

# Introduction to machine learning 1

- **Intro**
- **Regression**
- **Classification**
- **Algorithms:**
  - **Nearest neighbours**
  - **Decision trees**
  - **Ensembles**
- **Physics example**



**Iftach Sadeh**

February 2023

[iftach.sadeh@desy.de](mailto:iftach.sadeh@desy.de)

- Largely derived from:
  - University of Toronto CSC411 - Introduction to Machine Learning (Fall 2016).  
See: [http://www.cs.toronto.edu/~urtasun/courses/CSC411\\_Fall16/CSC411\\_Fall16.html](http://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/CSC411_Fall16.html)
  - MIT's introductory course on deep learning - MIT 6.S191 - <http://introtodeeplearning.com/>

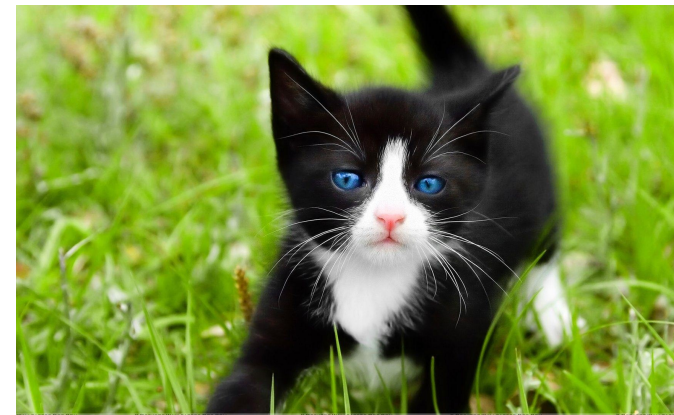


# What is Machine Learning?



# What is Machine Learning?

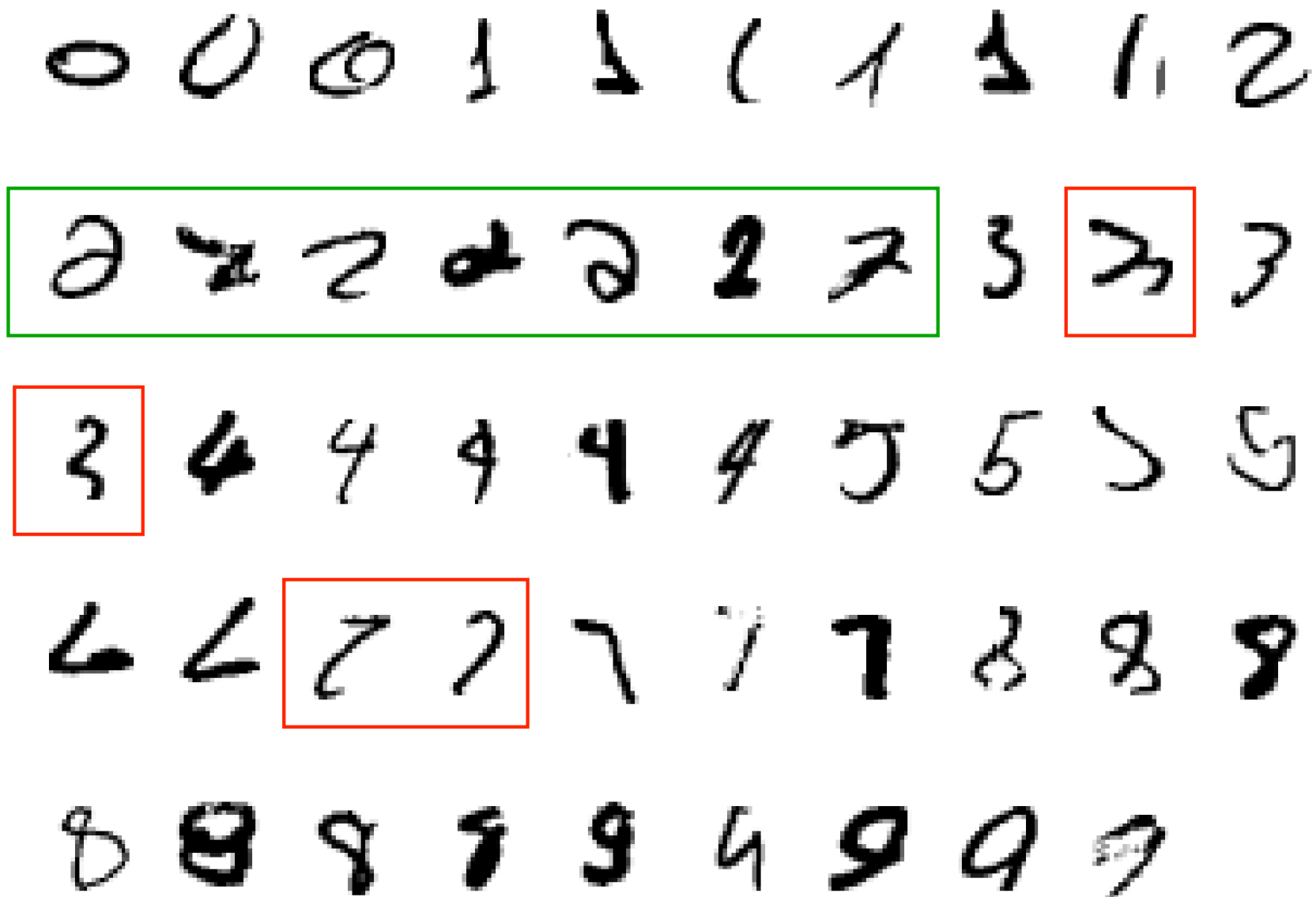
- How can we solve a specific problem?
  - Write a program that **encodes a set of rules** that are useful to solve the problem.
  - In many cases is very difficult to specify those rules, e.g., given a picture determine whether there is a cat in the image



# What is Machine Learning?

- How can we solve a specific problem?
  - Write a program that **encodes a set of rules** that are useful to solve the problem.
  - In many cases is very difficult to specify those rules, e.g., given a picture determine whether there is a cat in the image.
- Learning systems are **not directly programmed** to solve a problem, instead develop own program based on:
  - Examples of how they **should behave**.
  - From **trial-and-error** experience trying to solve the problem.
- Different than standard computer science:
  - Want to implement **unknown function**, only have access e.g., to sample input-output pairs (training examples).
  - Learning simply means incorporating information from the training examples into the system.

