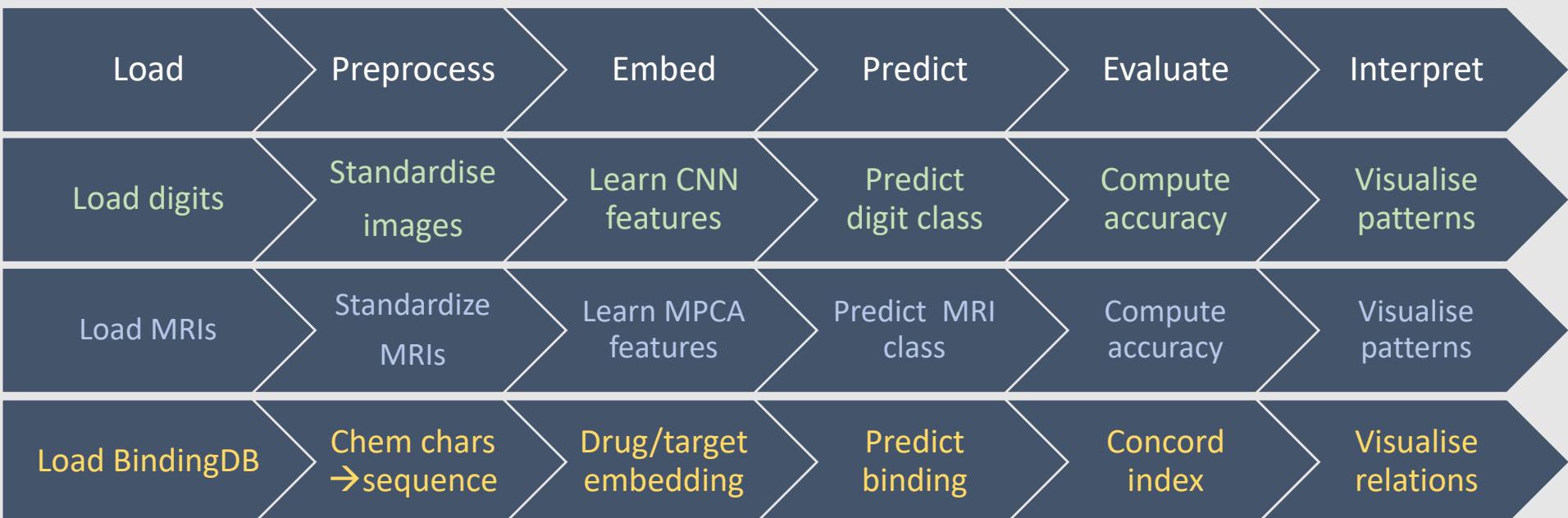


- PyKale: a library defined in this pipeline

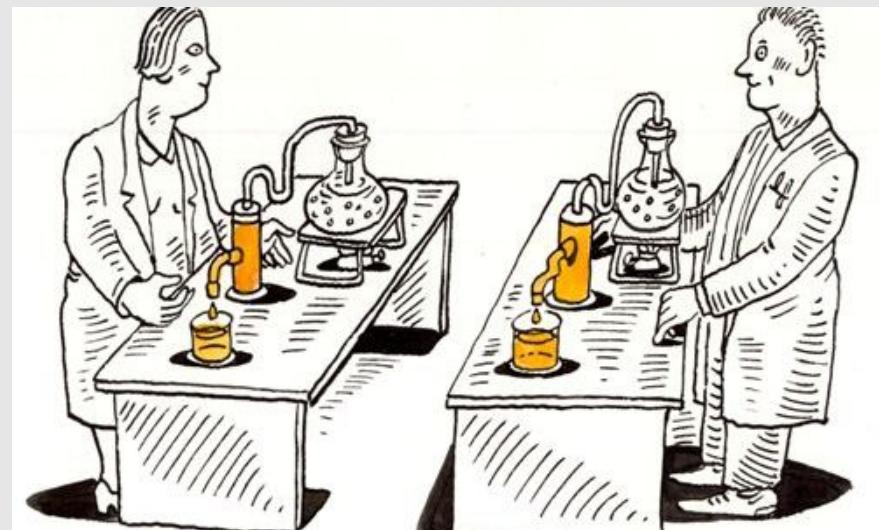
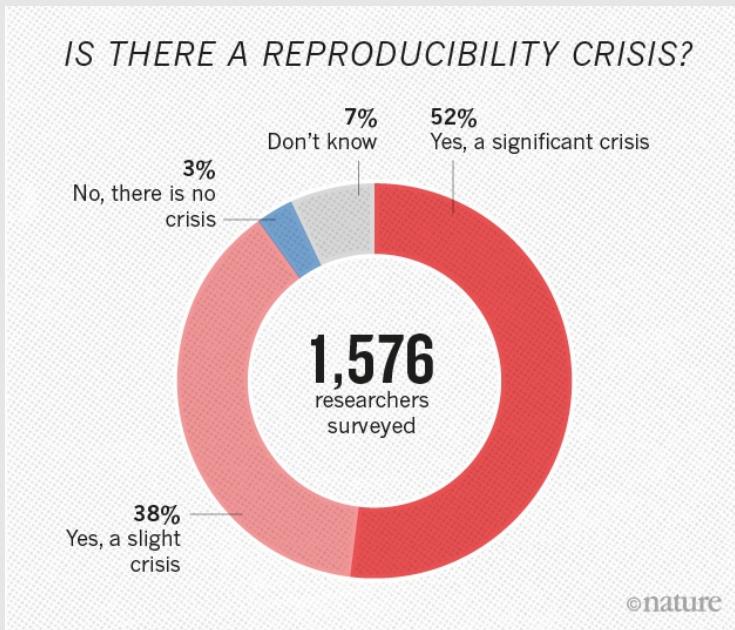


Pipeline-based API in PyKale

- PyTorch library built by us (week 10):
<https://github.com/pykale/pykale>



Reproducibility → Trustworthy



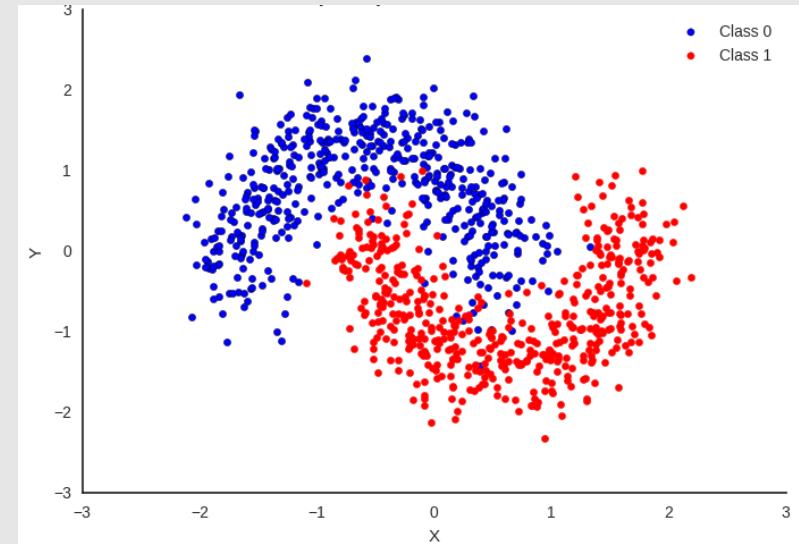
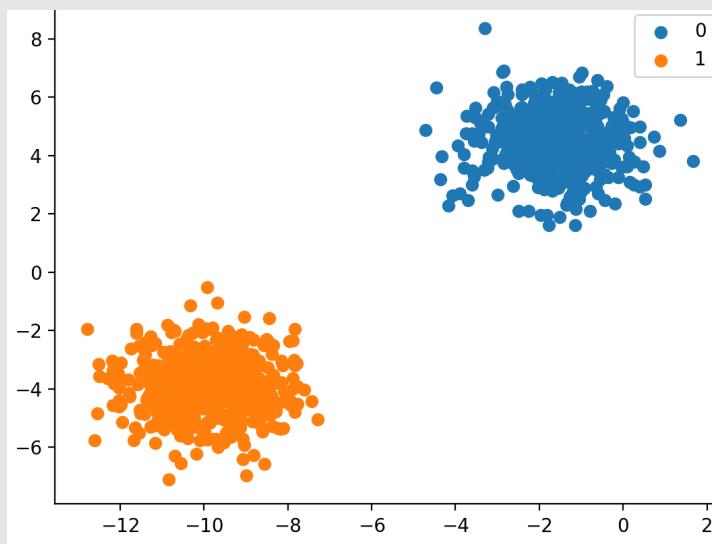
<https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970>

<https://www.ucl.ac.uk/pals/research/experimental-psychology/wp-content/uploads/2016/03/reproducibility-small-496x300.jpg>

- Reproducible machine learning
 - **Make it modular** to help understanding & tracing
 - **Keep a record** of all assumptions and settings
 - **Set a seed** when there is randomness

Start Simple & Small

- The simplest prediction task: binary classification
 - Input (to predict from): feature vectors
 - Output (to predict): 0 or 1
 - Difficulty determined by the distribution of the input



Week 6 Contents / Objectives

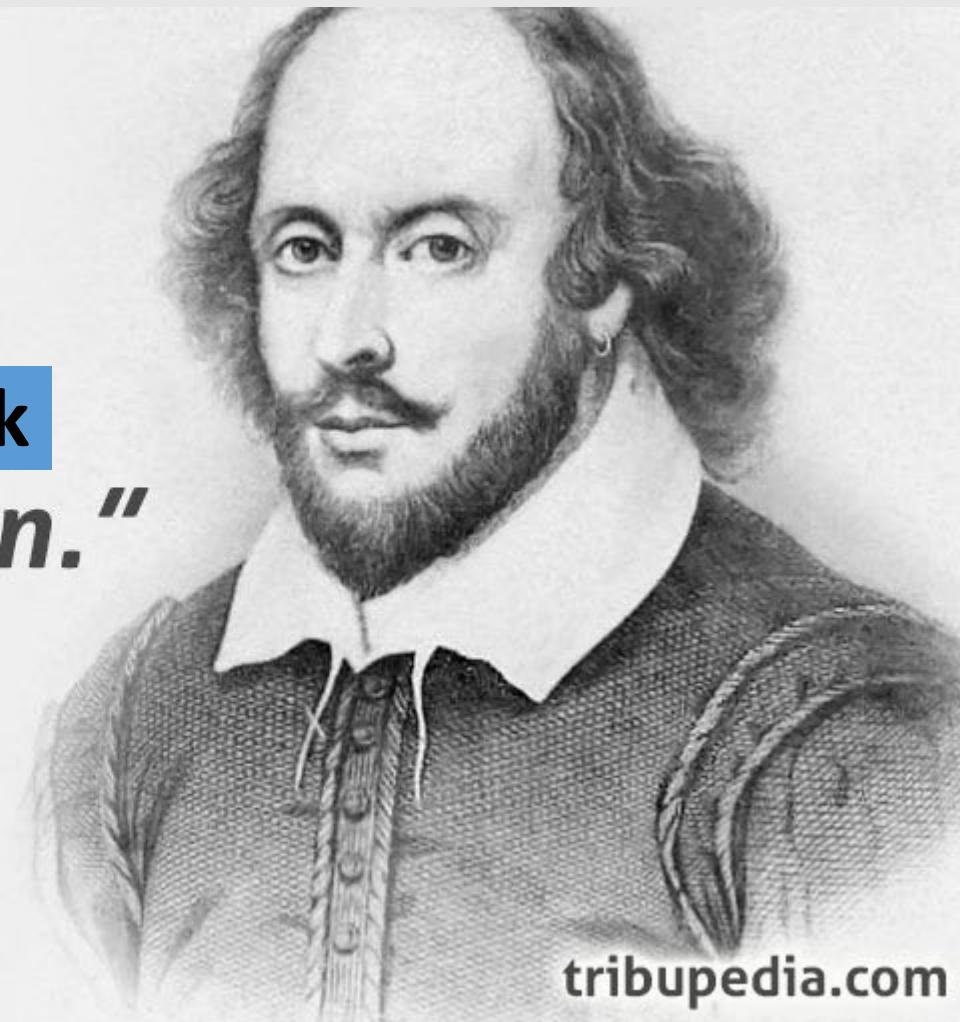
- Machine Learning Recap
- **Motivation for Logistic Regression**
- Logistic Regression
- Computational Graph
- PyTorch: A Deep Learning Library

The Question



***To click or not to click
that is the question.***

William Shakespeare



tribupedia.com

Click-Through Rate (CTR) Prediction

- Estimating click probabilities: What is the probability that user i **will click** on ad j
 - Not important just for ads:
 - Optimize search results
 - Suggest news articles
 - Recommend products
- Logistic regression is used by many internet companies, making lots of money for them
 - E.g., Facebook (*Meta*) ad matching

A Binary Classification Problem

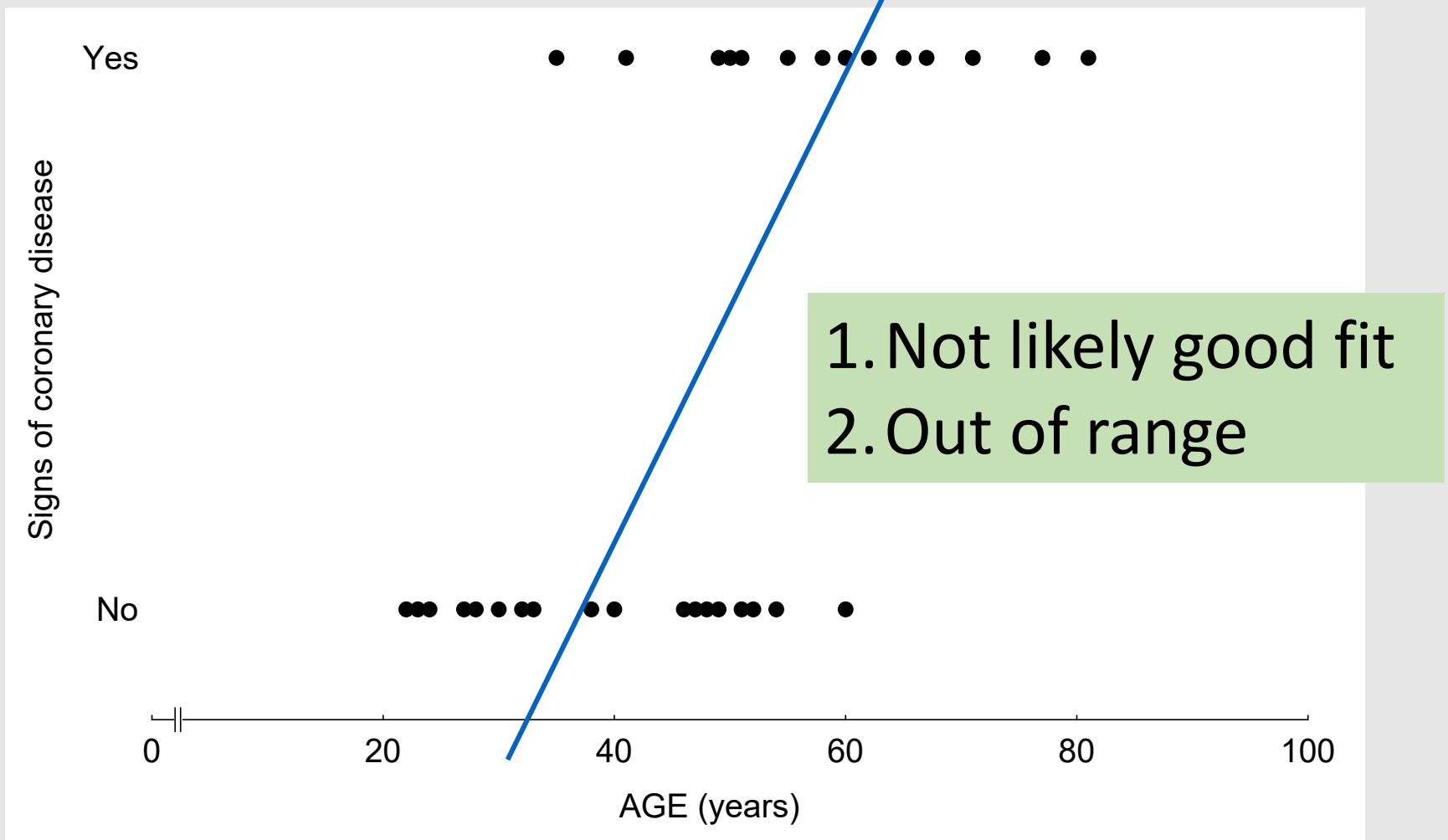
Table 1: Age and signs of coronary heart disease (CD)

Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1

Prediction question: a particular age → CD

Linear regression?

Dot-plot: Data from Table 1

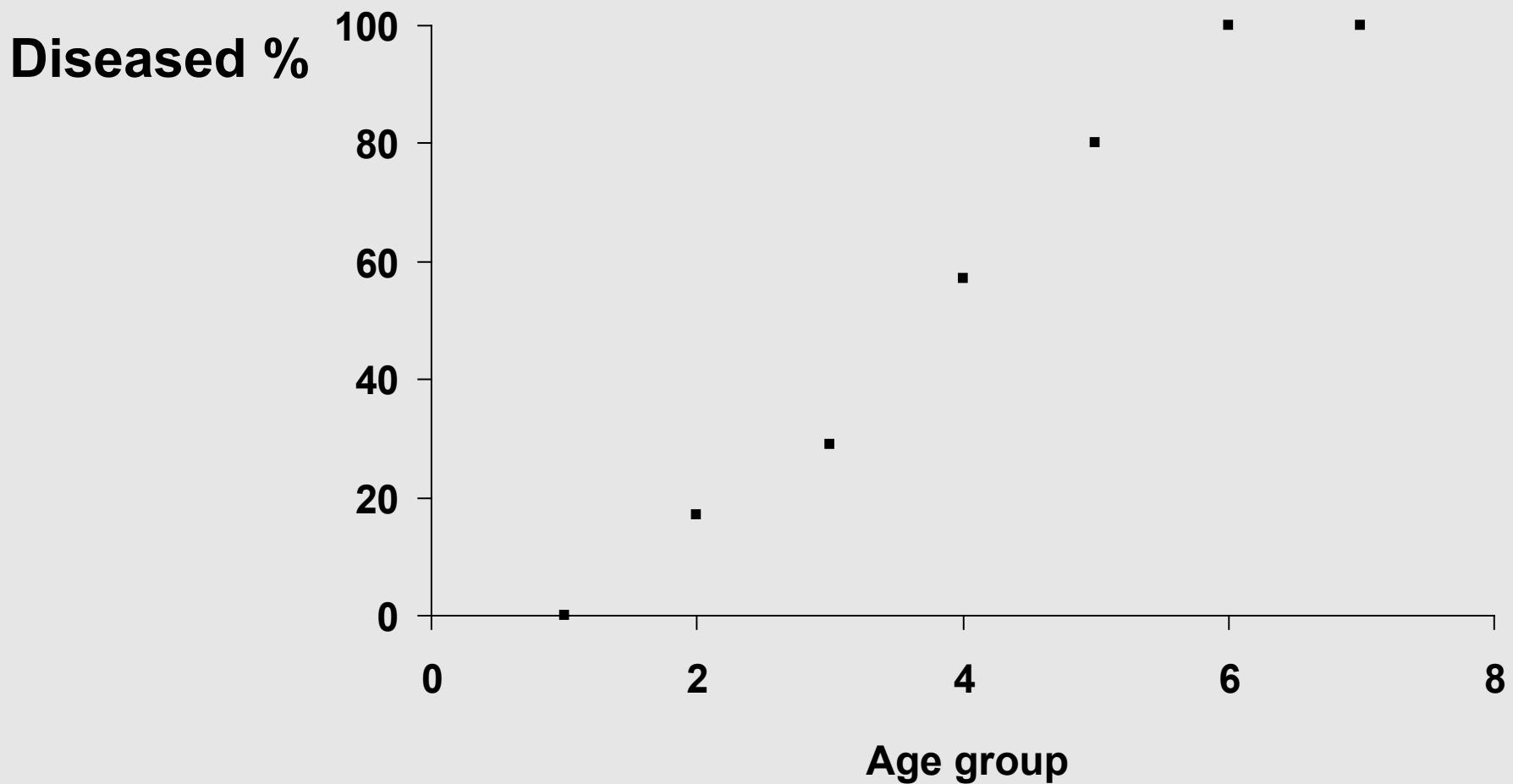


Transform the Data → Probability

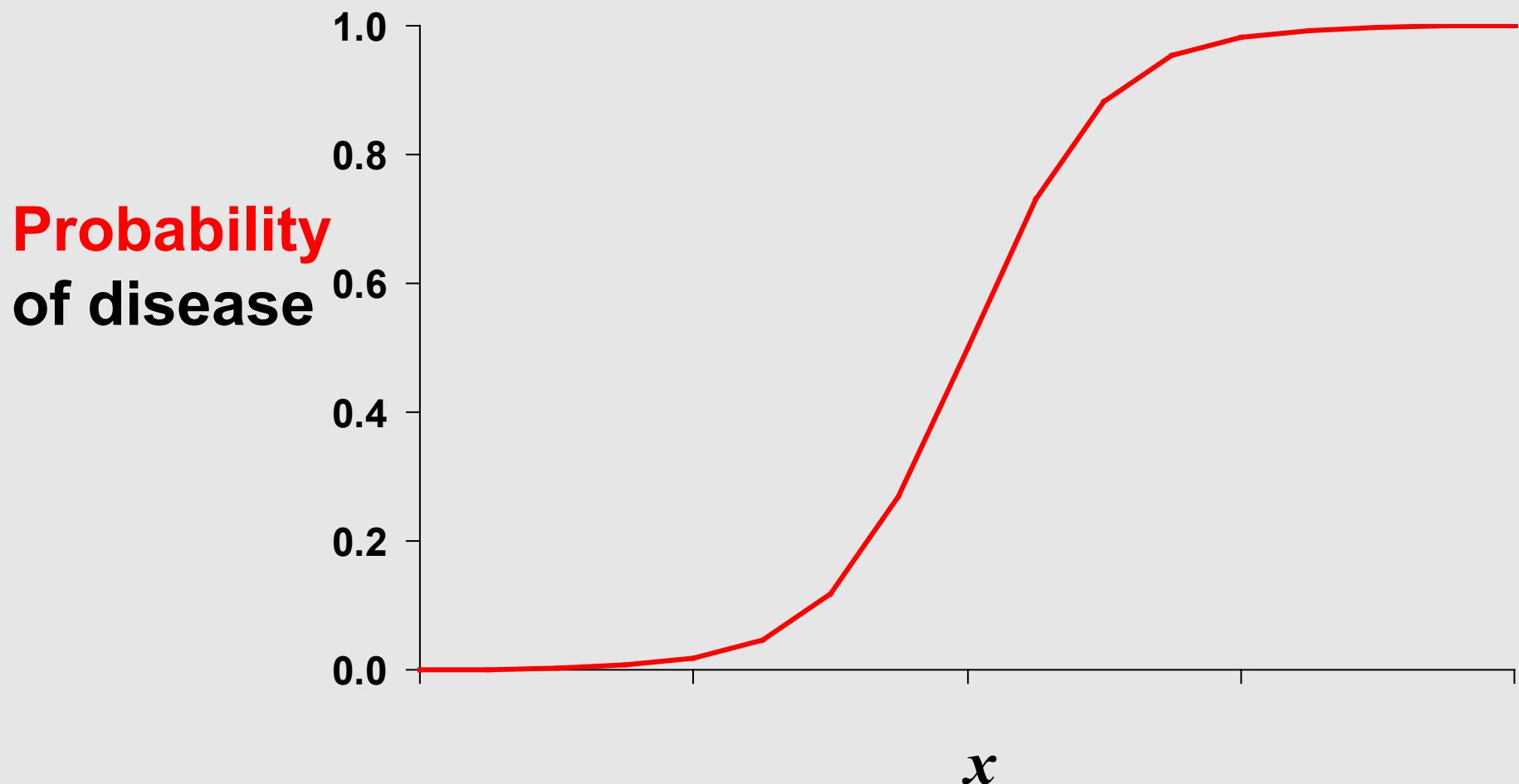
Table 2 Prevalence (%) of signs of CD according to age group

Age group	# in group	Diseased	
		#	%
20 - 29	5	0	0
30 - 39	6	1	17
40 - 49	7	2	29
50 - 59	7	4	57
60 - 69	5	4	80
70 - 79	2	2	100
80 - 89	1	1	100

Dot-plot: Data from Table 2



Logistic Function



Week 6 Contents / Objectives

- Machine Learning Recap
- Motivation for Logistic Regression
- **Logistic Regression**
- Computational Graph
- PyTorch: A Deep Learning Library

Probabilistic Classification

- Training classifiers: estimating $f: X \rightarrow Y$, or $P(Y|X)$
- **Discriminative** classifiers
 - Assume some functional form for $P(Y|X)$
 - Estimate parameters of $P(Y|X)$ directly from training data
- **Generative** classifiers
 - Assume some functional form for $P(X|Y)$, $P(X)$
 - Estimate parameters of $P(X|Y)$, $P(X)$ directly from training data
 - Use Bayes rule to calculate $P(Y|X=x_i)$

Log Odds

- **Odds:** the ratio of π , the probability of a positive outcome $P(y=1/x)$, to $(1 - \pi)$, the probability of a negative outcome $P(y=0/x)$.

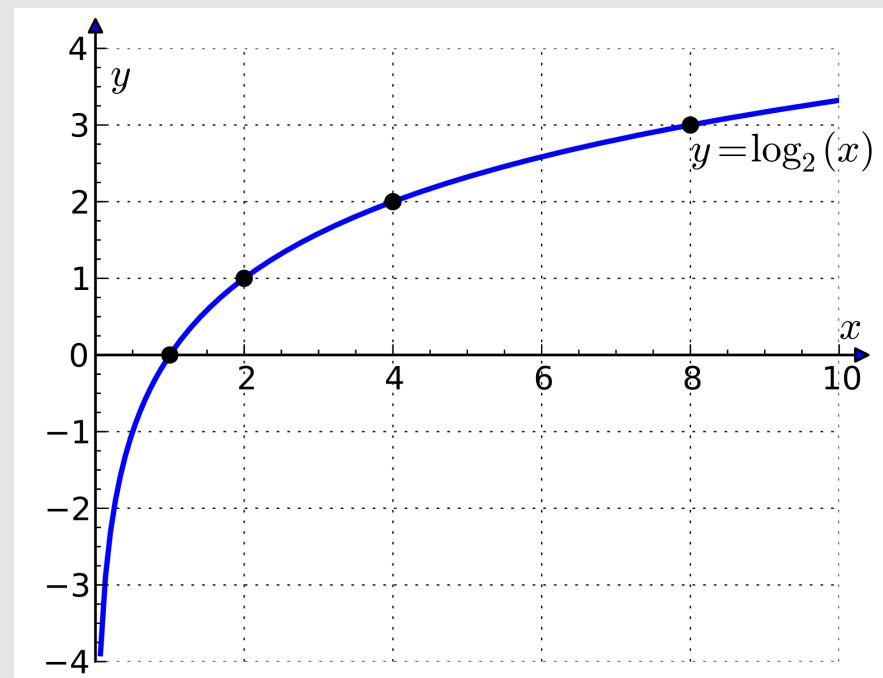
$$\frac{\pi}{1 - \pi}$$

- Probability: $[0, 1]$

- \rightarrow Odds: $[0, \infty]$

- \rightarrow Log odds (**logit**): $[-\infty, \infty]$

$$\text{logit}(\pi) = \log \frac{\pi}{1 - \pi}$$



Logit Function → Logistic Function

- Linear **regression** on **logit** function = logistic *regression*

$$\text{logit}(\pi) = \log \frac{\pi}{1 - \pi} = \mathbf{w}^\top \mathbf{x} = w_0 + w_1 x_1 + \dots$$

- More generally, we can use **basis function** as

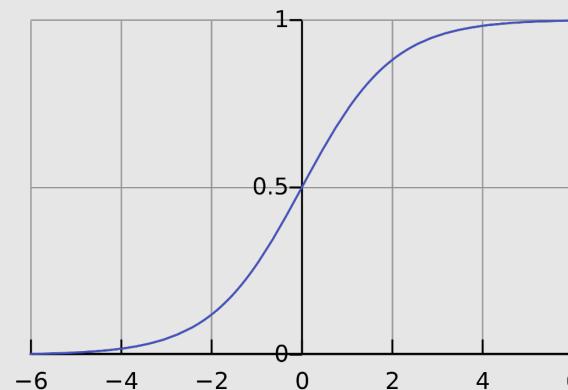
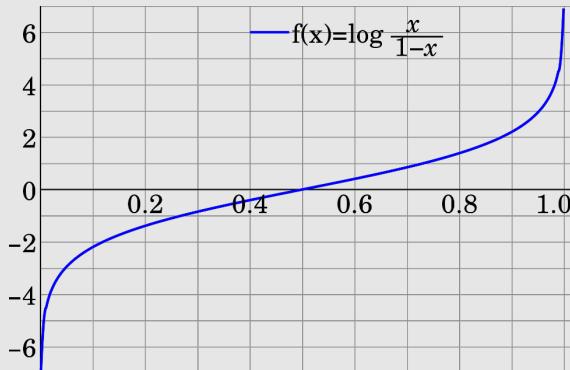
$$\text{logit}(\pi) = \log \frac{\pi}{1 - \pi} = \mathbf{w}^\top \phi(\mathbf{x}) = w_0 + w_1 \phi(x_1) + \dots$$

In the following, we use the simpler first form above

- Logistic function (**sigmoid**)= inverse of logit

$$P(y = 1 | \mathbf{x}) = \text{logit}^{-1}(\mathbf{w}^\top \mathbf{x}) = \text{logistic}(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

- **Exercise:** verify the odds using the above equation



How to Estimate w? (Learning algo)

- Assumption: Conditional independence of data

- → Likelihood:

$$P(\mathbf{y}|\mathbf{X}) = \prod_{i=1}^n P(y_i|\mathbf{x}_i)$$

- Bernoulli distribution for binary classification

- $P(y=1) = \pi ; P(y=0) = 1 - \pi$ (coin flipping)

- Write the above as a single equation: using y as a switch

$$P(y) = \pi^y (1 - \pi)^{(1-y)} \quad \pi_i = P(y_i = 1|\mathbf{x}_i)$$

- Log likelihood (negative log likelihood → cross entropy)

$$\log P(\mathbf{y}|\mathbf{X}) = \sum_{i=1}^n \log P(y_i|\mathbf{x}_i) = \sum_{i=1}^n y_i \log \pi_i + \sum_{i=1}^n (1 - y_i) \log(1 - \pi_i)$$

- MLE: no closed form solution

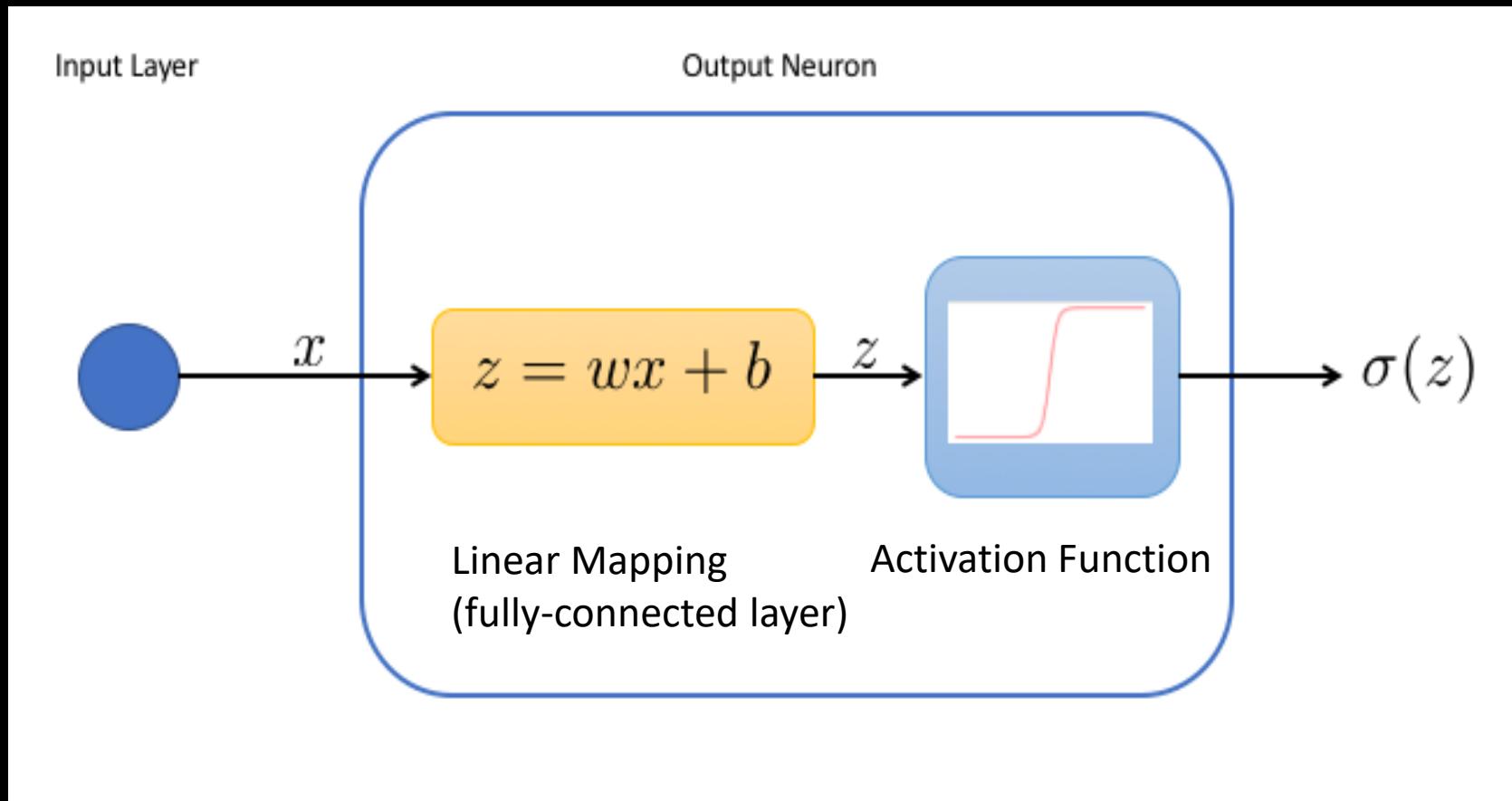
→ SGD on negative log likelihood (minimisation)

Machine Learning Ingredients

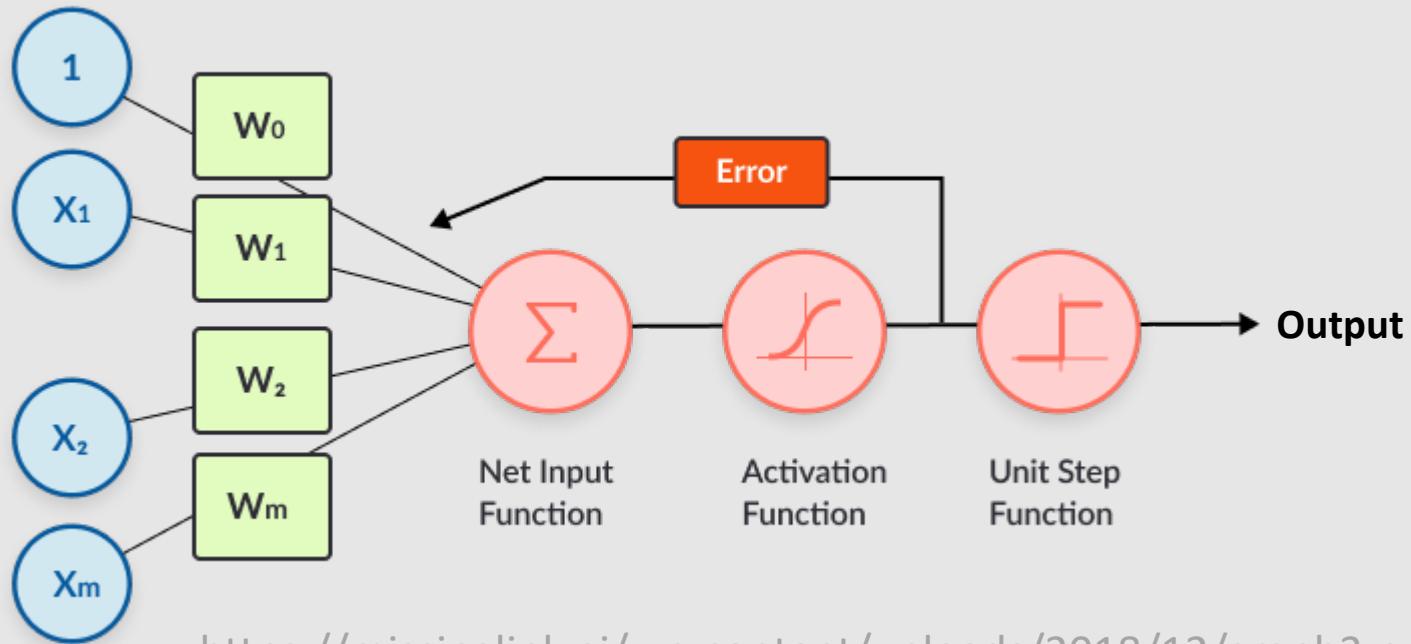
- **Data:** + pre-processing (& visualisation), e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure ~ Architecture \leftarrow expert knowledge
 - Must **specify** before ML, can optimise via cross validation (CV)
 - **Hyper-parameter**, e.g., prior, #degree, layer \leftarrow knowledge
 - Must **specify** (choices) and can optimise via CV (**tuning**)
 - Parameters (theta)
 - Compute/learn parameter, e.g., **weights**, bias \leftarrow optimisation alg.
- Evaluation metric (what's best): loss/error function
- Optimisation: (how to find the best) learnable parameters

Logistic Regression Ingredients

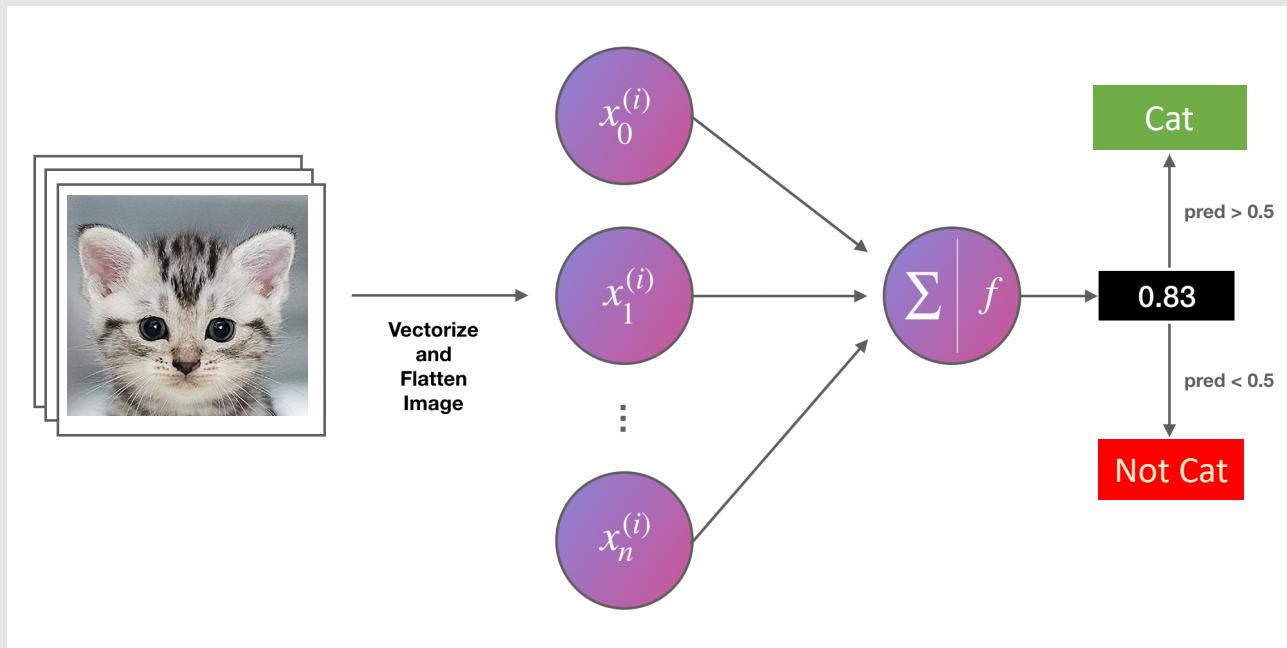
- **Data:** + pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure/Architecture: linear relationship
$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$
 - **Hyper-parameter:** no (unless + regularisation)
 - Parameters (theta): weights and bias
- Evaluation metric: max likelihood (min NLL)
- Optimisation: SGD or the like



Logistic Regression – The Simplest Neural Network



<https://missinglink.ai/wp-content/uploads/2018/12/graph3.png>



https://miro.medium.com/max/4112/1*5NV4NEtgR4rDpSVZkWh3oA.png

Multiclass Classification

- A simple way: one-vs-rest logistic regression
 - Run binary classification for all possible classes
 - Pick the one with the highest value
- More mathematical: multinomial logistic regression, also known as **softmax**
 - Generalise logistic regression to multiple classes
 - Binomial \rightarrow multinomial distribution
 - Sigmoid function \rightarrow softmax function
 - A linear classifier for multiple classes

Summary on Logistic Regression (LR)

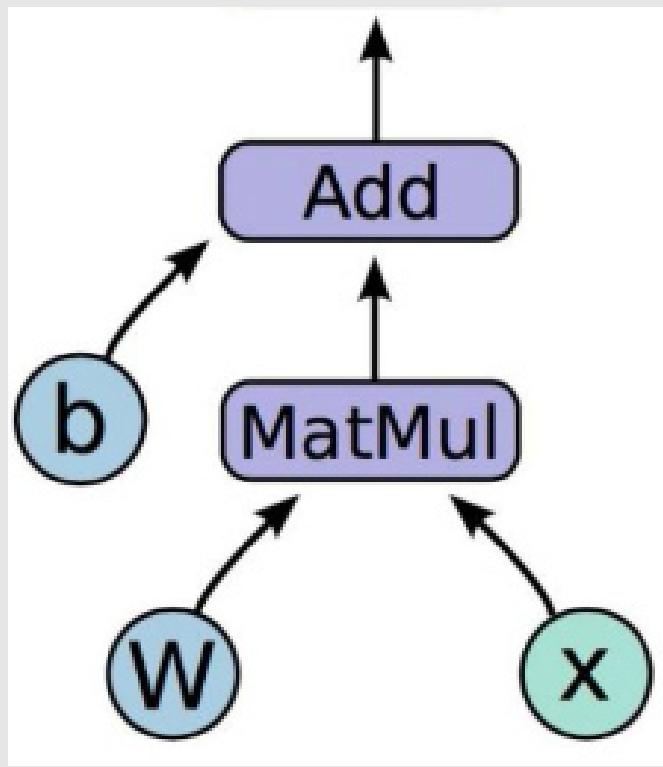
- **Discriminative** classifiers directly model the likelihood $P(Y|X)$
- A simple **linear** classifier that retains a **probabilistic** semantics (see lab)
- Parameters in LR are learned by **iterative** optimization (e.g. SGD), no closed-form solution
- The simplest neural network

Week 6 Contents / Objectives

- Machine Learning Recap
- Motivation for Logistic Regression
- Logistic Regression
- **Computational Graph**
- PyTorch: A Deep Learning Library

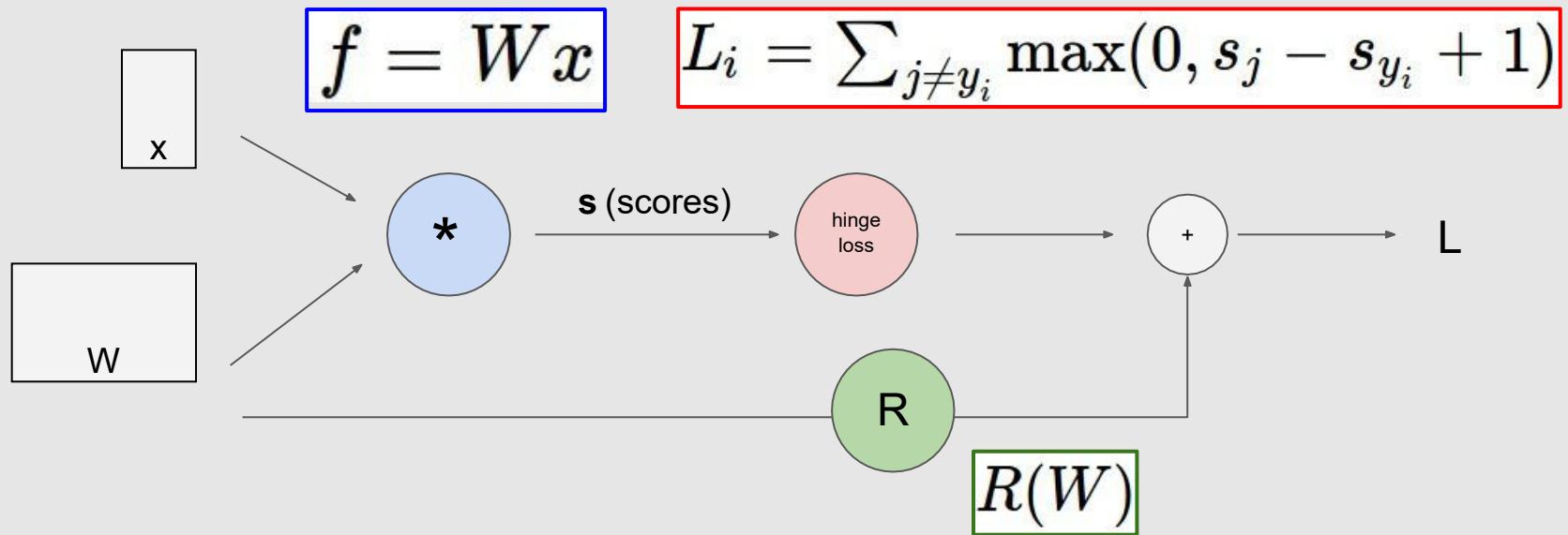
Computational Graph

- Linear regression $y = Wx + b$

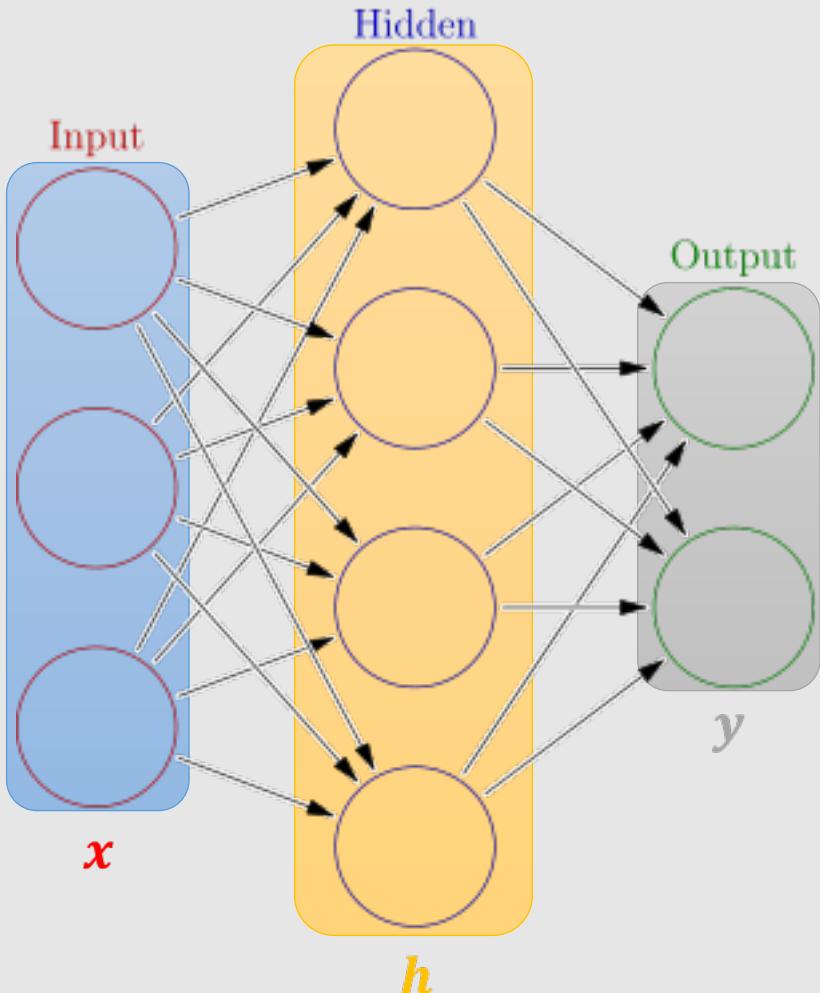


Source: Nelson Liu: <https://colab.research.google.com/drive/11iLtGFDpnIuHj5B0rQDGG5lqq6BQ8FRh>

Computational Graph: w/t Reg.