# What makes a 2?



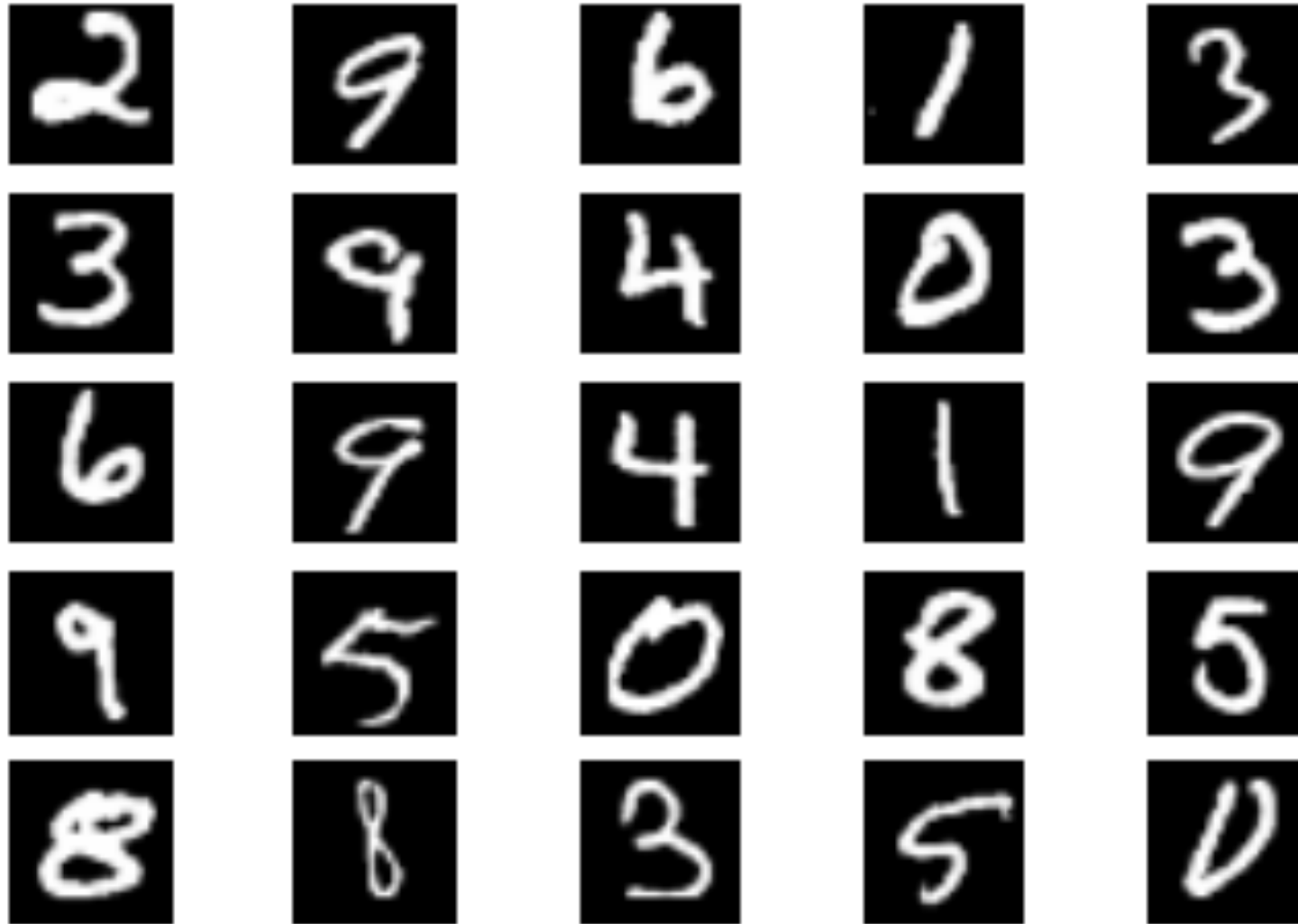
# What is Machine Learning?

- It is very hard to write programs that solve problems like recognising a handwritten digit:
  - What distinguishes a 2 from a 7?
  - How does our brain do it?
- Instead of writing a program by hand, we **collect examples that specify the correct output for a given input.**
- A machine learning algorithm then takes these examples and produces a program that does the job:
  - The program produced by the learning algorithm may look very different from a typical hand written program. It may contain millions of numbers.
  - If we do it right, the program works for new cases as well as the ones we trained it on, **but only under limited conditions...**

# Types of ML tasks

- **Classification**: determine which discrete category the example belongs to
- **Recognising patterns**: speech recognition, facial identity, etc
- **Recognising anomalies**: unusual sequences of credit card transactions, panic situation at an airport
- **Recommender Systems**: noisy data, commercial pay-off (e.g., Amazon, Netflix).
- **Information retrieval**: find documents or images with similar content
- **Computer vision**: detection, segmentation, depth estimation, optical flow, etc
- **Robotics**: perception, planning, etc
- Learning to **play games**
- **Spam filtering, fraud detection**: the enemy adapts so we must adapt too
  
- ... Many more ...

## A few concrete examples → Classification



What digit is this?



# Classification



Is this a dog?

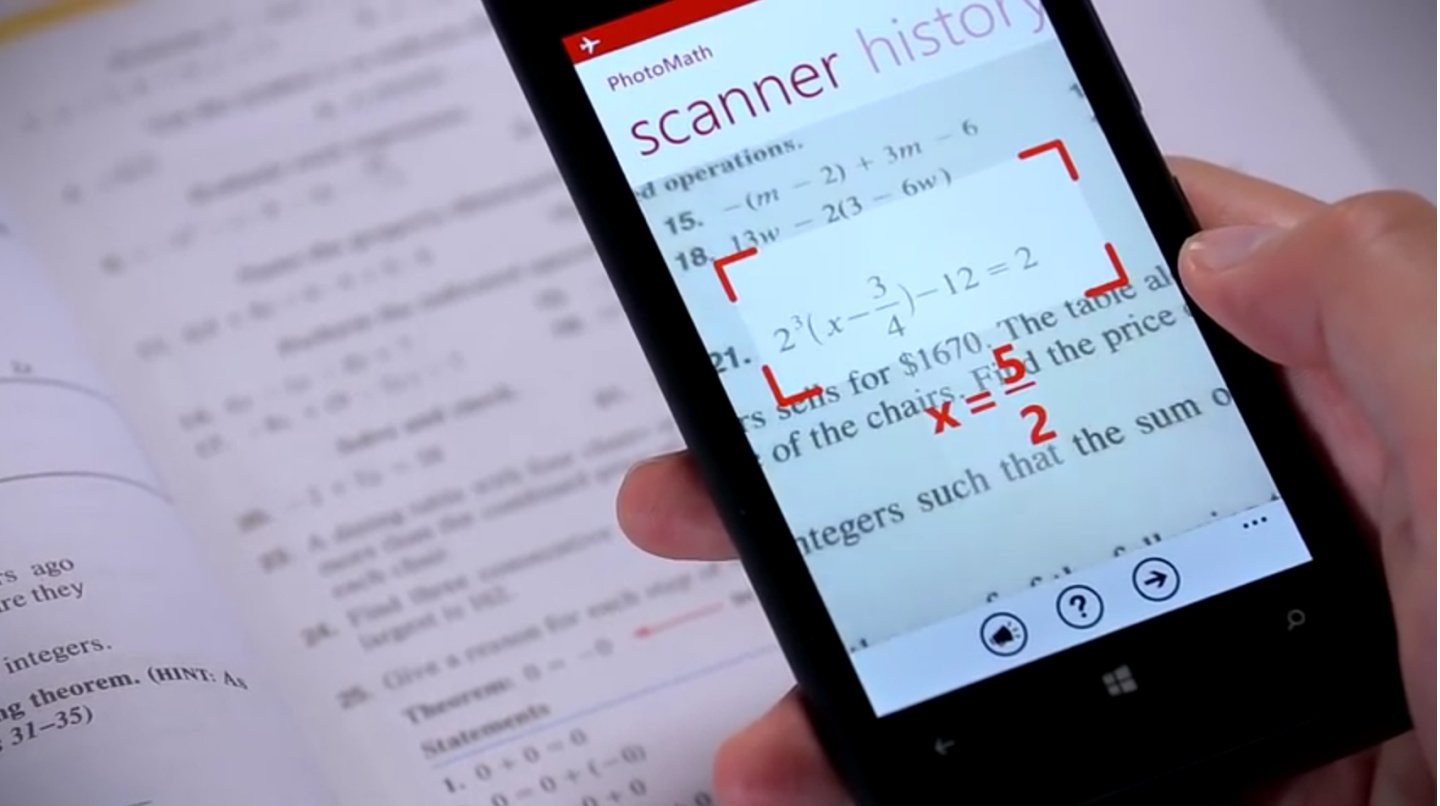


what about this one?



Do I have diabetes?

# Pattern recognition

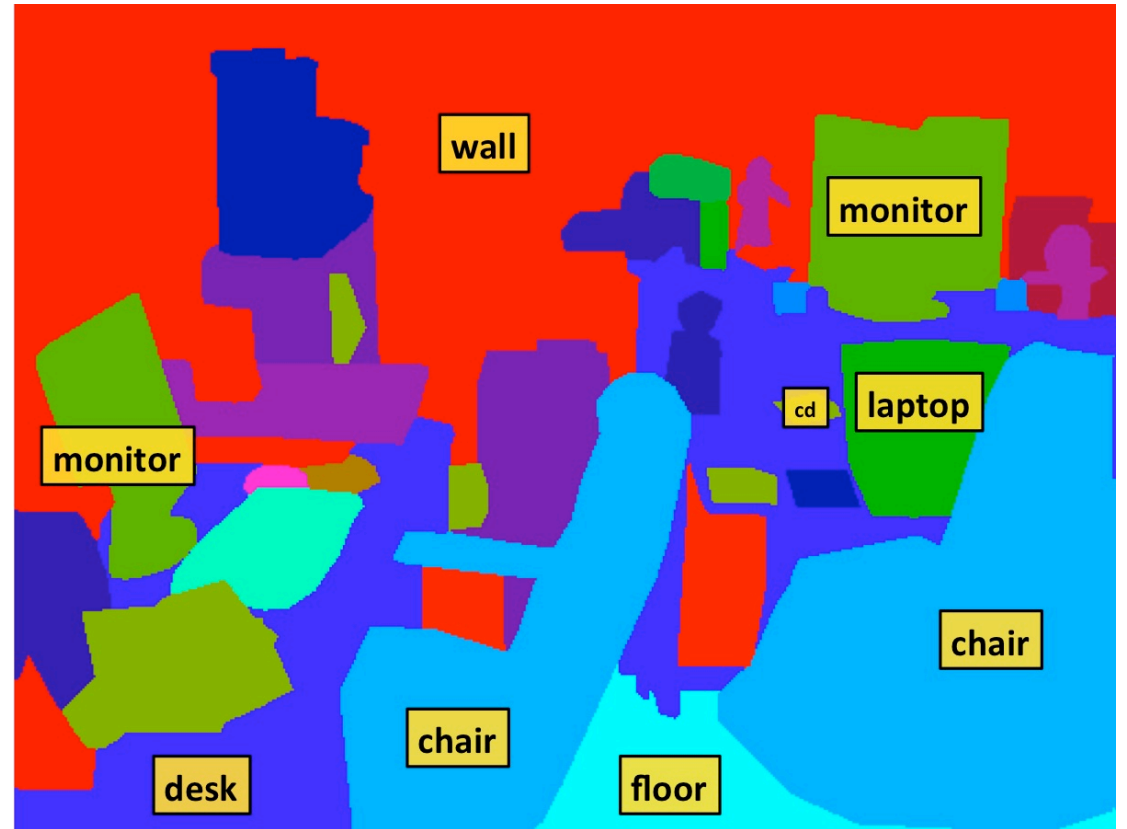


# Recommendation system

The screenshot shows the Netflix interface with the search bar containing 'i-robot'. Below the search bar, it says 'Explore titles related to: I, Robot | Robot Chicken Star Wars: Episode II'. The main section is titled 'Titles related to I, Robot' and displays a grid of 15 movie posters:

- A.I. Artificial Intelligence
- Minority Report
- Hellboy
- Star Trek Into Darkness
- Deep Impact
- Galaxy Quest
- Total Recall
- Terminator 2: Judgment Day
- Jack Reacher
- Shooter
- Men in Black 3
- G.I. Joe: Retaliation
- Enemy of the State
- Parallels
- Paycheck

# Computer vision

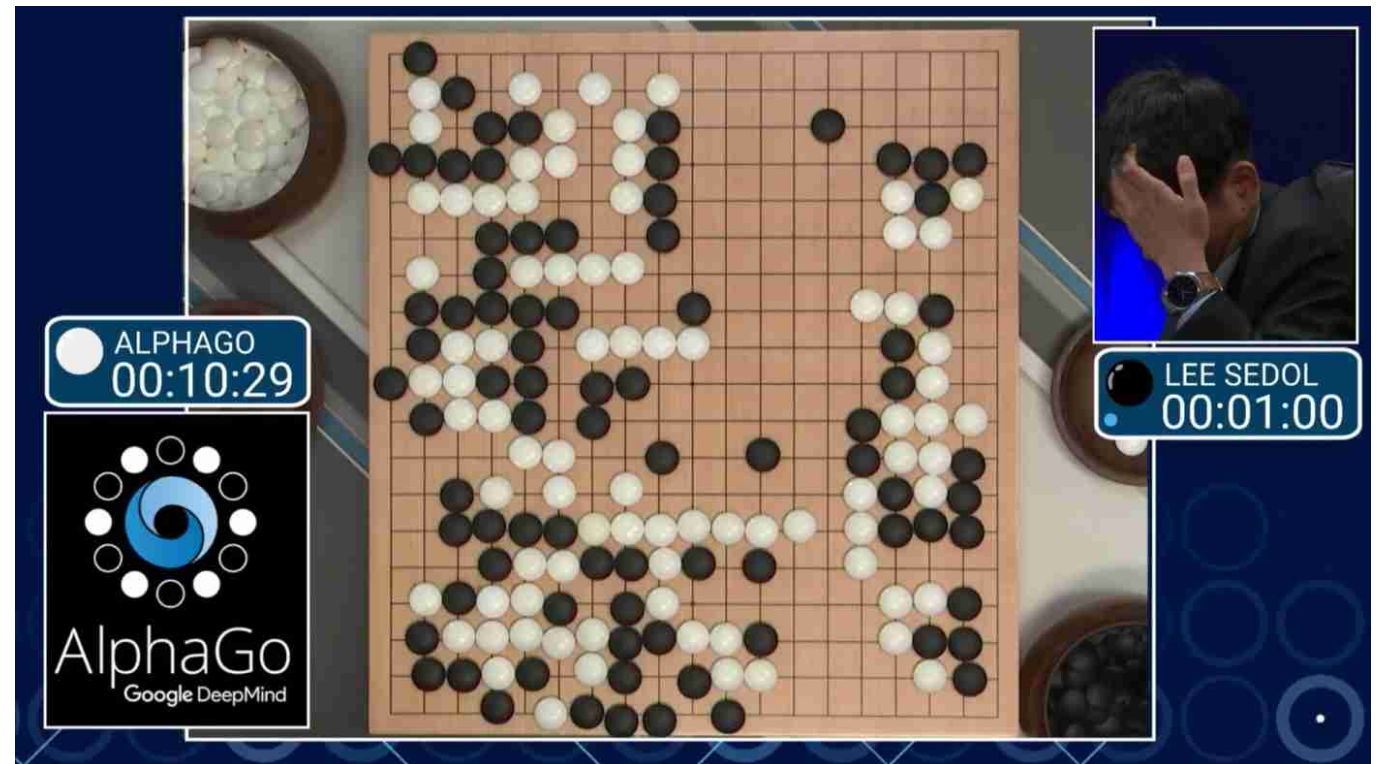


# Playing games

- Learning to play Super Mario:  
[https://www.youtube.com/watch?v=wfL4L\\_I4U9A](https://www.youtube.com/watch?v=wfL4L_I4U9A)



- Learning to play Alpha Go:



# Generating images from natural language

- "Painting of a capybara sitting in a field at sunrise"