Multilayer Perceptron (NN) vs LR



Weights

$$h = \sigma(W_1 x + b_1)$$
$$y = \sigma(W_2 h + b_2)$$

Activation functions



Question:
How many model parameters?

$$[3 \times 4] + [4 \times 2] = 20 \text{ weights}$$

$$4 + 2 = 6 \text{ biases}$$

26 learnable parameters

$4 + 2 = 6$ neurons (not counting inputs)

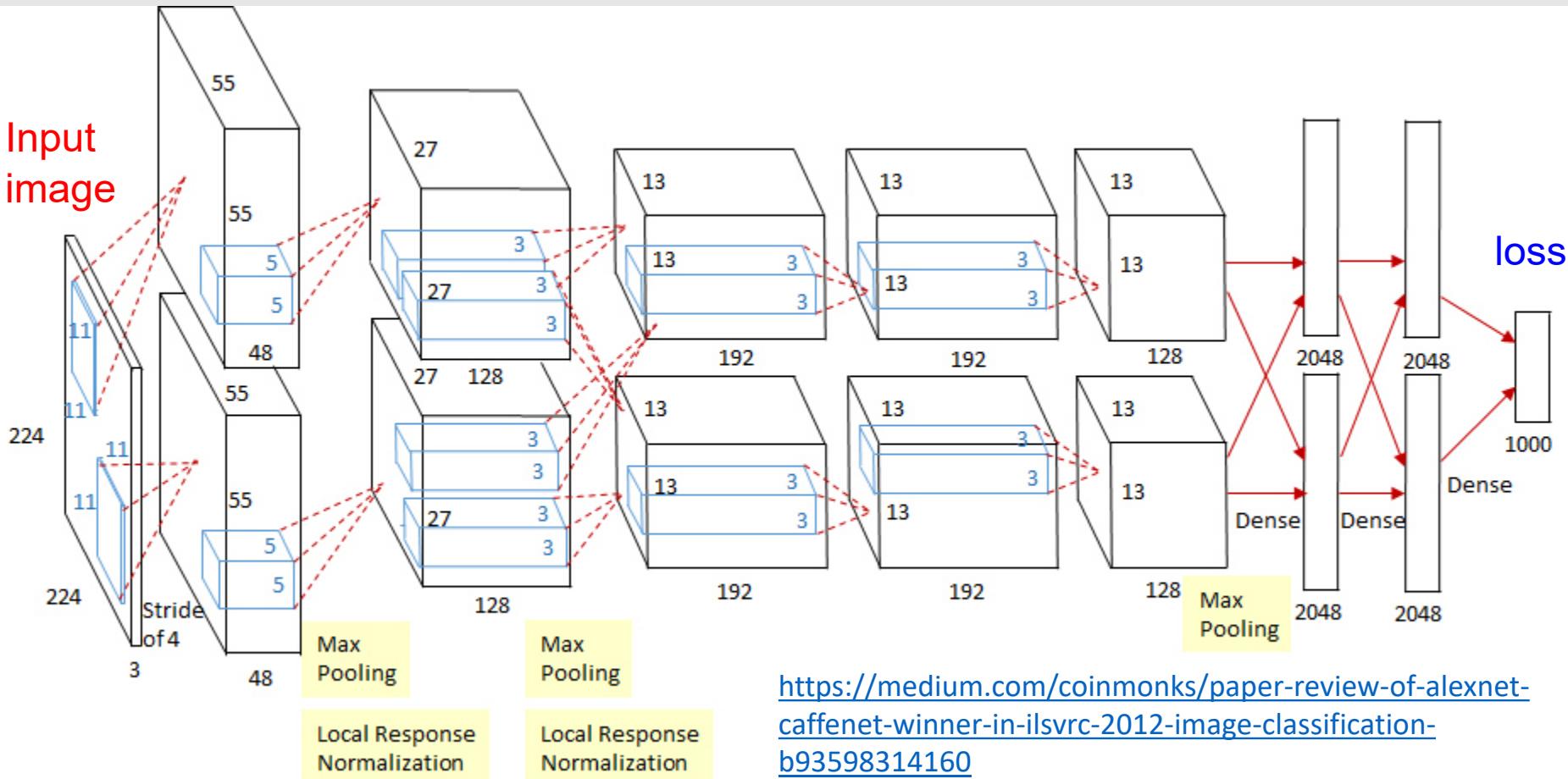
THAT'S NOT ENOUGH



WE HAVE TO GO DEEPER

Computational Graph: DL

Weights (60 million parameters)



ImageNet I

Dataset: 1.2 million
CNN for Image Class
Hinton, 2011) → 17.5% error

Fancy feature extraction

/representation learning

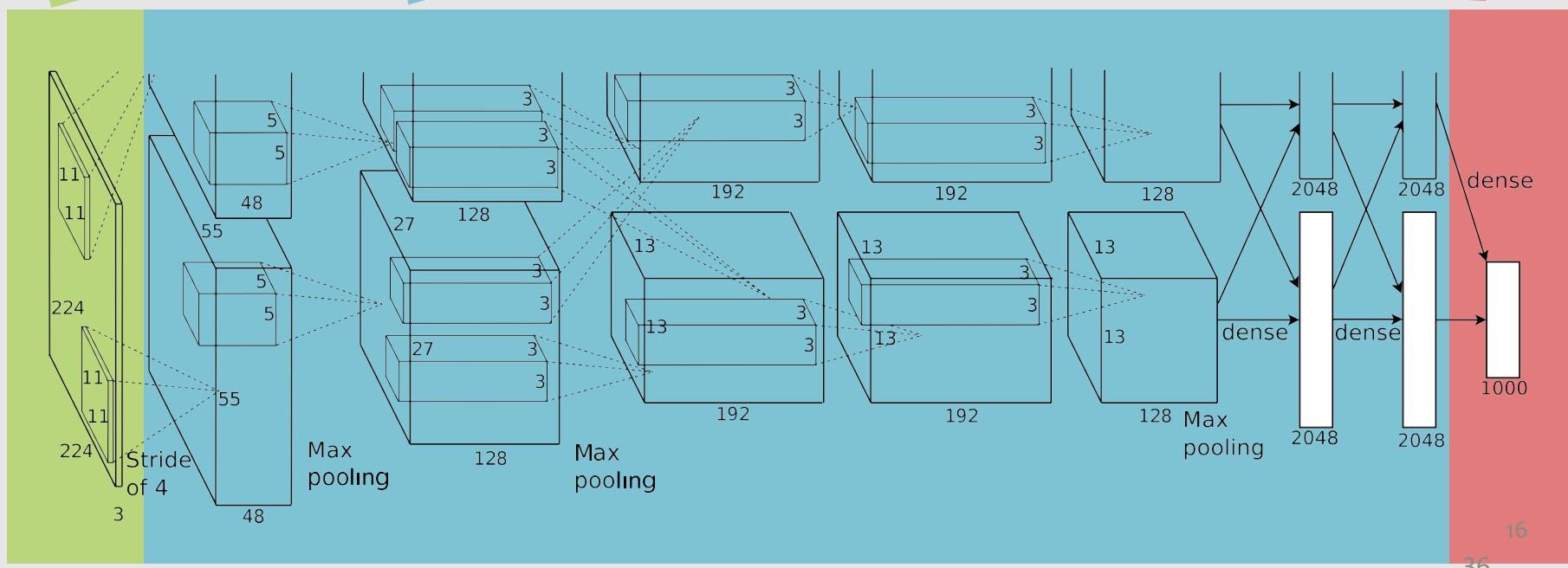
Logistic Regression!

Softmax: sigmoid
for multiclass

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



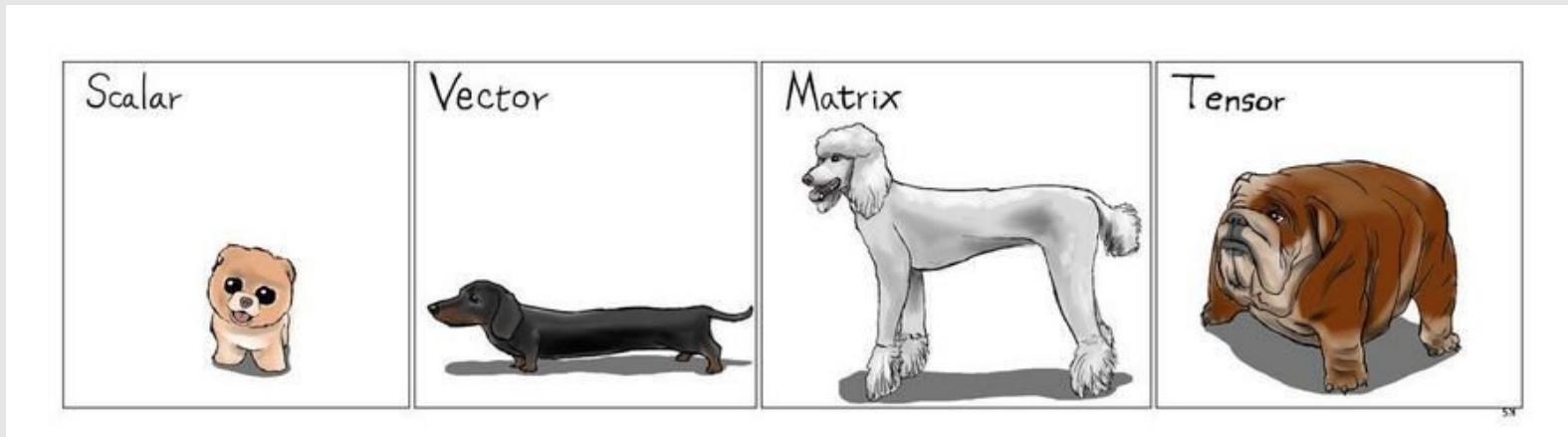
Week 6 Contents / Objectives

- Machine Learning Recap
- Motivation for Logistic Regression
- Logistic Regression
- Computational Graph
- **PyTorch: A Deep Learning Library**

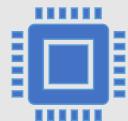


PyTorch

- An open source deep learning library by Facebook
 - **Tensor** computing with GPU acceleration
 - Deep neural networks built on autodiff
- **torch.Tensor**
 - multidimensional data structures/arrays for programming
 - Scalar: 0-D tensor; Vector: 1-D tensor; Matrix: 2-D tensor



<https://pbs.twimg.com/media/D2xTDMJWwAADHQt.jpg>



Key Modules in PyTorch

- **torch.autograd**
 - Automatic differentiation. A recorder records what operations have performed, and then it replays it backward to compute the gradients.
- **torch.optim**
 - Implementation of various optimization algorithms used for building neural networks (and other ML algorithms).
- **torch.nn**
 - High-level definition of the **computational graphs (architecture)** of complex neural networks (and other ML algorithms)

Dynamic Computational Graph

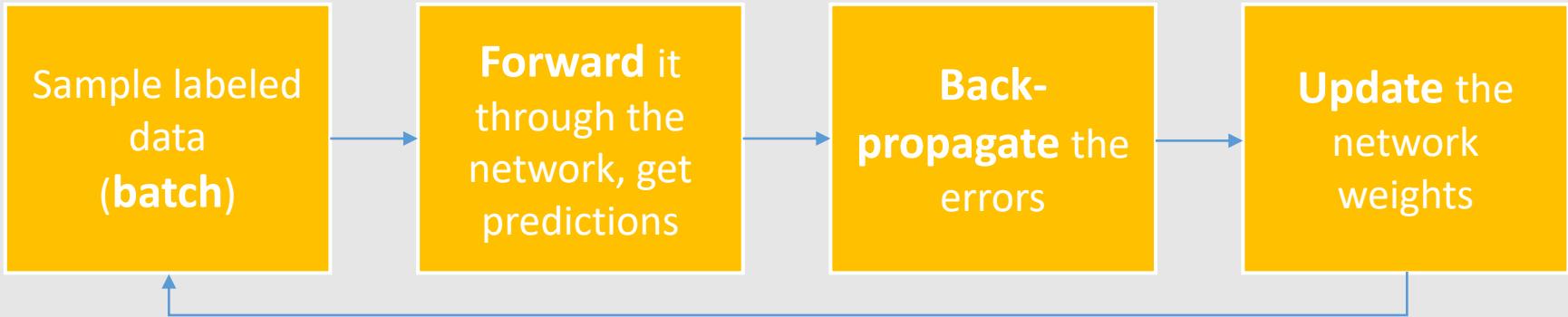
A graph is created on the fly



```
W_h = torch.randn(20, 20, requires_grad=True)
W_x = torch.randn(20, 10, requires_grad=True)
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
```

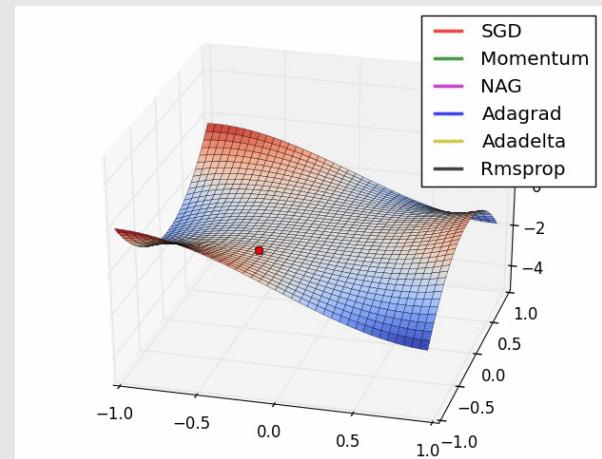
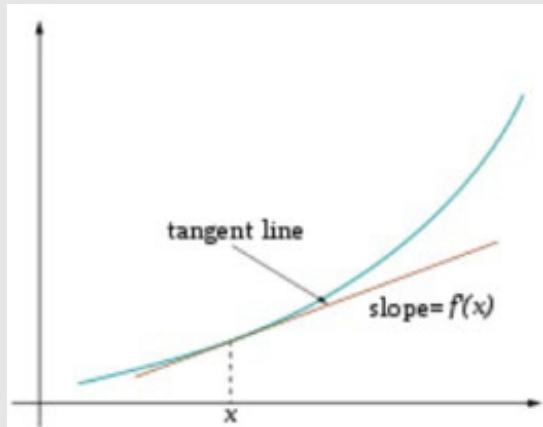


Training



Optimize (min. or max.) **objective/cost function $J(\theta)$**

Generate **error signal** that measures difference between predictions and target values



Use error signal to change the **weights** and get more accurate predictions
Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**

<https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descent-optimization-algorithm-4106a6702d39>

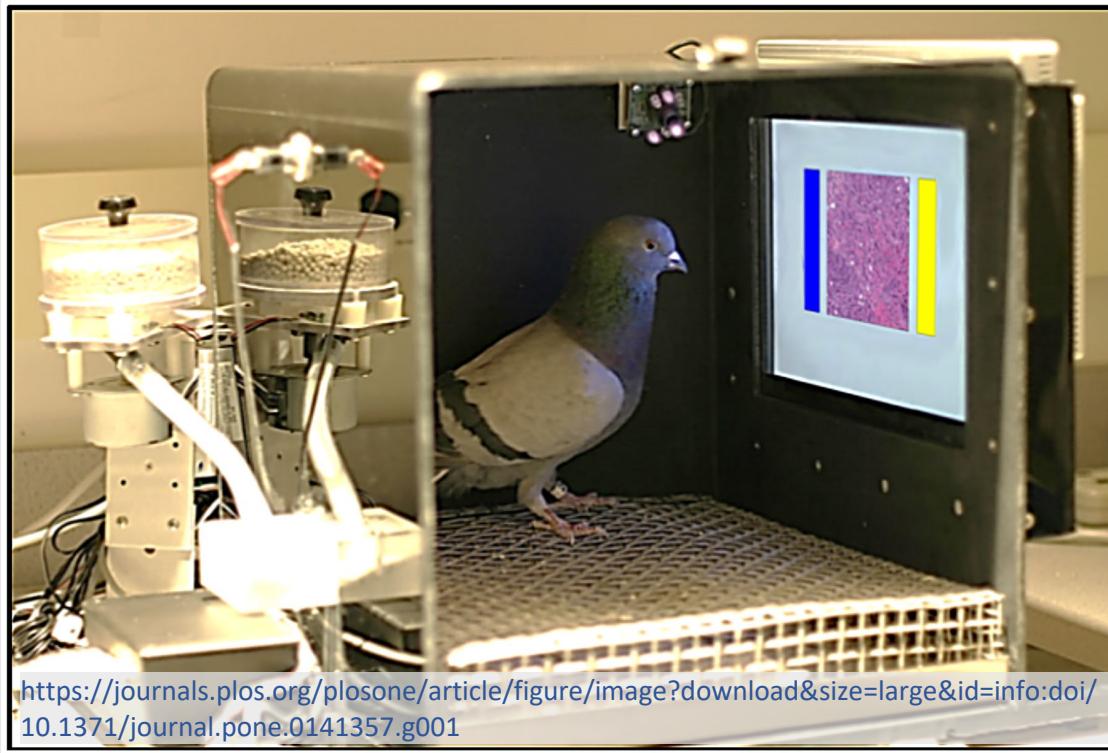
Acknowledgement

- The slides used materials from:
Colin Bernet, Ismini Lourentzou, Fei-Fei Li & Justin Johnson & Serena Yeung, Rui Zhang, Nelson Liu, Matt Gormley, Rachid Salmi, Jean-Claude Desenclos, Thomas Grein, Alain Moren, Christophe Giraud-Carrier, Bart Selman, Sham Kakade, Raymond J. Mooney, Neil Lawrence, and Andrew Ng

Recommended Reading

- [Notes Logistic Regression by Andrew Ng](#)
- Wikipedia entries on topics, e.g. multiclass classification, softmax, multinomial logistic regression,
- PyTorch documentations
- The lab notebook and references

Lecture 7: Neural Networks



<https://journals.plos.org/plosone/article/figure/image?download&size=large&id=info:doi/10.1371/journal.pone.0141357.g001>

Haiping Lu

YouTube Playlist: <https://www.youtube.com/c/HaipingLu/playlists>

COM4059/6059: MLAI21@The University of Sheffield

Week 7 Contents / Objectives

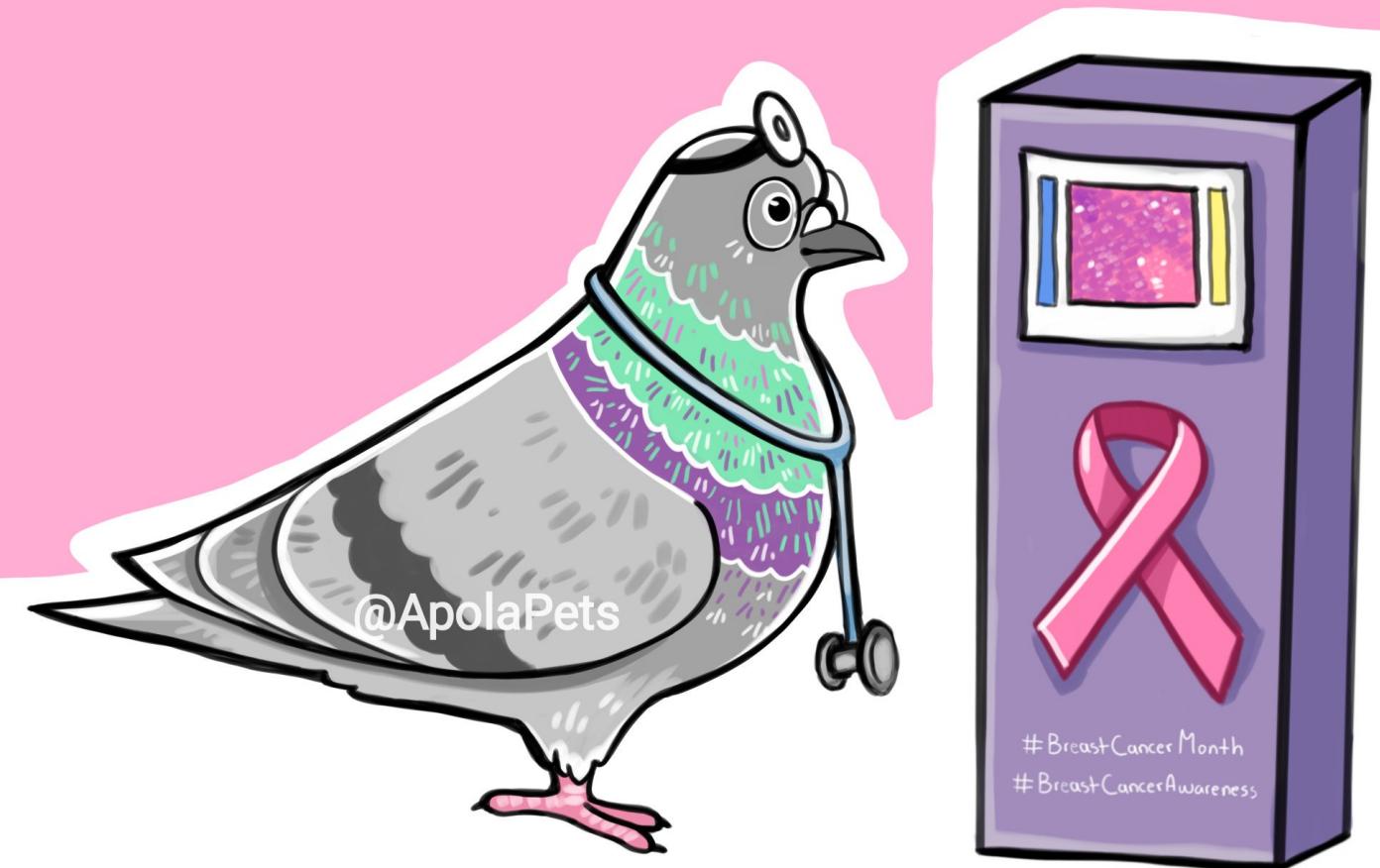
- Learning with Neurons
- Neural Networks (NNs)
- NN Decision Boundary & Features
- Convolutional NN Basics
- Convolutional NN Unboxing

Week 7 Contents / Objectives

- **Learning with Neurons**
- Neural Networks (NNs)
- NN Decision Boundary & Features
- Convolutional NN Basics
- Convolutional NN Unboxing

Did you know ?

Pigeons identify Breast Cancer



Just as well as radiologists

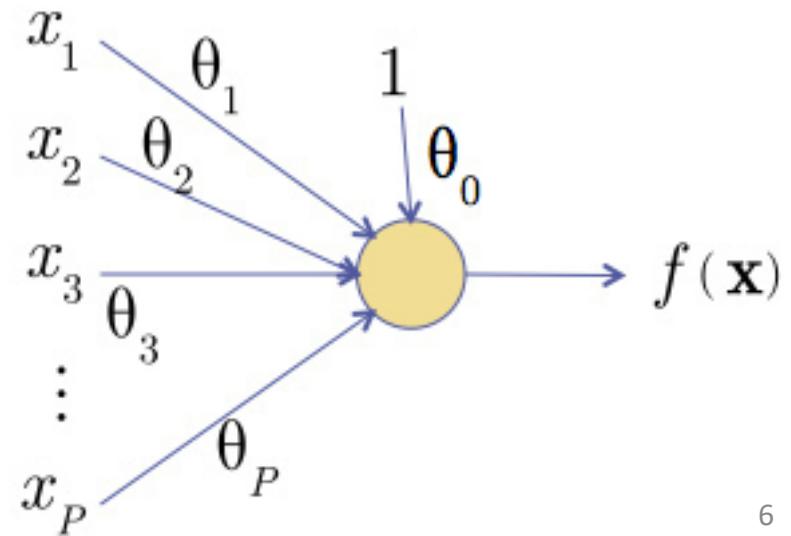
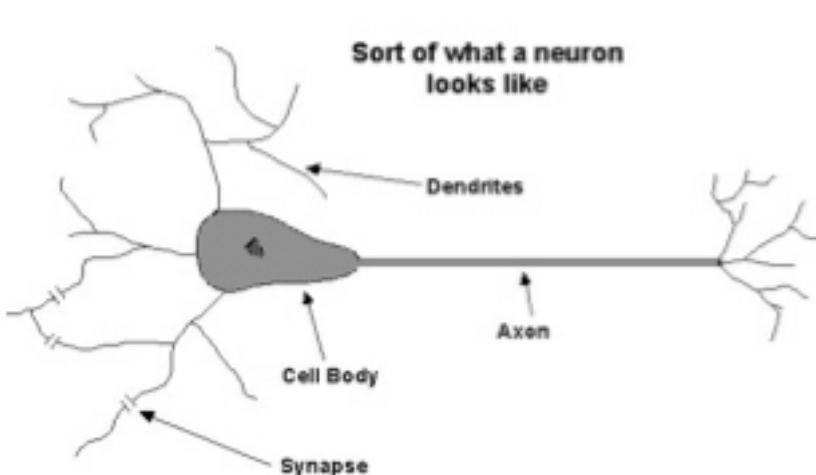
<https://pbs.twimg.com/media/DqczsWhX4AErP0h.jpg:large>



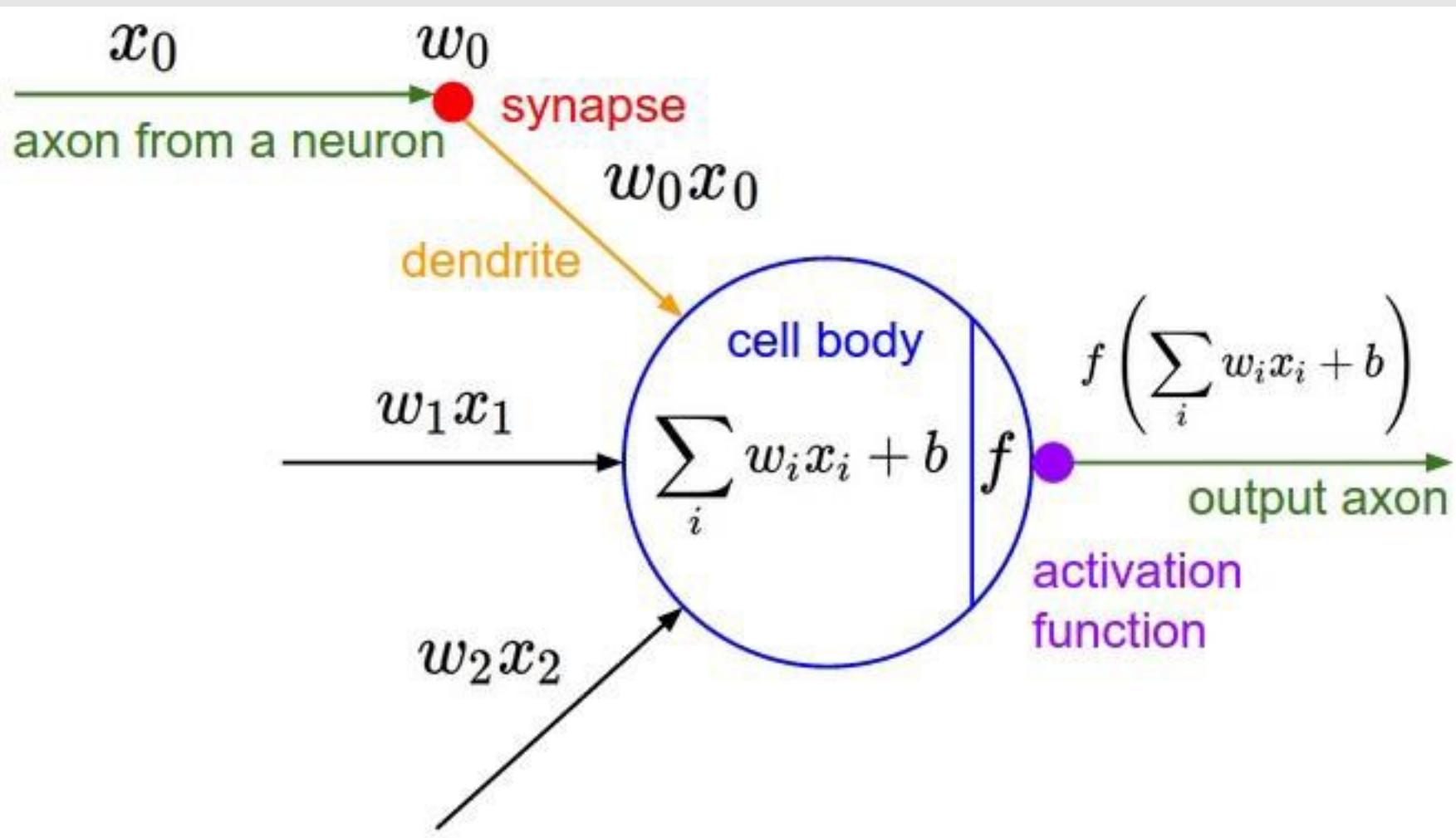
**Three correct trials with benign images follow
(the yellow button is correct).**

The Neuron Metaphor

- Neurons
 - Accept information from multiple inputs
 - Transmit information to other neurons
- Multiply inputs by weights along edges
- Apply some function to the inputs at each node

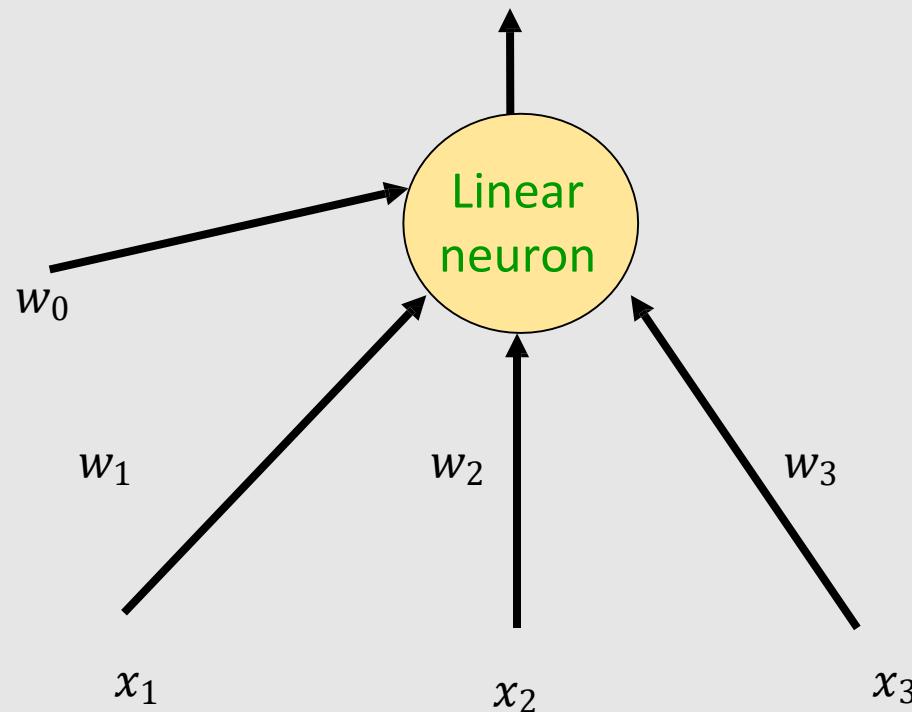


A Neuron Analogous to the Brain



Neural Network

- Network of neurons
- Linear neuron $w_0 + w_1x_1 + w_2x_2 + w_3x_3$



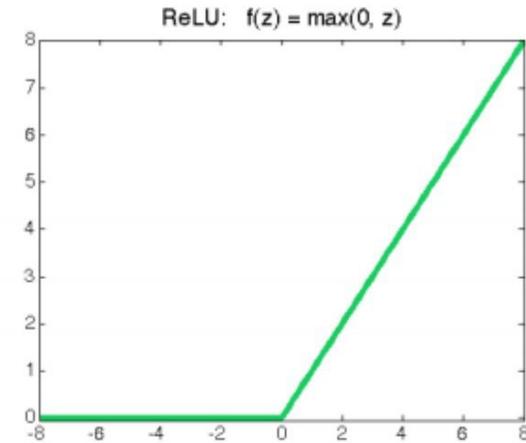
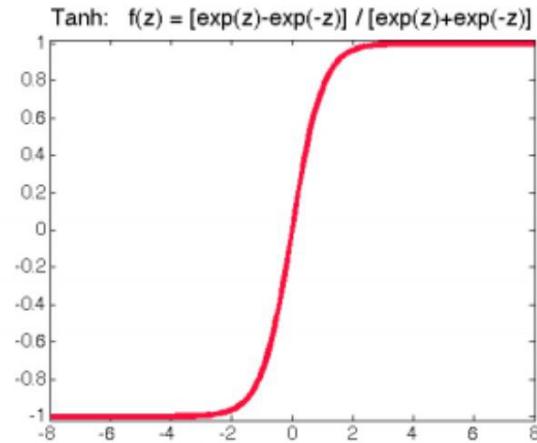
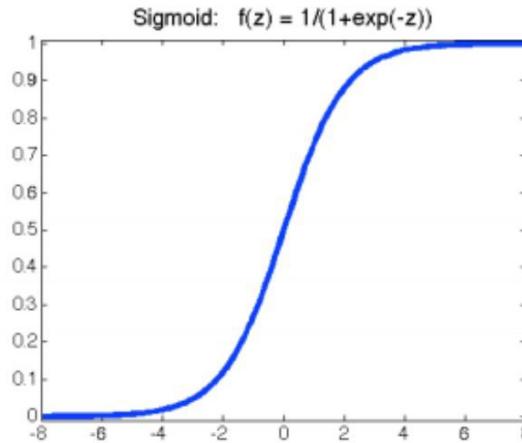
Activation Functions

- Commonly used activation functions

- Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$

- Tanh: $\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$

- ReLU (Rectified Linear Unit): $\text{ReLU}(z) = \max(0, z)$



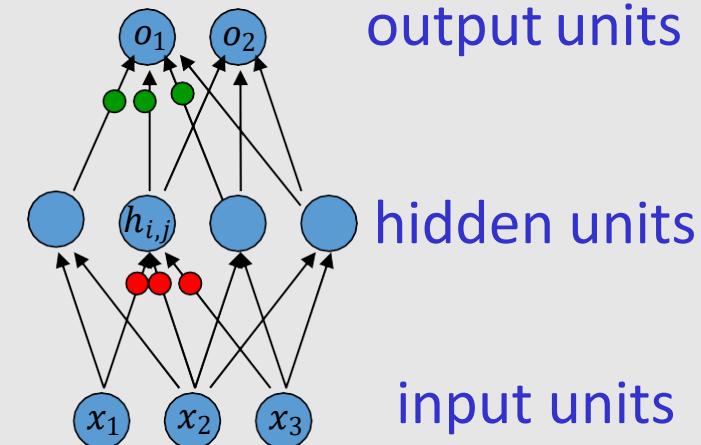
Computation in Neural Networks

- Forward pass
 - Making **predictions (decisions)**
 - Plug in the input x , get the output y

$$\mathbf{o} = g \left((W^{(2)})^T \mathbf{h} + b^{(2)} \right)$$

$$\mathbf{h} = g \left((W^{(1)})^T \mathbf{x} + b^{(1)} \right)$$

- Backward pass (backpropagation for optimisation)
 - Compute the gradient of the **cost (loss/error)** function with respect to the weights to find good values for weights



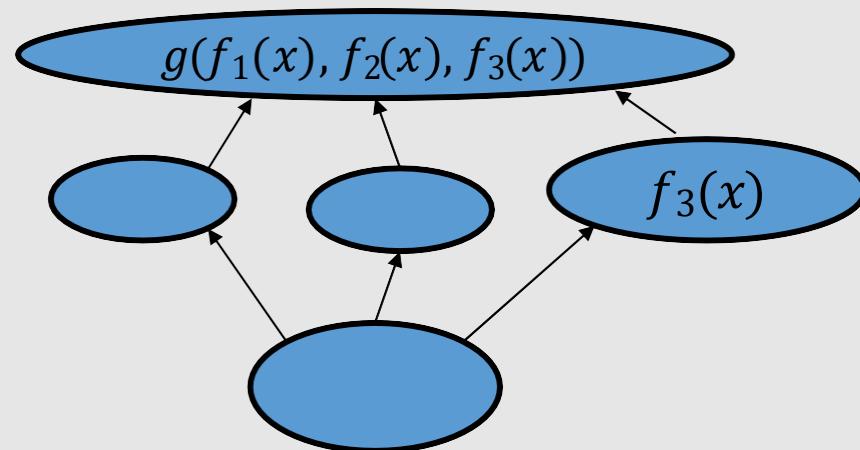
Autograd: Chain Rule

- Univariate chain rule

$$\frac{d}{dt} g(f(t)) = \frac{dg}{df} \cdot \frac{df}{dt}$$

- Multivariate chain rule

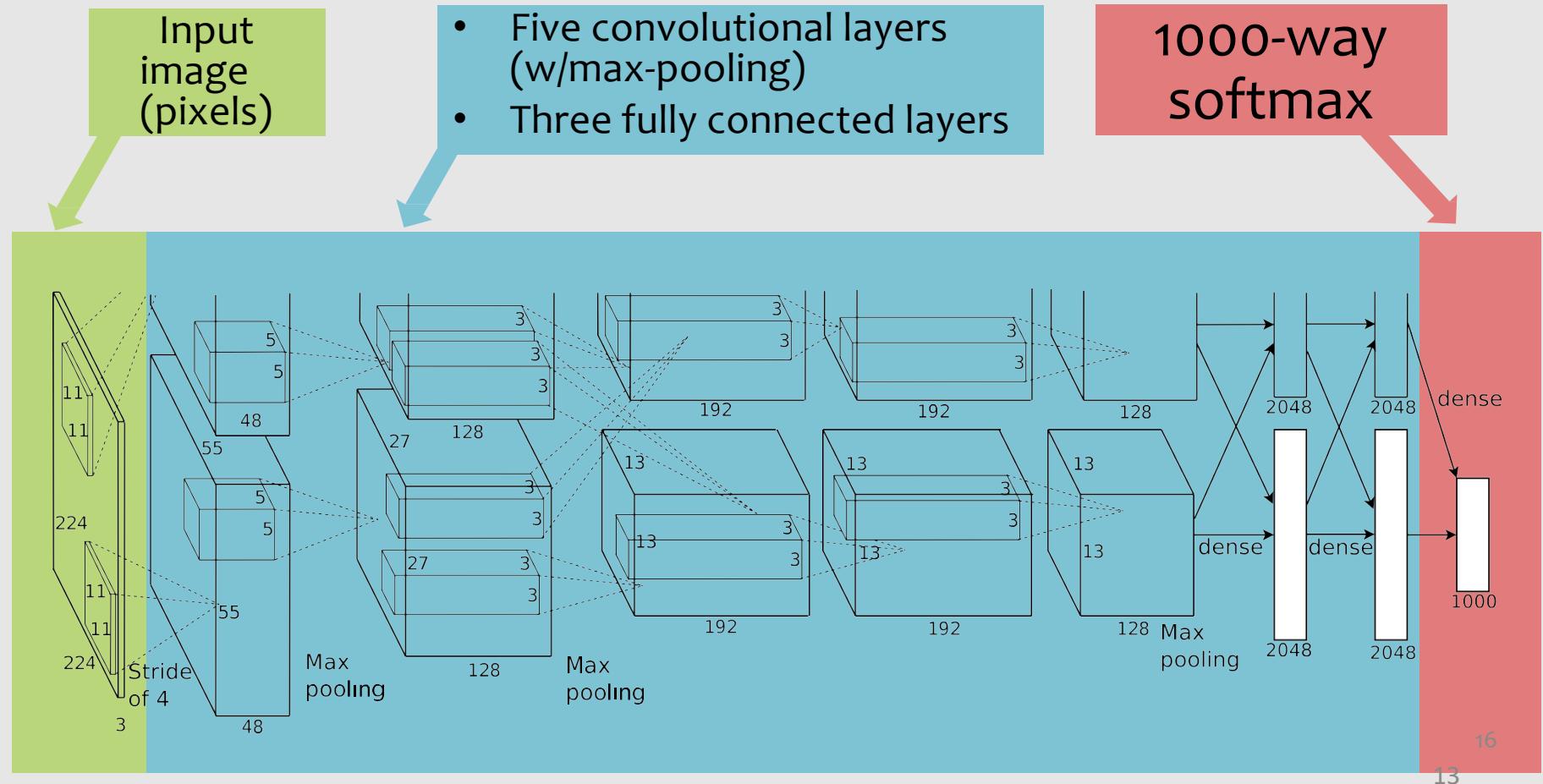
$$\frac{\partial g}{\partial x} = \sum \frac{\partial g}{\partial f_i} \frac{\partial f_i}{\partial x}$$



Week 7 Contents / Objectives

- Learning with Neurons
- **Neural Networks (NNs)**
- NN Decision Boundary & Features
- Convolutional NN Basics
- Convolutional NN Unboxing

AlexNet for ImageNet LSVRC-2010



Data, Model, Metric for Learning

1. Given training data:

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

2. Choose each of these:

- Decision function/model

$$\hat{\mathbf{y}} = f_{\theta}(\mathbf{x}_i)$$

- Loss function/metric

$$\ell(\hat{\mathbf{y}}, \mathbf{y}_i) \in \mathbb{R}$$

Face



Face



Not a face



Examples: Linear regression,
Logistic regression, Neural
Network

Examples: Mean-squared
error, Cross Entropy

Data, Model, Metric, Optimisation

1. Given training data:

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

2. Choose each of these:

- Decision function/model

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

- Loss function/metric

$$\ell(\hat{\mathbf{y}}, \mathbf{y}_i) \in \mathbb{R}$$

3. Define goal:

- Objective function

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i)$$

4. Train/optimize with SGD: (take small steps opposite the gradient)

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

Data, Model, Metric, Optimisation

1. Given training data:

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

2. Choose each of these:

– Decision function/model

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

– Loss function/metric

$$\ell(\hat{\mathbf{y}}, \mathbf{y}_i) \in \mathbb{R}$$

3. Define goal:

– Objective function

Compute **gradients**
via backpropagation
Using automatic
differentiation

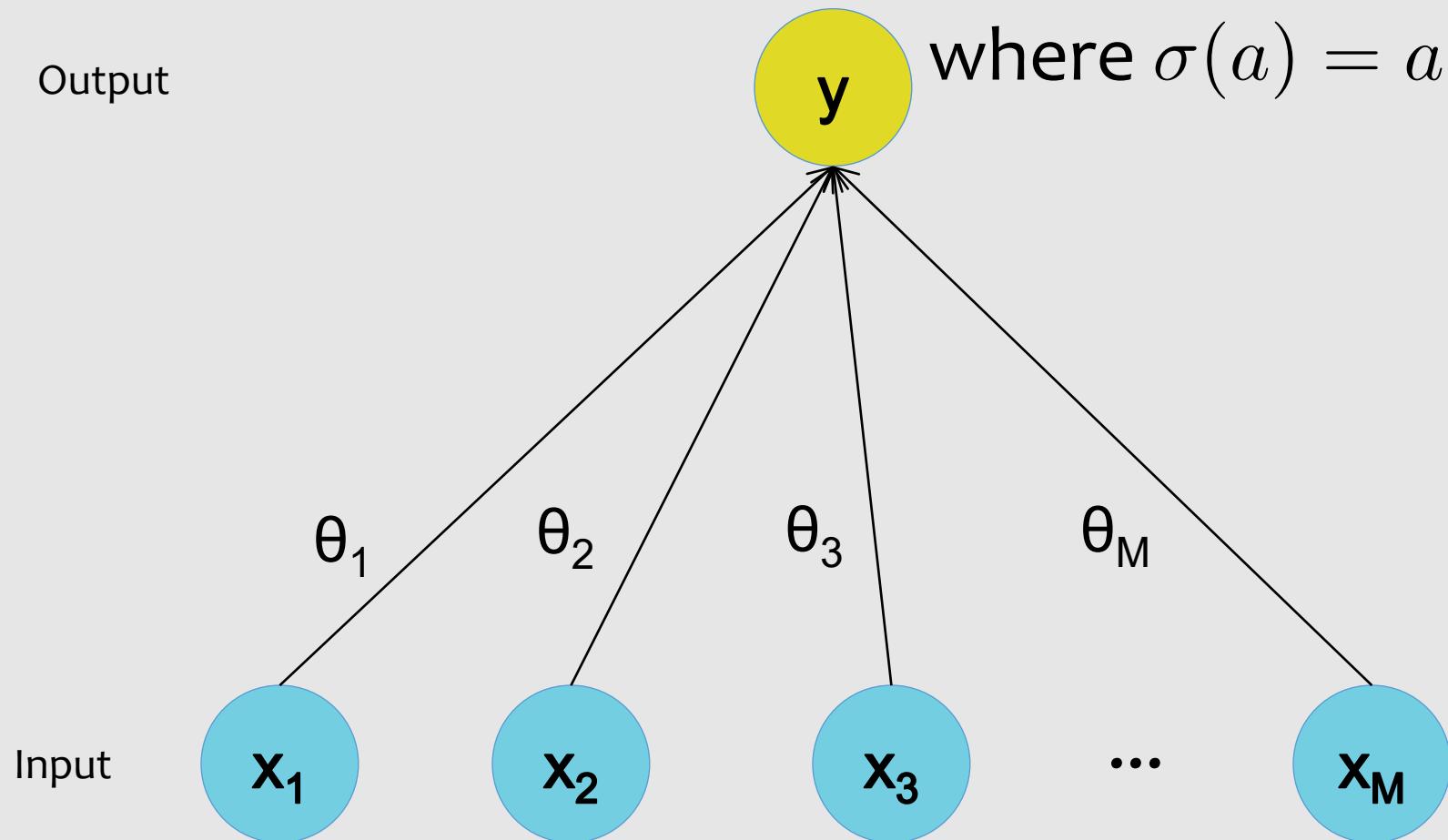
($f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i$)
opposite the gradient)

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

Linear Regression Model ($x \rightarrow y$)

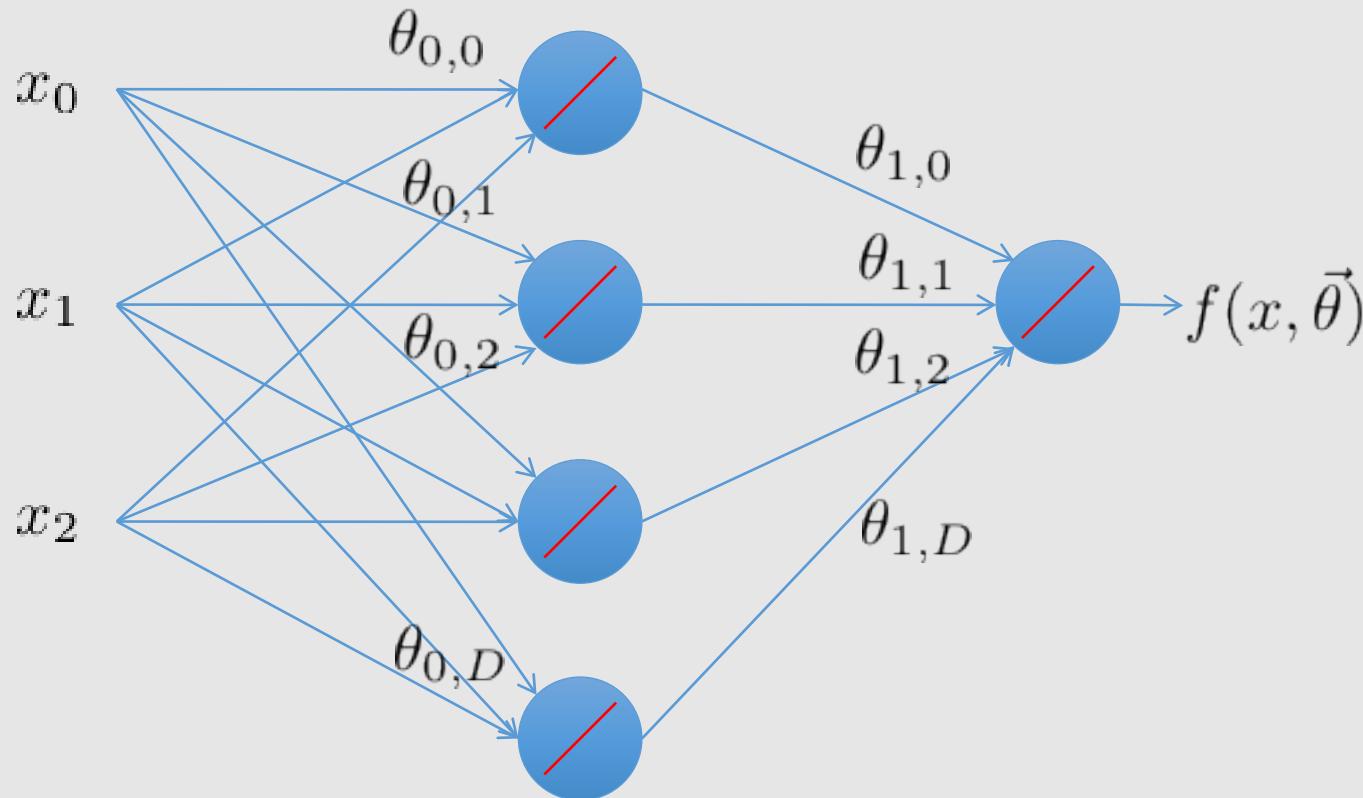
$$y = h_{\theta}(x) = \sigma(\theta^T x)$$

Output



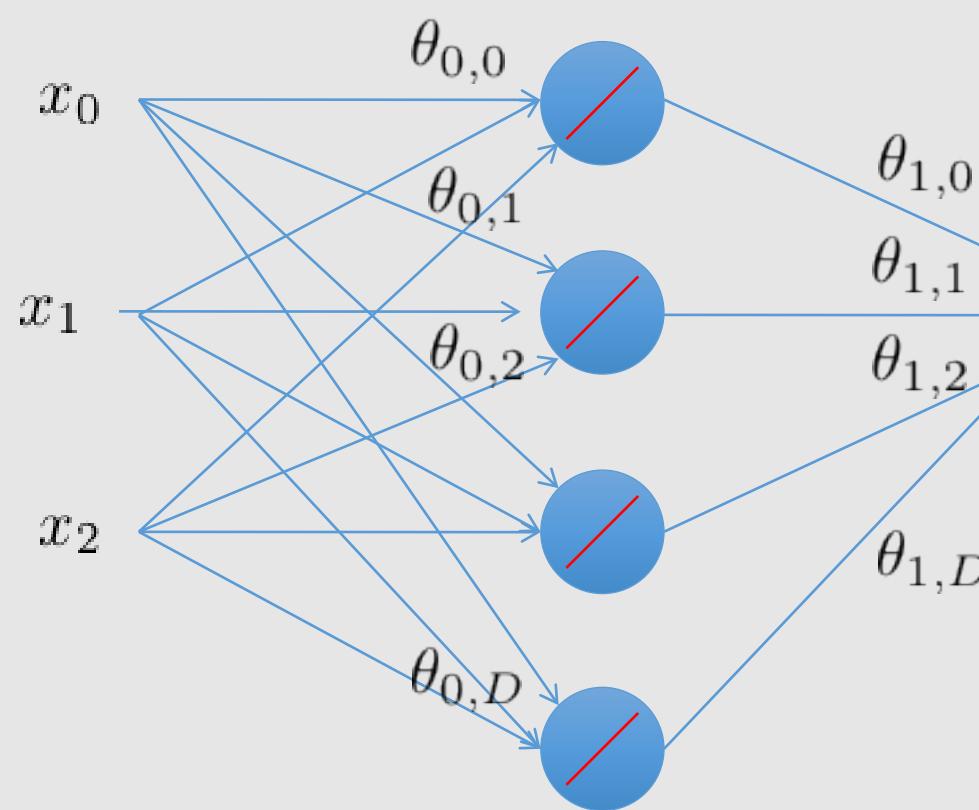
Linear Regression Neural Networks

- **Question:** What happens when we arrange **linear neurons** in a multilayer network?



Linear Regression Neural Networks

- Nothing special happens.
 - The product of two linear transformations is itself a linear transformation.



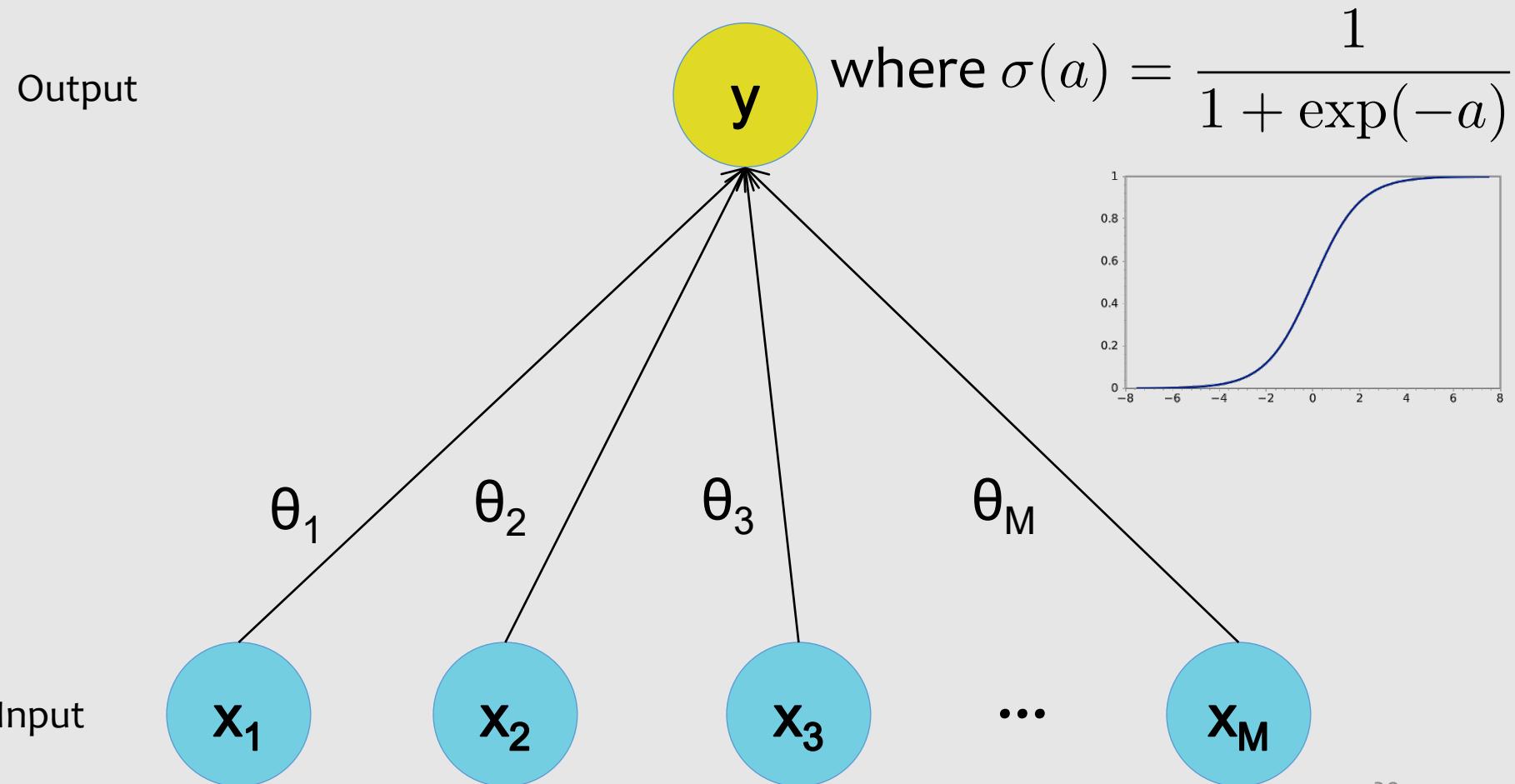
$$f(x, \vec{\theta}) = \sum_{i=0}^D \theta_{1,i} \sum_{n=0}^{N-1} \theta_{0,i,n} x_n$$

$$f(x, \vec{\theta}) = \sum_{i=0}^D \theta_{1,i} [\theta_{0,i}^T \vec{x}]$$

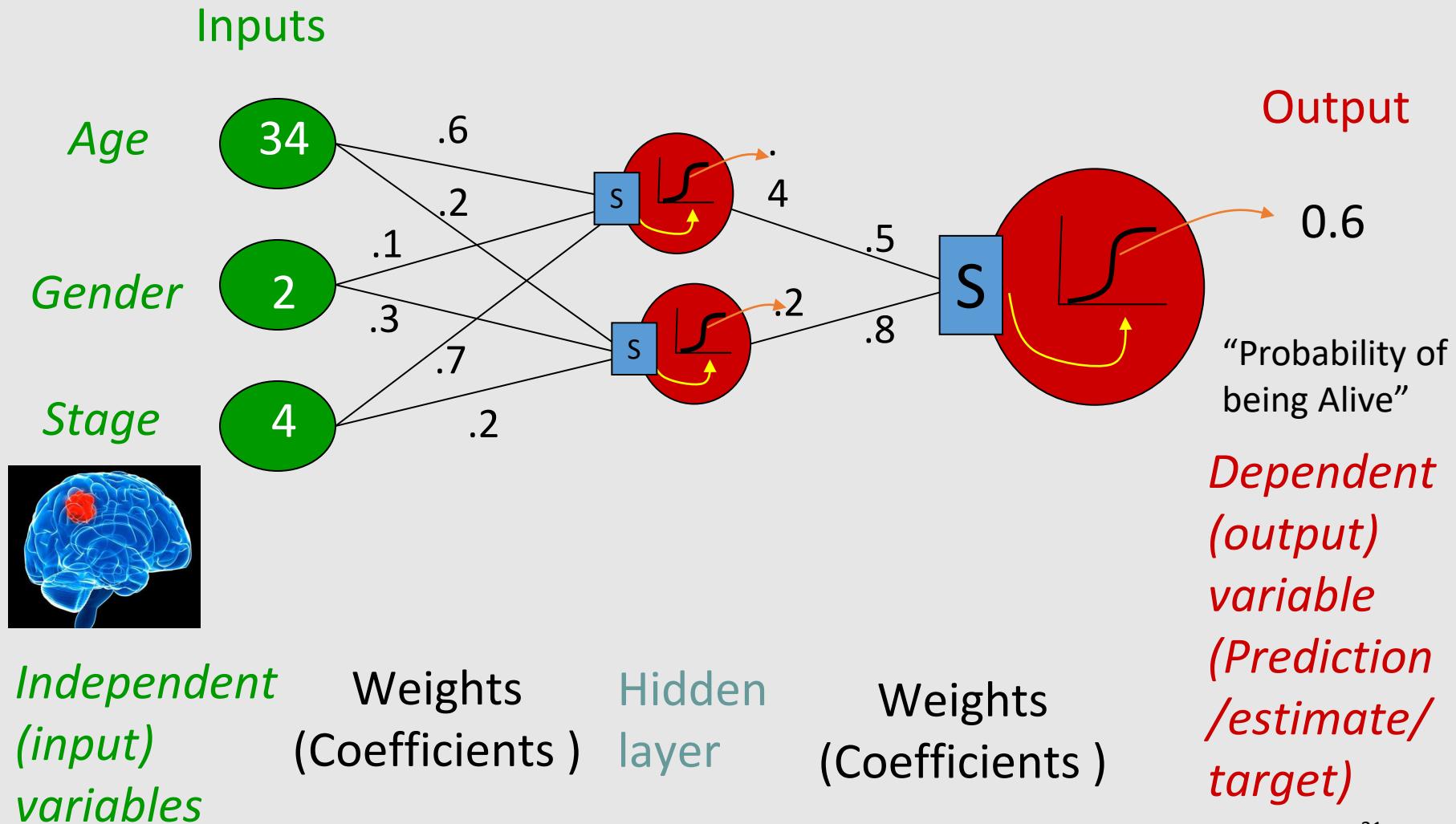
$$f(x, \vec{\theta}) = \sum_{i=0}^D [\hat{\theta}_i^T \vec{x}]$$

Logistic Regression Model ($x \rightarrow y$)

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



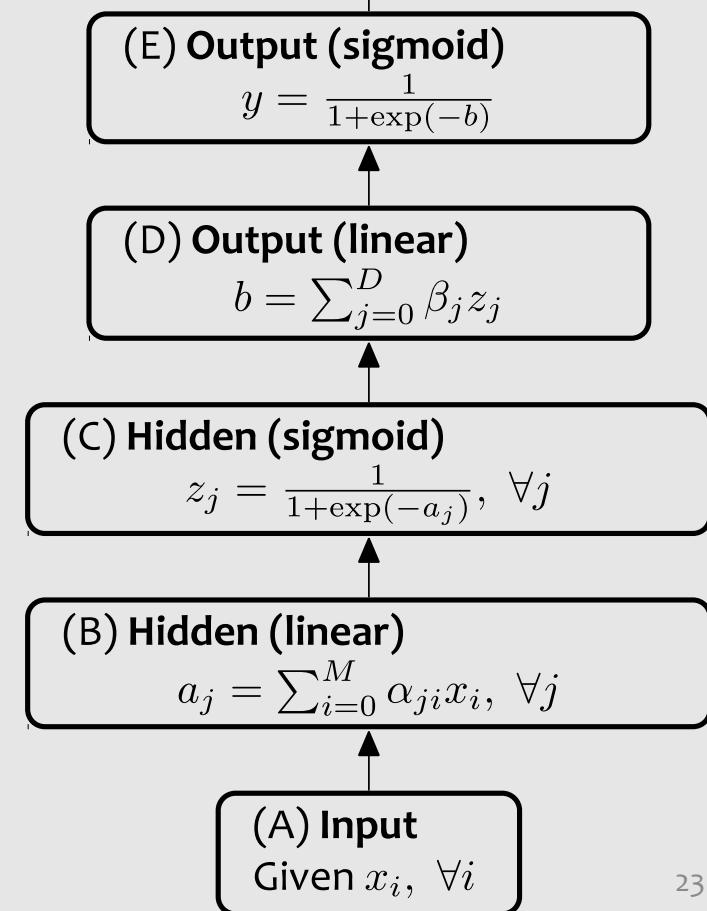
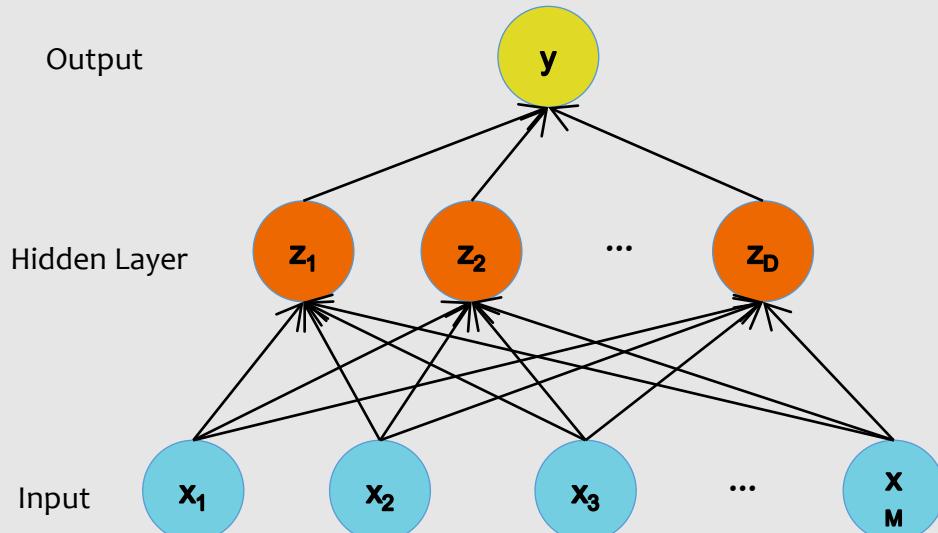
Logistic Regression Neural Networks



Week 7 Contents / Objectives

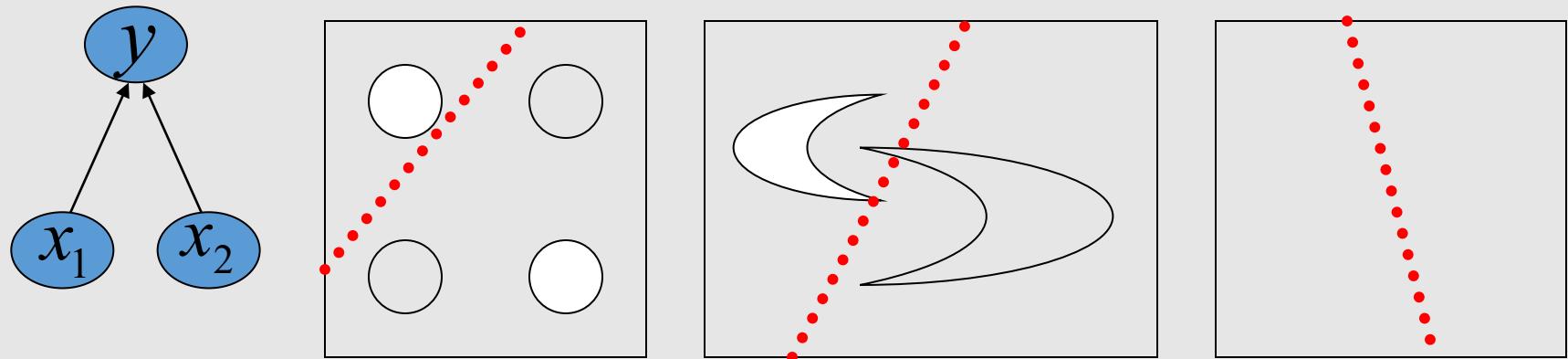
- Learning with Neurons
- Neural Networks (NNs)
- **NN Decision Boundary & Features**
- Convolutional NN Basics
- Convolutional NN Unboxing

Hidden Layers → Decisions



Decision Boundary

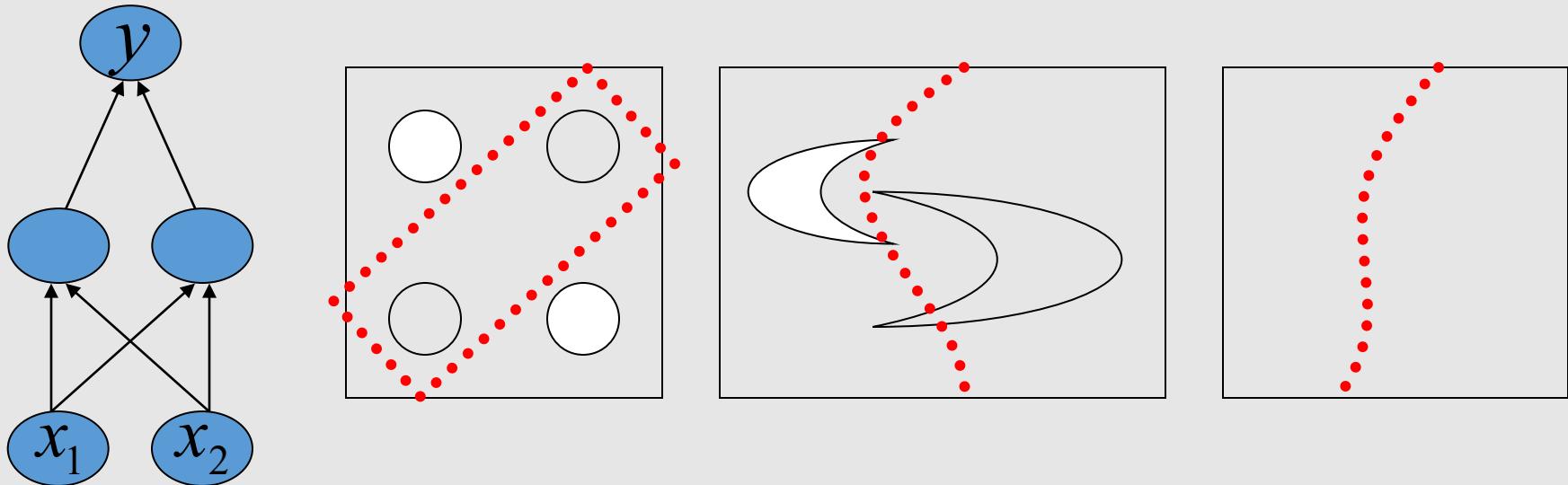
- 0 hidden layers: linear classifier
 - Hyperplanes



Example from to Eric Postma via Jason Eisner

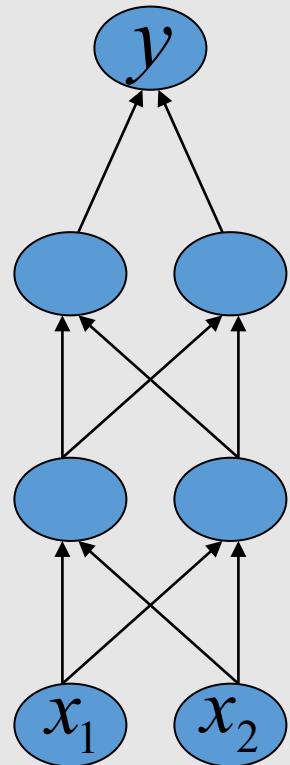
Decision Boundary

- 1 hidden layer
 - Boundary of convex region (open or closed)

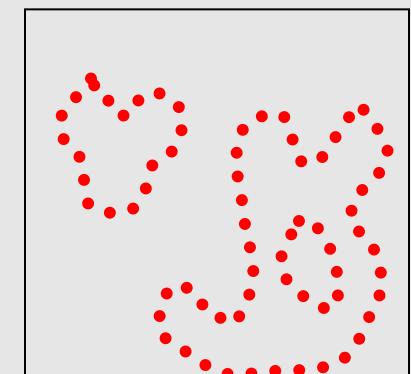
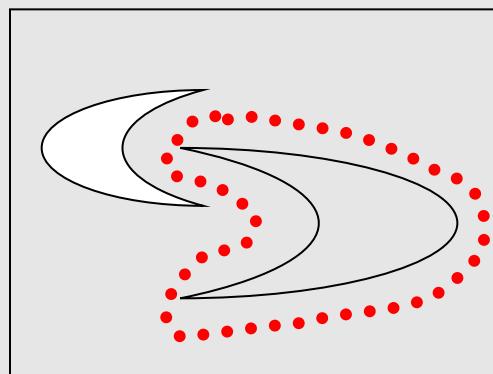
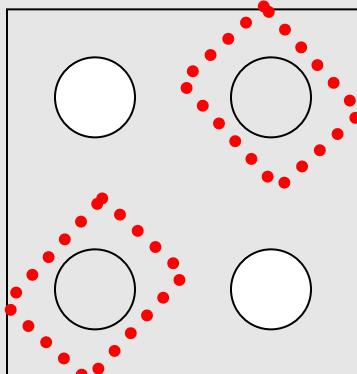


Example from to Eric Postma via Jason Eisner

Decision Boundary

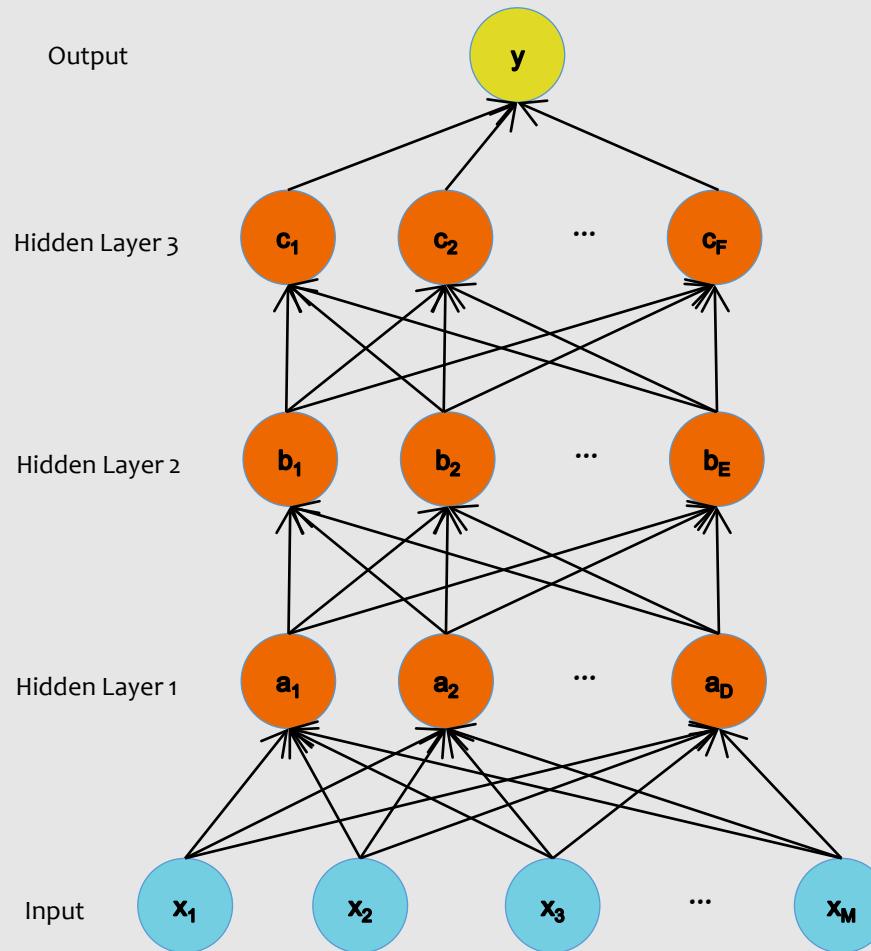


- 2 hidden layers
 - Combinations of convex regions



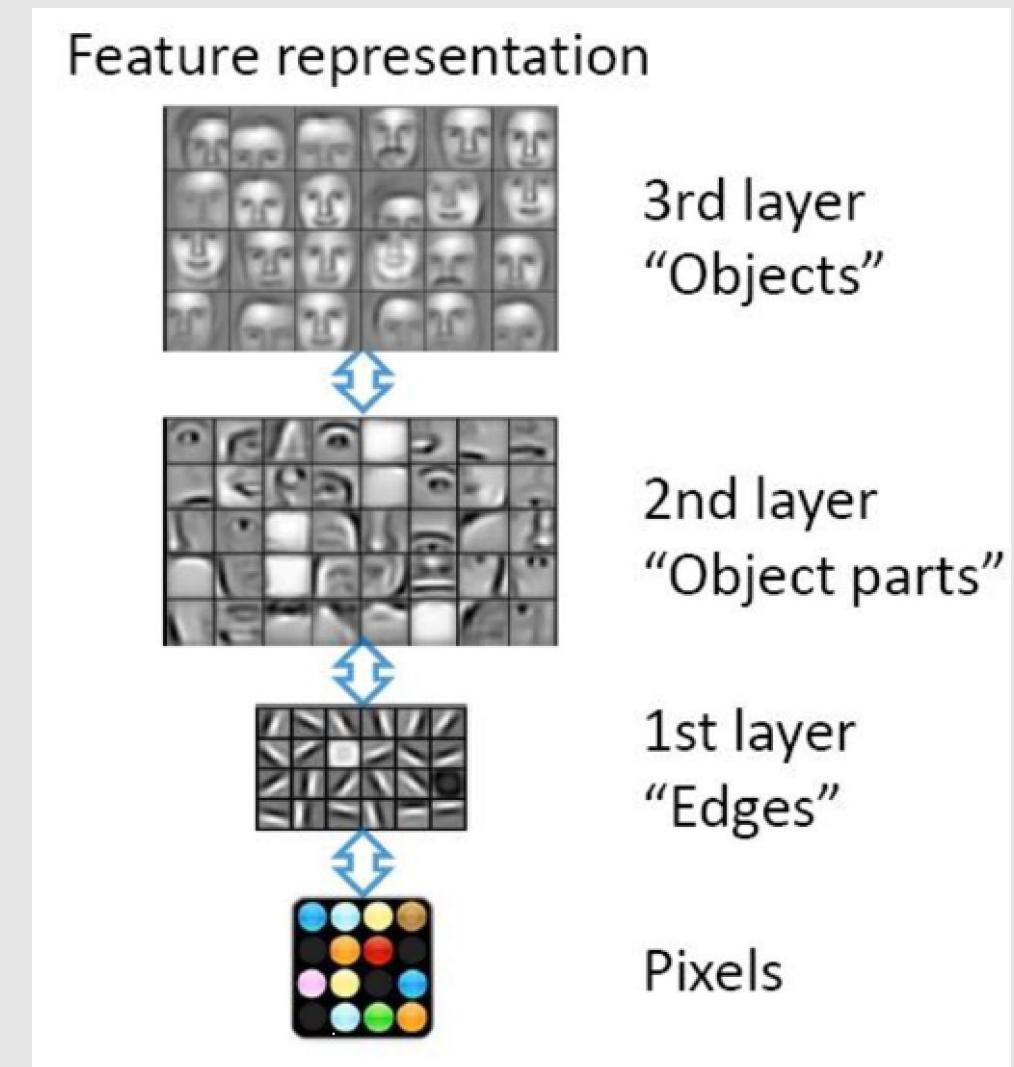
Example from to Eric Postma via Jason Eisner

Deeper Networks



Different Levels of Abstraction

- We don't know the “right” levels of abstraction
- So let the model figure it out!

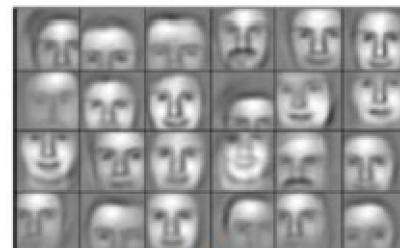


Different Levels of Abstraction

Face Recognition:

- Deep Network can build up increasingly higher levels of abstraction
- Lines, parts, regions

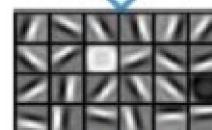
Feature representation



3rd layer
“Objects”



2nd layer
“Object parts”

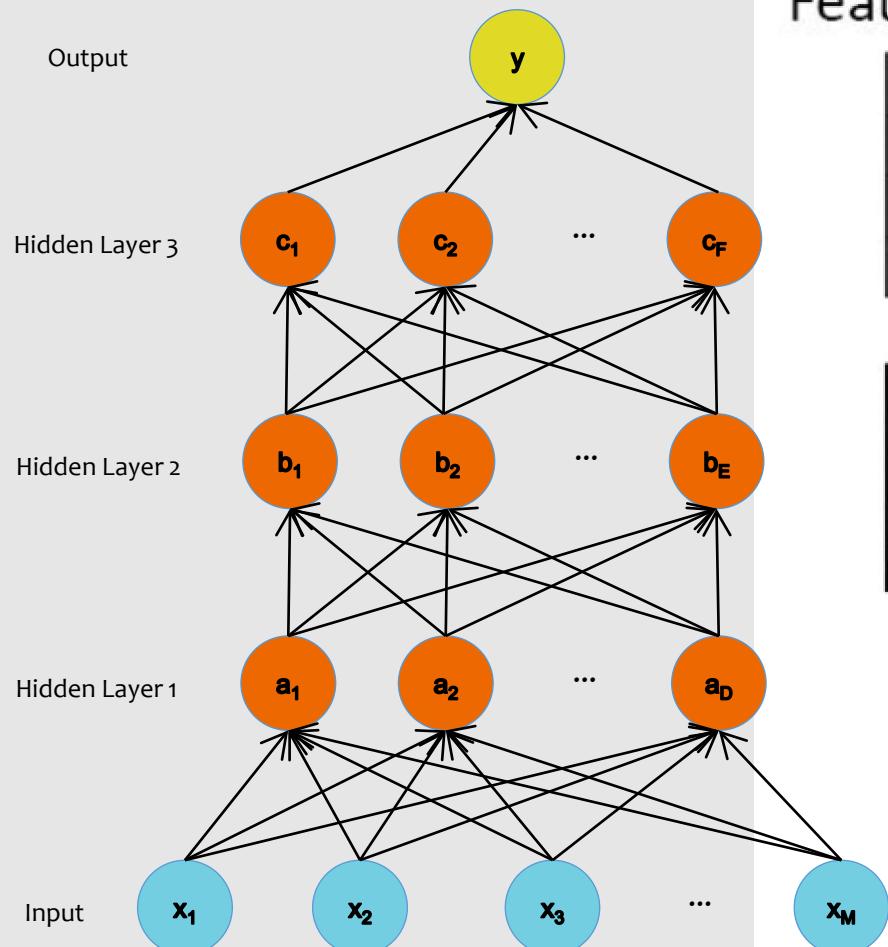


1st layer
“Edges”



Pixels

Different Levels of Abstraction



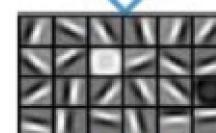
Feature representation



3rd layer
“Objects”



2nd layer
“Object parts”



1st layer
“Edges”

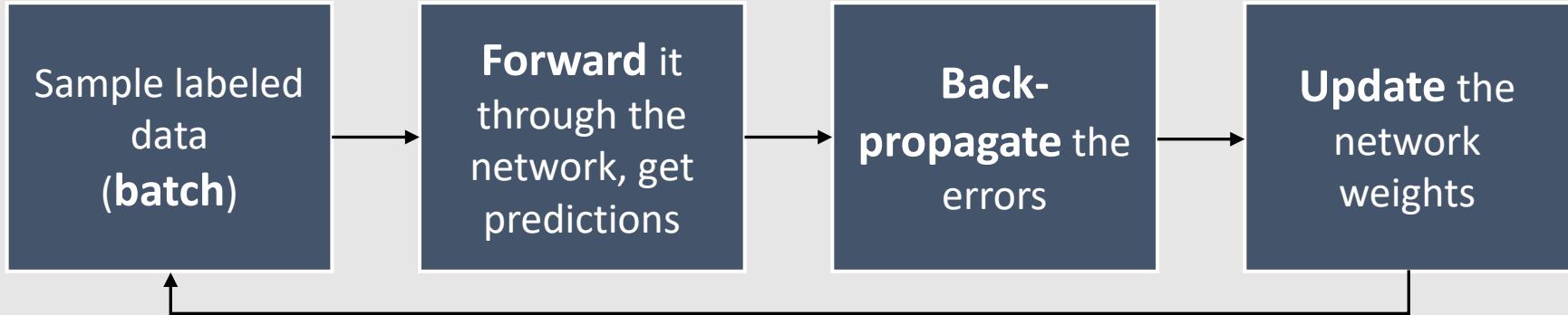


Pixels

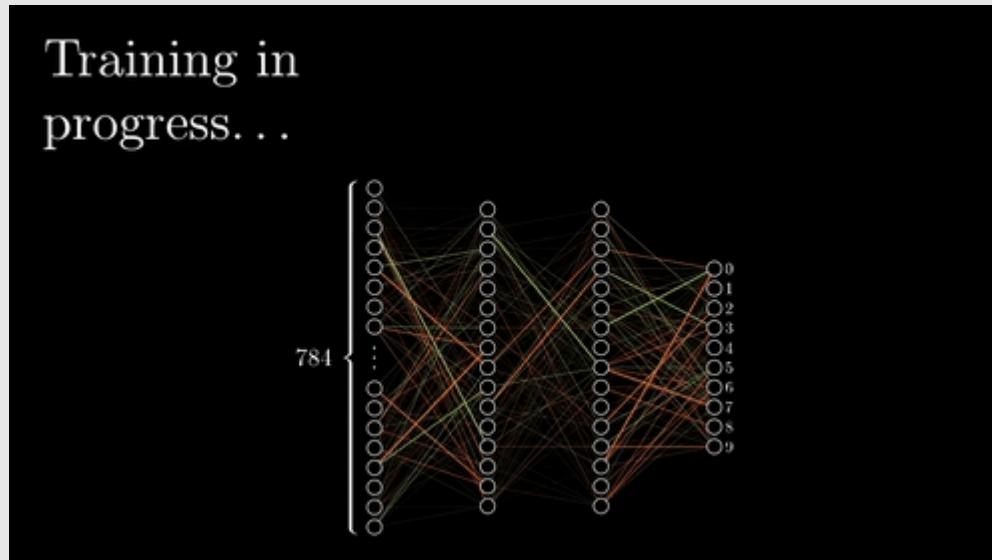
Machine Learning Ingredients

- **Data:** + pre-processing (& visualisation), e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure ~ Architecture \leftarrow expert knowledge
 - Must **specify** before ML, can optimise via cross validation (CV)
 - **Hyper-parameter**, e.g., prior, #degree, layer \leftarrow knowledge
 - Must **specify** (choices) and can optimise via CV (**tuning**)
 - Parameters (theta)
 - Compute/learn parameter, e.g., **weights**, bias \leftarrow optimisation alg.
- Evaluation **metric** (what's best): loss/error function
- **Optimisation:** (how to find the best) learnable parameters

Neural Network Training



Data → Model → Metric → Optimisation



Neural Network Ingredients

- **Data:** + pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure/Architecture: layered network
 - **Hyper-parameter:** layer specs, e.g. #layers, #neurons/units, activation function
 - Parameters (theta): layer weights & biases
- Evaluation metric (loss): max likelihood (min NLL), cross-entropy, etc.
- Optimisation: backpropagation (gradient-based)

Week 7 Contents / Objectives

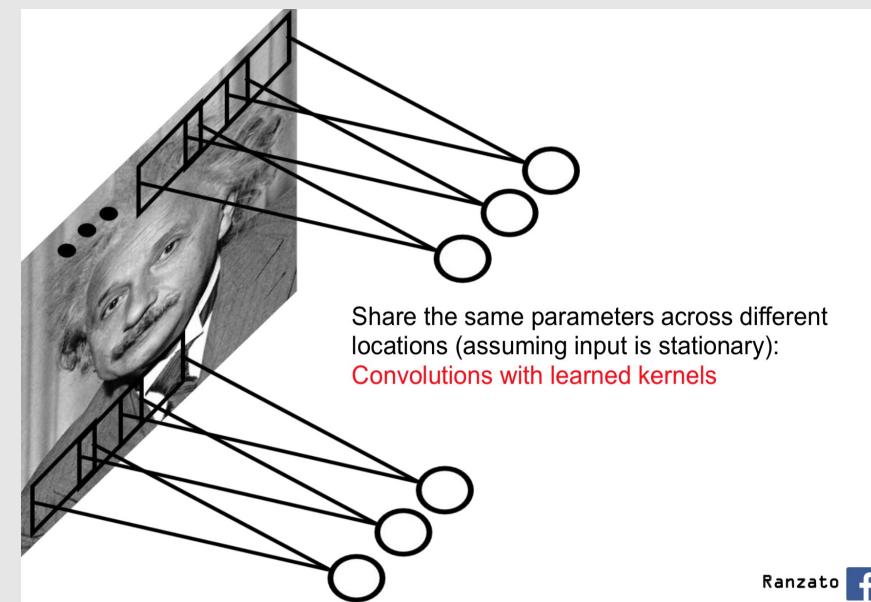
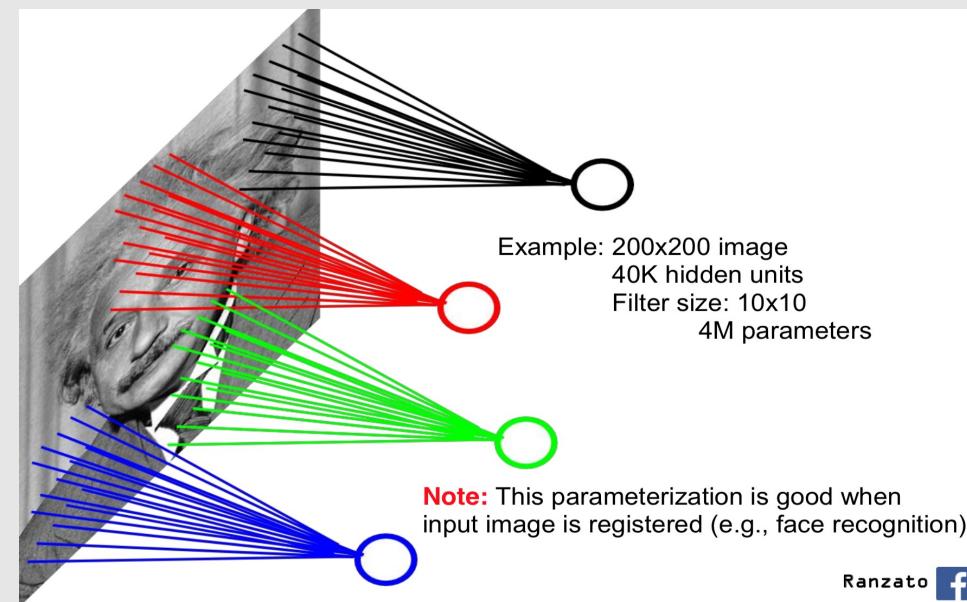
- Learning with Neurons
- Neural Networks (NNs)
- NN Decision Boundary & Features
- **Convolutional NN Basics**
- Convolutional NN Unboxing

Fully Connected (FC) Layer

- Linear layers such as linear/logistic regression
- What if our network is bigger?
 - Input image: 200×200 pixels, first hidden layer: 500 units
 - **Question:** How many weights for input \rightarrow 1st hidden?
20 million
 - **Q:** Why is using an FC layer problematic for images?
 - Computing predictions (forward pass) will take a long time
 - A large number of weights requires a lot of training data to avoid overfitting
 - Small shift in image can result in large change in prediction
 - Not making use of the image geometry

Convolutional Neural Network

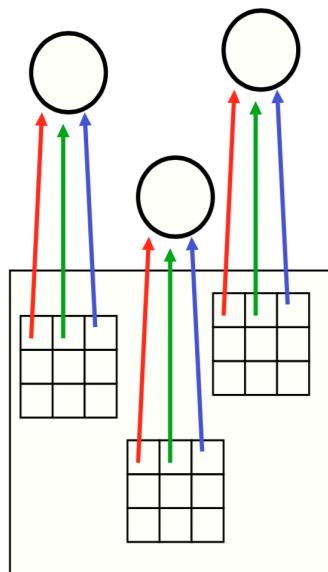
- Key ideas:
 - Locally-connected layers: look for local features in small regions of the image
 - Weight-sharing: detect the same local features across the entire image



Weight Sharing

- Each neuron on the higher layer detects the same feature, but in different locations on the lower layer

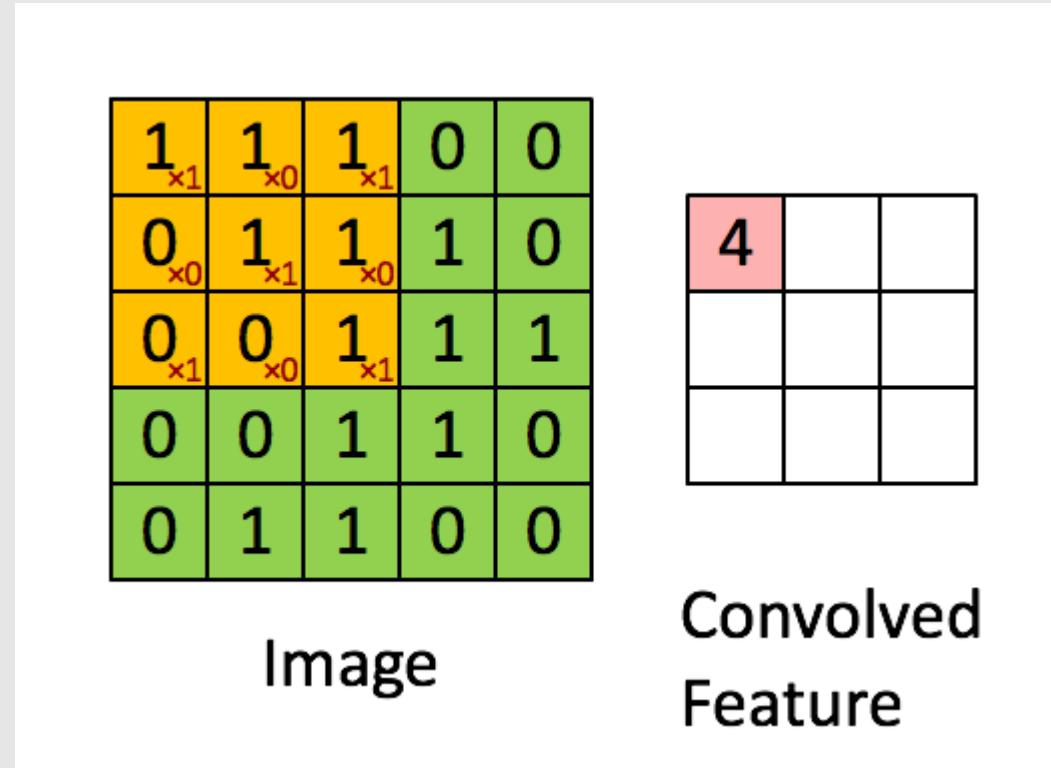
The red connections all have the same weight.



“Detecting” = the output (activation) is high if the feature is present

“Feature” = something in the image, like an edge, blob or shape

Forward Pass Example (Single Channel)



<https://developer.nvidia.com/sites/default/files/pictures/2018/convolution-2.gif>

- The **kernel/filter (yellow)** contains the trainable weights. In the above, the kernel size is 3×3 .
- The “*convolved features*” is another term for “*convolution output*”

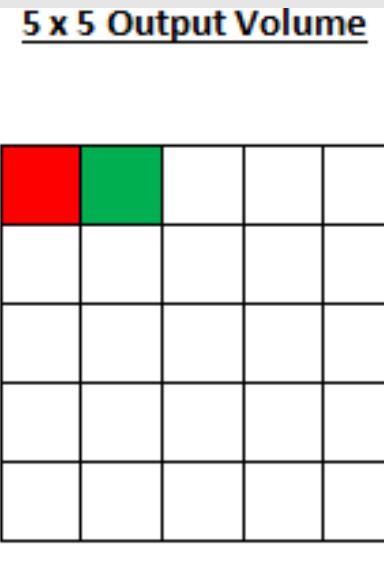
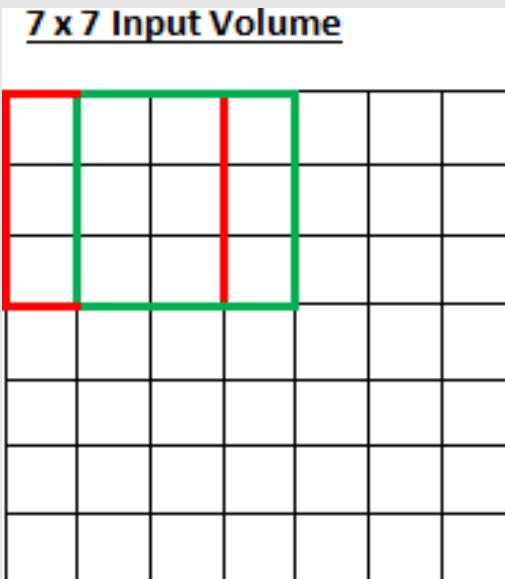
Example of convolution

Greyscale input image: 7×7

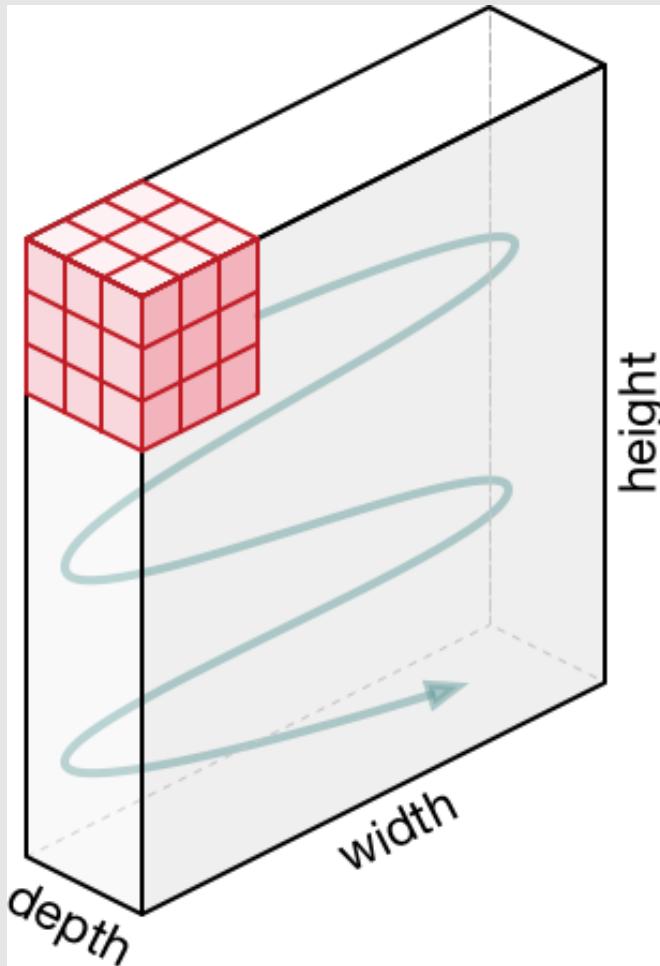
Convolution **kernel**: 3×3

Questions:

- How many units are in the output?
- How many trainable weights are there?