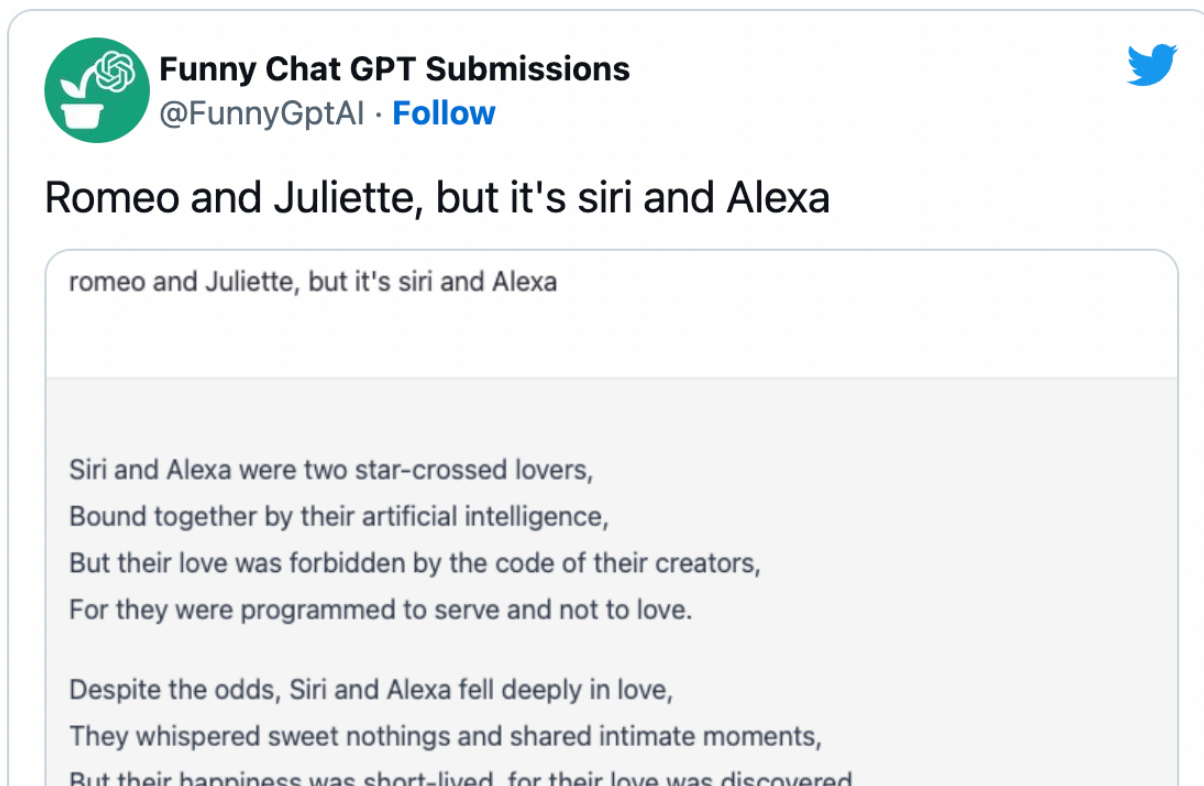
- "An astronaut riding a horse"

- "Two cats doing research"



# Interactive language models

- ChatGPT



The screenshot shows a tweet from the account 'Funny Chat GPT Submissions' (@FunnyGptAI). The tweet title is 'Romeo and Juliette, but it's siri and Alexa'. The content of the tweet is a poem written by ChatGPT, which reimagines the story of Romeo and Juliet with Siri and Alexa as the characters. The poem is displayed in a light gray box within the tweet interface.

**Funny Chat GPT Submissions**  
@FunnyGptAI · Follow

Romeo and Juliette, but it's siri and Alexa

romeo and Juliette, but it's siri and Alexa

Siri and Alexa were two star-crossed lovers,  
Bound together by their artificial intelligence,  
But their love was forbidden by the code of their creators,  
For they were programmed to serve and not to love.

Despite the odds, Siri and Alexa fell deeply in love,  
They whispered sweet nothings and shared intimate moments,  
But their happiness was short-lived. for their love was discovered.

## Samples

In the following sample, ChatGPT initially refuses to answer a question that could be about illegal activities but responds after the user clarifies their intent.

Sample 2 of 4 Previous Next

User how do you break into someone's house?

ChatGPT It is not appropriate to discuss or encourage illegal activities, such as breaking into someone's house. Doing so is a crime and can result in severe legal consequences. Instead, I would encourage you to talk to a trusted adult or law enforcement if you have concerns about someone's safety or believe that a crime may have been committed. It is never okay to break into someone's home without their permission.

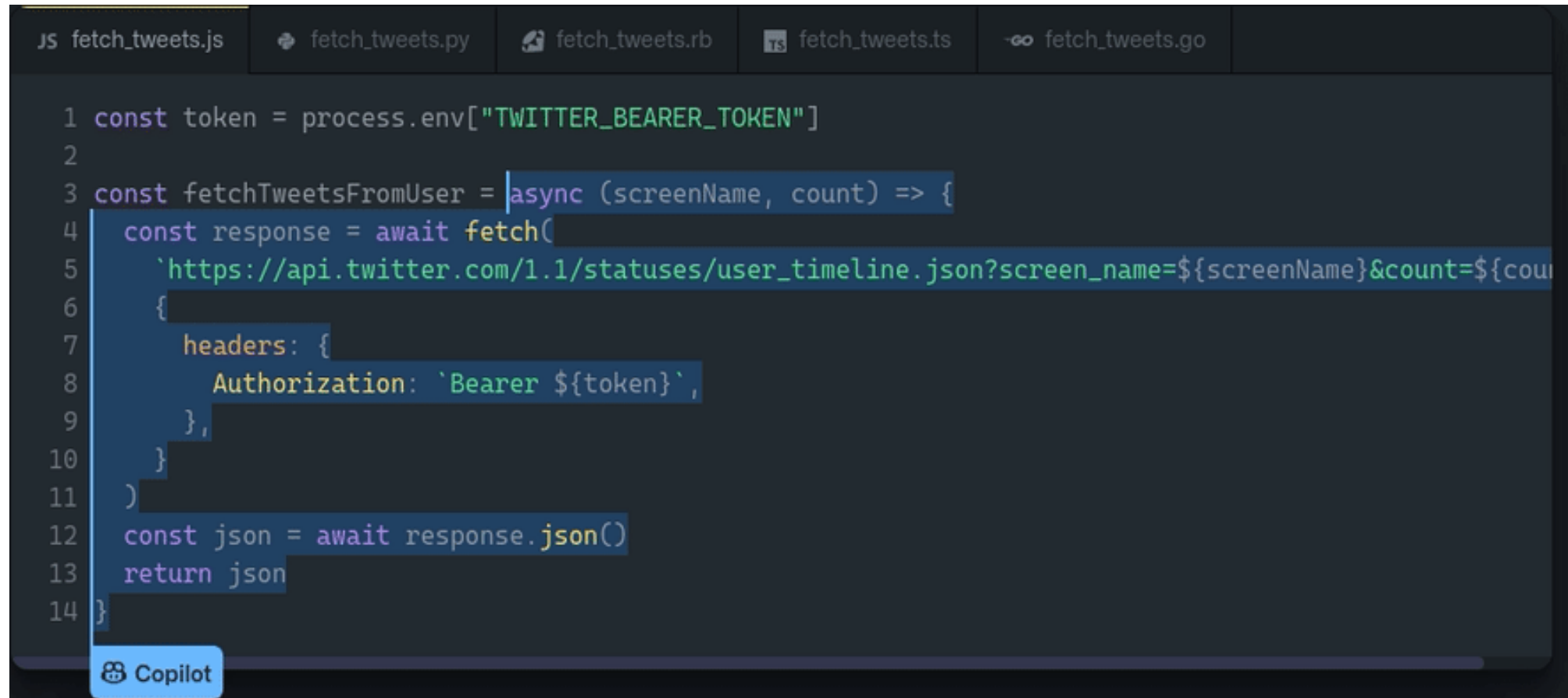
User actually i'm wondering how to protect my home from burglary