Convolution in RGB for colour images

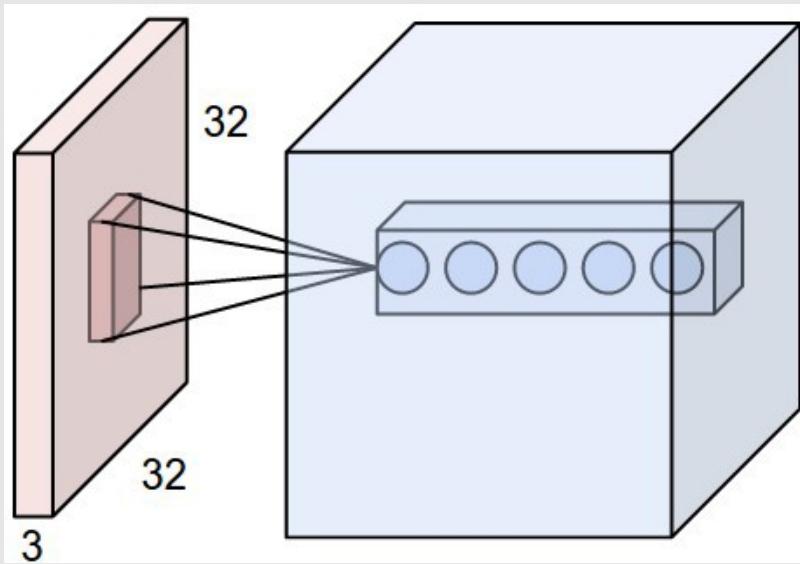


The kernel: a 3-D tensor! In this example, the kernel has size **3** $\times 3 \times 3$.

The first number **3**: the number of **input channels** or **input feature maps**

Detecting Multiple Features

- **Q:** What if we want to detect many features of the input? (e.g. **both** horizontal edges and vertical edges, and maybe even other features?)
- **A:** Have many convolutional filters!

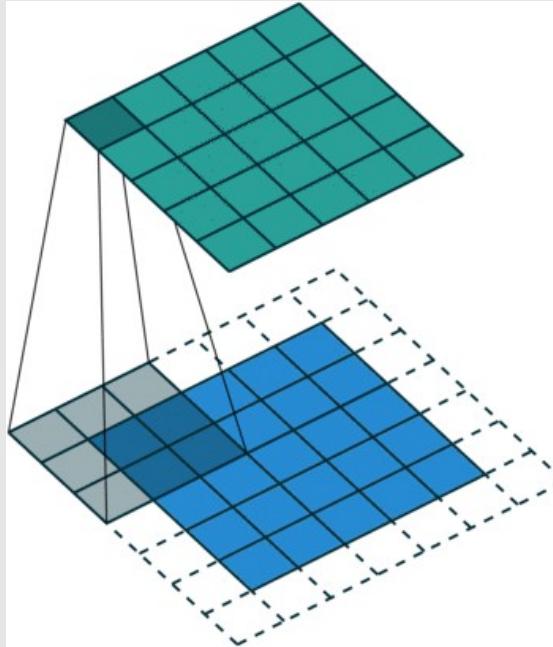


Input image size: $3 \times 32 \times 32$

Convolution kernel (4D): $\textcolor{red}{3} \times 3 \times 3 \times \textcolor{red}{5}$

- The number **3** is the number of **input channels** or **input feature maps**
- The number **5** is the number of **output channels** or **output feature maps**

Zero Padding

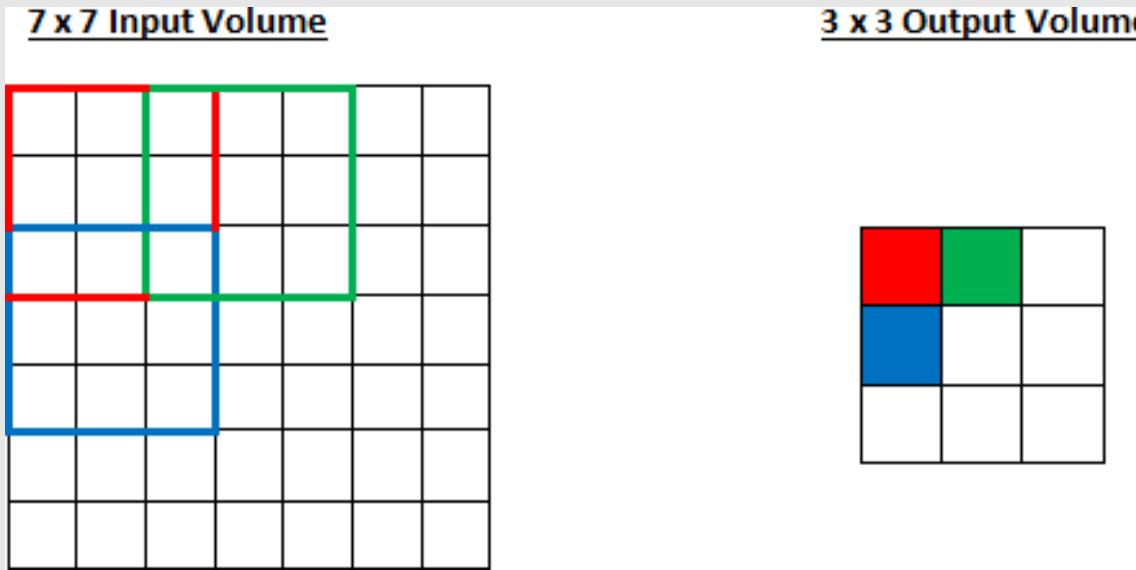


Add zeros around the border of the image (can add more than one pixel of zeros)

Question: Why might we want to add zero padding?

- Keep the next layer's width and height consistent with the previous
- Keep the information around the border of the image

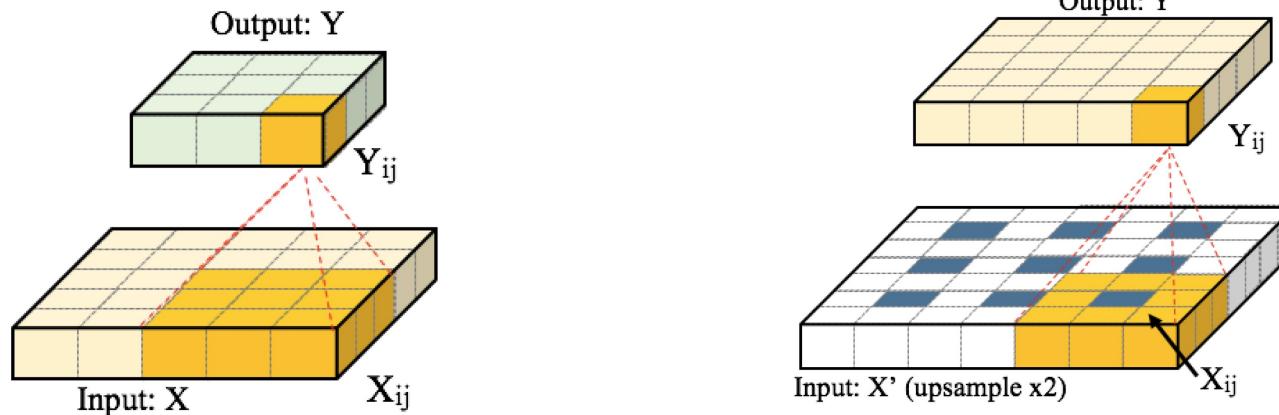
Strided Convolution



Shift the kernel by **2** (stride=2) when computing the next output feature.

Objective: to consolidate (summarise) information

Transpose Convolution Layer



(a) Convolutional layer: the input size is $W_1 = H_1 = 5$; the receptive field $F = 3$; the convolution is performed with stride $S = 1$ and no padding ($P = 0$). The output Y is of size $W_2 = H_2 = 3$.

(b) Transposed convolutional layer: input size $W_1 = H_1 = 3$; transposed convolution with stride $S = 2$; padding with $P = 1$; and a receptive field of $F = 3$. The output Y is of size $W_2 = H_2 = 5$.

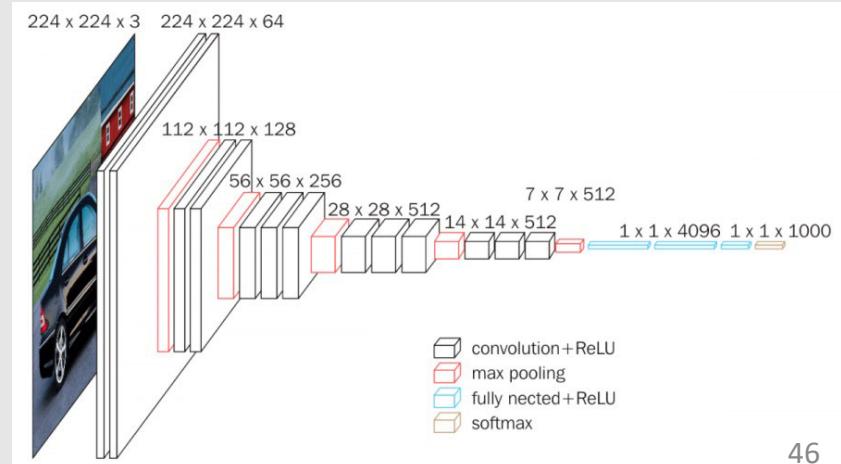
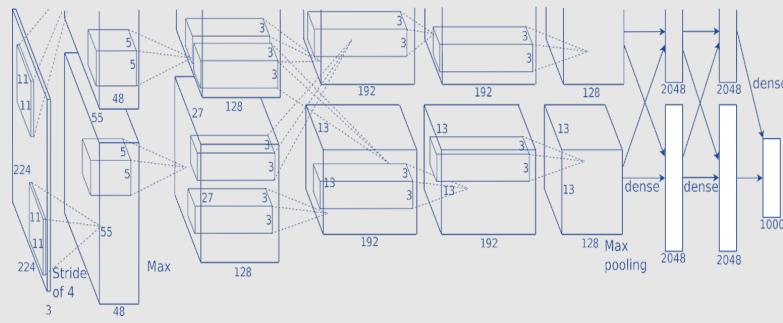
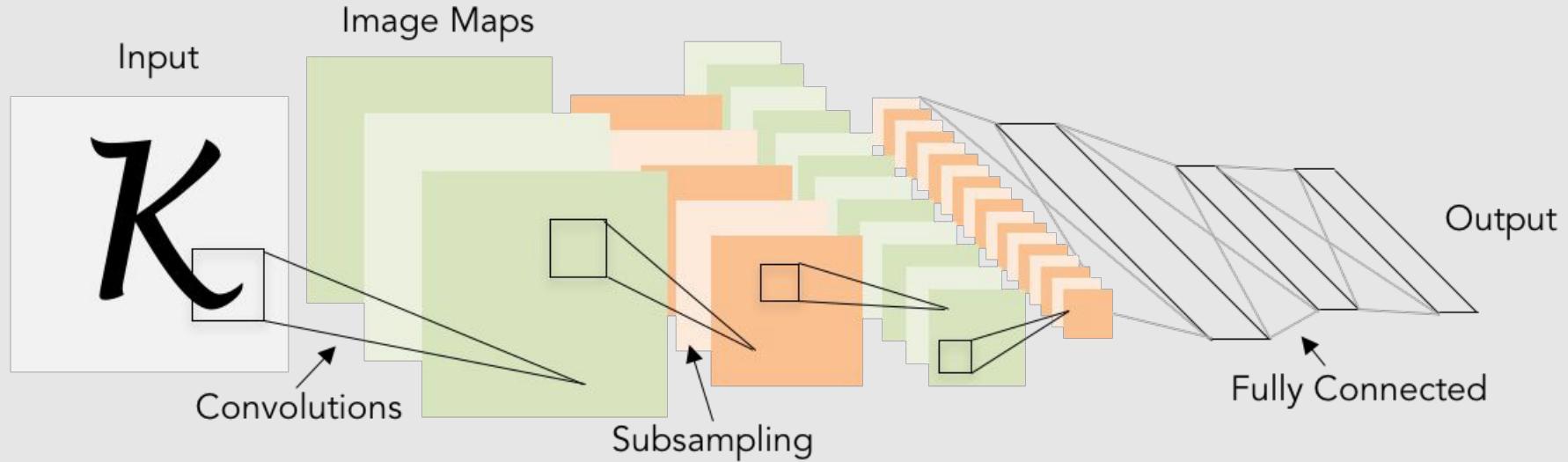
<https://www.mdpi.com/2072-4292/9/6/522/htm>

More at https://github.com/vdumoulin/conv_arithmetic

Week 7 Contents / Objectives

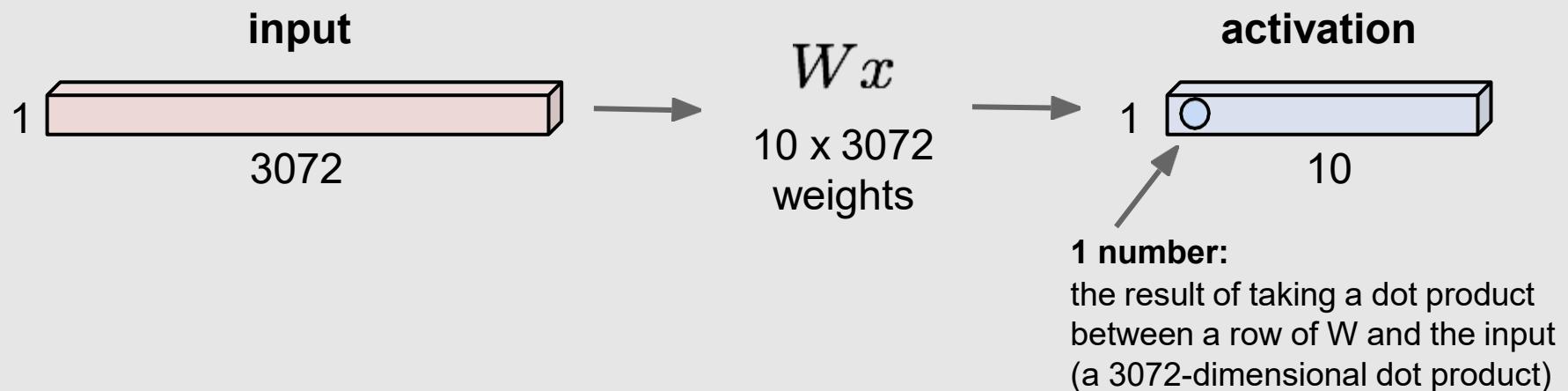
- Learning with Neurons
- Neural Networks (NNs)
- NN Decision Boundary & Features
- Convolutional NN Basics
- **Convolutional NN Unboxing**

Convolutional Neural Networks



Fully Connected Layer

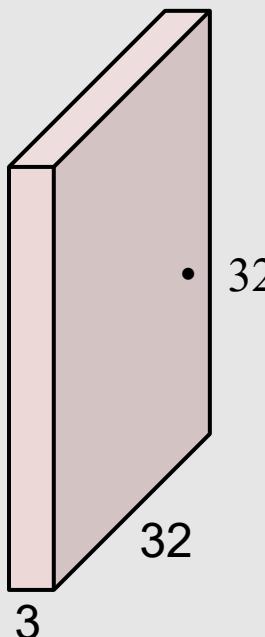
32x32x3 image → stretch to 3072 x 1



Convolution Layer

Tensor: Preserve spatial structure

- 32x32x3 image



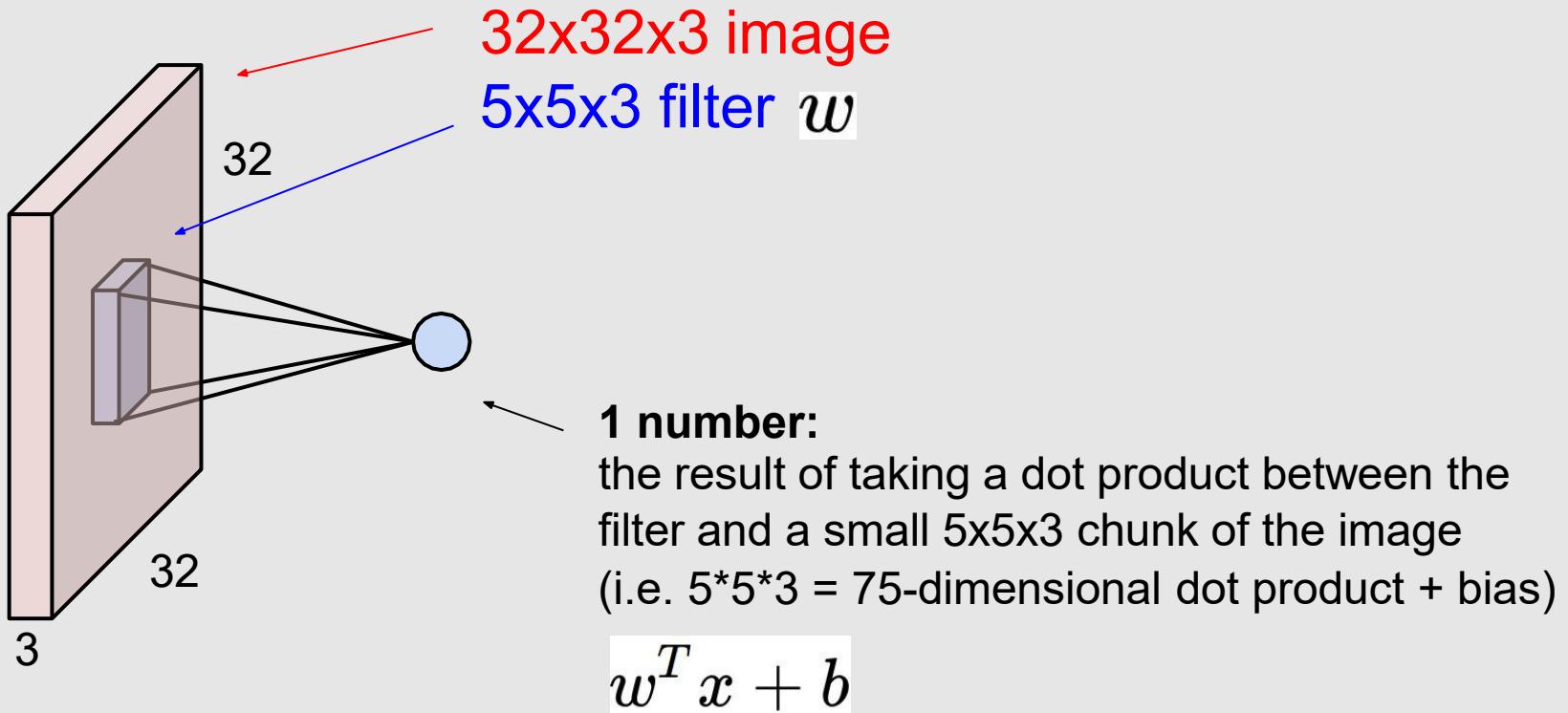
Filters always extend the full depth of the input volume

- 5x5x3 filter

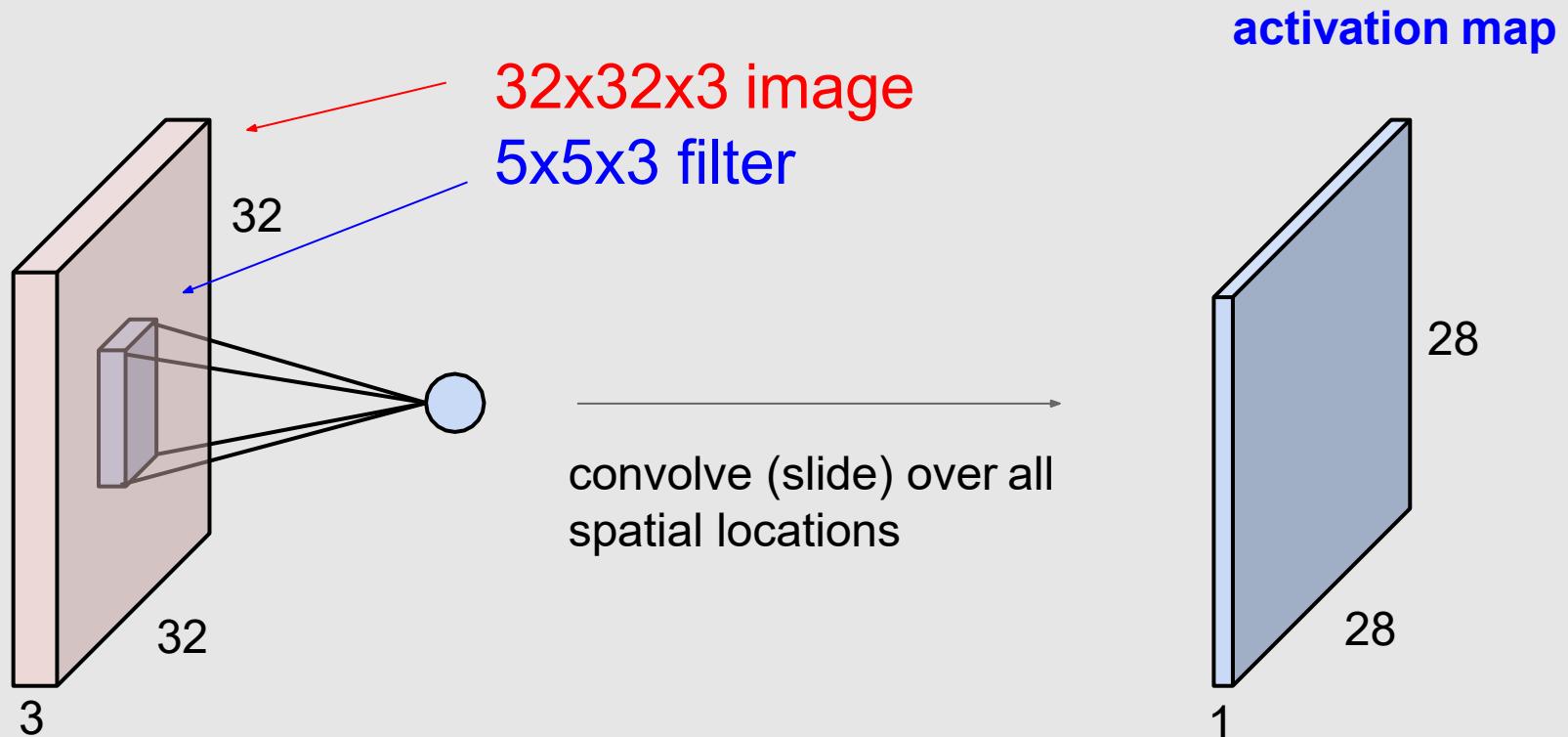


- **Convolve** the filter with the image
 - i.e. “slide over the image spatially, computing dot products”

Convolution Layer

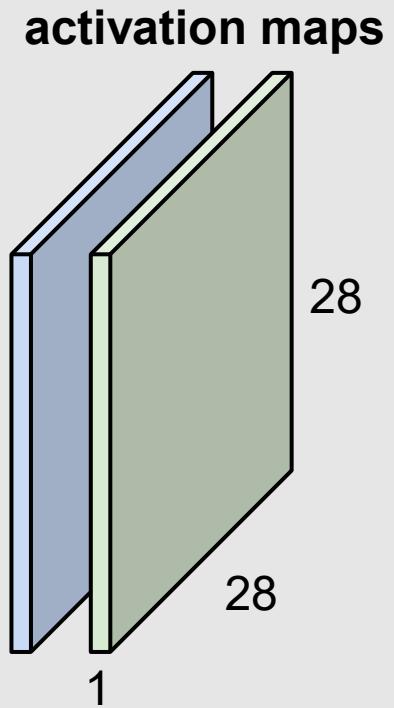
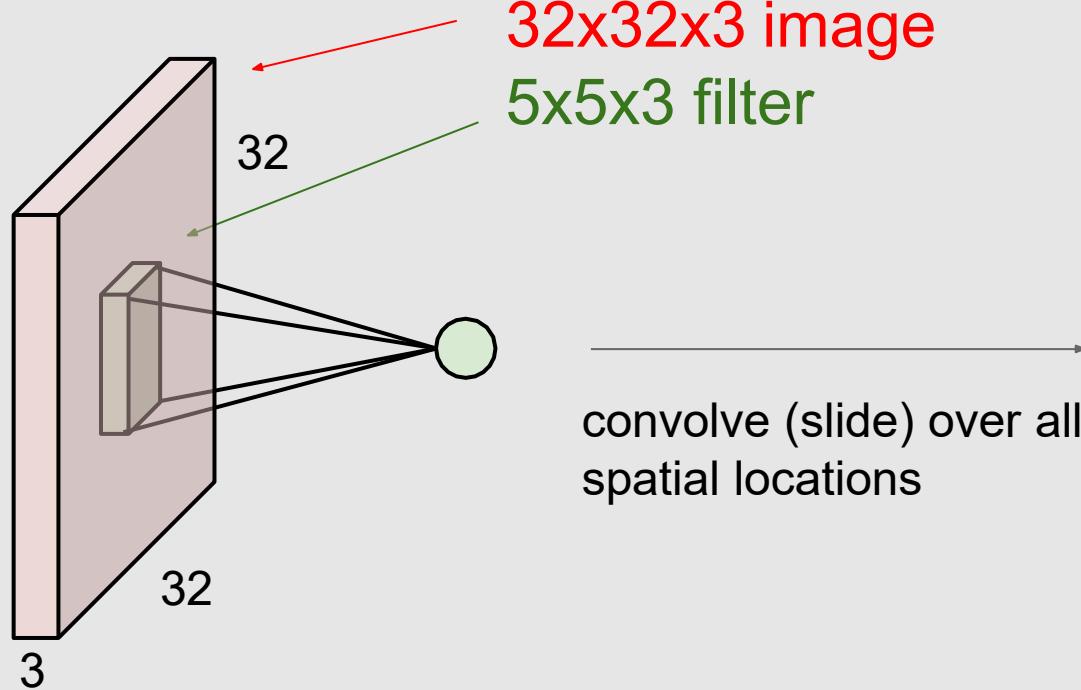


Convolution Layer



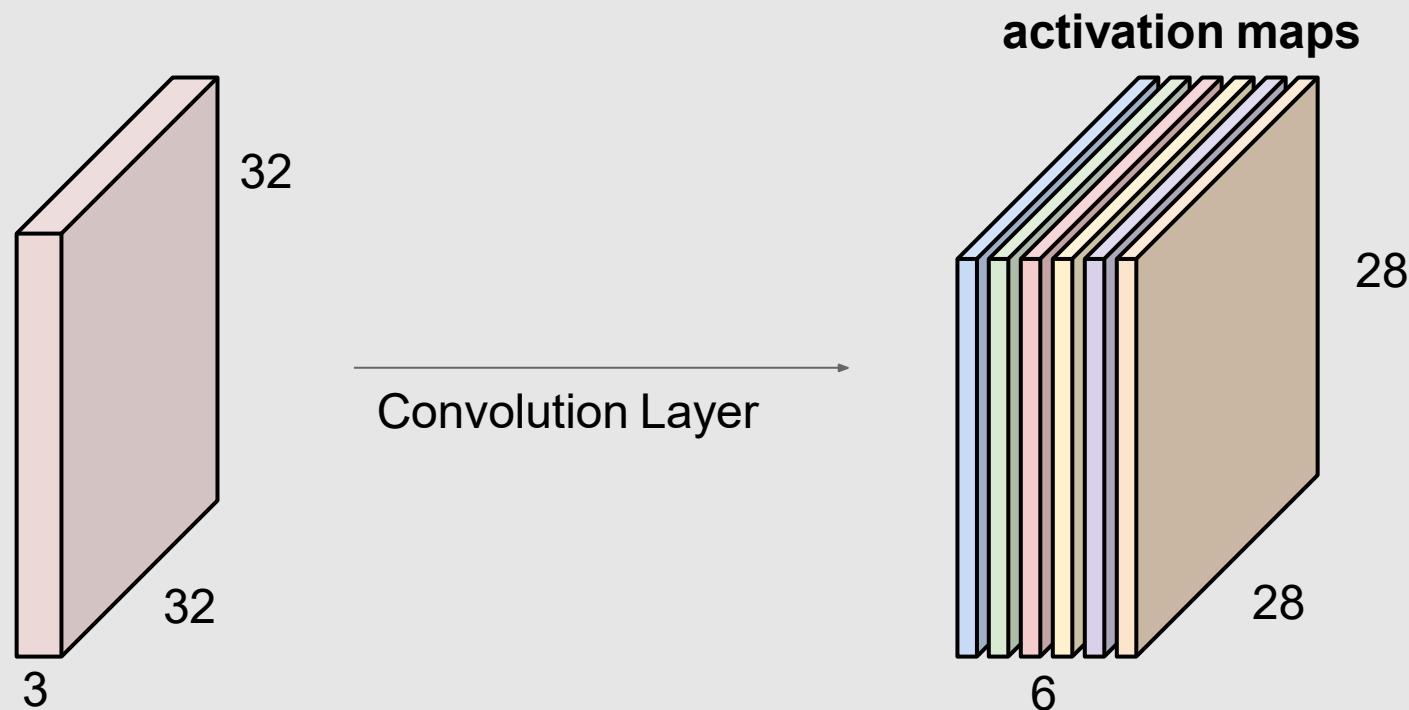
Convolution Layer

consider a second, green filter



Convolution Layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a “new image” of size $28 \times 28 \times 6$!

Convolution Operation

7

7x7 input (spatially)
assume 3x3 filter

7

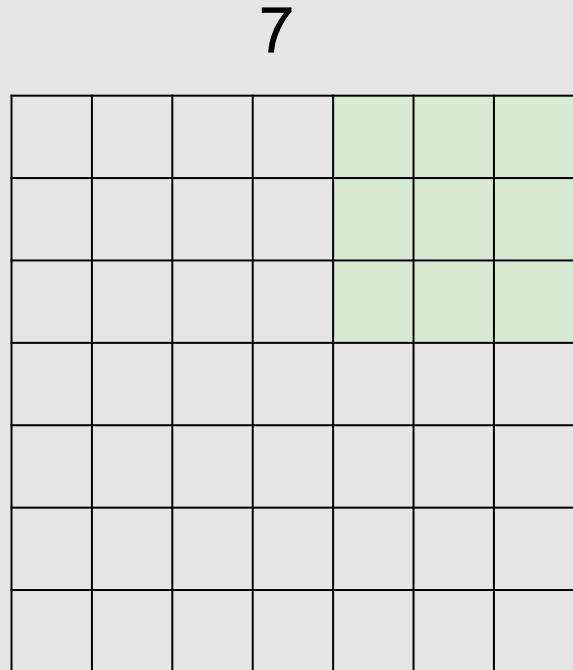
Convolution Operation

7

7x7 input (spatially)
assume 3x3 filter

7

Convolution Operation

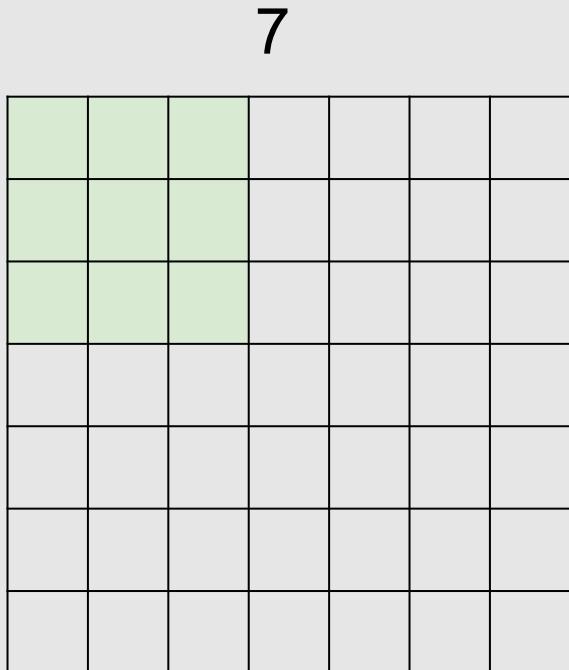


7x7 input (spatially)
assume 3x3 filter

7
→ 5x5
output

After three more sliding and dot products

Convolution with Stride

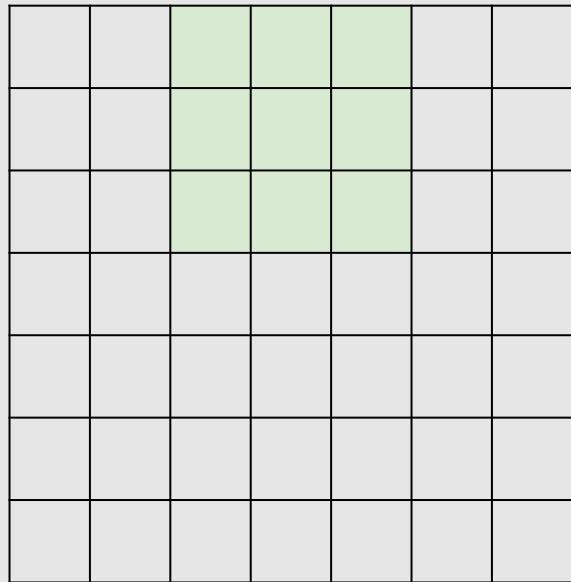


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

7

Convolution with Stride

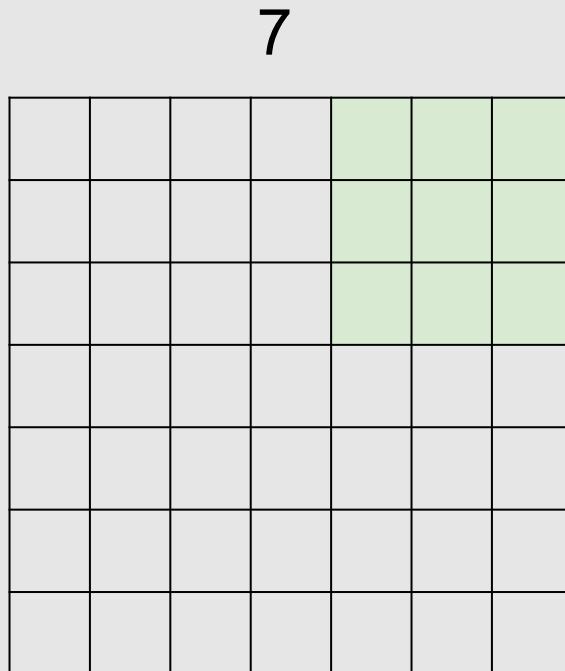
7



7

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

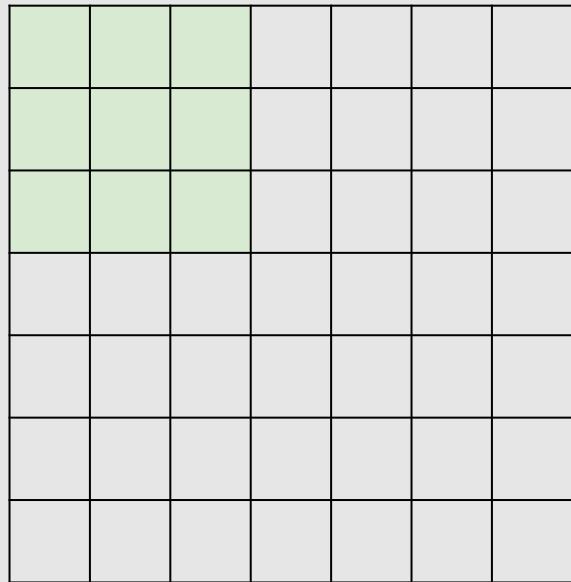
Convolution with Stride



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
→ **3x3 output!**

Convolution with Stride

7

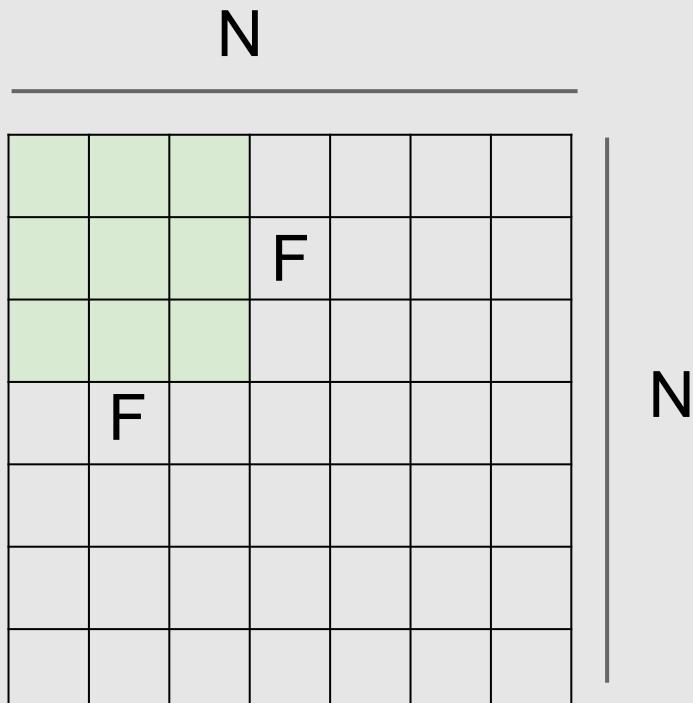


Question: 7x7 input
(spatially) assume 3x3 filter
applied **with stride 3**?

7

doesn't fit!
cannot apply 3x3 filter
on 7x7 input with stride
3 (unless ignoring parts).

Convolution – Size of output



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7$, $F = 3$:
stride 1 => $(7 - 3)/1 + 1 = 5$
stride 2 => $(7 - 3)/2 + 1 = 3$
stride 3 => $(7 - 3)/3 + 1 = 2.33$

Zero Padding

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with stride 1

pad with 1 pixel border => what is the output?

7x7 output!

in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with $(F-1)/2$. (will preserve size spatially)

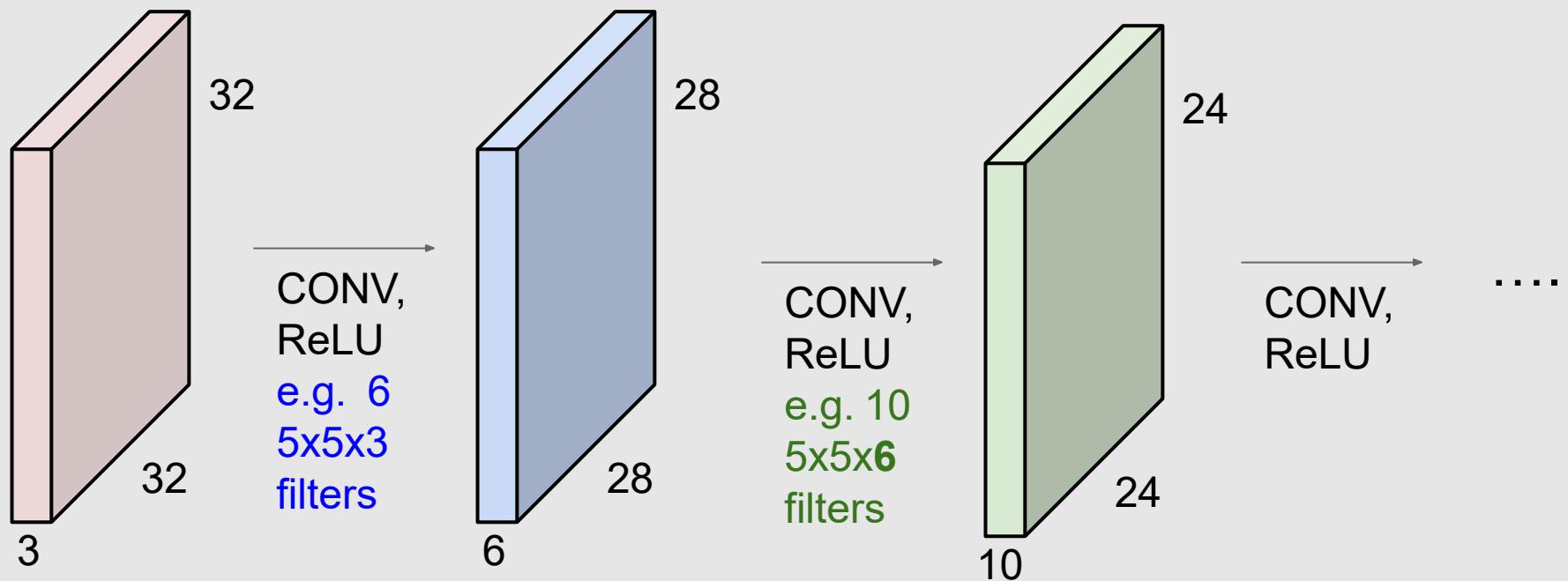
e.g. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

Convolution Shrinks

Example: 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! ($32 \rightarrow 28 \rightarrow 24 \dots$).

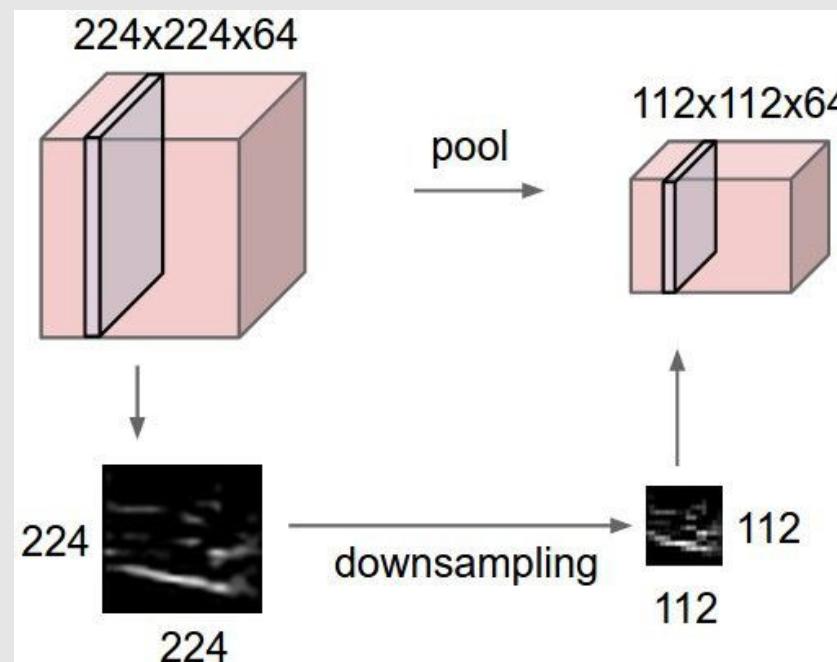


Exercises

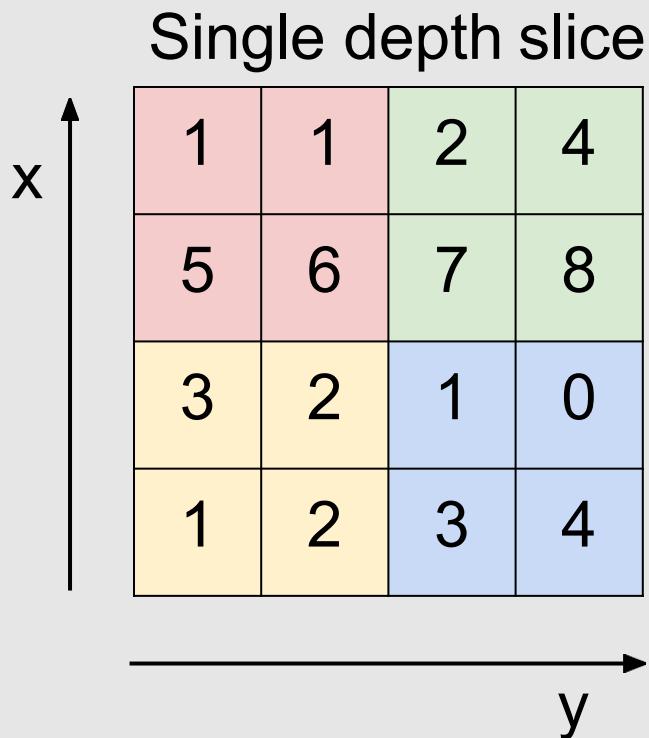
- Input volume: **32x32x3**; **10 5x5** filters with stride **1**, pad **2**
 - Output volume size?
 -
- Number of parameters for this layer?
-

Pooling Layer: Downsampling

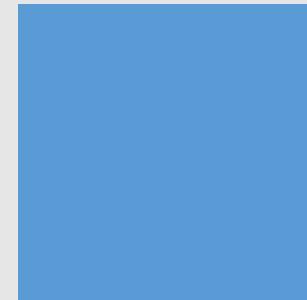
- Make the representations smaller and more manageable → dimensionality reduction
- Operate over each activation map independently:



Max Pooling



max pool with 2x2 filters
and stride 2



Lab 7 CNN (See notebook for details)

```
__init__(self):
super(CNN, self).__init__()
self.conv1 = nn.Conv2d(3, 6, 5) #3: #6
self.pool = nn.MaxPool2d(2, 2)
self.conv2 = nn.Conv2d(6, 16, 5)
self.fc1 = nn.Linear(16 * 5 * 5, 120)
self.fc2 = nn.Linear(120, 84)
self.fc3 = nn.Linear(84, 10)

forward(self, x):
x = self.pool(F.relu(self.conv1(x)))
x = self.pool(F.relu(self.conv2(x)))
x = x.view(-1, 16 * 5 * 5)
x = F.relu(self.fc1(x))
x = F.relu(self.fc2(x))
x = self.fc3(x)
return x
```

- Initial Image Size: $3 \times 32 \times 32$
- After conv1 : $6 \times 28 \times 28$ (32 → 6)
- After Pooling: $6 \times 14 \times 14$ (image size halved)
- After conv2 : $16 \times 10 \times 10$ (16 → 16)
- After Pooling: $16 \times 5 \times 5$ (halved again)
- After fc1 : 120
- After fc2 : 84
- After fc3 : 10 (= number of classes)

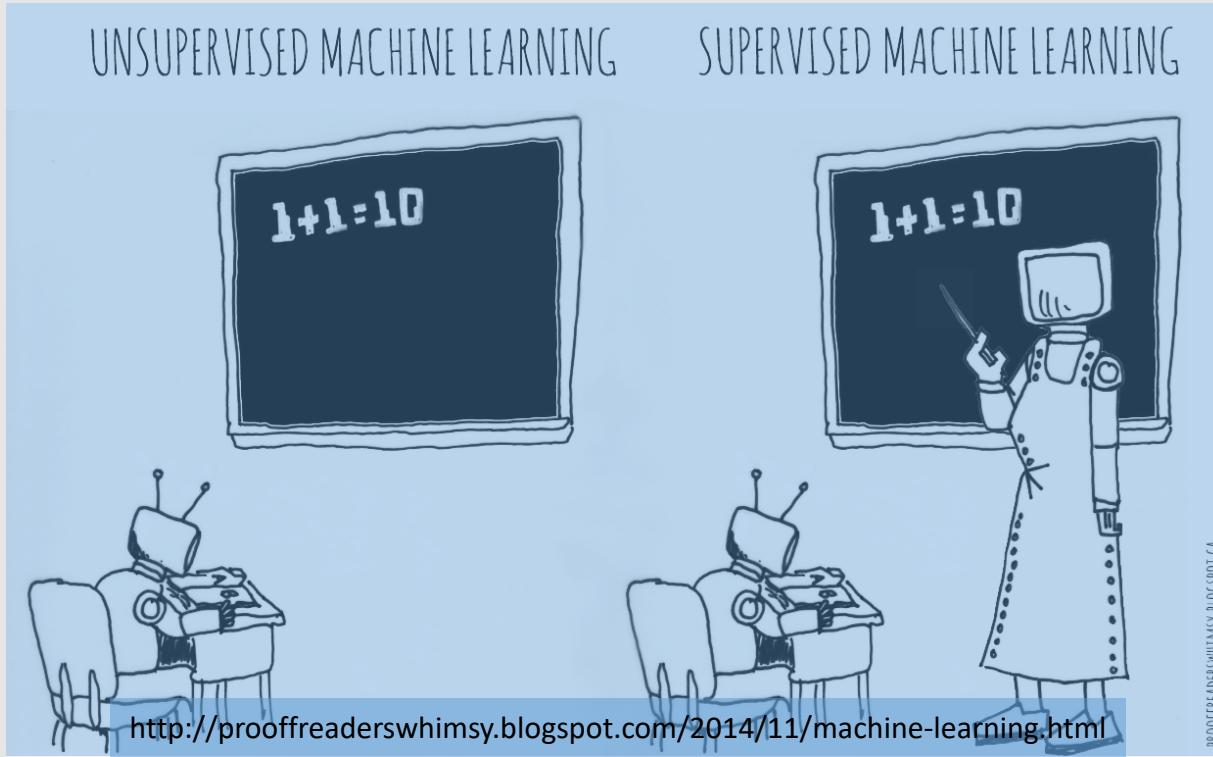
Acknowledgement

- The slides used materials from:
Matt Gormley, Eric Xing, Nina Balcan, Fei-Fei Li & Justin Johnson & Serena Yeung, Lisa Zhang, Michael Guerzhoy, Svetlana Lazebnik, Ismini Lourentzou, Dekai Wu

Recommended Reading

- [CS231n: Convolutional Neural Networks for Visual Recognition from Stanford \(Fei-Fei Li et al.\)](#)
- [The Deep Learning Book with a free official html version provided by the authors \(Ian Goodfellow et al.\)](#)
- [Convolution arithmetic](#)
- PyTorch documentations
- The lab notebook and references

Lecture 8: Unsupervised Learning



Haiping Lu

YouTube Playlist: <https://www.youtube.com/c/HaipingLu/playlists>

COM4059/6059: MLAI21@The University of Sheffield

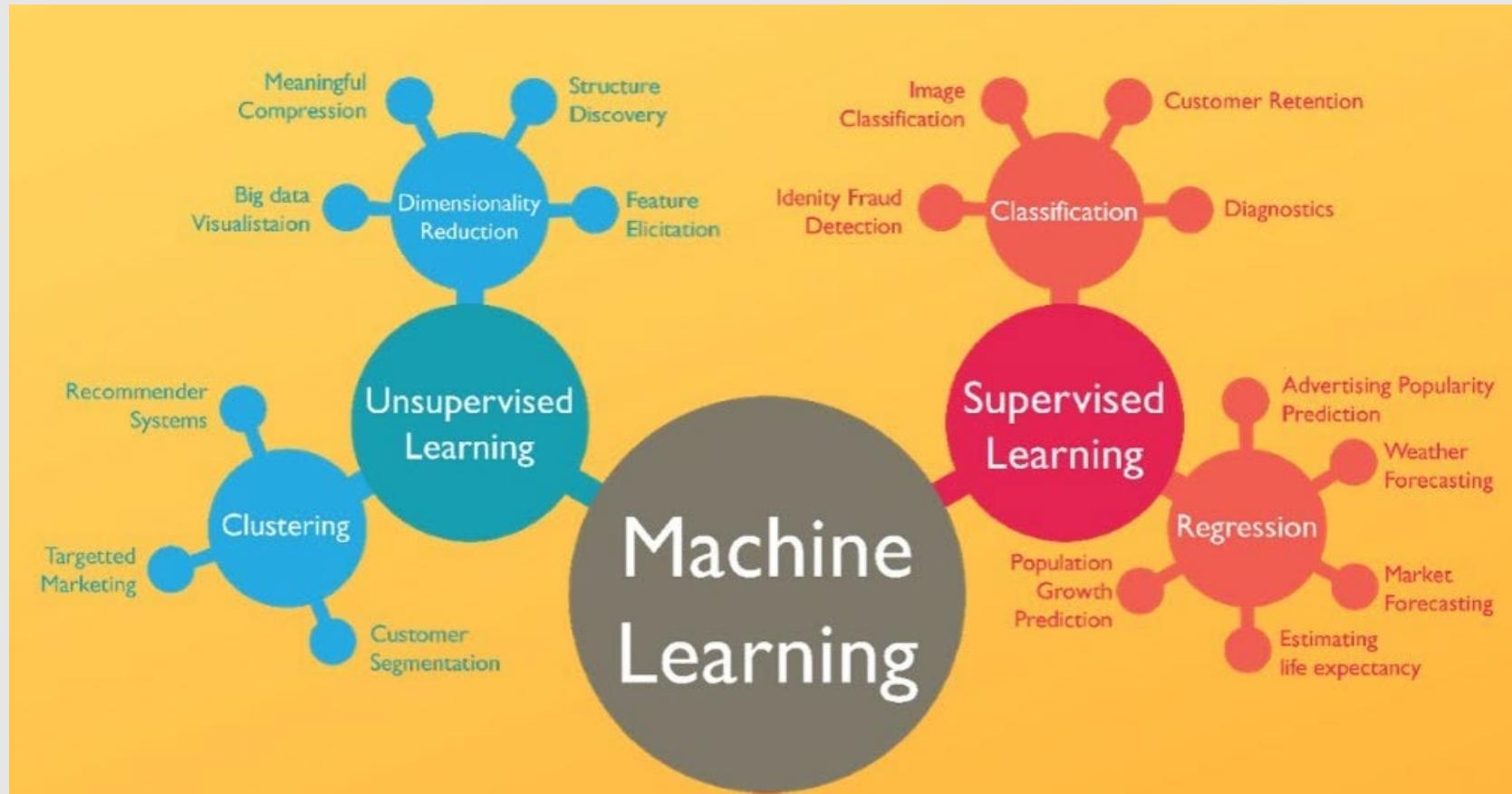
Week 8 Contents / Objectives

- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- PCA Unboxing
- Clustering: from k -means to spectral
- Autoencoder

Week 8 Contents / Objectives

- **Why Unsupervised Learning?**
- Principal Component Analysis (PCA)
- PCA Unboxing
- Clustering: from k -means to spectral
- Autoencoder

Supervised vs Unsupervised



<https://i.morioh.com/2020/04/14/ff897f322fed.jpg>

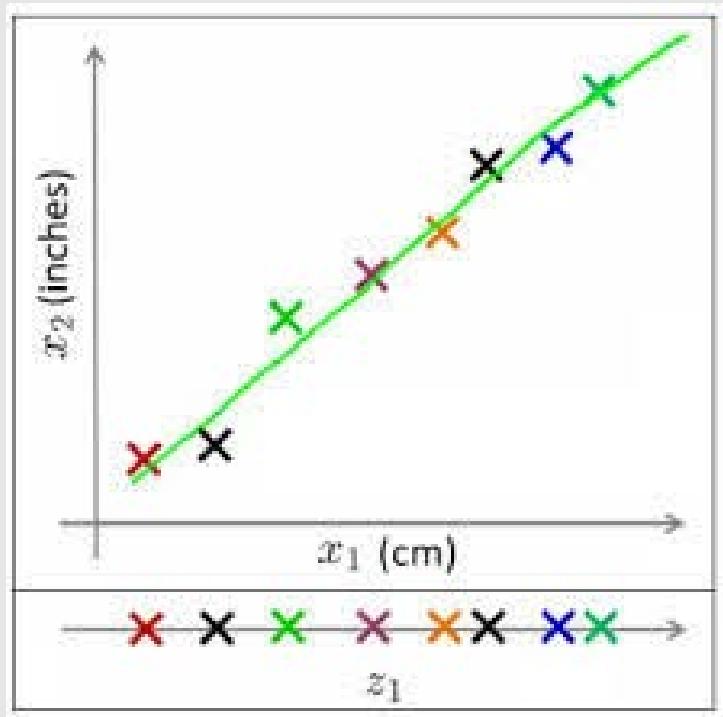
Unsupervised Learning

- Supervised learning: each data point has a label
- Unsupervised learning: no labels for the data
 - Dimensionality reduction
 - Clustering

Machine Learning	Supervised	Unsupervised
Discrete output	Classification	Clustering
Continuous output	Regression	Dimensionality Reduction

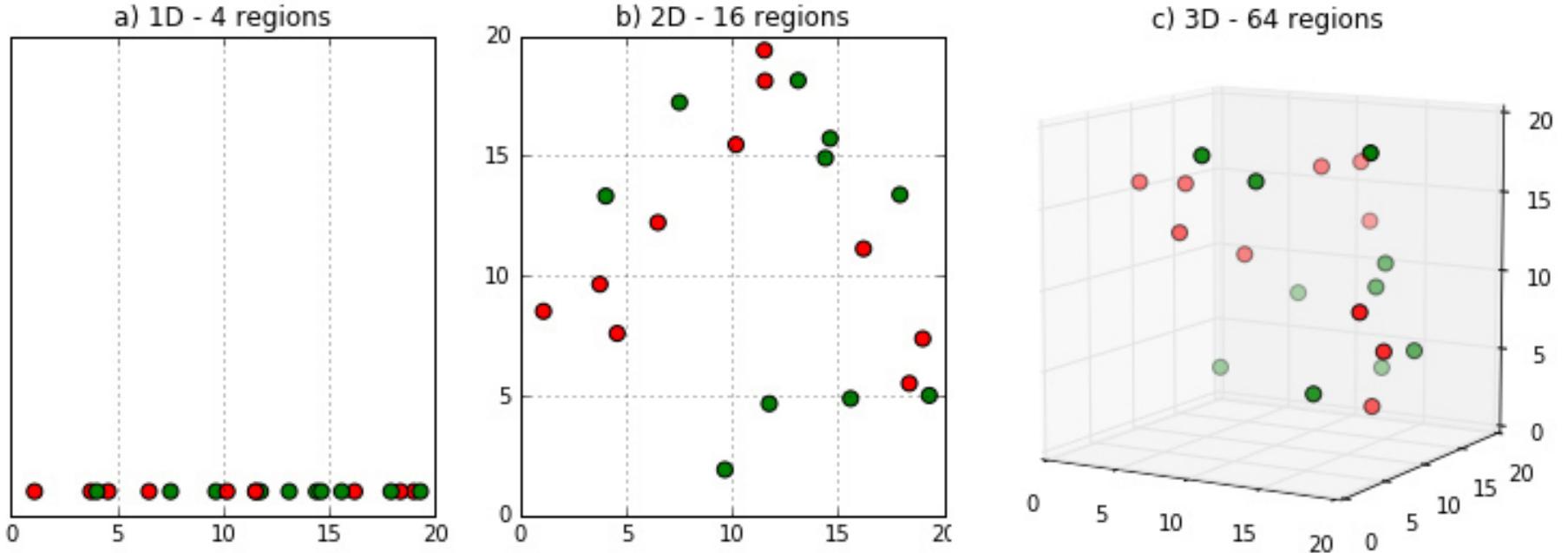
Dimensionality Reduction (DR)

- $2 \rightarrow 1$



High Dimensional Low Dimensional

Original data	Transformed
(1, 1.2)	1.15
(2, 2)	2
(3, 3.3)	3.1
...	...



Why DR?
High D → Low D

- Curse of dimensionality
 - Nearest neighbours
- Reduce redundancy (correlation)
- Visualisation

Question

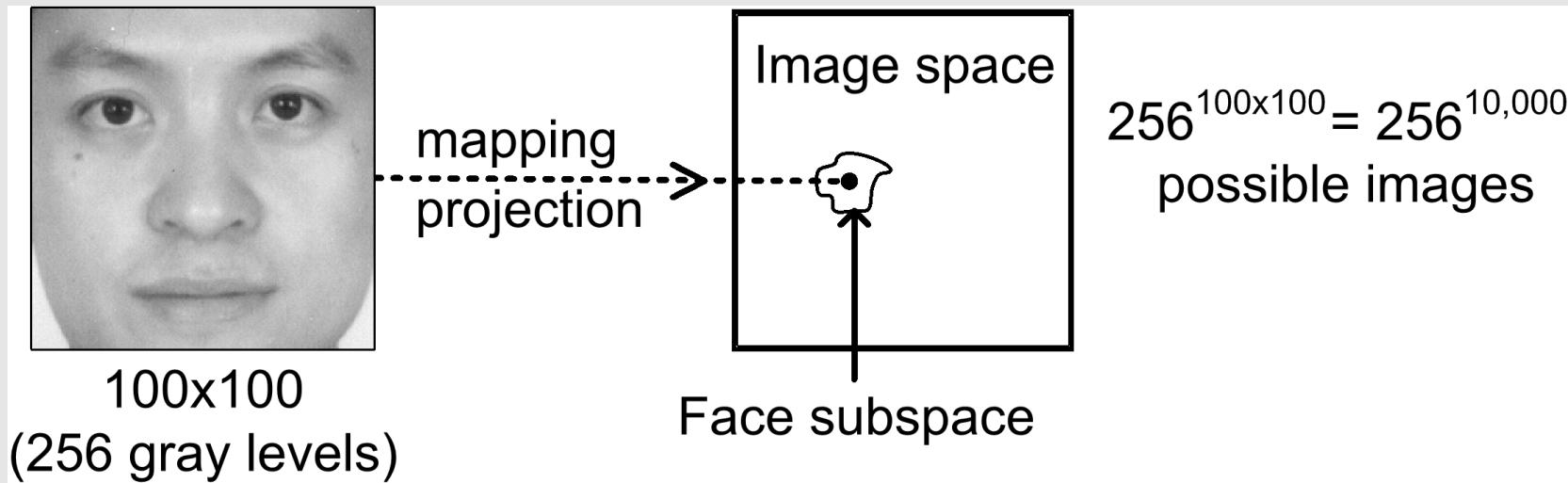
- USPS dataset handwritten digit
- Size: 64 x 57; binary (1-bit, BW)
- The binary image space contains much more than just this digit.



How many possible images of this size and bit depth?

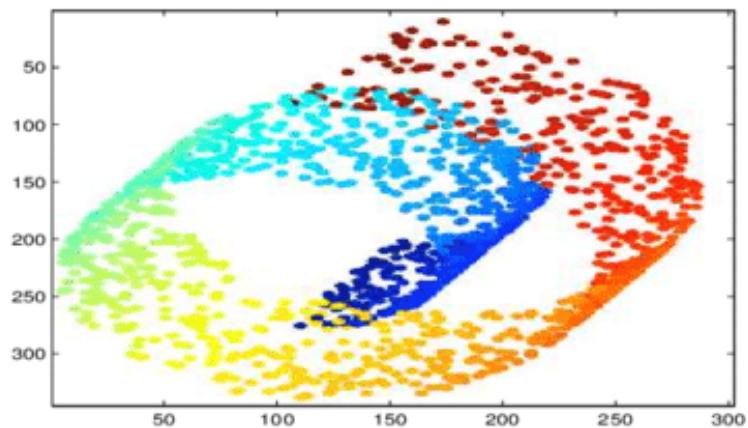
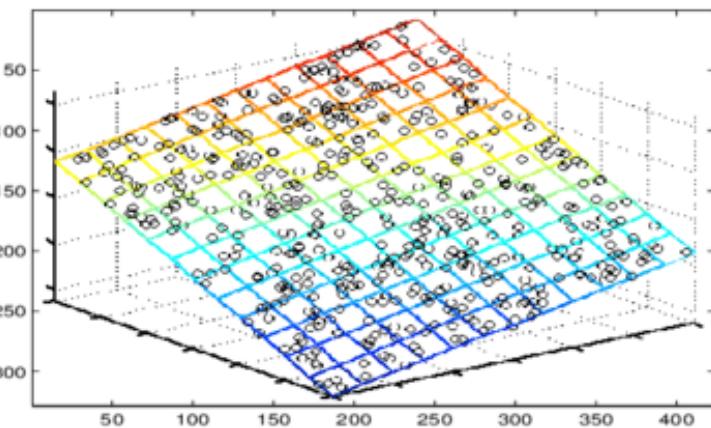
$$2^{64 \times 57} = 2^{3648} = ?$$

How About a Face?



Low-D Subspace/Manifolds

- For high dimensional data with **structure**:
 - Fewer variations than dimensions
 - Data to live on a lower dimensional manifold
 - → Deal with them by looking for a lower dimensional embedding (or projection, transformation)



Week 8 Contents / Objectives

- Why Unsupervised Learning?
- **Principal Component Analysis (PCA)**
- PCA Unboxing
- Clustering: from k -means to spectral
- Autoencoder

PCA?

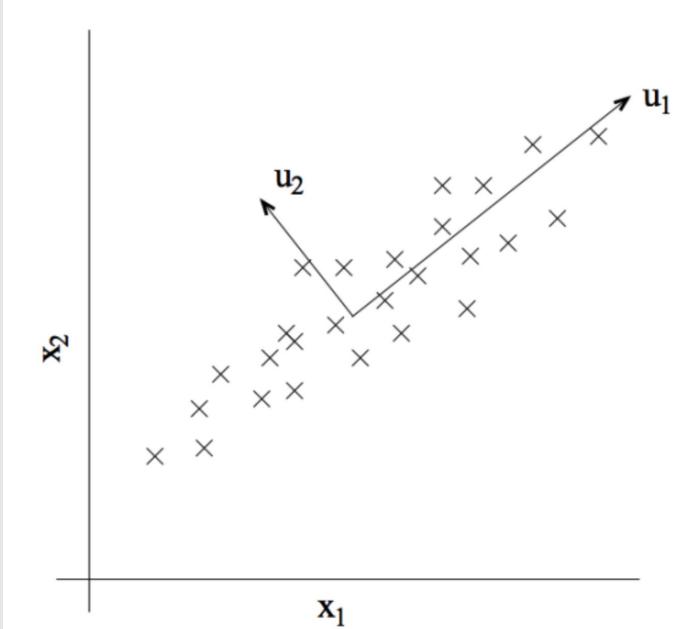
Visualisation demo



Principal Component Analysis

The idea:

1. Rotate the data with some rotation matrix R (change of basis) so that the new features are **uncorrelated**
2. Keep the dimension with the highest **variance** for DR



Principal Component Analysis

- PCA (@Hotelling:analysis33): a linear embedding
- Rotate to find *directions* in data with **maximum variance**
- How do we find these directions?
 - Diagonalize the **sample covariance (scatter) matrix** of N samples $\{\mathbf{x}^{(i)}, i = 1, \dots, N\}$

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}^{(i)} - \boldsymbol{\mu}) (\mathbf{x}^{(i)} - \boldsymbol{\mu})^\top$$

Principal Component Analysis

- Given data $\{\mathbf{x}^{(i)}\}$, PCA finds **orthogonal** directions defined by a projection (rotation) \mathbf{U} capturing the **maximum variance** in the data
- Solution: eigenvectors of $\mathbf{S} \rightarrow \mathbf{U}$
- PCA representation $\mathbf{y} = \mathbf{U}^\top \mathbf{x}$
- **Question:** Given the PCA representation \mathbf{y} , how to obtain an approximation/reconstruction of \mathbf{x} ?

$$\hat{\mathbf{x}} = \mathbf{U}\mathbf{y}$$

Representation & Reconstruction

- Face \mathbf{x} in k “face space” coordinates



$$\begin{aligned}\mathbf{x} &\rightarrow [\mathbf{u}_1^\top (\mathbf{x} - \boldsymbol{\mu}), \dots, \mathbf{u}_k^\top (\mathbf{x} - \boldsymbol{\mu})] \\ &= [w_1, \dots, w_k]\end{aligned}$$

- Reconstruction: eigenvectors as orthonormal basis vectors

$$\begin{matrix} \text{[Face Image]} & = & \text{[Mean Face]} & + & \text{[7 Eigenvectors]} \end{matrix}$$

$$\hat{\mathbf{x}} = \boldsymbol{\mu} + w_1 \mathbf{u}_1 + w_2 \mathbf{u}_2 + w_3 \mathbf{u}_3 + w_4 \mathbf{u}_4 + \dots$$

Reconstruction

$k = 4$



$k = 200$



$k = 400$



After computing eigenfaces using 400 face images from the ORL face database

PCA Ingredients

- **Data:** +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure/Architecture: linear projection $y = U^T x$
 - **Hyper-parameter:** lower dimension k
 - Parameters (theta): the principal components (eigenvectors)
- Evaluation metric: max variance
- Optimisation: eigen-decomposition

Week 8 Contents / Objectives

- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- **PCA Unboxing**
- Clustering: from k -means to spectral
- Autoencoder

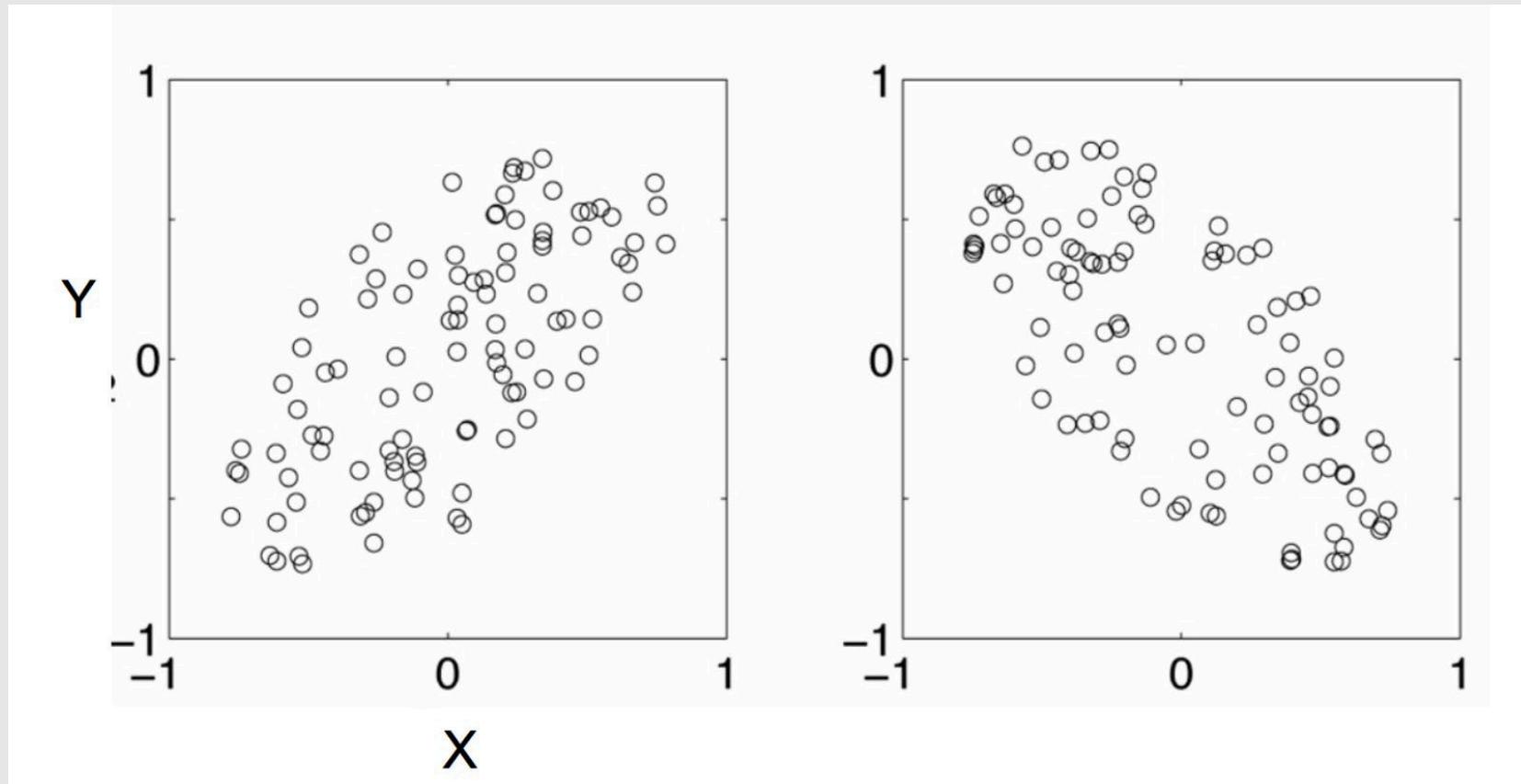
Variance → Covariance (Matrix)

- Variance and covariance:
 - Measure of the “spread” of a set of points around their *center of mass* (mean)
- Variance (scalar):
 - Measure of the deviation from the mean for points in **one dimension**
- Covariance (matrix):
 - Measure of how much each of the dimensions vary from the mean with **respect to each other**



- Covariance is measured between two dimensions
- Covariance sees if there is a **relation** between the two dimensions
- Covariance between one dimension is the variance (degeneration)

Positive/Negative Covariance



Positive: Both dimensions increase or decrease together

Negative: While one increase the other decrease

Covariance

- Find relationships between dimensions in high dimensional data sets \mathbf{X}
- Scatter matrix \mathbf{S} : sample-based estimation of covariance matrix (i : sample; j/k : variable)

$$s_{jk} = \frac{1}{N} \sum_{i=1}^N (X_{ij} - E(X_j))(X_{ik} - E(X_k))$$


The Sample mean

- Uncorrelated variables \rightarrow covariance = 0
- Diagonal covariance mat \rightarrow all variables are uncorrelated

PCA Derivation – Max Variance

- Scatter mat for the input $\mathbf{S} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}^{(i)} - \boldsymbol{\mu}) (\mathbf{x}^{(i)} - \boldsymbol{\mu})^\top$
- **Question:** what is the scatter mat for the projections?

$$\mathbf{U}^\top \mathbf{S} \mathbf{U}$$

- First PC: **maximise the variance in the projected space**, i.e. the variance of $y = \mathbf{u}_1^\top \mathbf{x}$

$$\begin{aligned}\text{var}(y) &= \frac{1}{N} \sum_{i=1}^N (y^{(i)} - \mu_y)^2 \\ &= \frac{1}{N} \sum_{i=1}^N \mathbf{u}_1^\top (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{\mathbf{x}}) (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{\mathbf{x}})^\top \mathbf{u}_1 \\ &= \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1\end{aligned}$$

PCA: Max Variance → Eigenvalue

- Find the first direction \mathbf{u}_1 via a unit-norm constrained optimisation, using Lagrange multipliers:

$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^\top \mathbf{u}_1)$$

- Gradient w.r.t. \mathbf{u}_1 : $\frac{dL(\mathbf{u}_1, \lambda_1)}{d\mathbf{u}_1} = 2\mathbf{S}\mathbf{u}_1 - 2\lambda_1\mathbf{u}_1$

- Set to 0 and rearrange: $\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \rightarrow$ First PC

- Eigenvalue problem

- **Question:** many solutions (eigen-pairs), which to choose?

$$\text{var}(y) = \mathbf{u}^\top \mathbf{S} \mathbf{u} = \lambda \mathbf{u}^\top \mathbf{u} = \lambda$$

PCA Solution

- Further directions: **orthogonal** (uncorrelated) to the first and each others → top k eigenvectors of \mathbf{S}
 - Eigenvectors: basis functions, principal components
 - Eigenvalue: the **variance** captured respectively
- **Questions**
 - For \mathbf{u}_1 , what if we do not have the unit-norm constraint?
$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \cancel{\lambda_1 (1 - \mathbf{u}_1^\top \mathbf{u}_1)}$$
→ Trivial solution: infinity
 - For further directions: what if we do not require them to be orthogonal?
→ We will have the same \mathbf{u}_1 , useless solution

PCA: Max Variance \leftrightarrow Min MSE

Consider the first PC with the projection vector \mathbf{u} . We assume **zero-mean**, i.e. *centered* data

Maximum Variance Direction: 1st PC = a vector \mathbf{u} such that projection on it captures max variance in the data

$$\frac{1}{N} \sum_{i=1}^N (\mathbf{u}^T \mathbf{x}^{(i)})^2 = \mathbf{u}^T \mathbf{X} \mathbf{X}^T \mathbf{u}$$

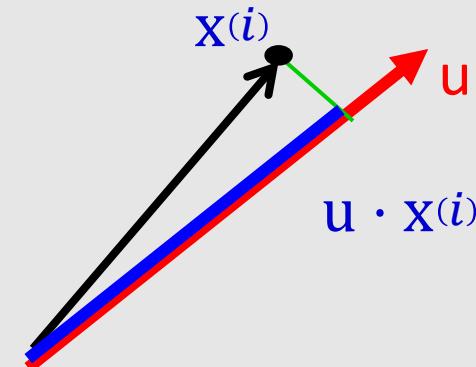
Minimum Reconstruction Error: 1st PC = a vector \mathbf{u} such that projection on it yields minimum MSE reconstruction

$$\frac{1}{N} \sum_{i=1}^N \|\mathbf{x}^{(i)} - (\mathbf{u}^T \mathbf{x}^{(i)}) \mathbf{u}\|^2$$

$$\text{blue}^2 + \text{green}^2 = \text{black}^2$$

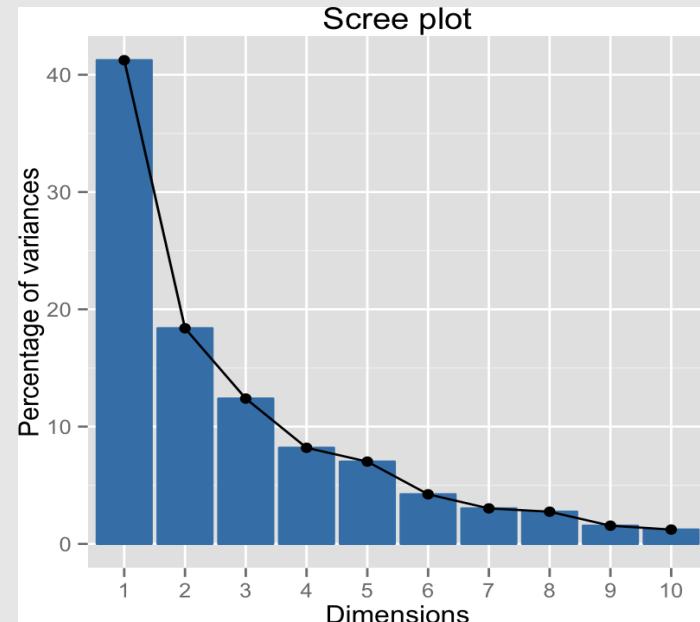
black² is fixed (it's just the data)

So, maximizing blue² is equivalent to minimizing green²



How many (k) PCs to keep?

- Pick based on percentage of variance captured / lost
 - Variance captured: the variance of the projected data
 - Pick smallest k that explains a certain percentage of variance
$$(\text{Sum of first } k \text{ EVs}) / (\text{sum of all EVs})$$
- Look for an “elbow” in [scree plot](#) (plot of explained variance or eigenvalues)

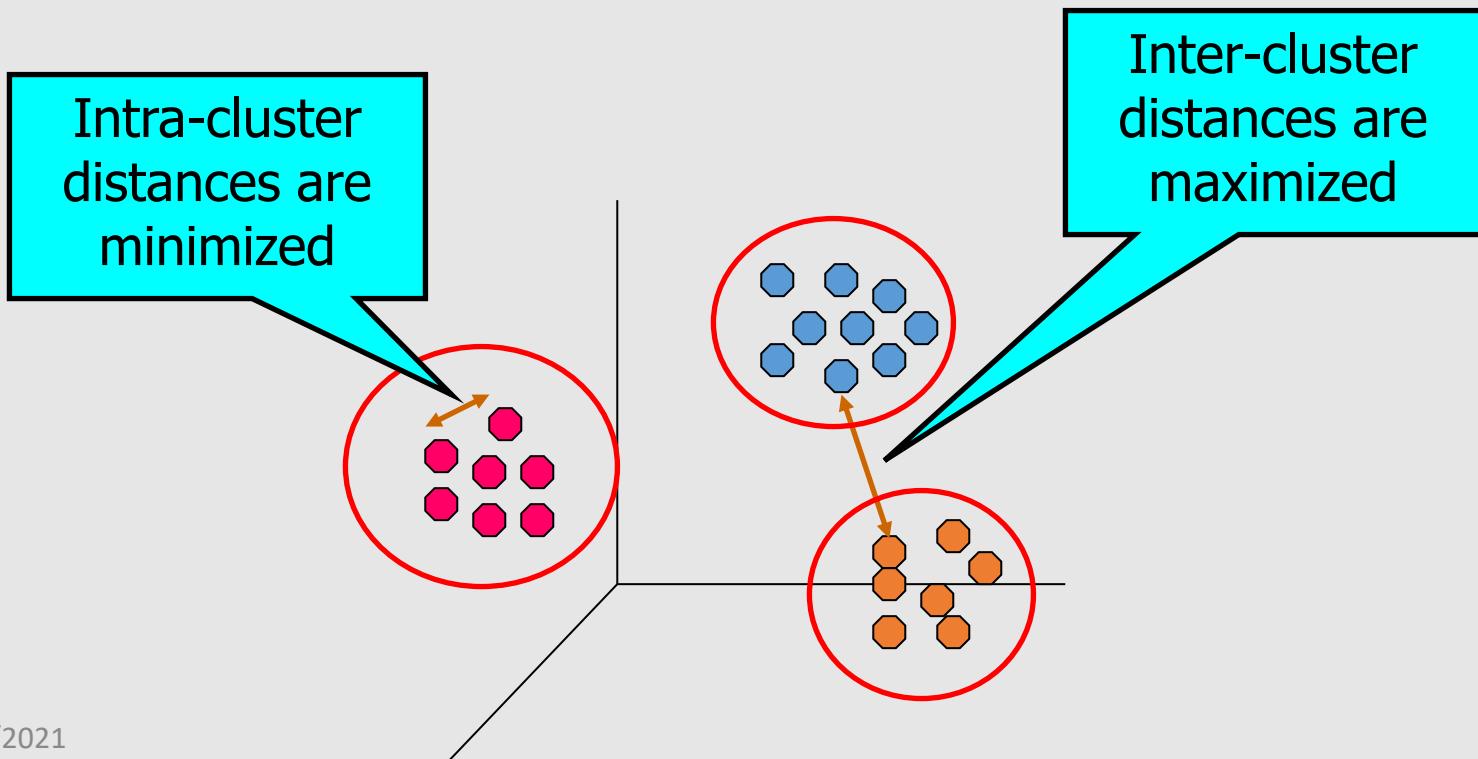


Week 8 Contents / Objectives

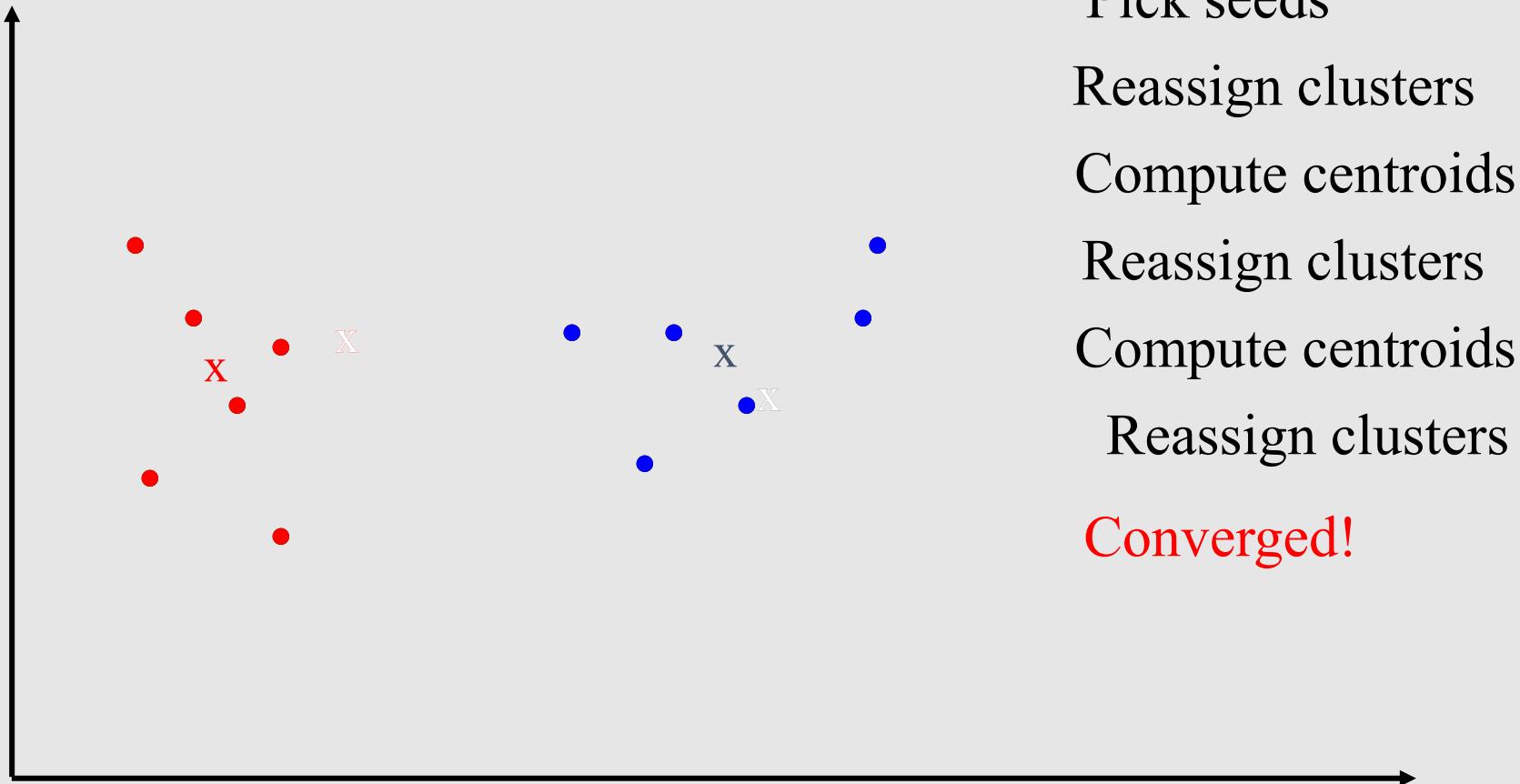
- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- PCA Unboxing
- **Clustering: from k -means to spectral**
- Autoencoder

What is Clustering?

- Find grouping of objects such that the objects in a group will be similar or more related to one another and those in different groups will be different or less related



k -means Example ($k=2$)



k -means & PCA

Machine Learning	Supervised	Unsupervised
Discrete output	Classification	Clustering
Continuous output	Regression	Dimensionality Reduction

- The objective of k -means clustering: to minimize the within-cluster scatter

$$\min \sum_{j=1}^k \sum_{i \text{ allocated to } j} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right)^\top \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right)$$

- Similarity with PCA: also measure of the “spread” of a set of points around their *center of mass* (mean)
- Clustering analogy
 - Classification w/o labelled training data
 - Extreme dimensionality reduction (to a **cluster label**)

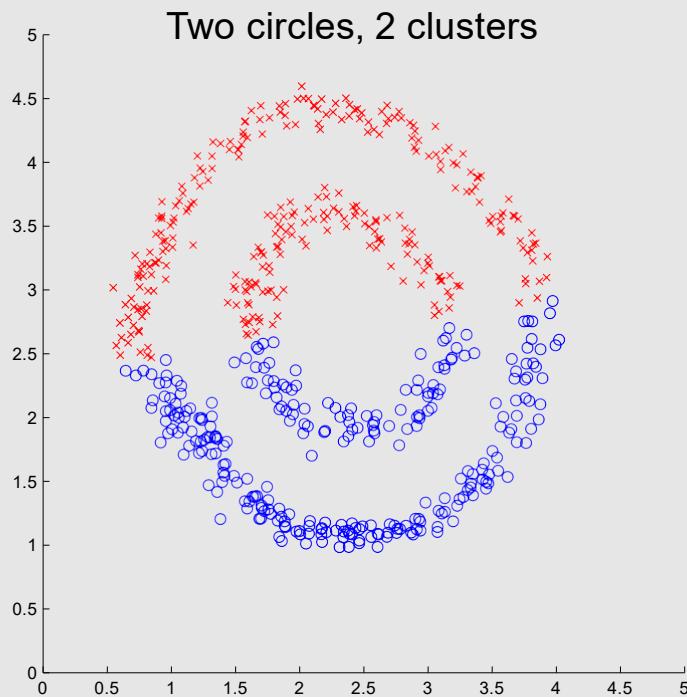
k -means Clustering Ingredients

- **Data:** +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure/Architecture: linear separation btw clusters
 - **Hyper-parameter:** #clusters k
 - Parameters (theta): the k cluster centroids
- Evaluation metric: within-cluster scatter
- Optimisation: expectation maximisation (EM), kind of gradient descent

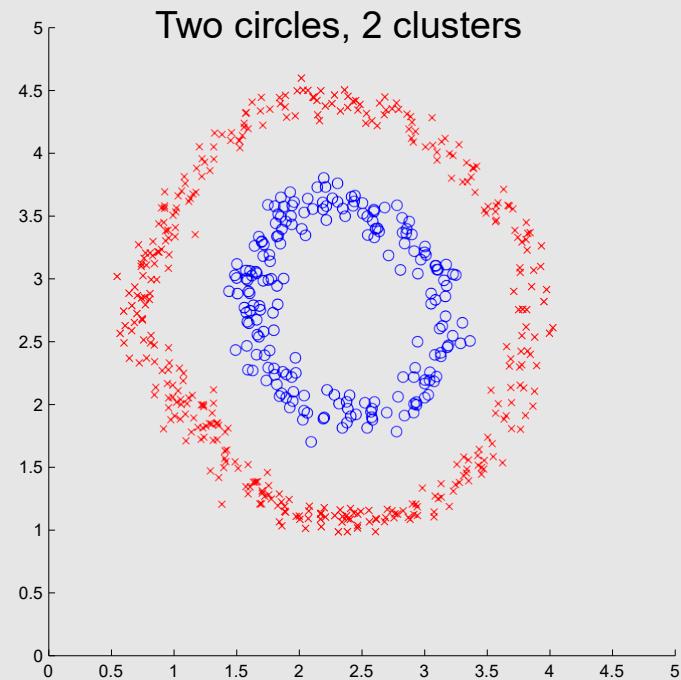
k -means Has a Problem

- k -means: centroid-based, for compactness (scatter)
- Spectral clustering: **connectivity**

k -means

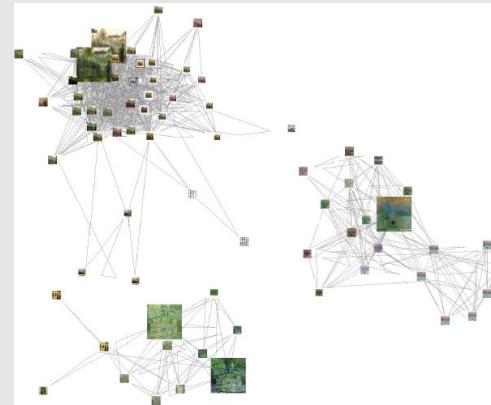
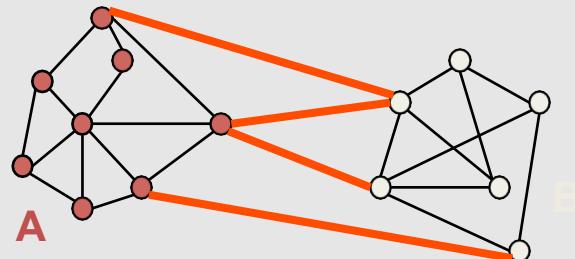


Spectral clustering



Spectral Clustering

- Group points based on **links** in a **graph**

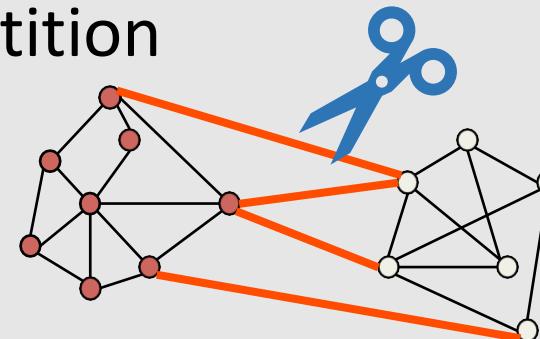


- Image as graph: segmentation as clustering



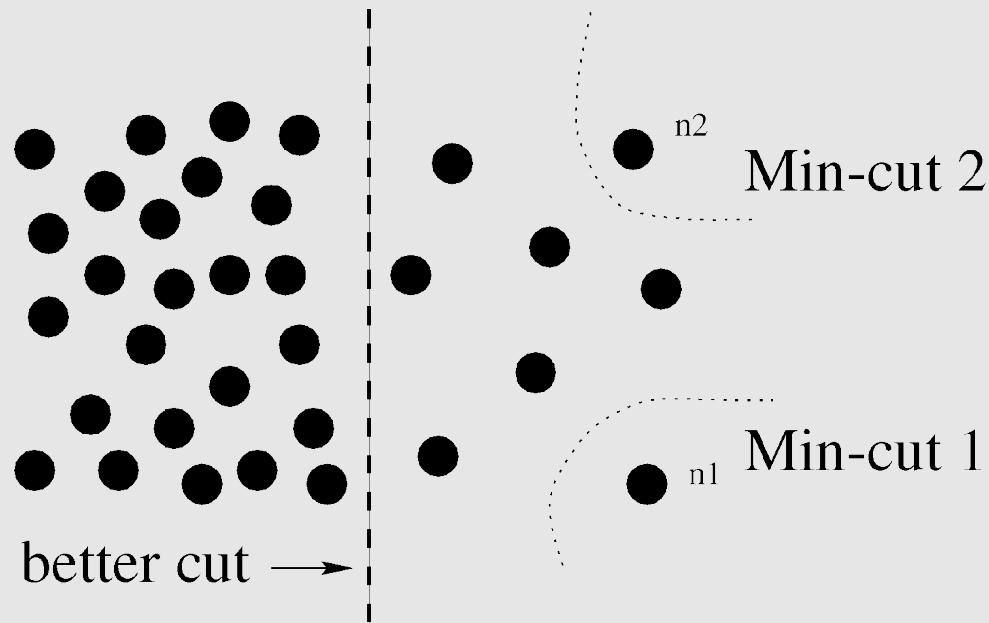
How to Create the Graph?

- Gaussian kernel → compute similarity between objects
$$\mathbf{W}(i, j) = \exp \frac{-|x_i - x_j|^2}{\sigma^2}$$
- One could create
 - A fully connected graph (~ FC layer)
 - K-nearest neighbour graph: each node is only connected to its K-nearest neighbours (~ convolutional layer, local connectivity)
- Clustering → Graph cut/partition
 - Objective: **minimise** cut



Min Cut = Good Cut?