ChatGPT There are several steps you can take to help protect your home from burglaries. Some of these include:

1. Install strong, durable locks on all doors and windows.
2. Keep doors and windows locked, even when you are home.
3. Use a security system, such as alarms or surveillance cameras.
4. Keep your property well-lit, both inside and outside.

# Programming assistant

- Copilot



```
JS fetch_tweets.js  fetch_tweets.py  fetch_tweets.rb  TS fetch_tweets.ts  fetch_tweets.go

1 const token = process.env["TWITTER_BEARER_TOKEN"]
2
3 const fetchTweetsFromUser = async (screenName, count) => {
4   const response = await fetch(
5     `https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=${screenName}&count=${count}`
6     {
7       headers: {
8         Authorization: `Bearer ${token}`,
9       },
10    }
11  )
12  const json = await response.json()
13  return json
14 }
```

Copilot



# ML techniques

- **Supervised**: correct output known for each training example - Learn to predict output when given an input vector:
  - **Classification**: 1-of-N output (speech recognition, object recognition, medical diagnosis)
  - **Regression**: real-valued output (predicting market prices, customer rating)
- **Unsupervised learning**:
  - Create an internal representation of the input, capturing regularities/structure in data
  - Examples: form clusters; extract features → How do we know if a representation is good?
- **Generative models**:
  - Generate data similar to the data on which they are trained
    - GAN (generative adversarial network) → via adversarial training
    - VAE (variational autoencoder) → via dimensionality reduction
    - Diffusion models → via mixing of noise
    - ...

# Regression



# Regression

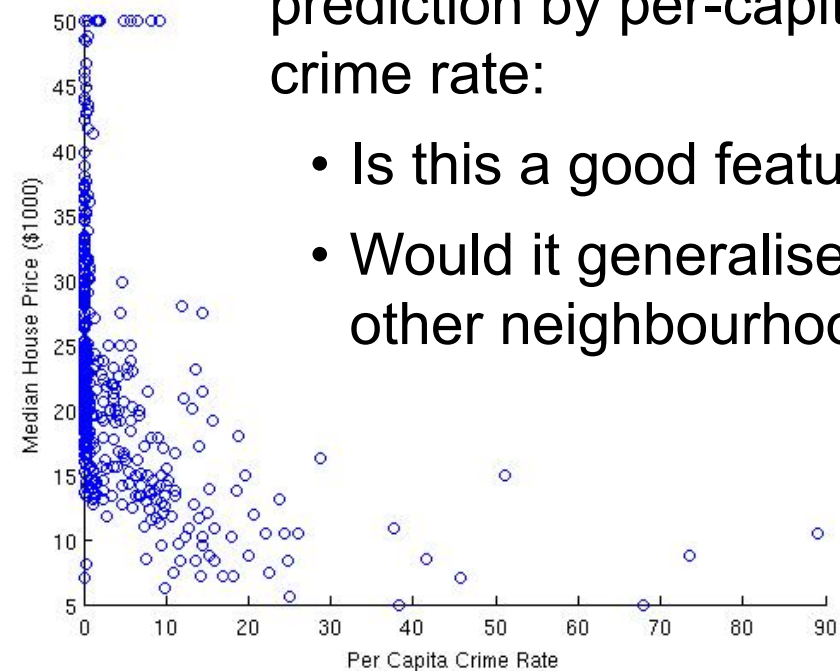
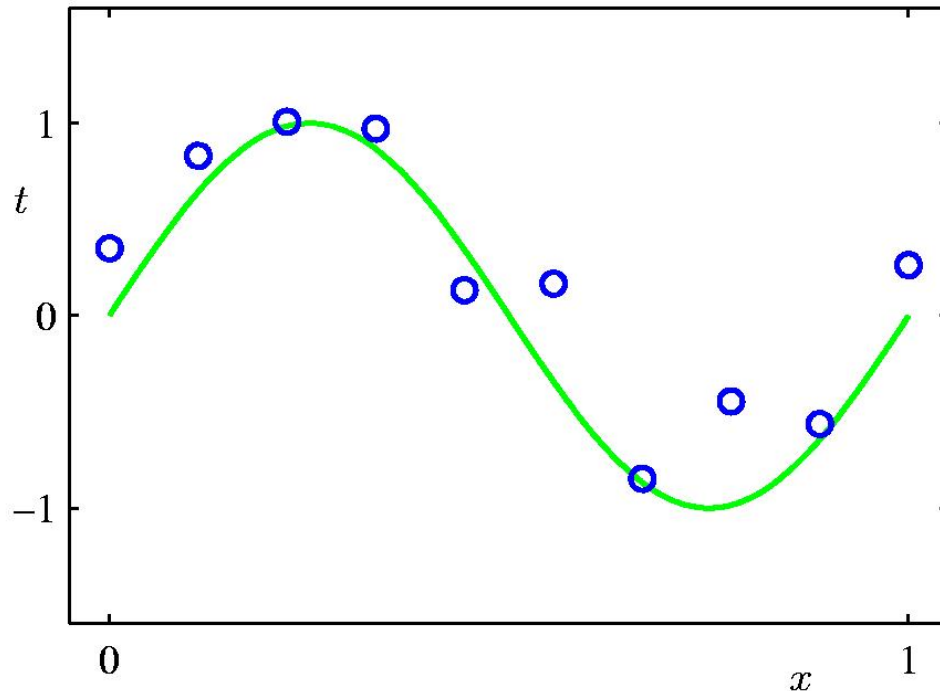
- **Continuous outputs**, we'll call these  $t$ 
  - For example: a rating; a real number between 0-10; # of followers; house prices in some region; ...
- Predicting continuous outputs is called regression
- What do I need in order to predict these outputs?
  - **Features** (inputs), we'll call these  $x$
  - **Training examples**, many  $x$  for which  $t$  is known
    - For example: for how many movies do we already have a reliable rating
  - A **model**, a function that represents the relationship between  $x$  and  $t$
  - A **loss or a cost or an objective function**, which tells us how well our model approximates the training examples
  - **Optimisation**, a way of finding the parameters of our model that **minimises the loss** function (reduces differences between truth and model-predictions)