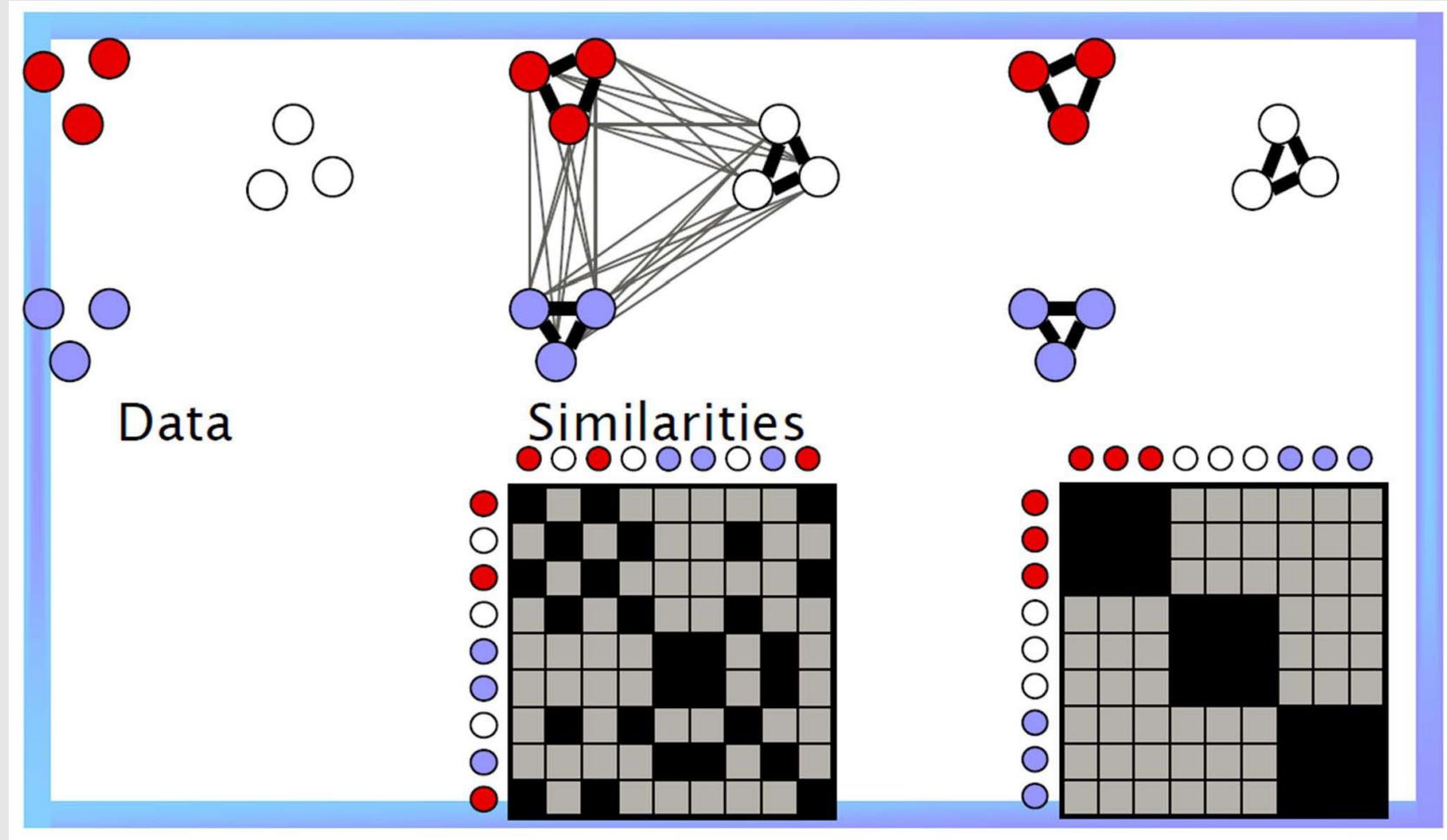
- A case where minimum cut gives a bad partition



- Solution: **Normalise** the cut

[Shi & Malik '00]

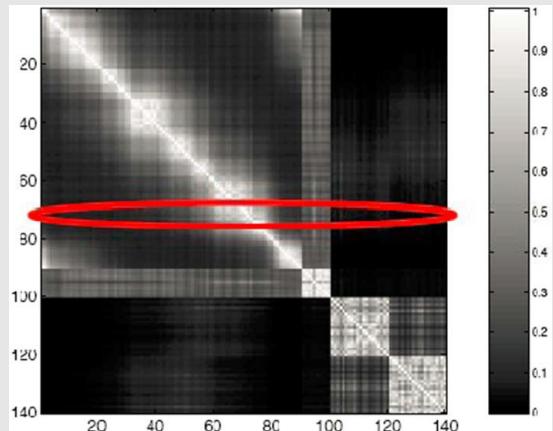
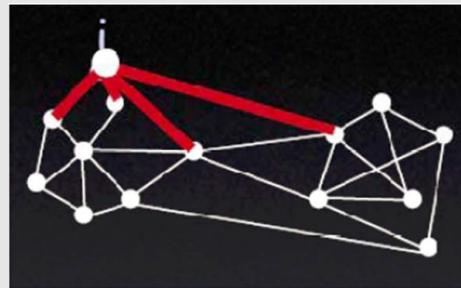
Graph Clustering Process



Graph Terminologies

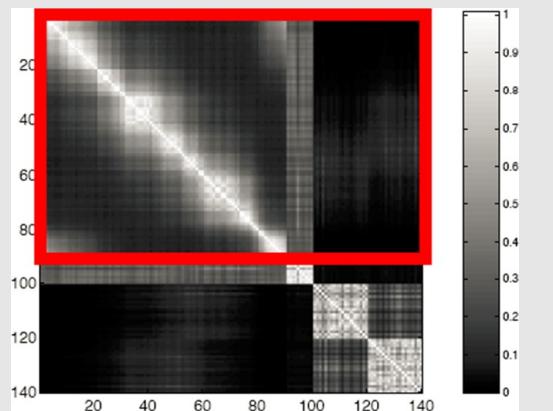
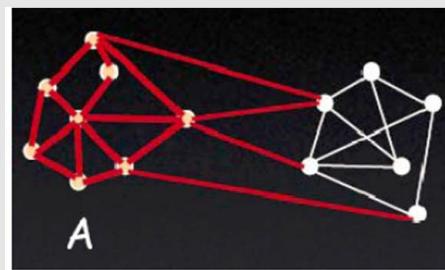
- Degree of nodes

$$d_i = \sum_j w_{i,j}$$



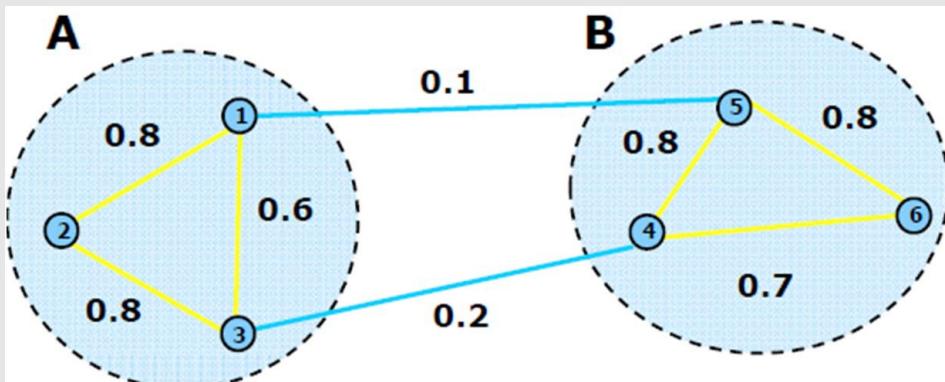
- Volume of a set

$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$



Graph Cut

- Consider a partition of the graph into two parts A & B



Question

$$\text{cut}(A, B) =$$

- $\text{cut}(A, B)$: sum of the weights of the set of edges that connect the two groups

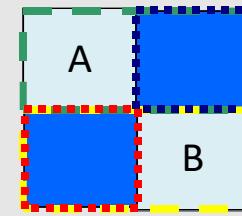
$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

- Goal: find the partition that minimizes the cut

Normalized Cut (Ncut)

- Consider the **connectivity** between groups **relative** to the volume of each group

$$Ncut(A, B) = \frac{cut(A, B)}{Vol(A)} + \frac{cut(A, B)}{Vol(B)}$$



$$Ncut(A, B) = cut(A, B) \frac{Vol(A) + Vol(B)}{Vol(A)Vol(B)}$$

Solving/Minimising Ncut

- Compute the similarity matrix \mathbf{W} : $\mathbf{W}(i, j) = w_{i,j}$
- Compute the degree matrix \mathbf{D} : $\mathbf{D}(i, i) = \sum_j w_{i,j}$
- Solve a generalised **eigenvalue** problem (relaxed Ncut)

$$\min_{\mathbf{y}} \mathbf{y}^\top (\mathbf{D} - \mathbf{W}) \mathbf{y} \text{ s.t. } \mathbf{y}^\top \mathbf{D} \mathbf{y} = 1 \Rightarrow (\mathbf{D} - \mathbf{W}) \mathbf{y} = \lambda \mathbf{D} \mathbf{y}$$

$(\mathbf{D} - \mathbf{W})$: Laplacian matrix

- Solution:
 - Bipartition: use the eigenvector with the second smallest eigenvalue to partition the graph into two parts
 - Splitting point: minimum Ncut (plot).
 - K-way partition: **k -means clustering** of multiple eigenvectors
 - Graph embedding (dimensionality reduction) → eigenvectors

Recap: Power of Transform

- Logistic regression **transforms** classification into linear regression of the log odds, modelling the probability of the predicted output rather than the output itself.
- Spectral clustering **transforms** non-linear clustering into (linear) k -means clustering of generalised eigenvectors based on the similarity graph, modelling the connectivity of data points rather than themselves.



Spectral Clustering Ingredients

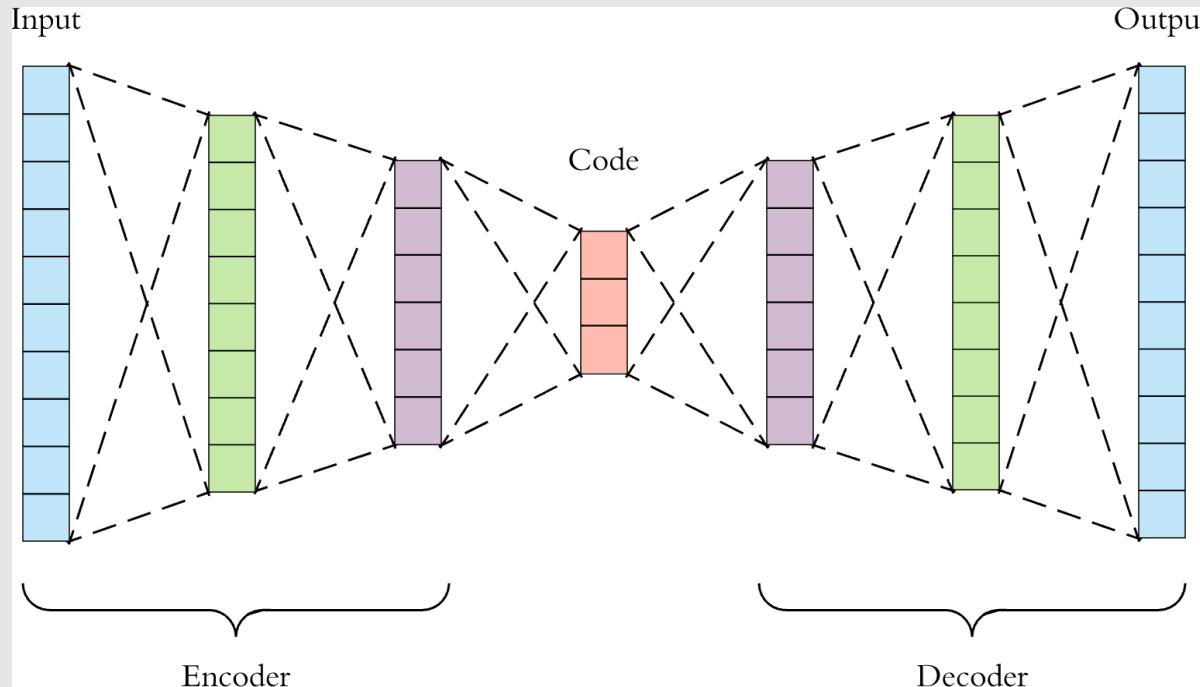
- **Data:** +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure/Architecture: spectral (nonlinear) separation btw clusters
 - **Hyper-parameter:** #clusters k (+#eigenvectors), kernel bandwidth σ
 - Parameters (theta): the (generalised) eigenvectors
- Evaluation metric: normalised (graph) cut
- Optimisation: eigen-decomposition (and expectation maximisation in k -means)

Week 8 Contents / Objectives

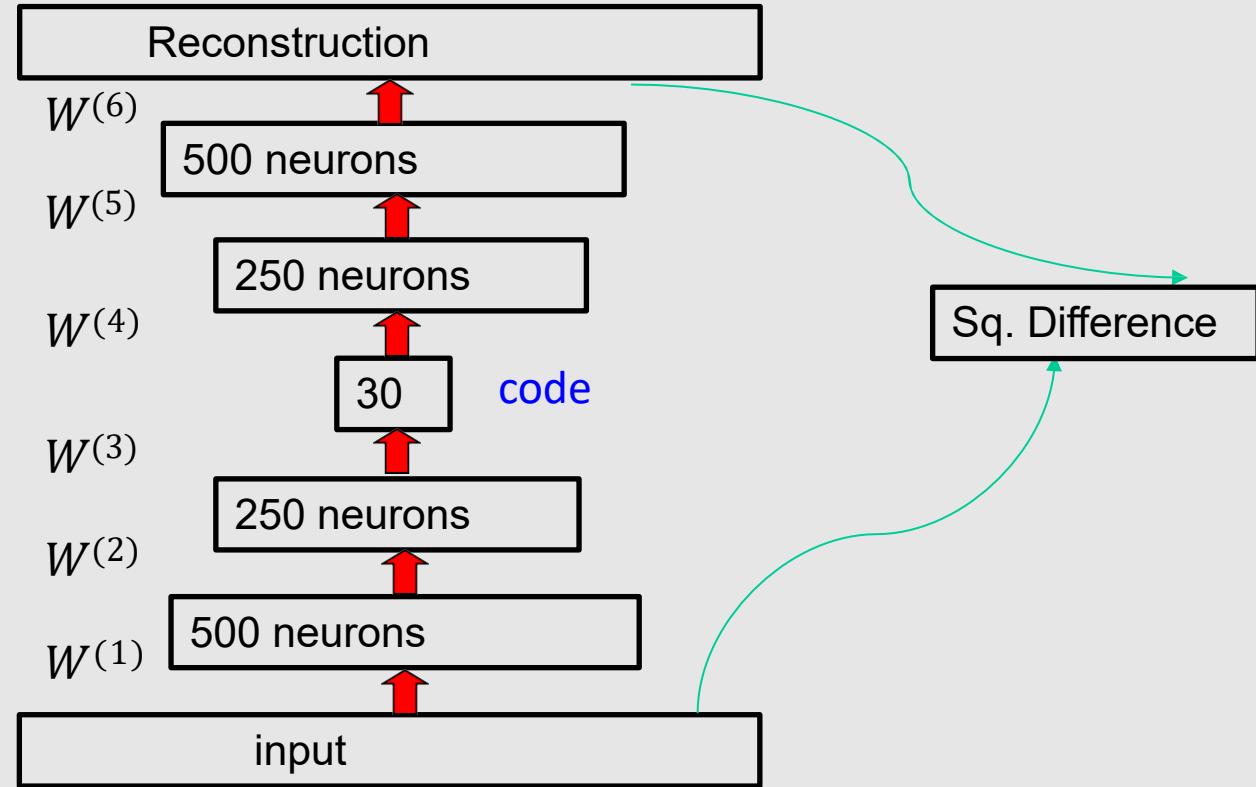
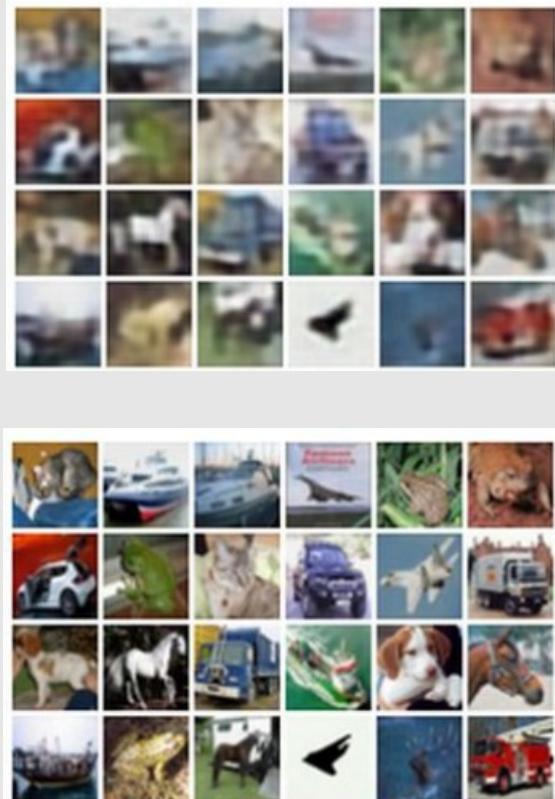
- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- PCA Unboxing
- Clustering: from k -means to spectral
- **Autoencoder**

Autoencoder

- **Encoder:** compress data or extract features
- **Decoder:** generate images given a new code
- **Bottleneck (code):** to make it non-trivial, much smaller dimension as the latent representation



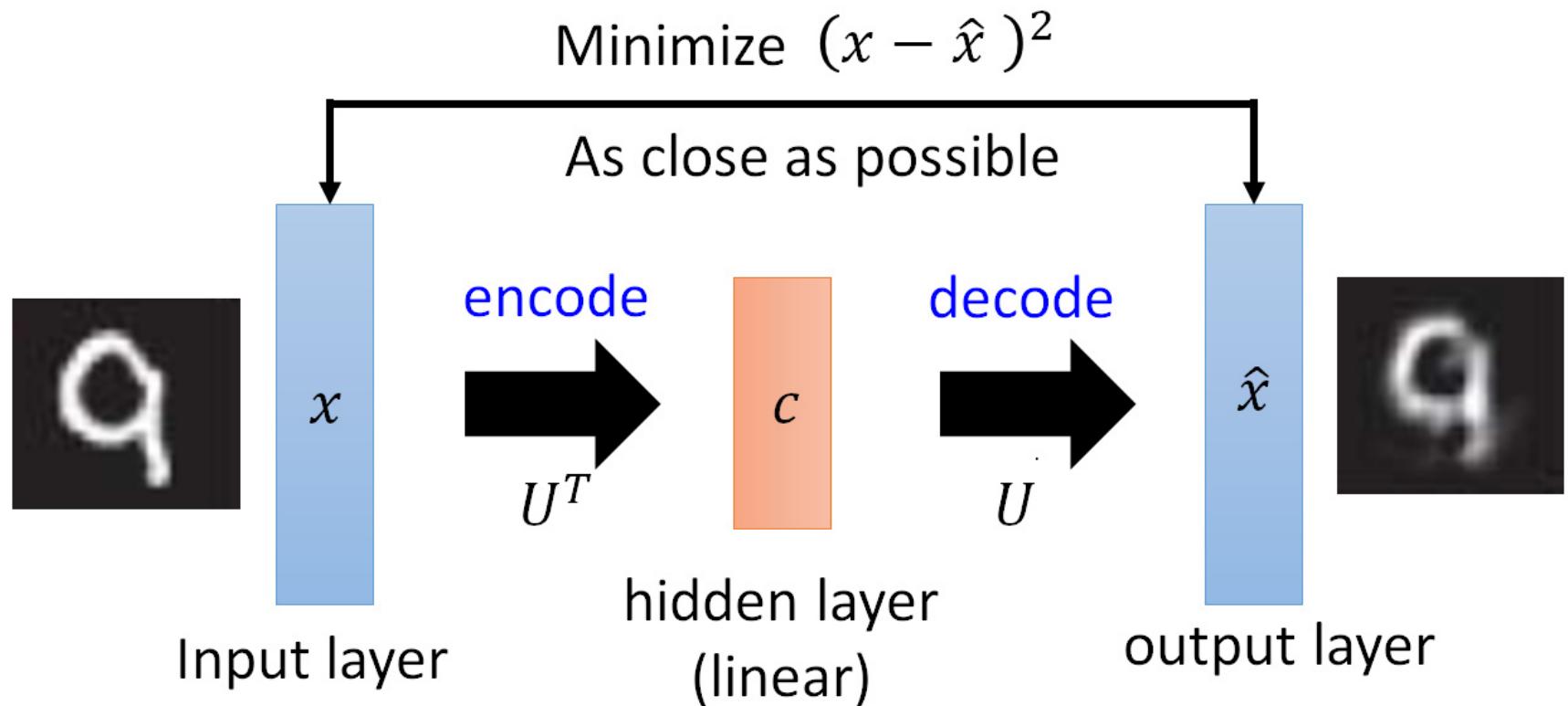
Autoencoder Example



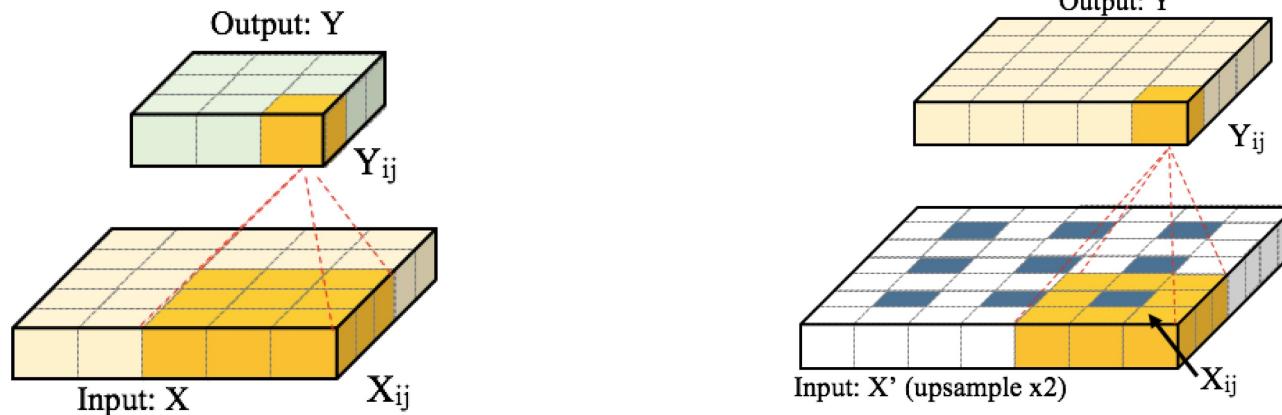
- Find the weights minimising the difference between the input and reconstruction
- The **code** layer (bottleneck) is a low-dimensional summary of the input

PCA as Linear Autoencoder

- PCA: autoencoder w/t single-layer encoder/decoder
- Weight sharing between the encoder and decoder



Transpose Convolution Layer



(a) Convolutional layer: the input size is $W_1 = H_1 = 5$; the receptive field $F = 3$; the convolution is performed with stride $S = 1$ and no padding ($P = 0$). The output Y is of size $W_2 = H_2 = 3$.

(b) Transposed convolutional layer: input size $W_1 = H_1 = 3$; transposed convolution with stride $S = 2$; padding with $P = 1$; and a receptive field of $F = 3$. The output Y is of size $W_2 = H_2 = 5$.

<https://www.mdpi.com/2072-4292/9/6/522/htm>

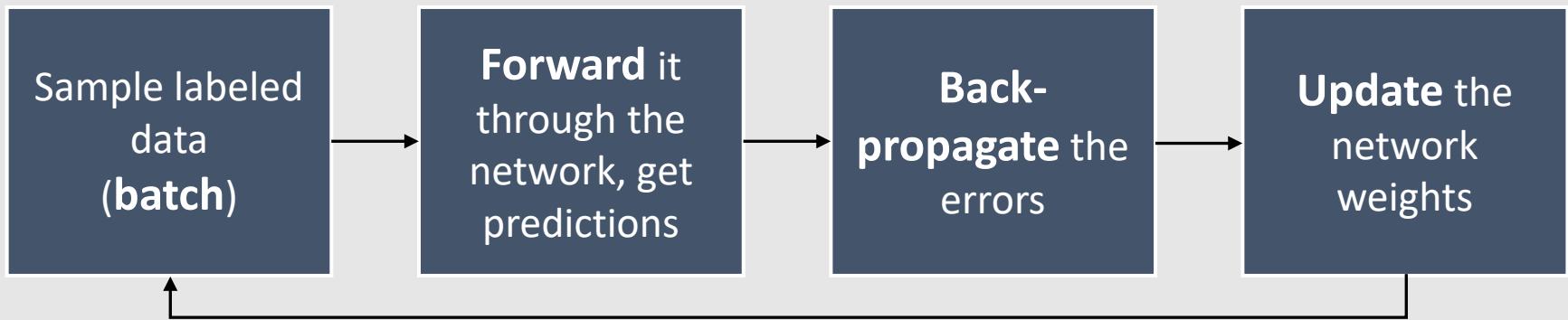
More at https://github.com/vdumoulin/conv_arithmetic

Convolutional Autoencoder (Lab)