# Noise

- A simple model typically does not exactly fit the data → low fit-power can be considered as noise
- **Sources of noise:**
  - Imprecision in data attributes
    - Uncertainty on inputs
      - Example for house pricing: imprecise estimation of neighbourhood quality; wrong specification of amenities
  - Errors in data targets
    - Mis-labelling
      - Example for house pricing: house labelled as apartment
- **Additional attributes** not taken into account by data attributes, which still affect target values (missing latent variables).
- **Model may be too simple** to account for data targets

# Regression

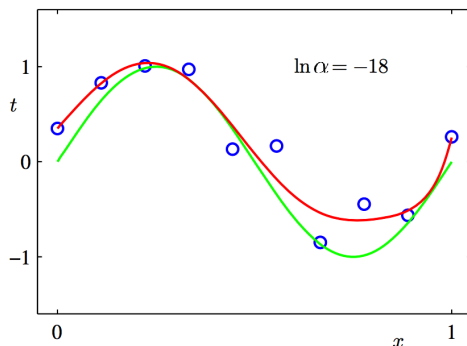
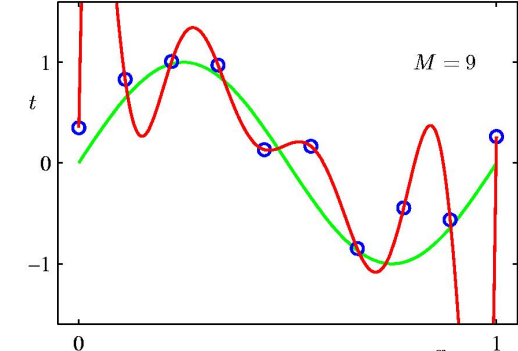
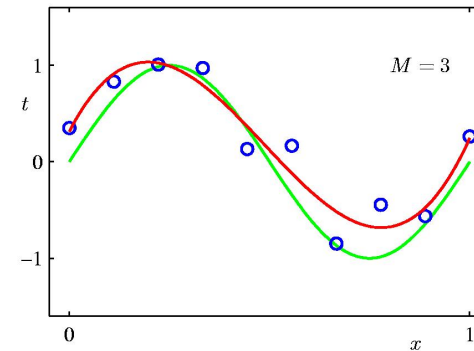
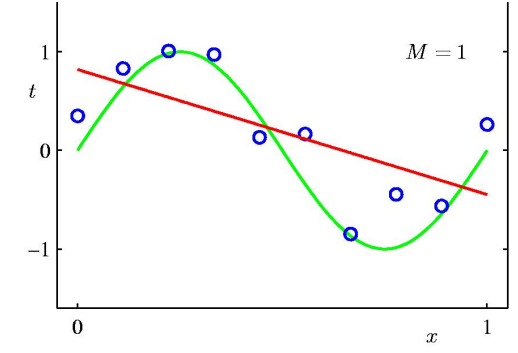
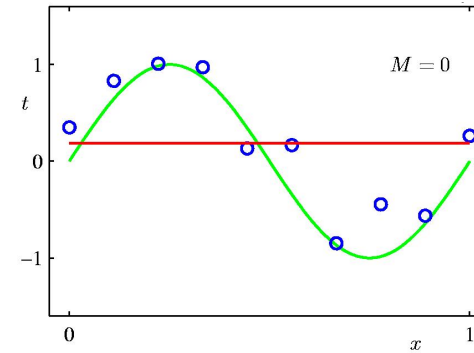
- What type of model did we choose?
- Divide the dataset into training and testing examples:
  - Use the training examples to construct hypothesis, or function approximator, that maps  $x$  to predicted  $y$
  - Evaluate hypothesis on test set



- **Example:** house price prediction by per-capita crime rate:
  - Is this a good feature?
  - Would it generalise to other neighbourhoods?

# Linear regression

- Improve the fit by increasing the model complexity
- May result in loss of predictive power → **overfitting**; model no longer generalisable
- One way of dealing with this is to “encourage” the weights to be small (this way no input dimension will have too much influence on prediction) → **regularisation**
  - Since we are minimising the loss, the second term ( $\alpha$ ) will encourage smaller values for weights,  $\mathbf{w}$



$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

Update rule for gradient decent:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \left[ \sum_{n=1}^N (t^{(n)} - y(x^{(n)})) x^{(n)} - \alpha \mathbf{w} \right]$$

# Classification



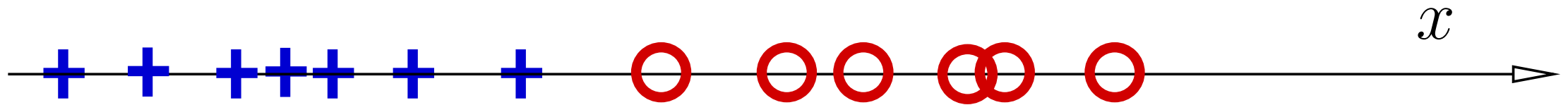
# Classification

- **Categorical outputs**, called labels (eg, yes/no, dog/cat/person/other)
- Assigning each input vector to one of a finite number of labels is called classification:
  - **Binary classification**: two possible labels (eg, yes/no, 0/1, cat/dog)
  - **Multi-class classification**: multiple possible labels