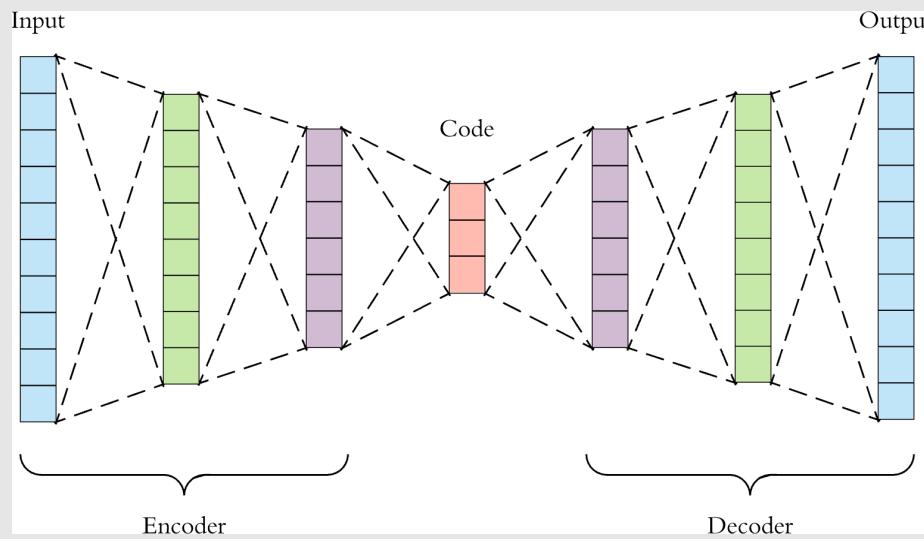
```
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            # 1 input image channel, 16 output channel, 3x3 square convolution
            nn.Conv2d(1, 16, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(16, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, 7)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, 7),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid() #to range [0, 1]
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Training



Data → Model → Metric → Optimisation



Autoencoder Ingredients

- **Data:** + pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
 - Structure/Architecture: layers defined in nn.module
 - **Hyper-parameter:** layer specs, e.g. #layers #channels, kernel size
 - Parameters (theta): layer weights and biases
- Evaluation metric: MSE or other
- Optimisation: backprop, SGD or the like

Acknowledgement

- The slides used materials from:
*Lisa Zhang, Michael Guerzhoy,
Neil Lawrence, Derek Hoiem,
Pandu Nayak, Prabhakar
Raghavan, Fereshteh Sadeghi,
Aarti Singh, David Sontag, James
Hays, Alan Fern, Tommi
Jaakkola, Jure Leskovec*

Recommended Reading

- The [PCA book](#) (from a UIUC link)
 - Chapter on PCA in most machine learning books
 - Chapter on clustering in most machine learning books
 - The [normalized cut paper](#) in 2000
 - Chapter on autoencoder in [the Deep Learning Book](#)
-
- Wikipedia entries on covered topics
 - Scikit-learn/PyTorch documentations
 - The lab notebook and references

Lecture 9

Generative Models

[Haiping Lu](#)

YouTube Playlist:

<https://www.youtube.com/c/HaipingLu/playlists>

[COM4059/6059: MLAI21@The University of Sheffield](#)

Week 9 Contents / Objectives

- Why Generative Models?
- Bayesian Inference
- Bayesian Linear Regression
- Variational Autoencoder (VAE)
- VAE Unboxing

Week 9 Contents / Objectives

- **Why Generative Models?**
- Bayesian Inference
- Bayesian Linear Regression
- Variational Autoencoder (VAE)
- VAE Unboxing

**“What I cannot create,
I do not understand.”**

- Richard Feynman

https://pbs.twimg.com/media/DmK_xgyXsAA2mo8.jpg

The **holy grail** in ML: understand data → create data

Generating Faces (VAE)



<https://www.youtube.com/watch?v=XNZIN7Jh3Sg>

Digital Generative Art (VAE)



<https://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/>

Generating Images (GAN)



<https://www.youtube.com/watch?v=XOxxPcy5Gr4>

DeepFakes

Which image is real?



DeepFakes

Neither!

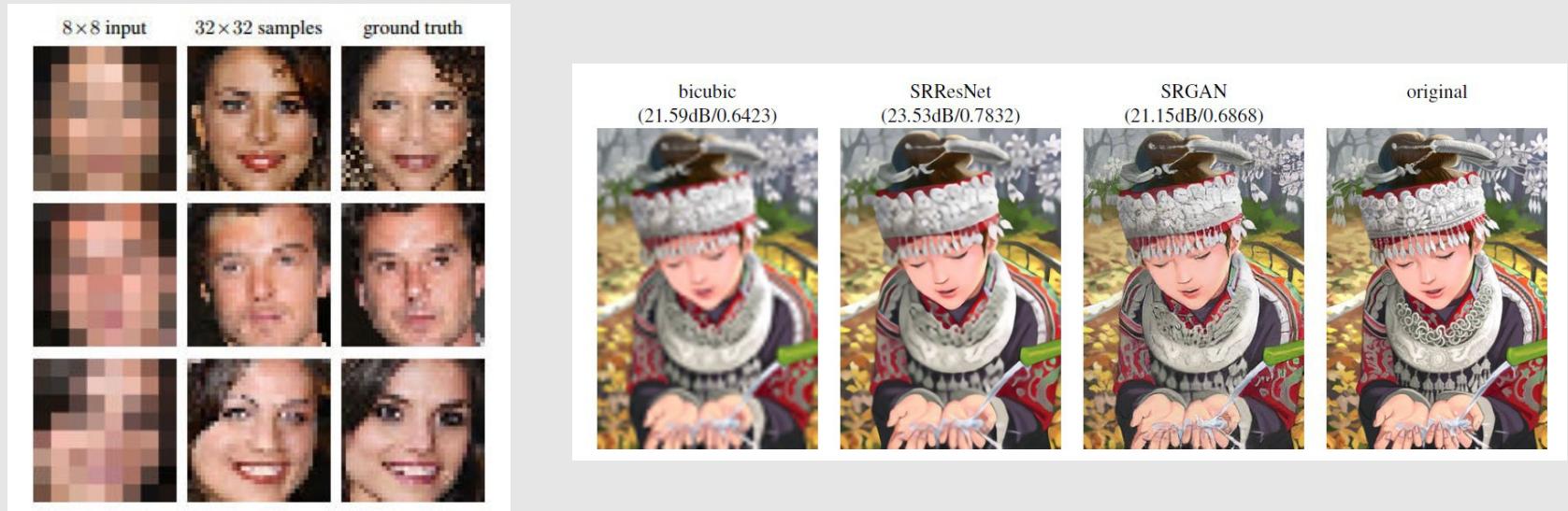


No glasses!

No smile!

Image Super Resolution

- Conditional generative model
 $P(\text{high res image} \mid \text{low res image})$



Ledig et al., 2017

Image Translation / Colorization

- Conditional generative model
 $P(\text{ zebra images} | \text{ horse images})$



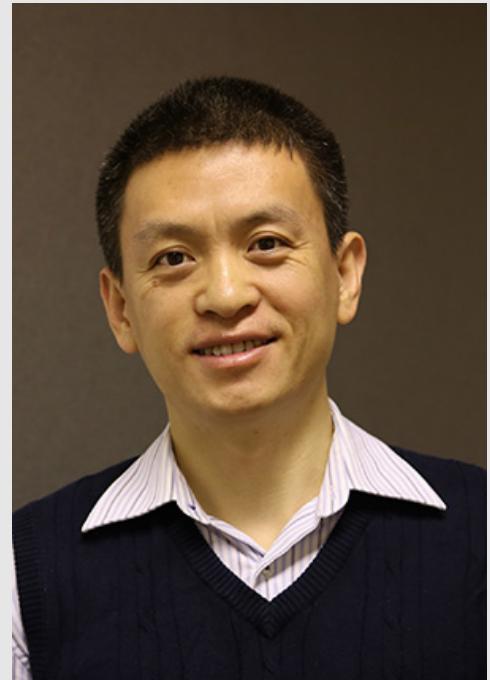
Zhu et al., 2017

Week 9 Contents / Objectives

- Why Generative Models?
- **Bayesian Inference**
- Bayesian Linear Regression
- Variational Autoencoder (VAE)
- VAE Unboxing

Question

- Which year was this photo taken?
 - A. 1996
 - B. 2006
 - C. 2016
 - D. 2026

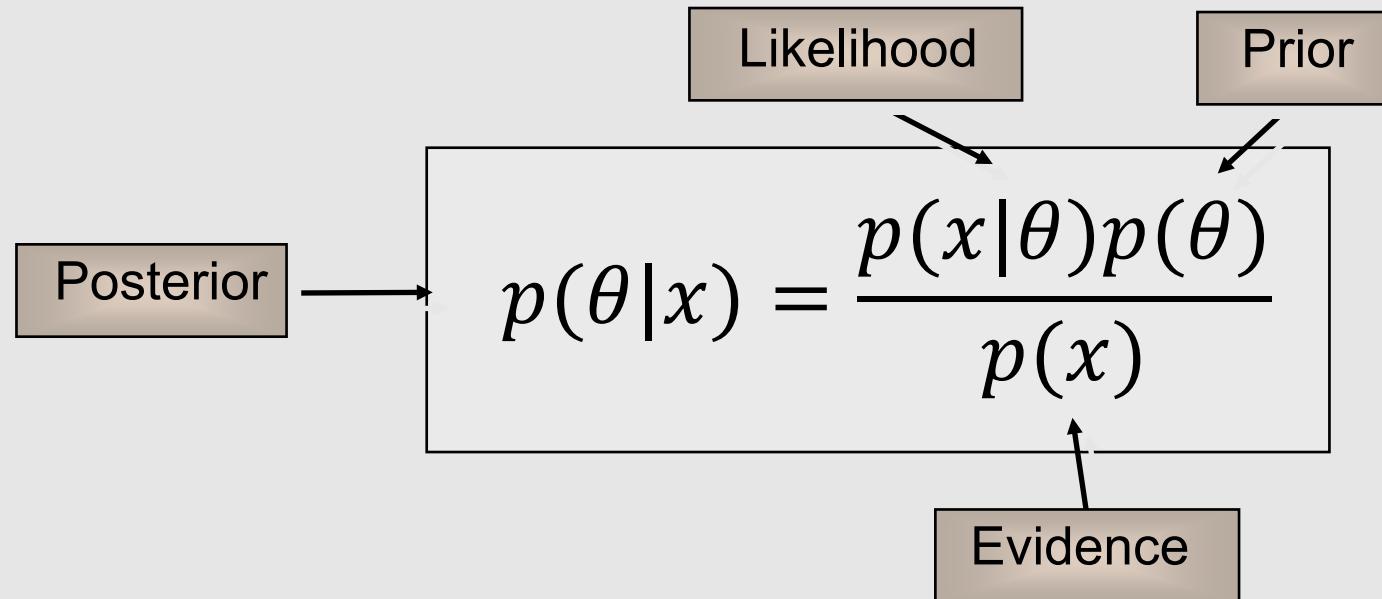


Bayes' Rule

Given data x and parameters θ , their joint probability can be written as

$$p(\theta|x)p(x) = p(x, \theta) \quad p(x, \theta) = p(x|\theta)p(\theta)$$

Eliminating $p(x, \theta)$ gives Bayes' rule:



Key Concepts

- **Prior** probability: the estimate of the probability of the model **before** the data (evidence) is observed
- **Posterior** probability: the probability of the model **after** observing the data (evidence)
- **Likelihood**: the probability of observing a (random) data point given a model (*fixed*) → the **compatibility** of the data (evidence) with the given model
- **Marginal likelihood**: "model **evidence**", the probability of observing a (random) data point under all possible model variations

Principles of Bayesian Inference

- ⇒ Formulation of a generative model

likelihood $p(x|\theta)$

prior distribution $p(\theta)$

- ⇒ Observation of data

x

- ⇒ Update of beliefs based upon observations, given a prior state of knowledge

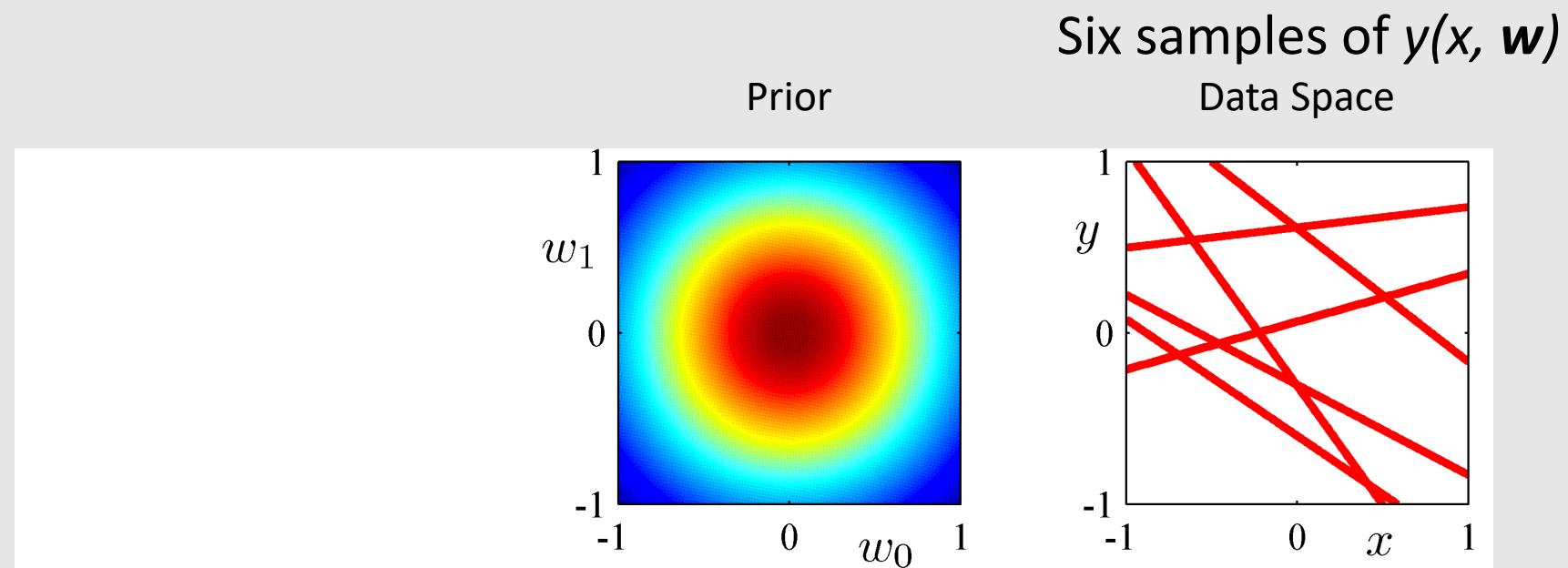
$$p(\theta|x) \propto p(x|\theta)p(\theta)$$

Week 9 Contents / Objectives

- Why Generative Models?
- Bayesian Inference
- **Bayesian Linear Regression**
- Variational Autoencoder (VAE)
- VAE Unboxing

Bayesian Linear Regression (1)

Aim: Estimate model parameters w_0 & w_1



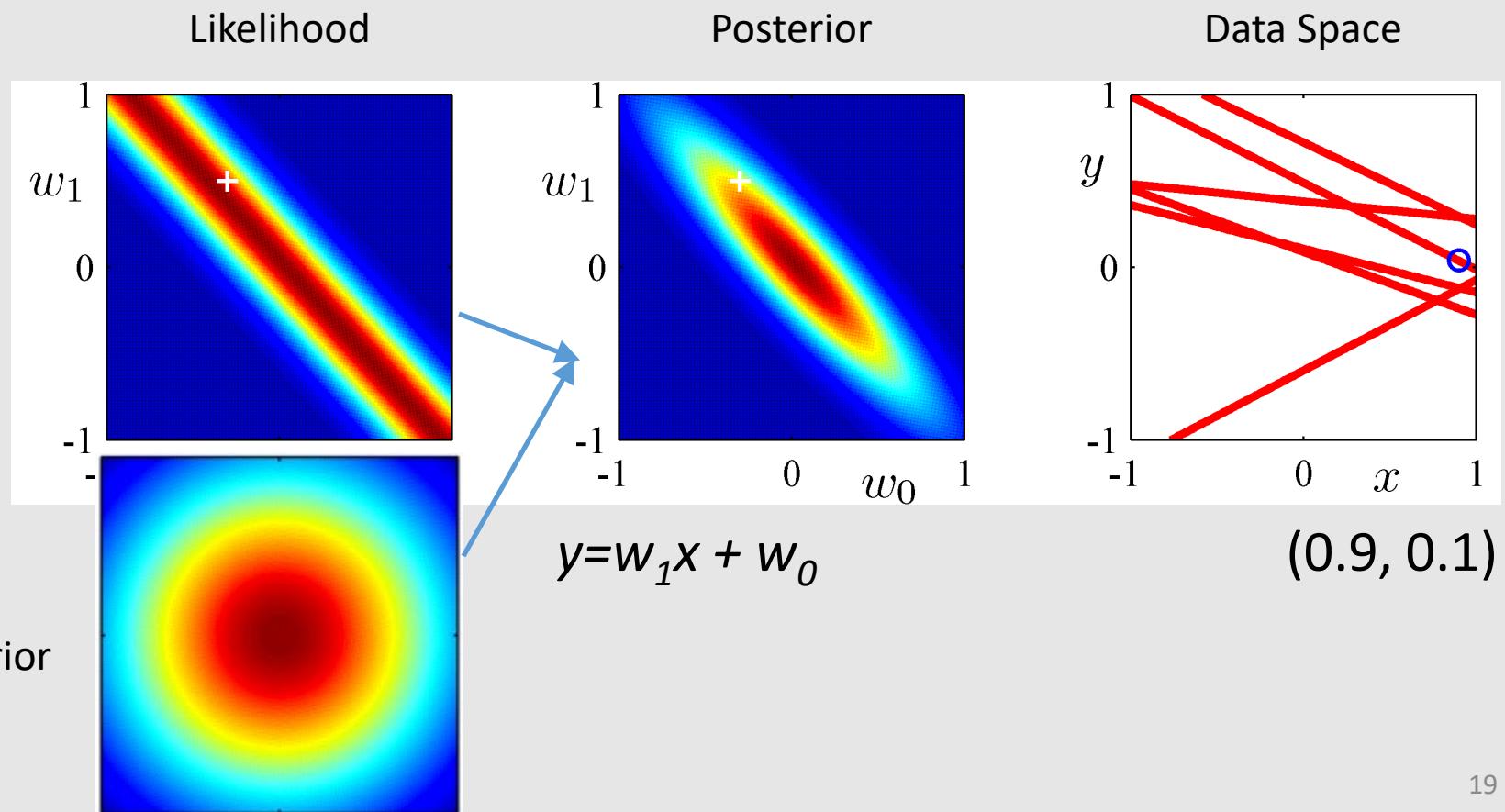
Bayesian inference: placing a probability distribution (prior density) over the model parameters w_0 & w_1

Now: No data points are observed.

Bayesian Linear Regression (2)

1 data point observed → soft constraint.

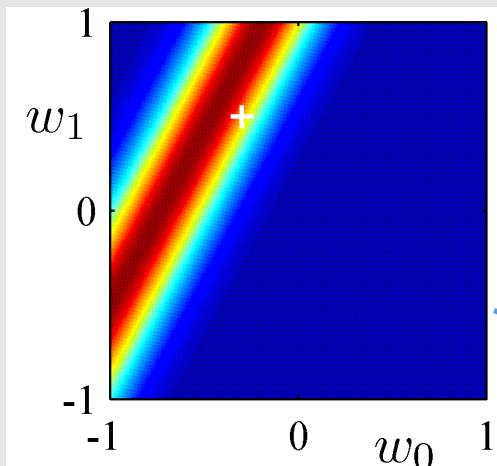
This posterior → prior for the next data point observed



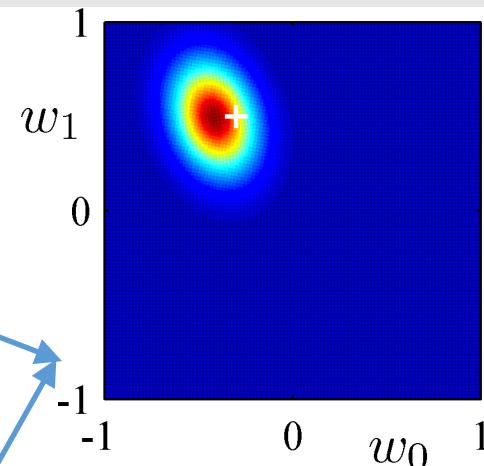
Bayesian Linear Regression (3)

A second data point observed

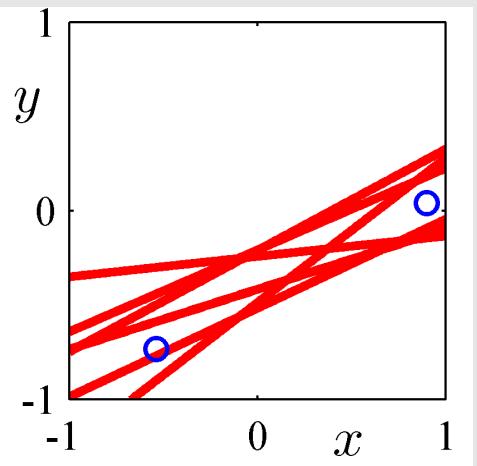
Likelihood of 2nd pt



Posterior



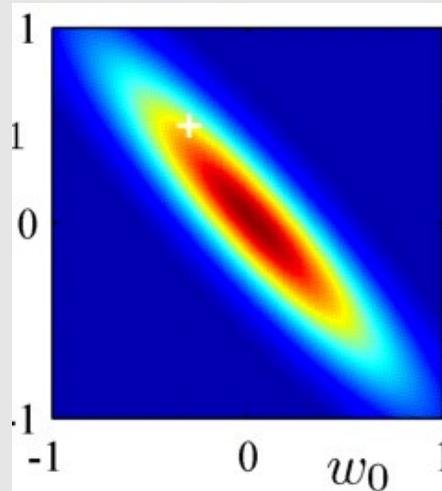
Data Space



$$y = w_1 x + w_0$$

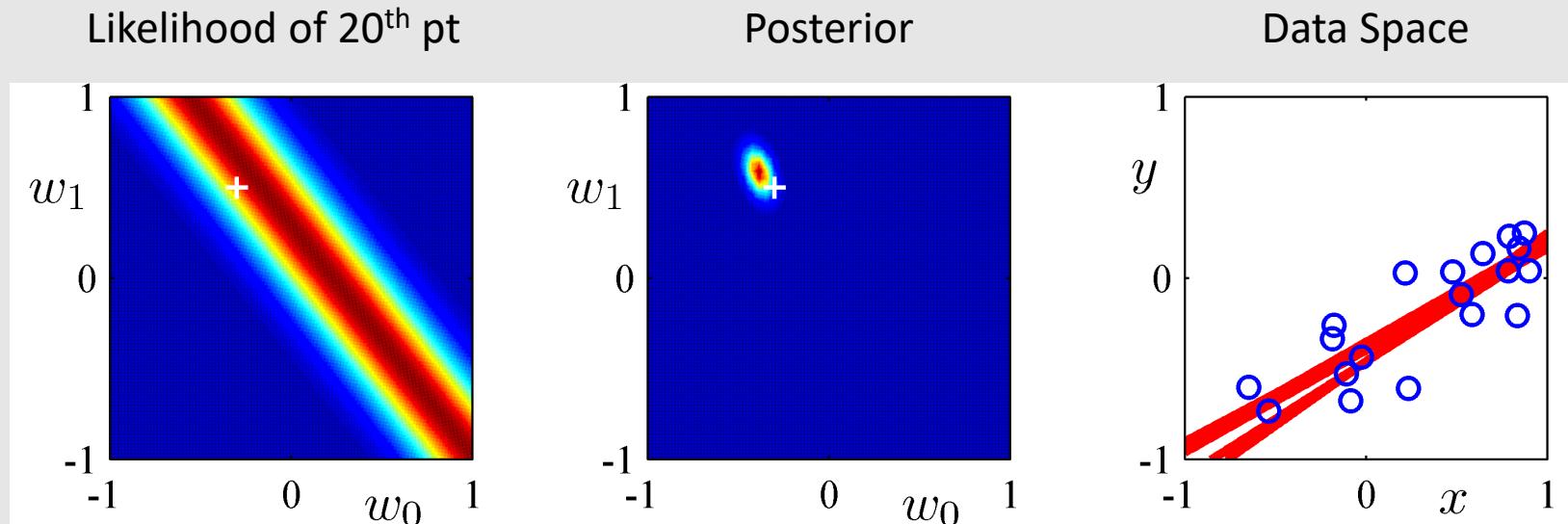
$(-0.7, -0.8)$

Current
Prior =
Previous
posterior



Bayesian Linear Regression (4)

20 data points → very close to true values of w_0 & w_1

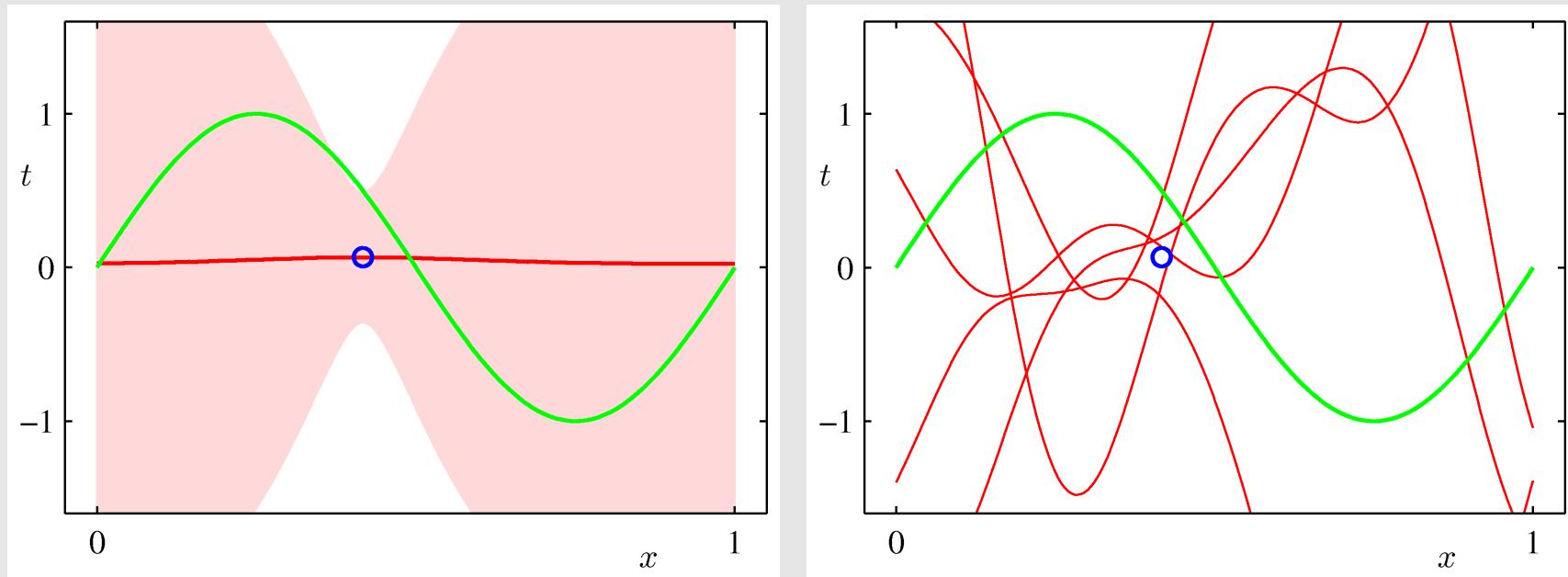


How about making **probabilistic** predictions for any x ?

Bayesian inference: Evaluate the predictive **distribution**

Predictive Distribution (1)

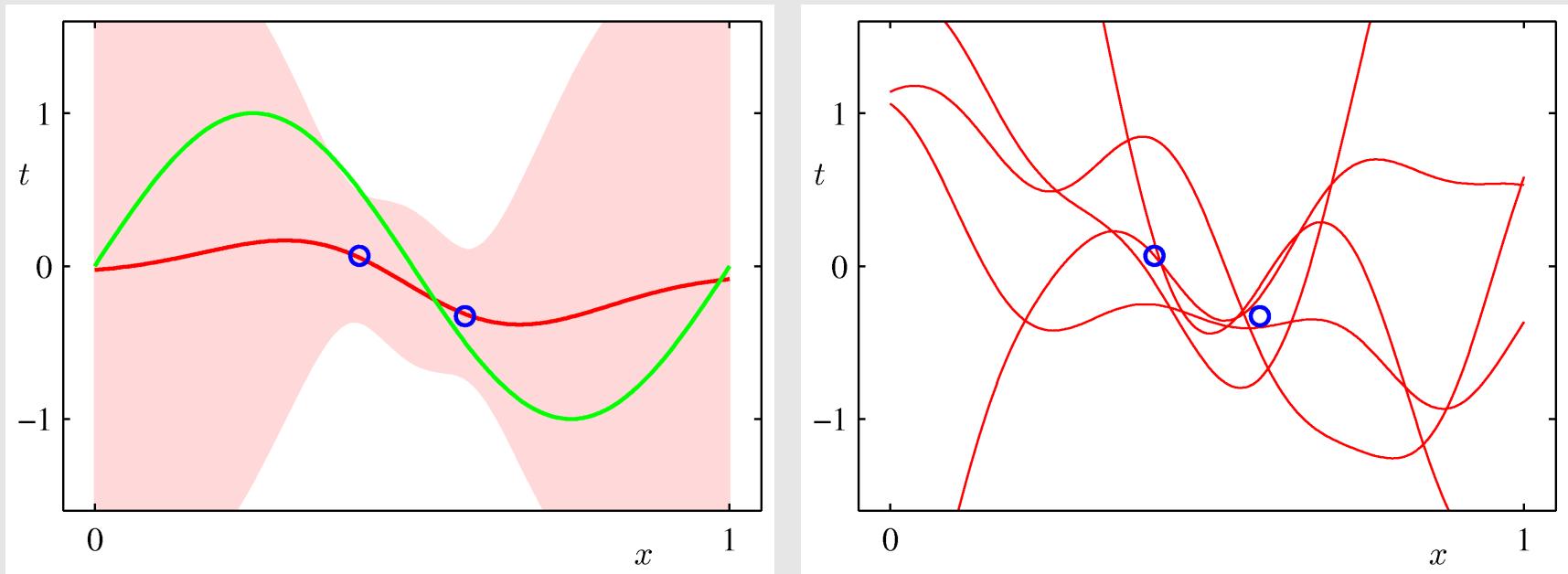
- Data: green curve + noise → sinusoidal data (blue circles)
- Model: 9 Gaussian basis functions



- Aim: Predict the output distribution
- Now: 1 data point. Red: model; shade: model uncertainty

Predictive Distribution (2)

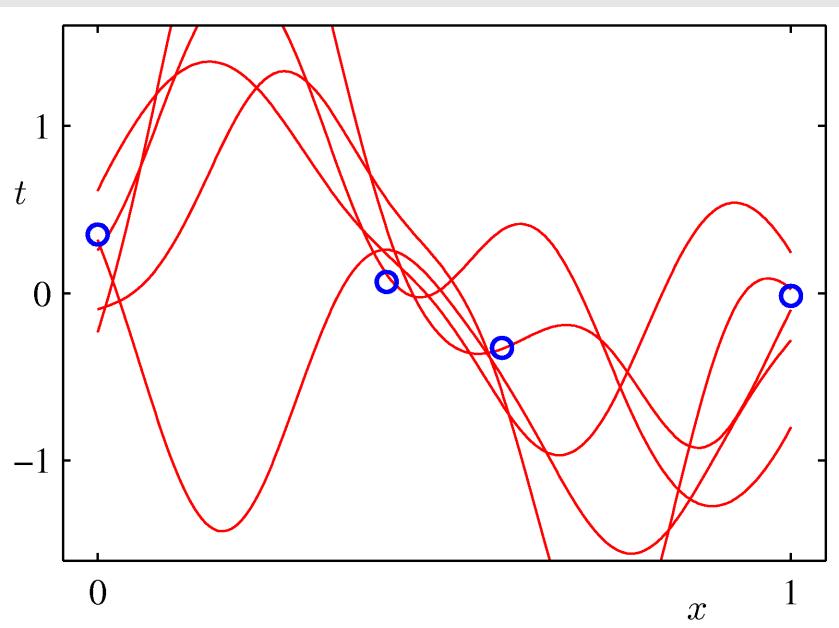
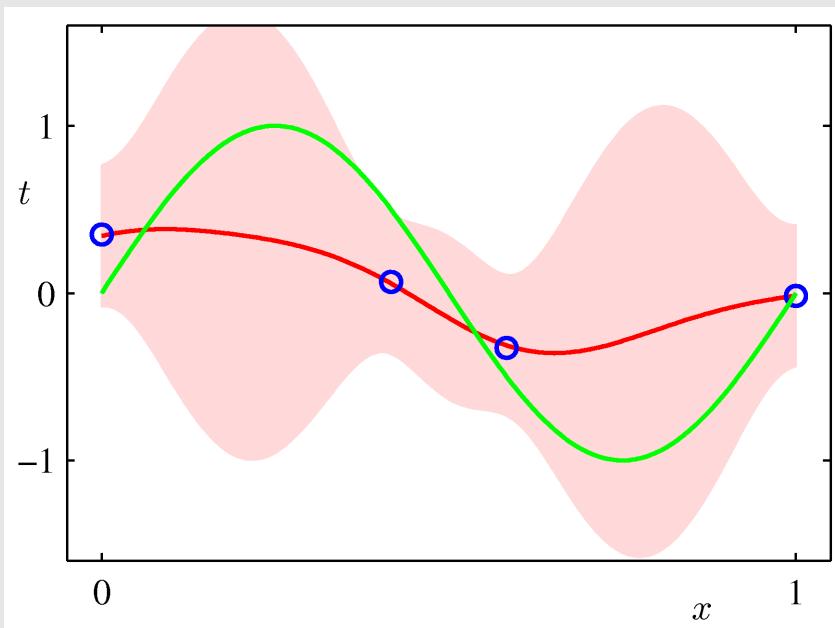
- 2 data points observed → reduced uncertainty near the points



- Left: the predictive distribution
- Right: samples from the predictive distribution

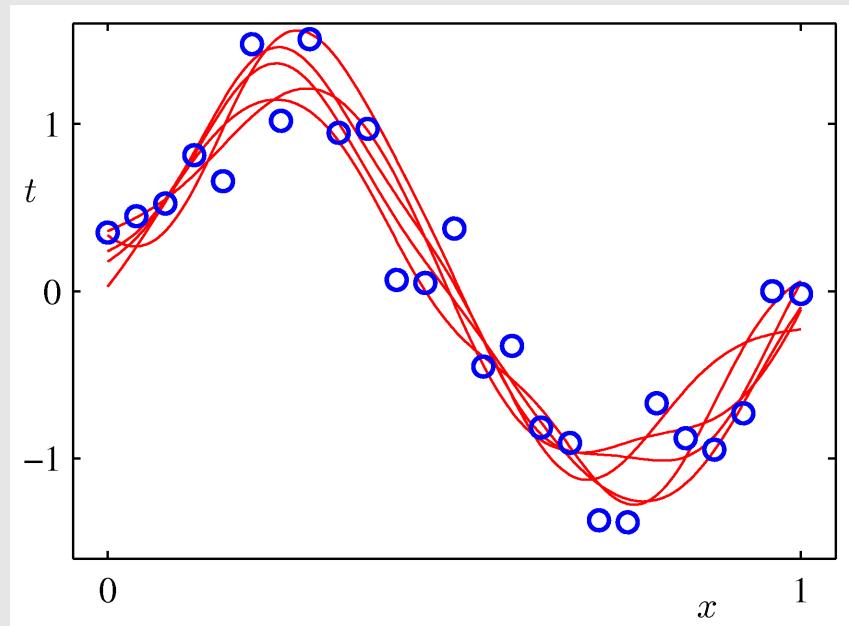
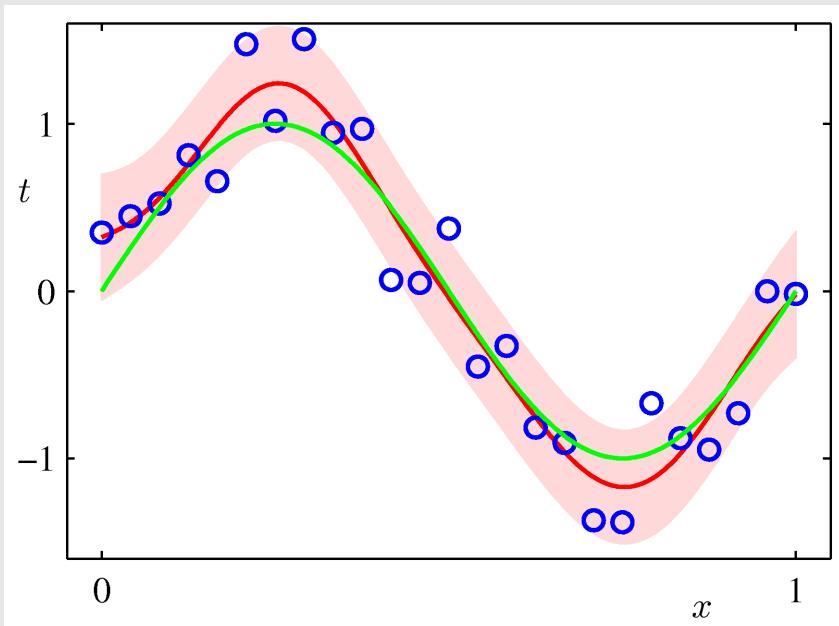
Predictive Distribution (3)

- 4 data points observed → further reduced uncertainty near the points



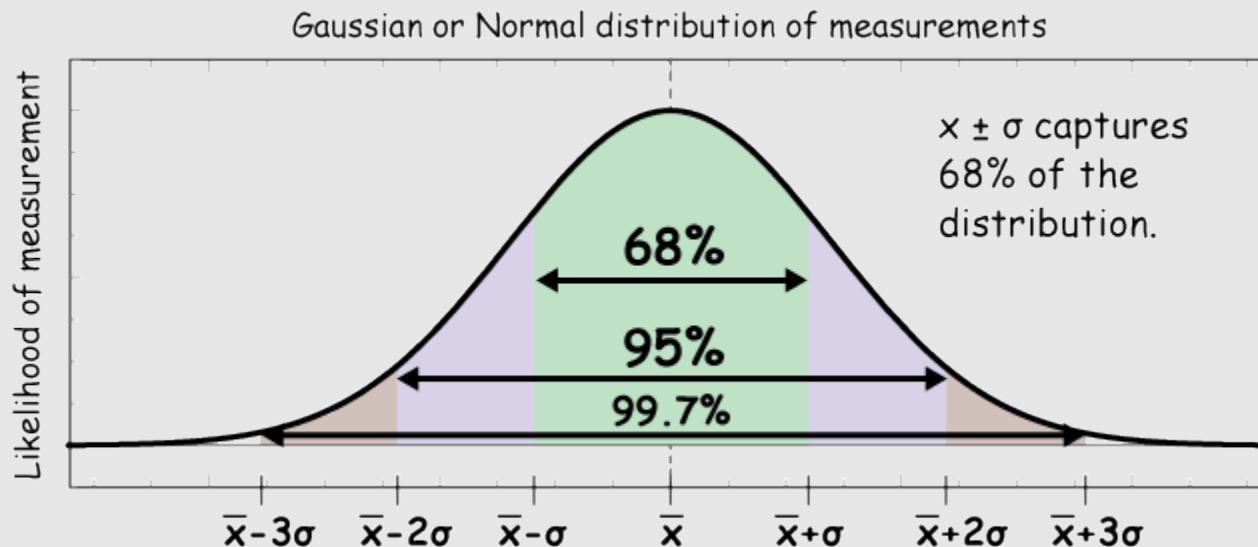
Predictive Distribution (4)

- 25 data points → significantly reduced uncertainty



Gaussian/Normal Distribution

- Knowing the **mean and (co)variance** (std) is sufficient to specify the distribution (*sufficient statistics*)
 - Closed form solution often feasible
- Density estimation: estimate mean and (co)variance



Bayesian Regression Ingredients

- Data: + pre-processing, e.g., $\mathcal{N}(0,1)$
- Model
 - Structure/Architecture: basis function chosen, e.g. poly, Gaussian
 - Hyper-parameter: for basis function (e.g., degree) & prior
 - Parameters (theta): weights and bias
- Evaluation metric: MSE
- Optimisation: closed form for Gaussian distributions, SGD etc. otherwise

Pros and Cons of Bayesian Methods

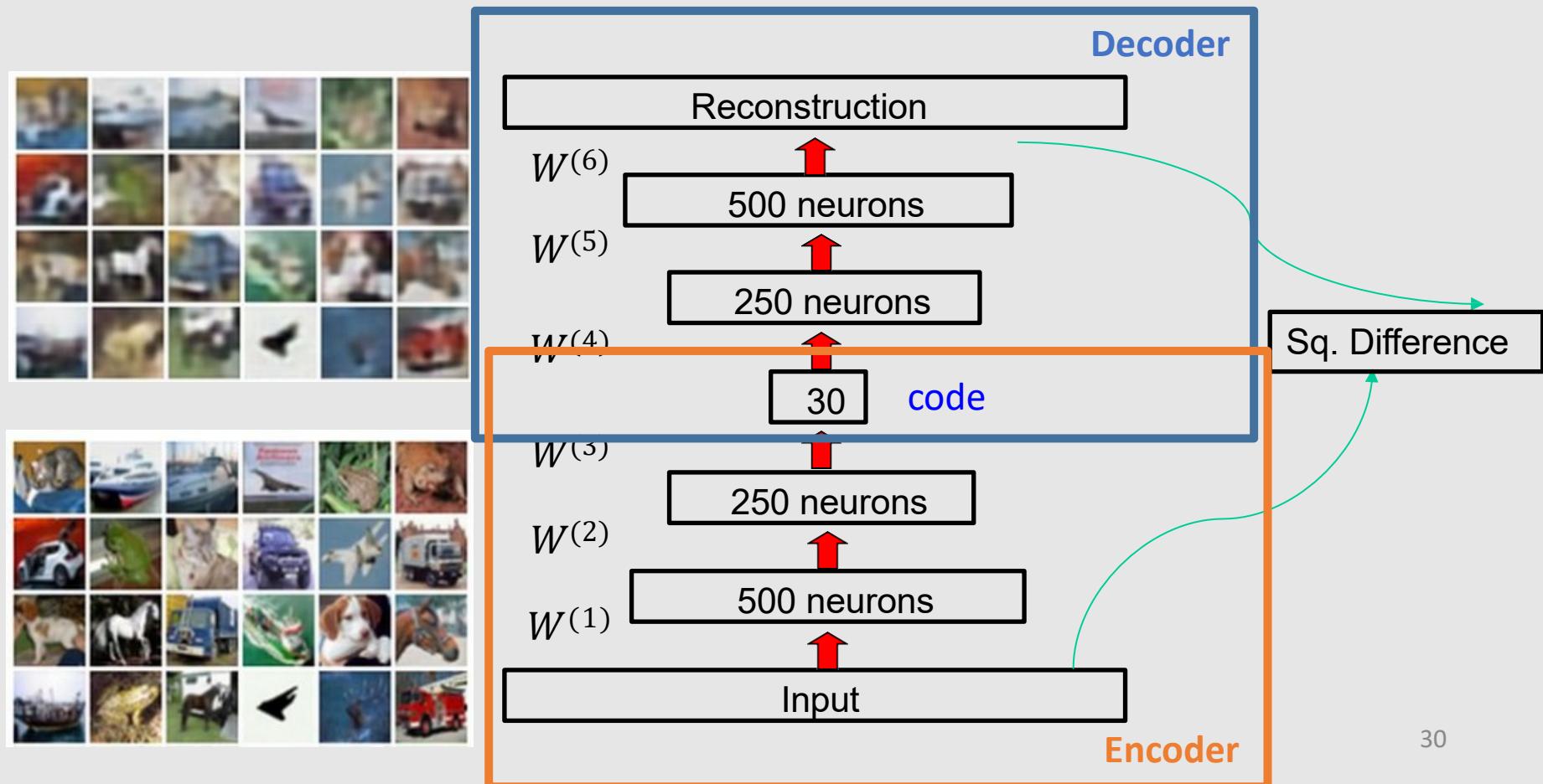
- Pros
 - Provide **uncertainty estimation**, e.g. predicting an output distribution with mean and (co)**variance**
 - Make use of more information (prior, if available)
 - Less overfitting in general
- Cons
 - Complexity
 - Subjectivity: all inferences are based on beliefs.
Which prior to choose? If prior is wrong, ...

Week 9 Contents / Objectives

- Why Generative Models?
- Bayesian Inference
- Bayesian Linear Regression
- **Variational Autoencoder (VAE)**
- VAE Unboxing

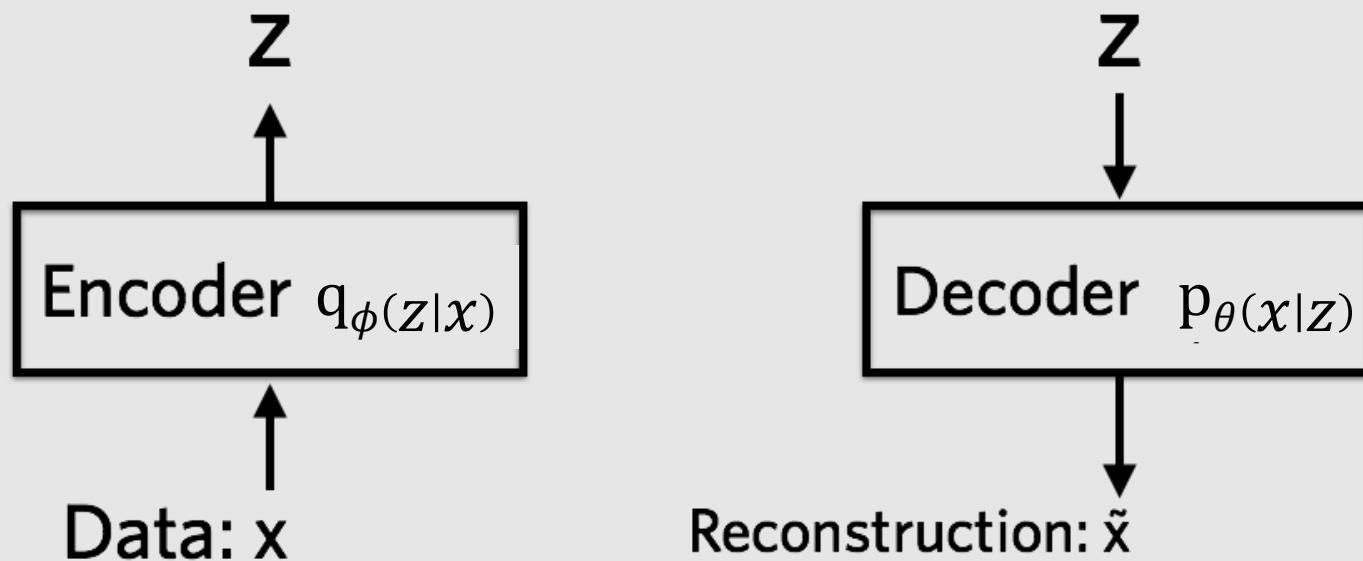
Autoencoders

- The **decoder** reproduces the input from a representation (the **code**) learned by the **encoder**



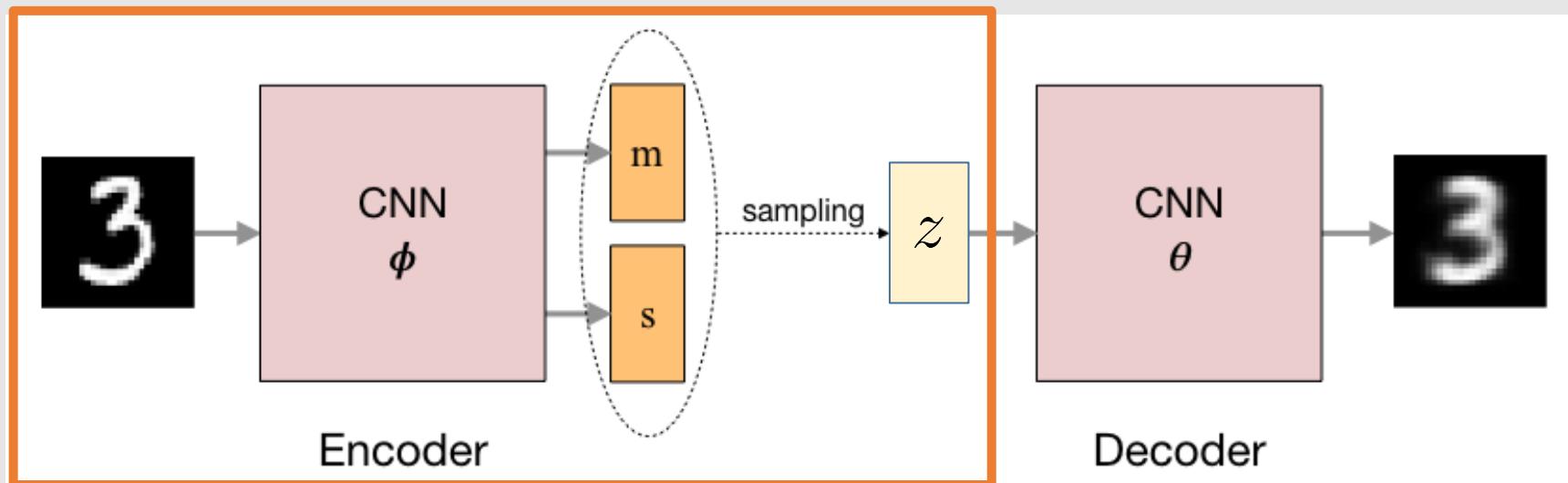
Variational Autoencoder (VAE)

- Make both the encoder and decoder **probabilistic**
- **Encoder:** draw latent variables z (the **code**) from a probability distribution conditioned on the input x
- **Decoder:** reconstruct x probabilistically conditioned on z



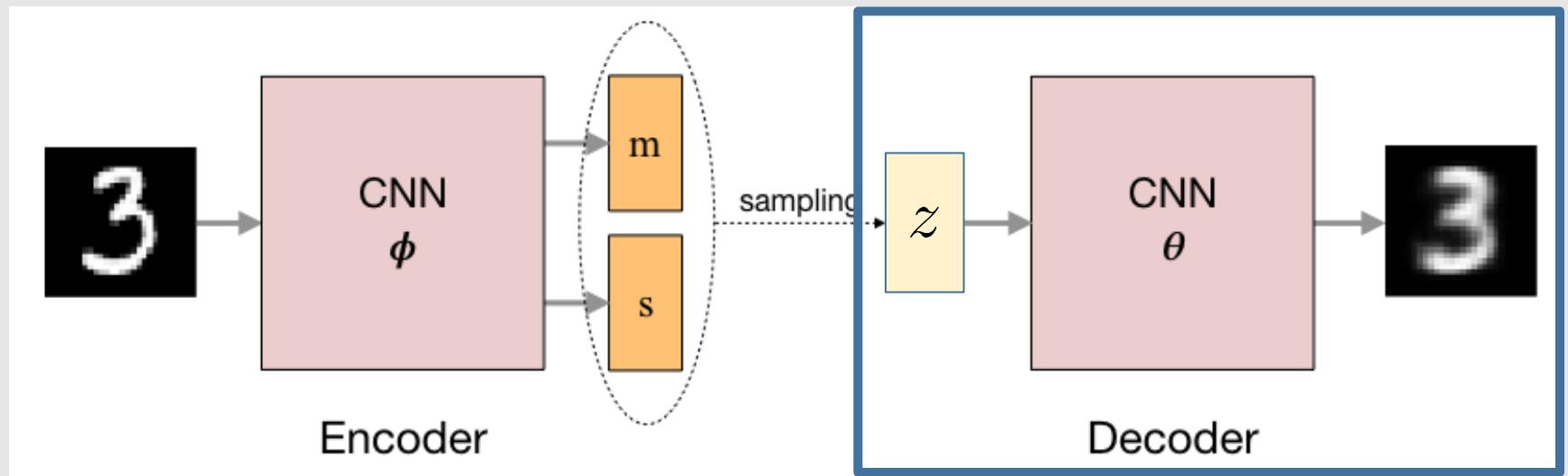
VAE Encoder

- Take the input x and output parameters for a probability distribution $q_\phi(z | x)$. For Gaussian: output the mean and standard deviation
 - Use a neural network with parameter ϕ to do this
- Sample from this distribution to get *random* values of the lower-dimensional representation z



VAE Decoder

- Takes latent variable z and out parameters for a distribution $p_\theta(x | z)$, e.g. the mean and standard deviation for each pixel in the output
 - Use a neural network with parameter θ to do this
- Sample $p_\theta(x | z)$ to get the reconstruction \tilde{x}



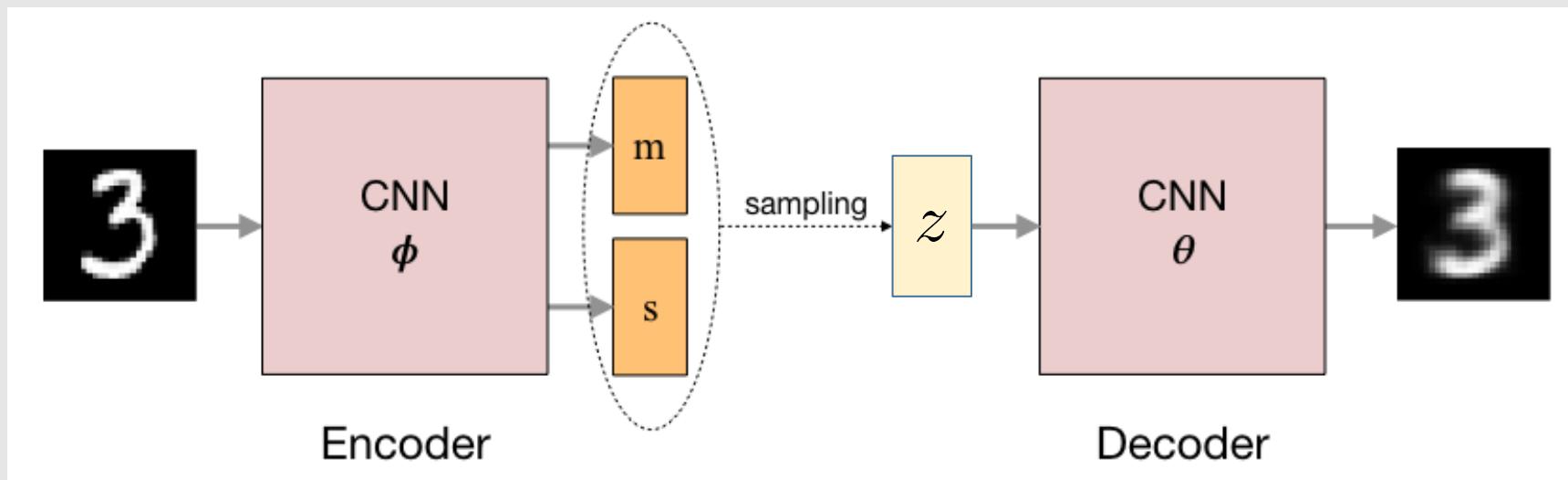
VAE Loss Function

- Objective: learn parameters of two probability distributions ϕ and θ
- For a single data point, the loss function is
$$l_i(\phi, \theta) = -\mathbb{E}_{z \sim q_\phi(z|x_i)} [\log p_\theta(x_i | z)] + \mathbb{KL}(q_\phi(z | x_i) || p(z))$$
- Term #1: the expected negative log-likelihood \rightarrow the reconstruction loss
- Term #2: a regularisation, the Kullback-Leibler divergence between the encoder's distribution $q_\phi(z | x)$ and the marginal distribution $p(z)$, measuring their mismatch
 - $q_\phi(z | x)$ is an approximation to the true posterior $p(z | x)$ based on variational inference, hence the name **variational**

Optimization Challenge

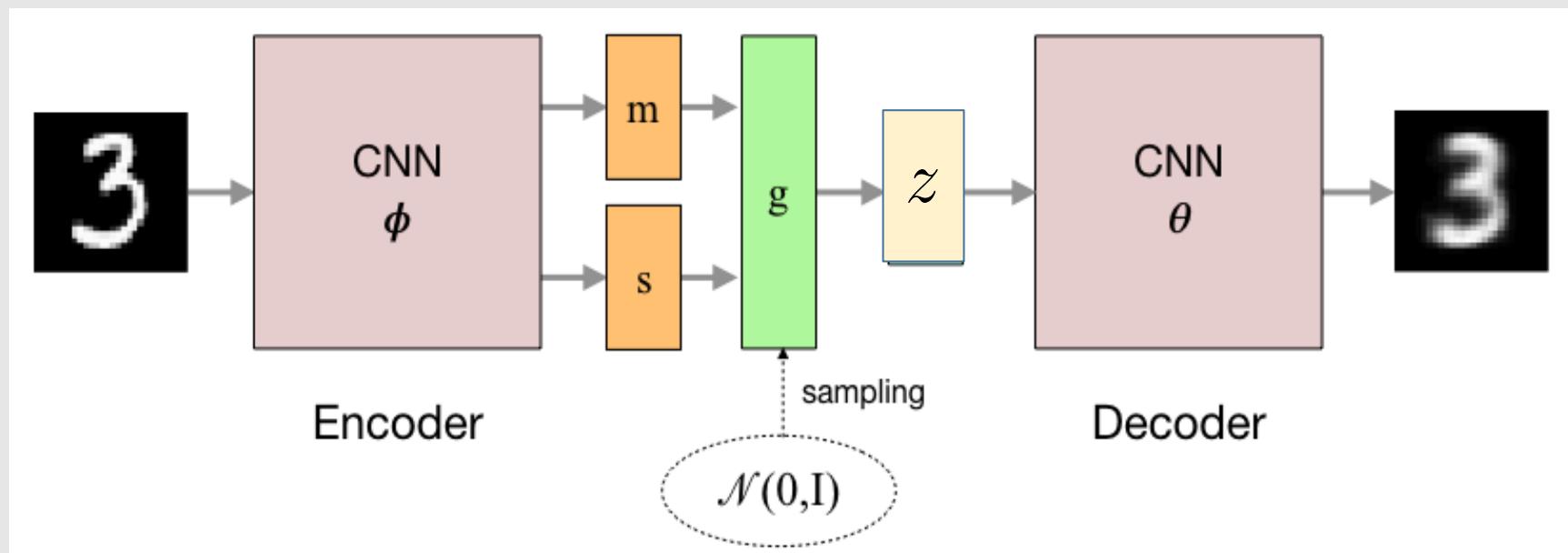
- The expectation in the loss function will be approximated by choosing samples and averaging. This is not differentiable w.r.t. ϕ and θ .

$$l_i(\phi, \theta) = -\mathbb{E}_{z \sim q_\phi(z|x_i)} [\log p_\theta(x_i | z)] + \mathbb{KL}(q_\phi(z | x_i) || p(z))$$



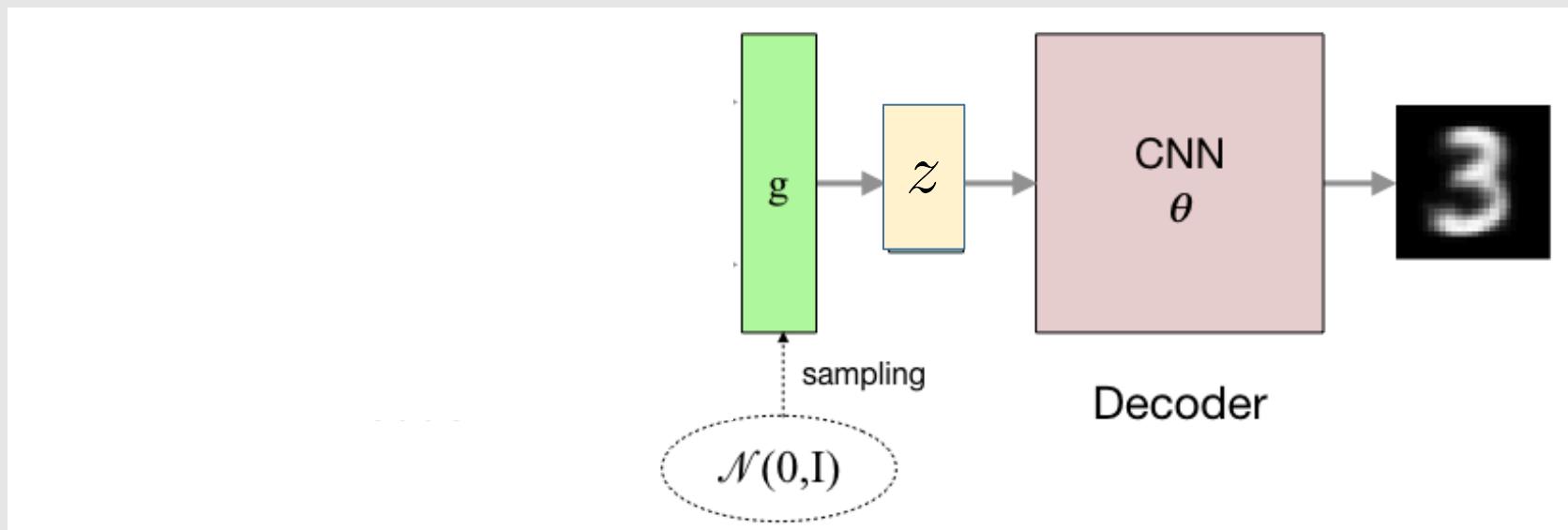
Reparameterization Trick

- If z is $N(\mu(x_i), \Sigma(x_i))$, then we can sample z using $z = \mu(x_i) + \sqrt{(\Sigma(x_i))} \epsilon$, where ϵ is $N(0,1)$. So we can draw samples from $N(0,1)$, which doesn't depend on the parameters.



Generative Mode of VAE

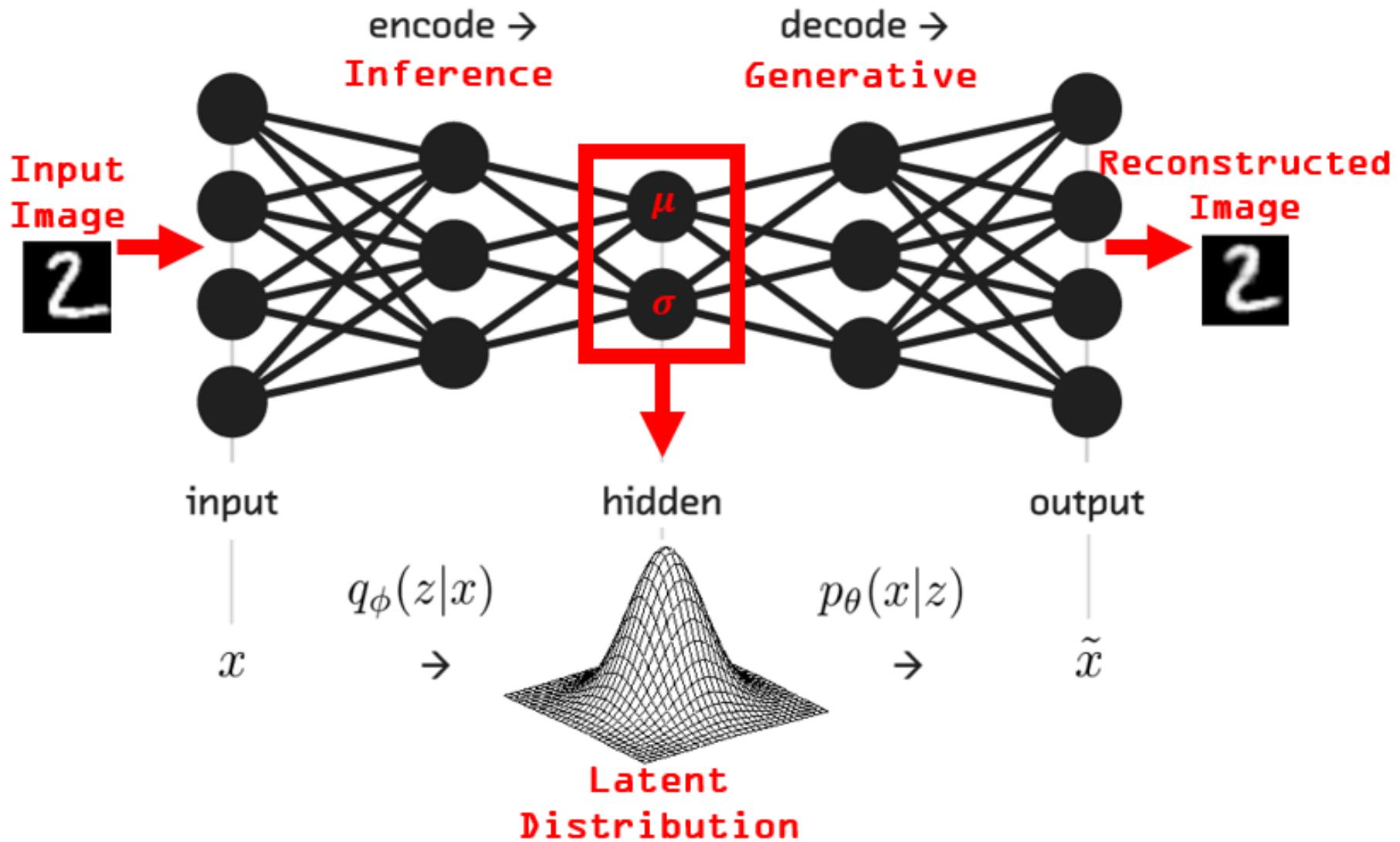
- After training, sample any z from $N(0,1)$ and decode it to get a sample of the entire data distribution $p(x)$
→ Generate new samples that look like the input but aren't in the input.



Week 9 Contents / Objectives

- Why Generative Models?
- Bayesian Inference
- Bayesian Linear Regression
- Variational Autoencoder (VAE)
- **VAE Unboxing**

Probabilistic Modelling in VAE



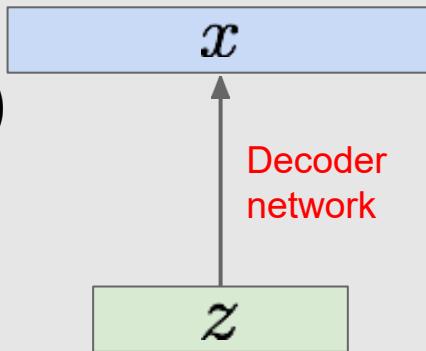
Generative Modelling in VAE

Sample from the
true likelihood

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from
the true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior $p(z)$ to be simple, e.g.
Gaussian.

Likelihood $p(x|z)$ is complex (generates
image) → represent with a neural network

Intractability Challenge

Evidence
(Marginal likelihood)

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Intractable to compute $p(x|z)$ for every z !

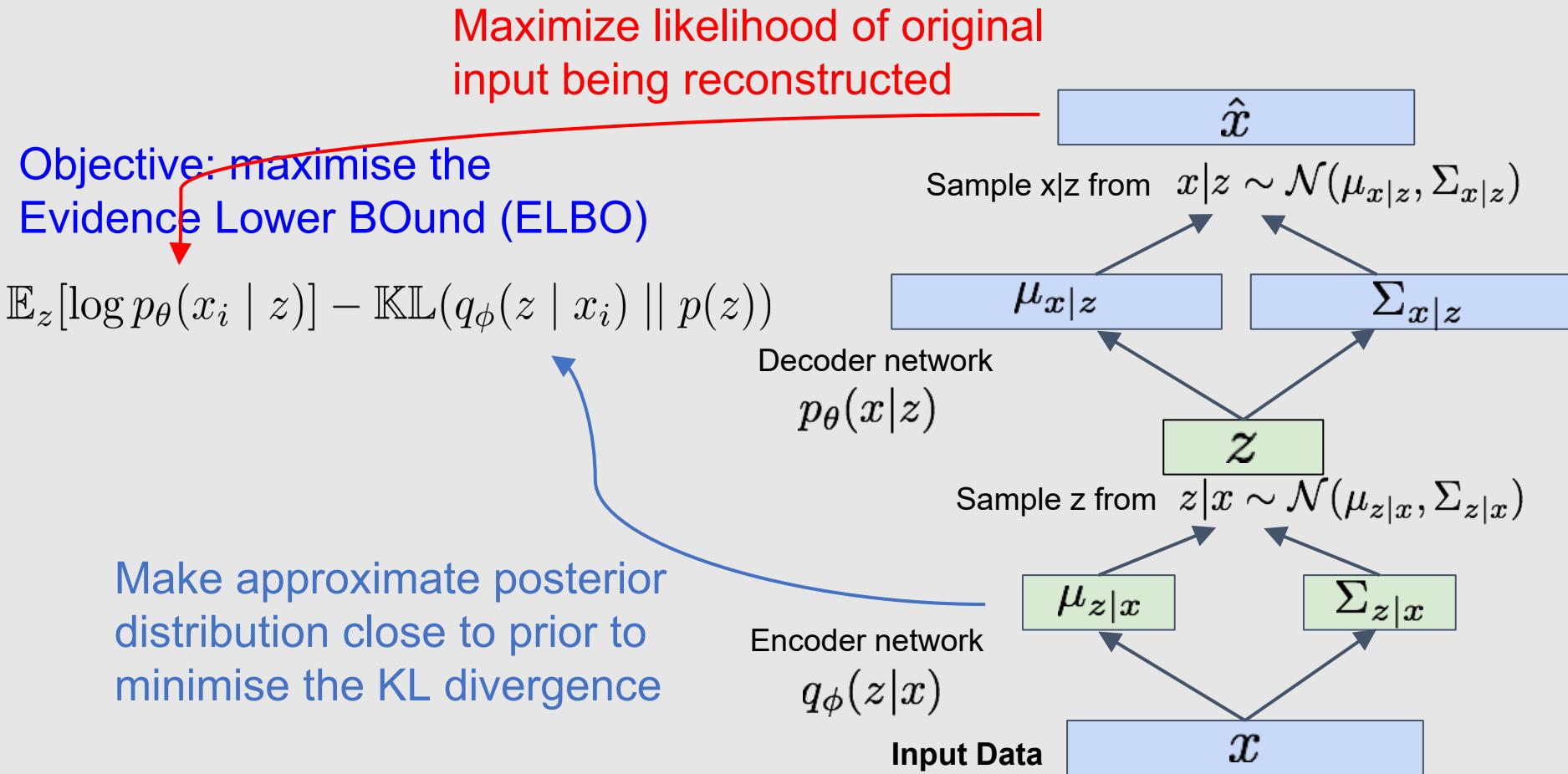
Posterior also intractable:

$$p_{\theta}(z|x) = p_{\theta}(x|z)p_{\theta}(z)/p_{\theta}(x)$$

Intractable evidence

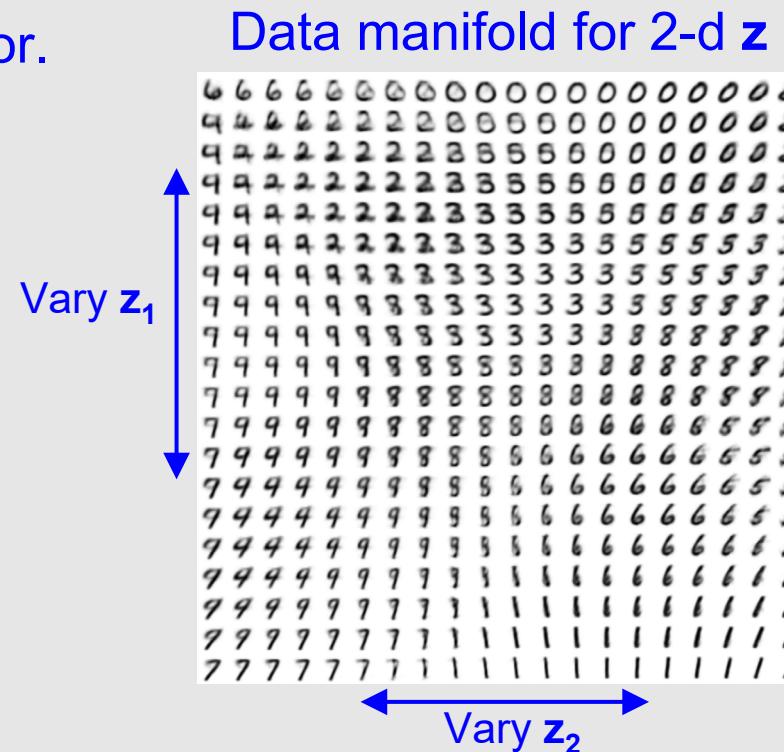
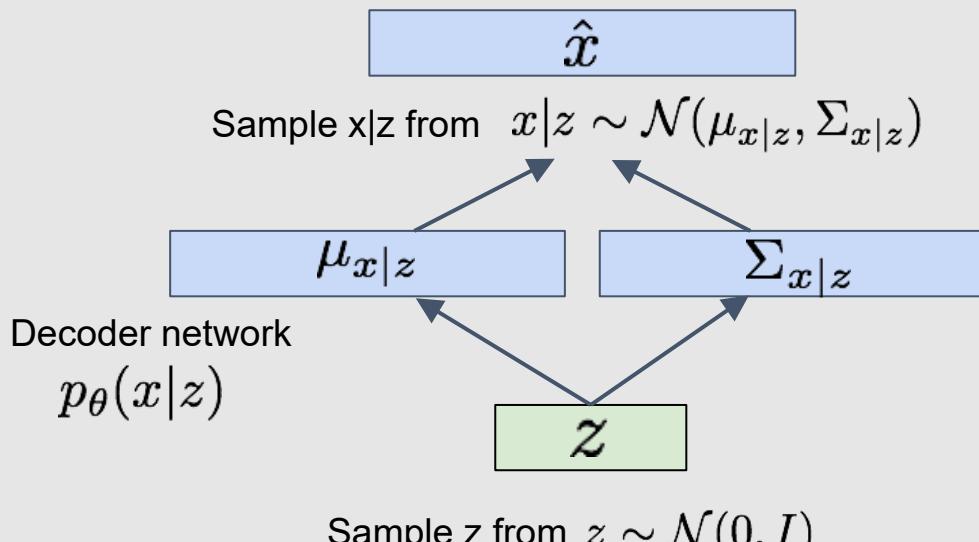
Solution: Define an additional encoder network $q_{\phi}(z | x)$ that approximates $p_{\theta}(z | x)$ to make the problem tractable → the **variational** inference method

Variational Autoencoder Construction



Generating Data with VAE

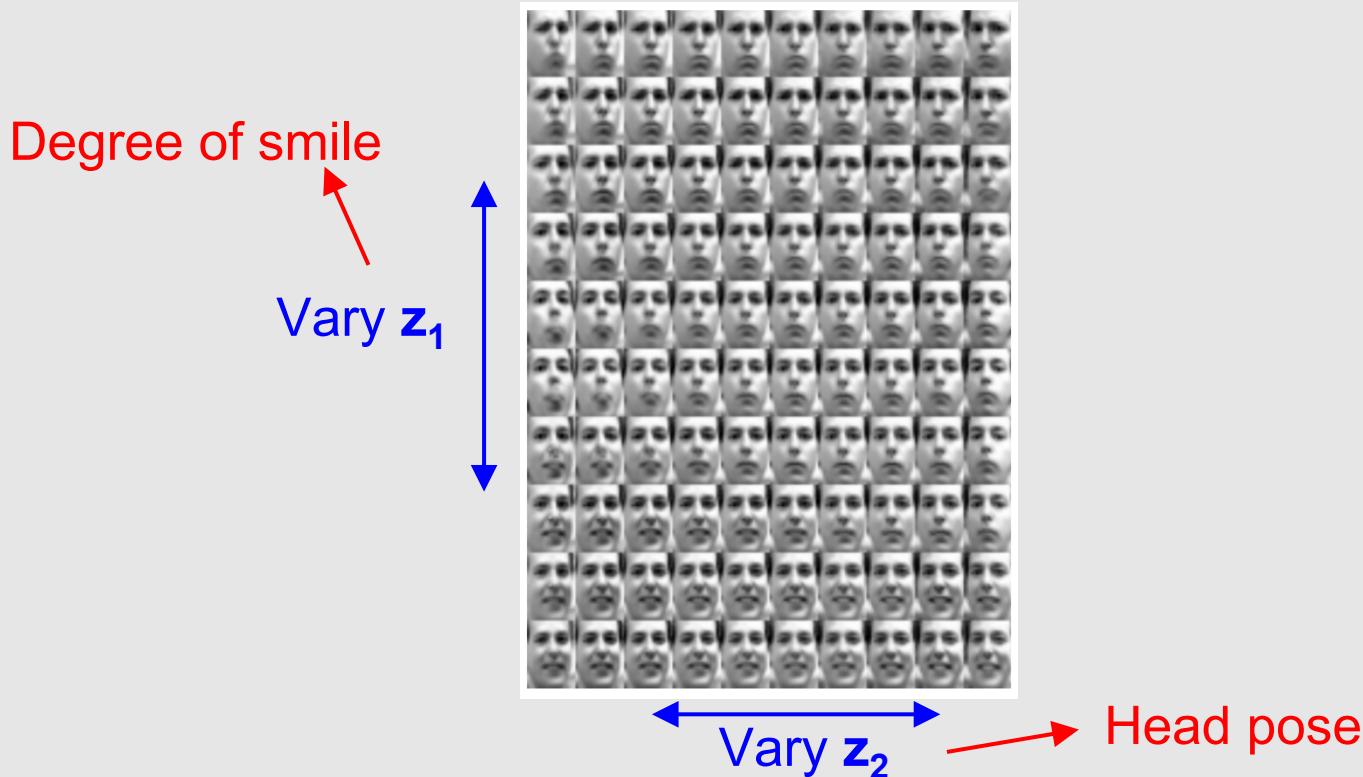
Use decoder network. Sample z from prior.



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Face Generation & Interpretation



- Diagonal prior on $\mathbf{z} \rightarrow$ independent latent variables
- Different dimensions of \mathbf{z} encode interpretable factors of variation

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Variational Autoencoder Ingredients

- Data: + pre-processing, e.g., $\mathcal{N}(0,1)$
- Model
 - Structure/Architecture: layered network
 - Hyper-parameter: layer specs, e.g. #layers
#units, (convolutional) kernel size
 - Parameters (theta): layer weights and biases
- **Evaluation metric: max evidence lower bound**
- Optimisation: backprop, SGD or the like

Pros and Cons of VAE

- Pros
 - Principled approach to generative models
 - Inference of $q(z|x)$ → useful feature representation for other tasks
- Cons
 - Samples blurrier and lower quality compared to state-of-the-art (GANs)

Acknowledgement

- The slides used materials from:
Christopher Bishop, Neil Lawrence, Lee Harrison, John Gosling, Chuck Huber, Greg Buzzard, Mike Mozer, Stefano Ermon, Aditya Grover, Martin Krasser, Dhruv Batra, Fei-Fei Li, Justin Johnson, Serena Yeung

Recommended Reading

- [PRML book](#): Section 3.3 on Bayesian Linear Regression
- [CS231n: Convolutional Neural Networks for Visual Recognition from Stanford](#) (Lecture 11-2020)
- [CS236: Deep Generative Models @Stanford](#)
- Wikipedia entries on related topics
- The lab notebook and references

Lecture 10

Accessible Multimodal Learning and Transfer Learning with PyKale



[Haiping Lu](#)

YouTube Playlist: <https://www.youtube.com/c/HaipingLu/playlists>

[COM4059/6059: MLAI21@The University of Sheffield](#)

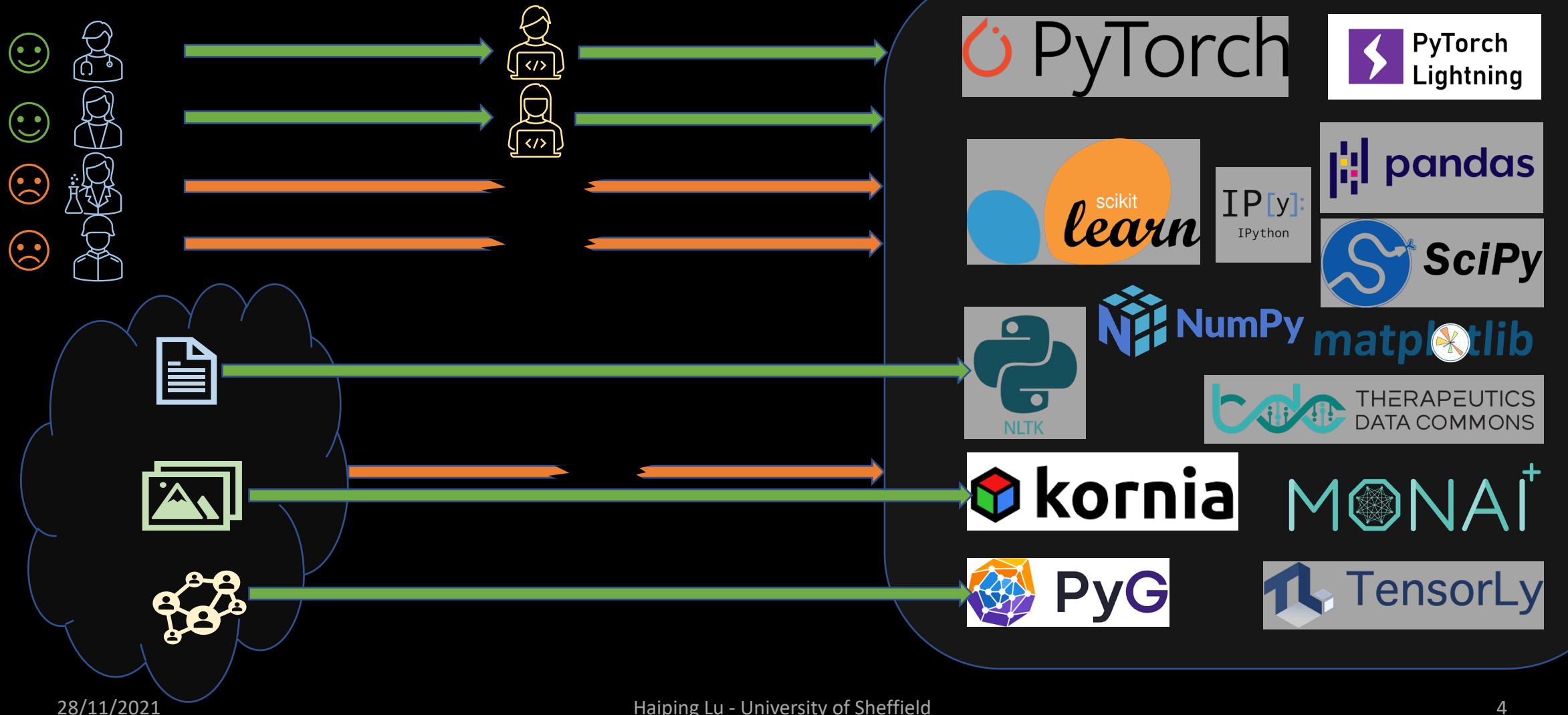
Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- What PyKale Looks Like
- Summary and Review

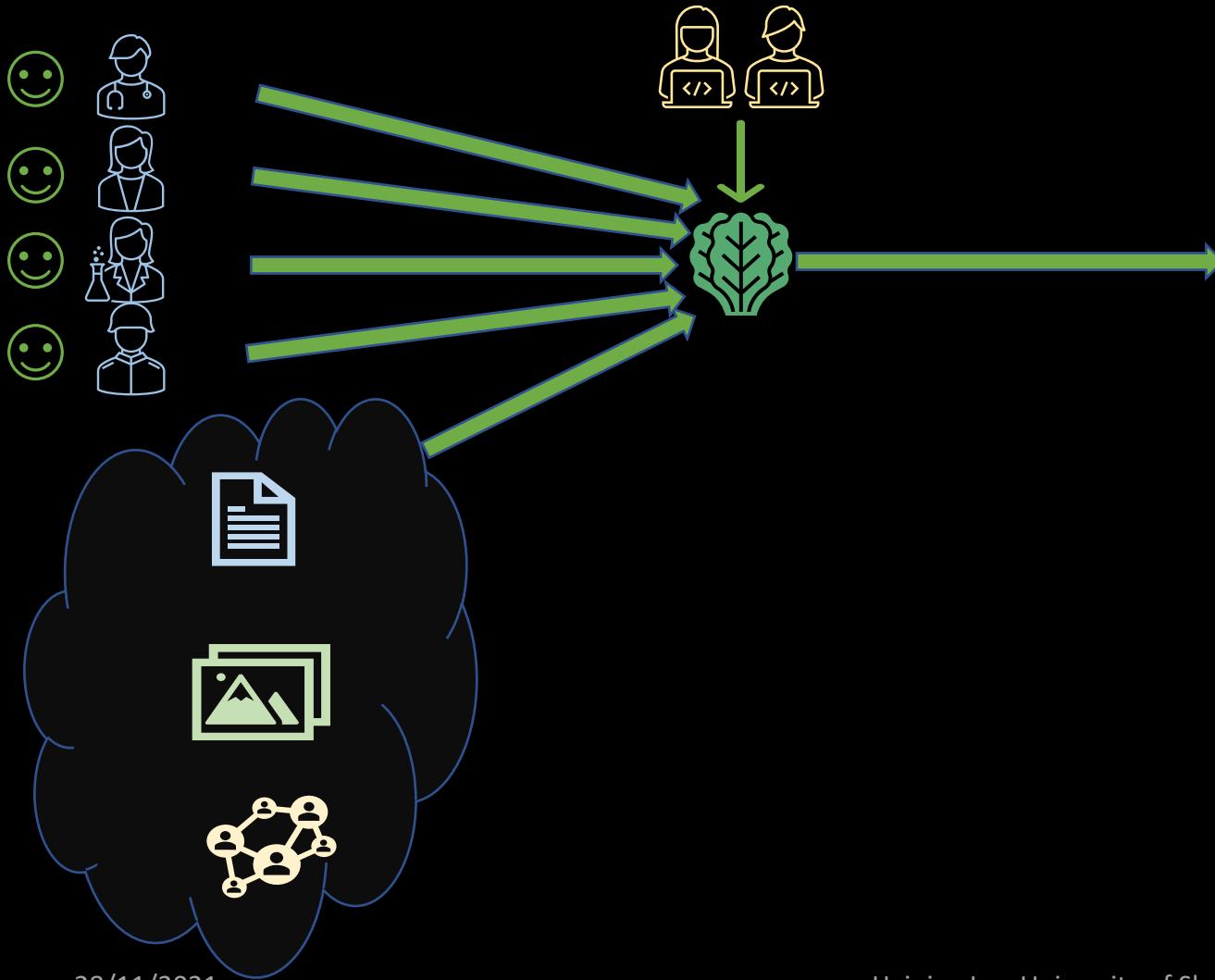
Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- What PyKale Looks Like
- Summary and Review

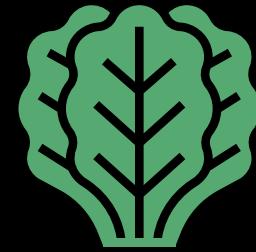
Accessibility issues in interdisciplinary research



Accessibility solution



Why build PyKale?



- Accessibility challenge in interdisciplinary research
- Abundant ML software vs unmet user demands



- Low-level ML software → focus on basics & fundamentals
- Single-domain ML software → tailor for single domains, varying APIs



- Our software → high-level, multi-source, unified API, domain-traversing
- Bridge gaps between data, software and end users for accessible, scalable, and sustainable research in machine learning

Week 10 Contents / Objectives

- Why PyKale
- **How PyKale was Built**
- What PyKale Looks Like
- Summary and Review

Is it FAIR?



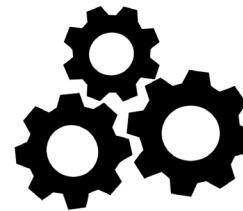
F
indable



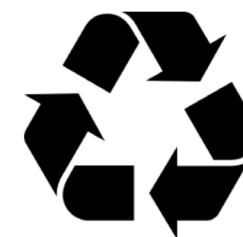
A
ccessible



I
nteroperable



R
Reusable

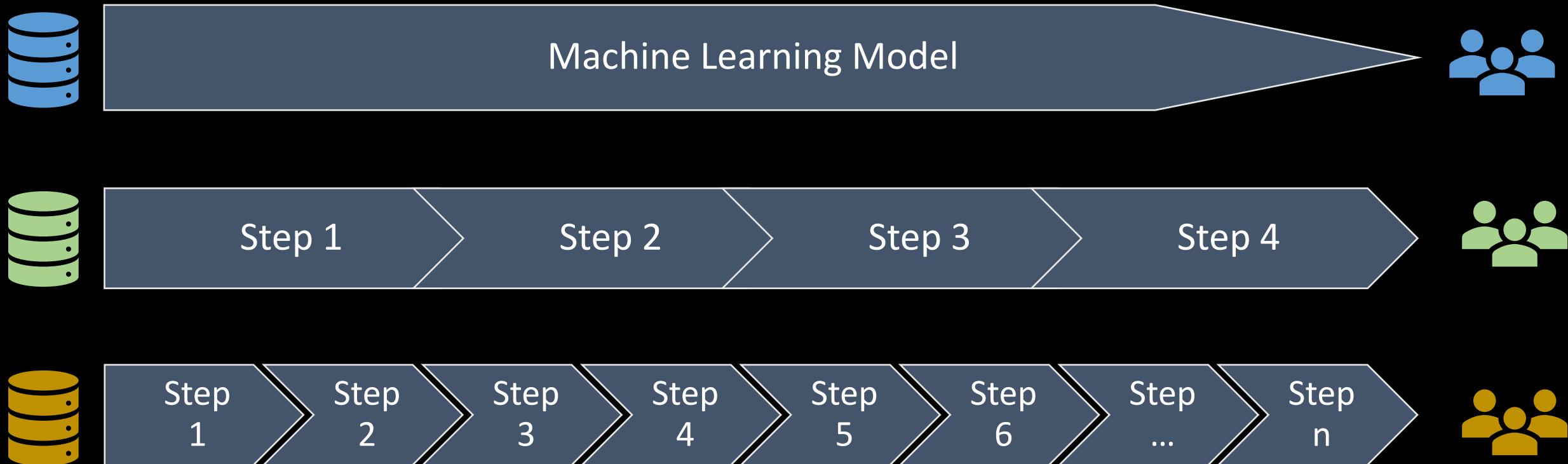


New principles: green machine learning

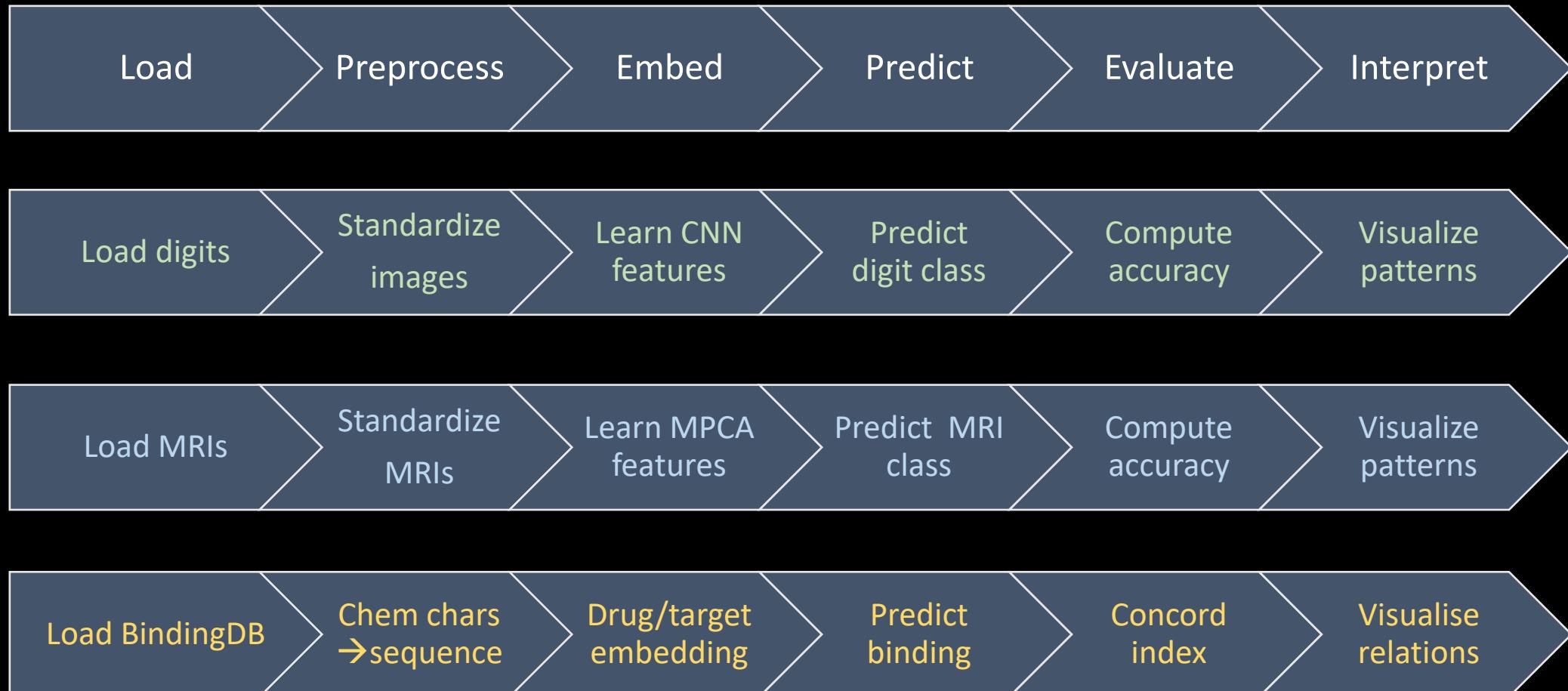
- **Reduce** repetition and redundancy
 - Refactor to standardize workflow and enforce styles
 - Identify and remove duplicated functionalities
- **Reuse** existing resources
 - Reuse the same machine learning pipeline for different data
 - Reuse existing software for available functionalities
- **Recycle** learning models across areas
 - Identify commonalities between applications
 - Recycle models for App A to App B



API without standardization 😞



Pipeline-based API



Accessible machine learning



- Bridge the gap: abundant ML software unmet user demands



- Separate code and configuration: pykale/examples

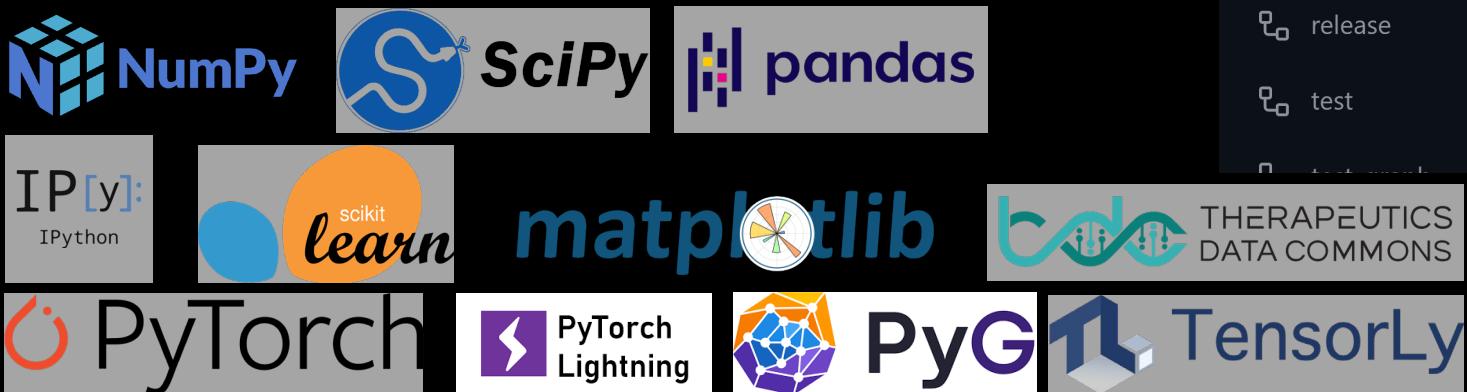


- End users (non-experts in ML)
 - Configure via YAML, no need to write Python/ML → Lower entry barriers
- ML experts
 - Manage config with min coding → Improve reproducibility/efficiency
 - Deliver accessible ML software → **Scalable** collaboration
→ Broad applications



Sustainable software development

- GitHub ecosystem → industrial standard
 - Automation & continuous integration
 - Read the Docs, PyPI release
 - Linting, Pre-commit, PyTest,Codecov
- Python/PyTorch ecosystem



A screenshot of a GitHub repository's Actions page. The top navigation bar shows Code, Issues (5), Pull requests (3), Discussions, and Actions (selected). Below the navigation is a search bar and a "New workflow" button. A sidebar on the left lists workflow names: assign project, changelog, lint, release, and test. The main area displays "All workflows" with 2,617 workflow runs. One run is marked as "test #840: Scheduled" with a green checkmark, and another is marked as "Multi source example" with a red X.

Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- **What PyKale Looks Like**
- Summary and Review

Kale API snapshot: pipeline-based

The screenshot shows a terminal window with four tabs, each displaying a list of files and a dropdown menu with user names.

- pykale / kale /**:
 - Xianyuari (selected)
 - ..
 - embed
 - evaluate
 - interpret
 - loaddata
 - pipeline
 - predict
 - prepdata
 - utils

Dropdown menu:
 - Load Data
 - Preprocess Data
 - Embed
 - Predict
 - Evaluate
 - Interpret
 - Pipeline
 - Utilities
- pykale / kale / loaddata /**:
 - sz144 fix digits_access
 - ..
 - README.md
 - __init__.py
 - cifar_access.py
 - dataset_access.py
 - digits_access.py
 - get_dicom.py
 - mnistm.py
 - multi_domain.py
 - office_access.py
- pykale / kale / utils /**:
 - XianyuLiu Delete csv_logger.py
 - ..
 - README.md
 - __init__.py
 - download.py
 - logger.py
 - print.py
 - seed.py
- pykale / kale / pipeline /**:
 - haipinglu define acronyms
 - ..
 - README.md
 - __init__.py
 - deep_dti.py
 - domain_adapter.py
 - mpca_trainer.py
 - multi_domain_adapter.py
 - video_domain_adapter.py

u - University of Sheffield

Standardized examples & code config separation

The image displays three GitHub commit screenshots illustrating the standardization of examples and code configuration separation.

Commit 1: [haipinglu add help on how to run](#)

- ..
- configs
- README.md
- __init__.py
- config.py
- main.py
- tutorial.ipynb

Commit 2: [haipinglu add help on how to run](#)

- ..
- configs
- figures
- README.md
- __init__.py
- config.py
- main.py
- model.py
- tutorial.ipynb

Commit 3: [haipinglu improve yaml doc](#)

- ..
- configs
- README.md
- __init__.py
- config.py
- main.py
- model.py
- tutorial.ipynb

YAML configuration

```
5 lines (4 sloc) | 49 Bytes

1 DAN:
2   METHOD: "DANN"
3
4 DATASET:
5   SOURCE: "svhn"
```

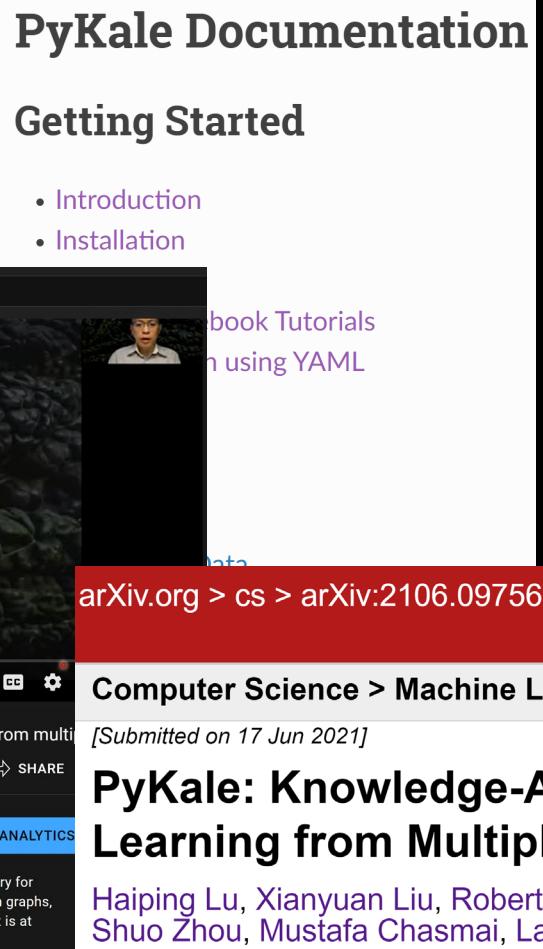
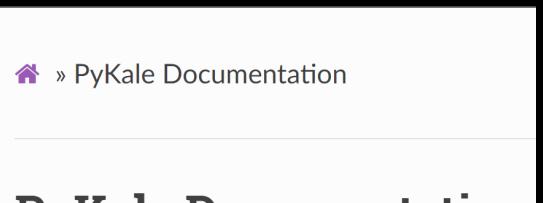
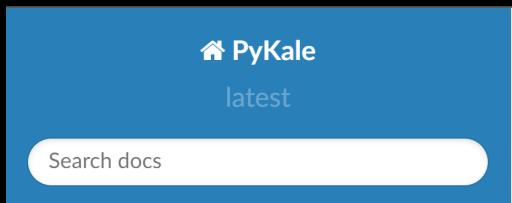
```
14 lines (11 sloc) | 155 Bytes

1 DAN:
2   METHOD: "CDAN"
3
4 DATASET:
5   NUM_REPEAT: 1
6   SOURCE: "svhn"
7   VAL_SPLIT_RATIO: 0.5
8
9 SOLVER:
10  MIN_EPOCHS: 0
11  MAX_EPOCHS: 3
12
13 OUTPUT:
14  PB_FRESH: None
```

```
# -----
# Dataset
# -----
_C.DATASET = CN()
_C.DATASET.ROOT = "../data" # Path to store data and results
_C.DATASET.NAME = "digits" # Name of datasets
_C.DATASET.SOURCE = "mnist" # The source dataset name
_C.DATASET.TARGET = "usps" # The target dataset name
_C.DATASET.NUM_CLASSES = 10
_C.DATASET.NUM_REPEAT = 10
_C.DATASET.DIMENSION = 784
_C.DATASET.WEIGHT_TYPE = "natural"
_C.DATASET.SIZE_TYPE = "source"
_C.DATASET.VAL_SPLIT_RATIO = 0.1
# -----
# Solver
# -----
_C.SOLVER = CN()
_C.SOLVER.SEED = 2020
_C.SOLVER.BASE_LR = 0.001 # Initial learning rate
_C.SOLVER.MOMENTUM = 0.9
_C.SOLVER.WEIGHT_DECAY = 0.0005 # 1e-4
_C.SOLVER.NESTEROV = True

_C.SOLVER.TYPE = "SGD"
_C.SOLVER.MAX_EPOCHS = 120 # "nb_adapt_epochs": 100,
# _C.SOLVER.WARMUP = True
_C.SOLVER.MIN_EPOCHS = 20 # "nb_init_epochs": 20,
_C.SOLVER.TRAIN_BATCH_SIZE = 150 # 150
_C.SOLVER.TEST_BATCH_SIZE = 200 # No difference in ADA
```

Community engagement



The screenshot shows the PyKale GitHub repository's contributing guide and a project board.

Contributing to PyKale

Light involvements (viewers/users) | Medium involvements (contributors) | Heavy involvements (maintainers)

Ask questions | Report bugs | Suggest improvements | Branch, fork & pull | Coding style | Test | Review & merge | Release & management

Project Board

The project board displays three columns: Overview & backlog, To do, and In progress. It includes cards for tasks like 'Added by haipinglu', 'v0.2.0', 'Remove dependency on csv logger and ...', and 'DA for action recognition TA3N'.

GitHub Issues

A list of issues is shown, including:

- Unique Selling Points (bobturneruk, 11 days ago)
- Should we have tests for tutorial notebooks? (bobturneruk, 12 Aug, Answered)
- Where should we publish on PyKale? (bobturneruk, 12 Aug, Unanswered)

GETTING STARTED

Introduction

Installation

Tutorial

Jupyter Notebook Tutorials

» Jupyter Notebook Tutorials

Jupyter Notebook Tutorials

- Image classification: Digits domain adaptation
- Drug discovery: BindingDB drug-target interaction prediction
- Medical image analysis: Cardiac MRI for MPCA-based diagnosis

← → ⌂ https://colab.research.google.com/github/pykale/pykale/blob/main/examples/digits_dann_lightn/tutorial.ipynb



tutorial.ipynb

File Edit View Insert Runtime Tools Help

+ Code

+ Text

Copy to Drive

PyKale Tutorial: Domain Adaptation on Digits with Lightning

| [Open in Colab](#) (click Runtime → Run all (Ctrl+F9) | [Launch Binder](#) (click Run → Run All Cells) |

If using [Google Colab](#), a free GPU can be enabled to save time via setting Runtime → Change runtime type

← → ⌂ https://hub.gke2.mybinder.org/user/pykale-pykale-77qjut67/doc/tree/examples/digits_dann_lightn/tutorial.ipynb

tutorial.ipynb

File Edit View Run Kernel Tabs Settings Help



PyKale Tutorial: Domain Adaptation on Digits with Lightning

| [Open in Colab](#) (click Runtime → Run all (Ctrl+F9) | [Launch Binder](#) (click Run → Run All Cells) |

If using [Google Colab](#), a free GPU can be enabled to save time via setting Runtime → Change runtime type → Hardware accelerator: GPU

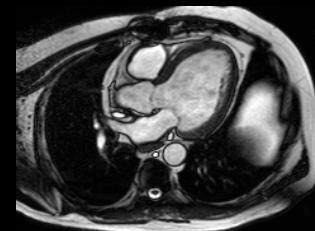
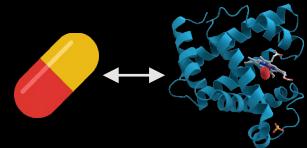
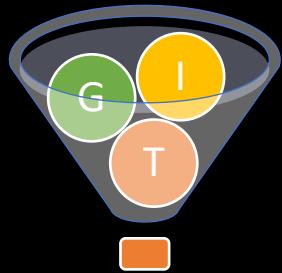
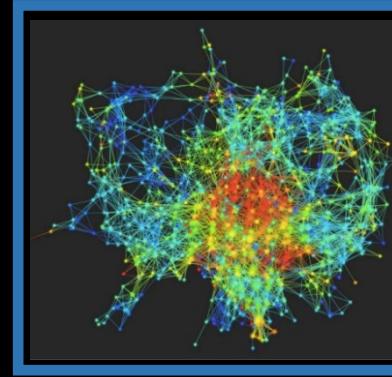


**LIVE
DEMO**

[livedemo-compressor.jpg \(1450x960\) \(posterno.com\)](http://livedemo-compressor.jpg (1450x960) (posterno.com))

Data, model, & app

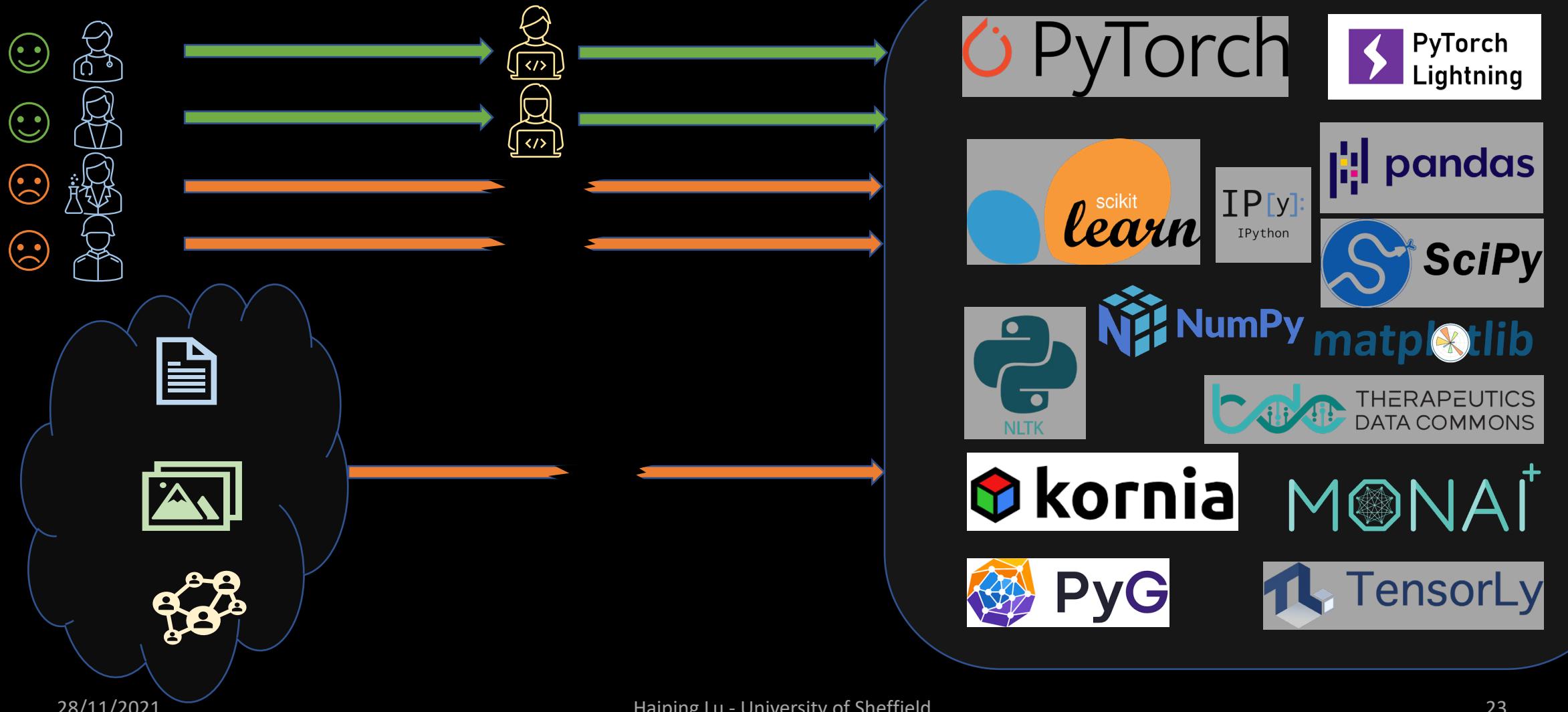
- Data: graph, image, text, video
- Machine learning models
 - Transfer learning: domain adaptation
 - Multimodal learning: integration of heterogeneous data
 - Deep learning: CNN, GNN/GCN, transformers
 - Dimensionality reduction: tensor-based
- Example applications
 - Image/video recognition: CIFAR, digits, office, action videos
 - Bioinformatics/graph analysis: BindingDB, knowledge graphs
 - Medical imaging: cardiac MRI, brain fMRI



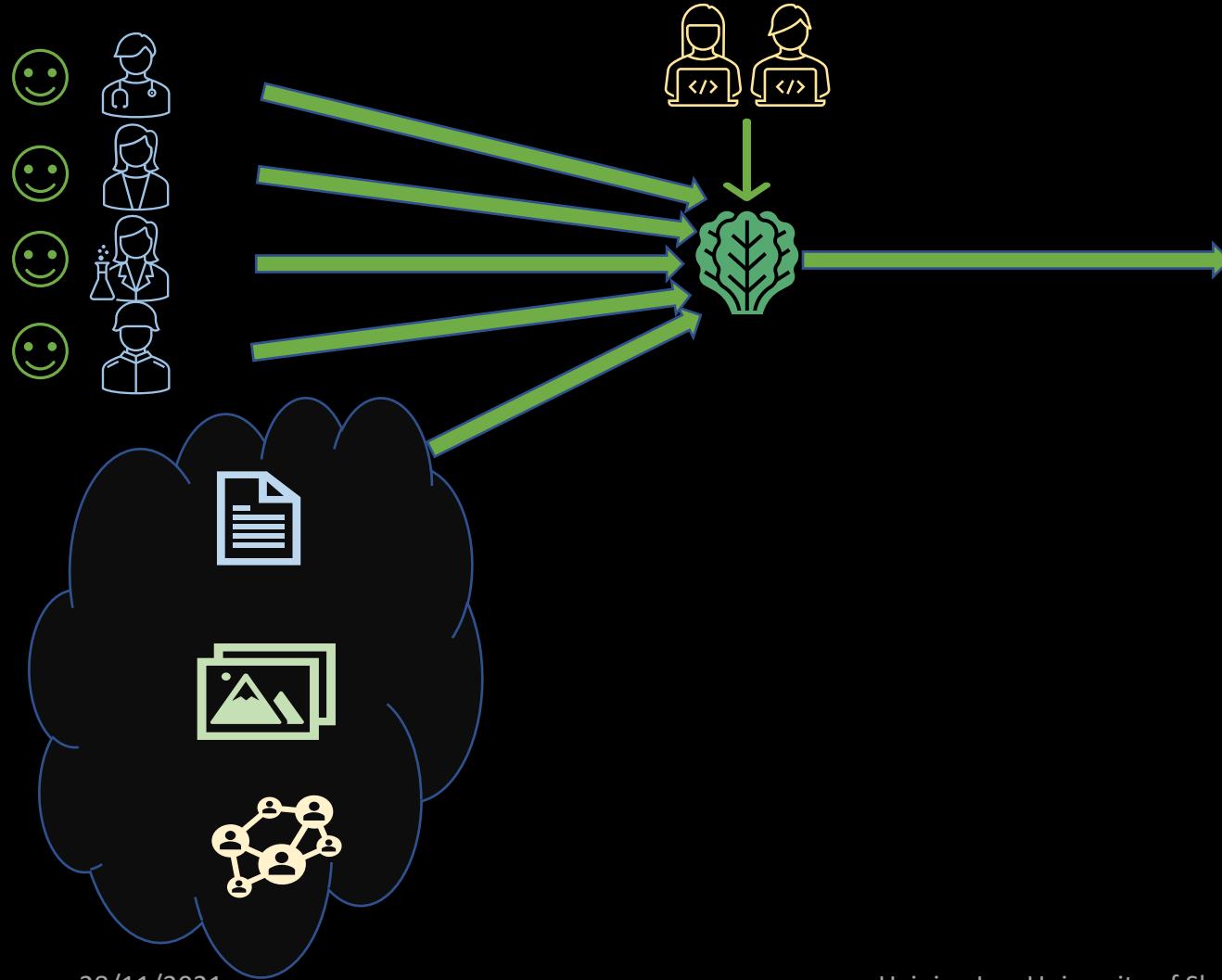
Week 10 Contents / Objectives

- Why PyKale
- How PyKale was Built
- What PyKale Looks Like
- **Summary and Review**

Accessibility issues in interdisciplinary research



PyKale fills the gaps



PyKale: accessible, scalable, sustainable → →

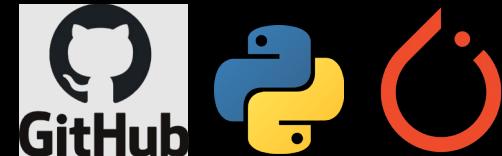
- **Green ML principles:** deliver FAIR ML software



- Pipeline-based API to unify workflow



- Industrial software engineering practices for **sustainability**

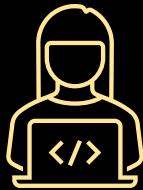


- **Accessible** and **scalable** with code-configuration separation

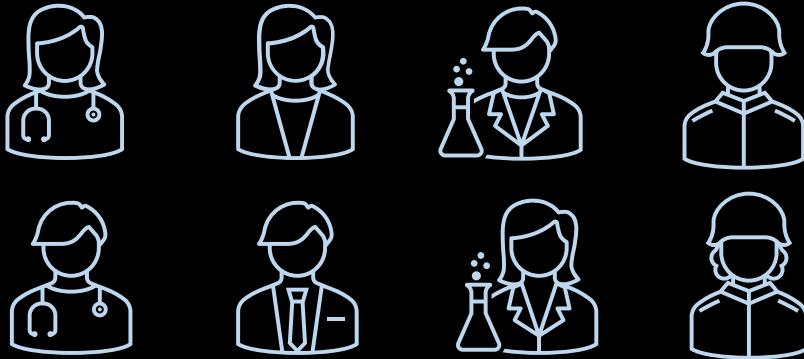


- Multimodal learning & transfer learning for graphs, images, videos, ...

Targeted audience



ML experts



End users

- Reproducible research
- Scalable collaboration
- Cross-boundary models

- Off-the-shelf ML pipelines
- Friendly configuration editing
- Multi modalities under one roof

Open with strong community engagement

Open & welcome

- Open source on GitHub (MIT License)
 - <https://github.com/pykale/pykale>
- Welcome feedback/contribution!

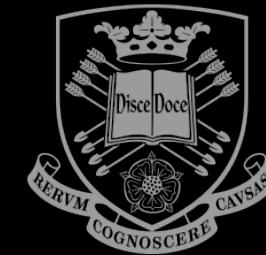
A screenshot of a web browser window. The address bar shows the URL <https://pytorch.org/ecosystem/>. The page content includes the PyTorch logo and the PyKale logo. A descriptive text block states: "PyKale is a PyTorch library for multimodal learning and transfer learning with deep learning dimensionality reduction on graphs, images, texts, and videos."



Acknowledgement



- Shef groups + community



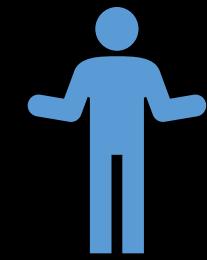
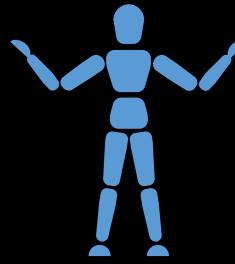
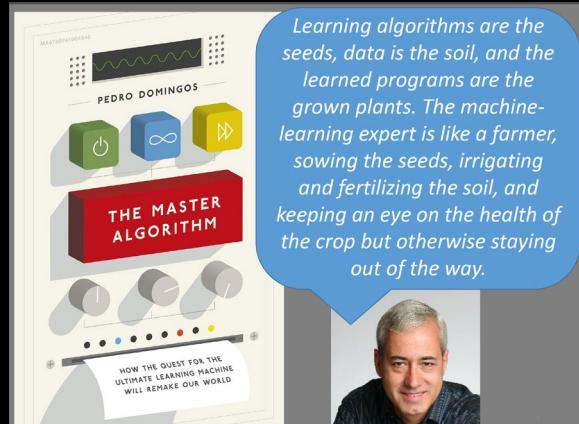
The
University
Of
Sheffield.



- Partially funded by Wellcome Trust via an Innovator Award

Review and Reflect on Machine Learning

- Training: model as well as people
 - You are a **model**
- Reflection and analogy
 - What you want to learn (objective function / goal)
 - How we teach you to learn (training / optimisation)
 - What you have learned (decision model)
- Key challenge: overfitting vs **generalisation**
- Post your questions
- Have fun



[kisspng-fashion-show-computer-icons-man-icon-5acd9cfaa3d9f5.3374529915234245066712.jpg \(900x900\) \(cleanpng.com\)](https://kisspng.com/fashion-show-computer-icons-man-icon-5acd9cfaa3d9f5.3374529915234245066712.jpg)

Data, Model, Metric, Optimisation

- | | |
|--|--|
| 1. Given training data:
$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ | 3. Define goal:
– Objective function
$\theta^* = \arg \min_{\theta} \sum_{i=1}^N \ell(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$ |
| 2. Choose each of these:
– Decision function/model
$\hat{\mathbf{y}} = f_{\theta}(\mathbf{x}_i)$ | 4. Train/optimize with SGD: (take small steps opposite the gradient)
$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla \ell(f_{\theta}(\mathbf{x}_i), \mathbf{y}_i)$ |