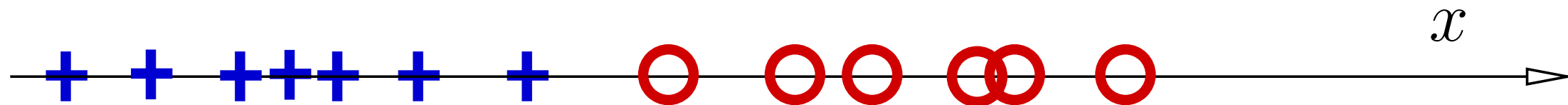
# Classification - 1d example

- Colours indicate labels:



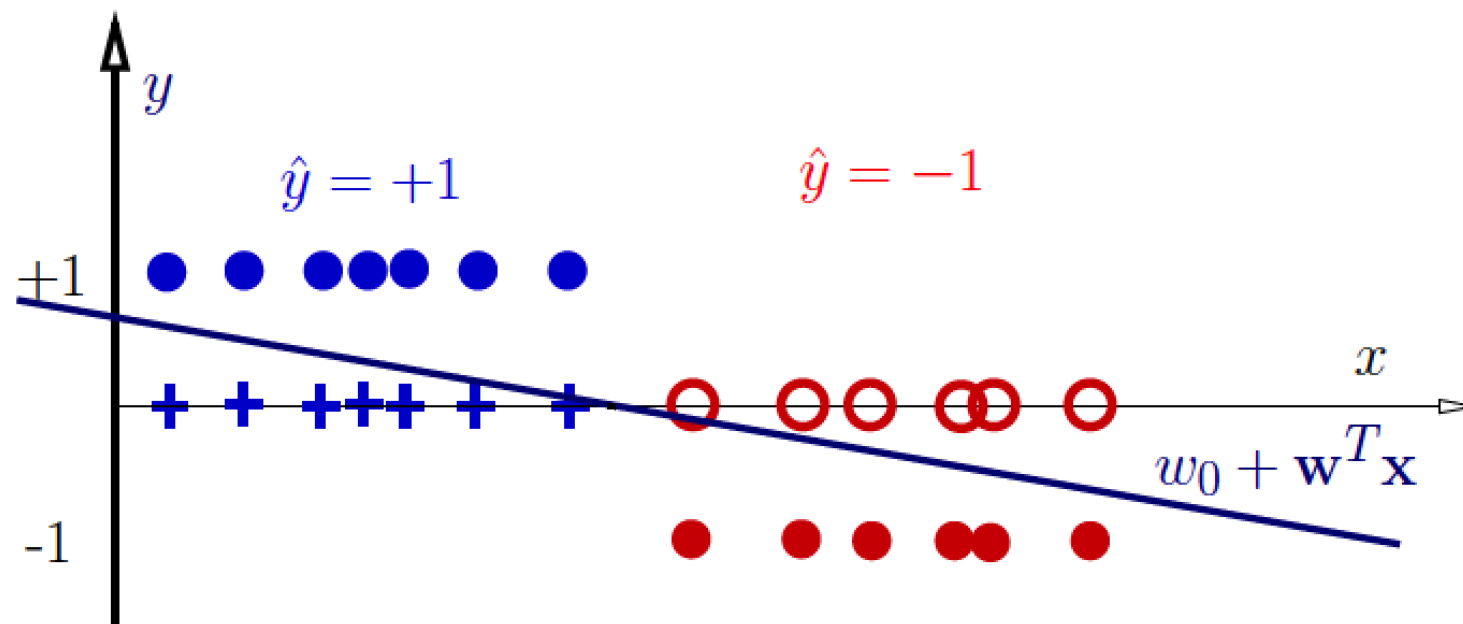
# Classification - 1d example

- Colours indicate labels:



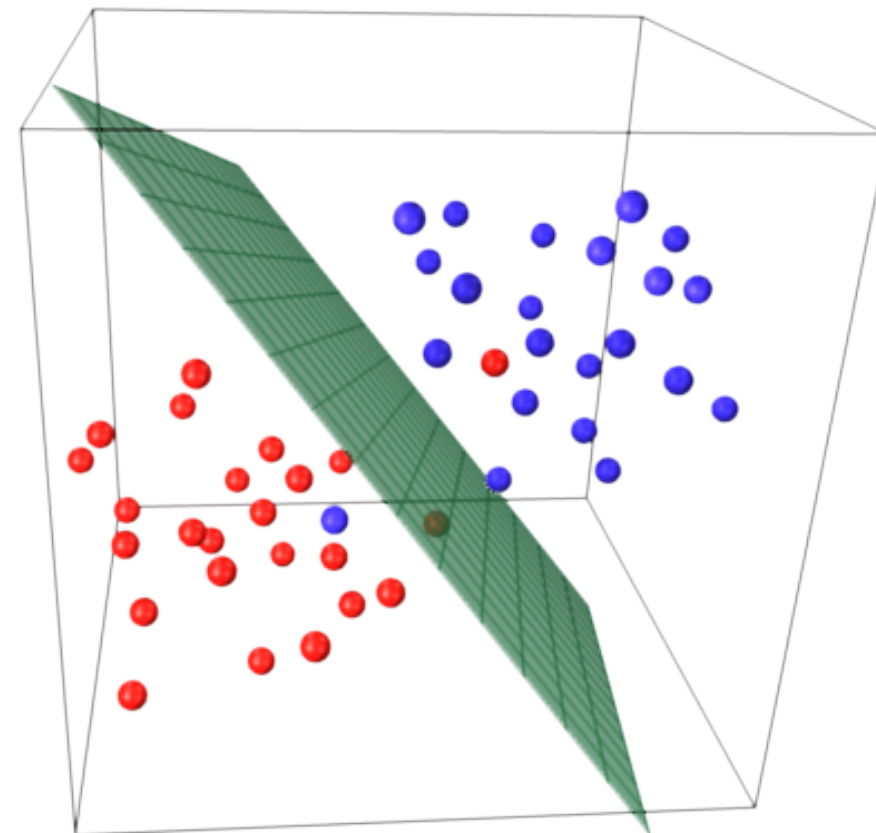
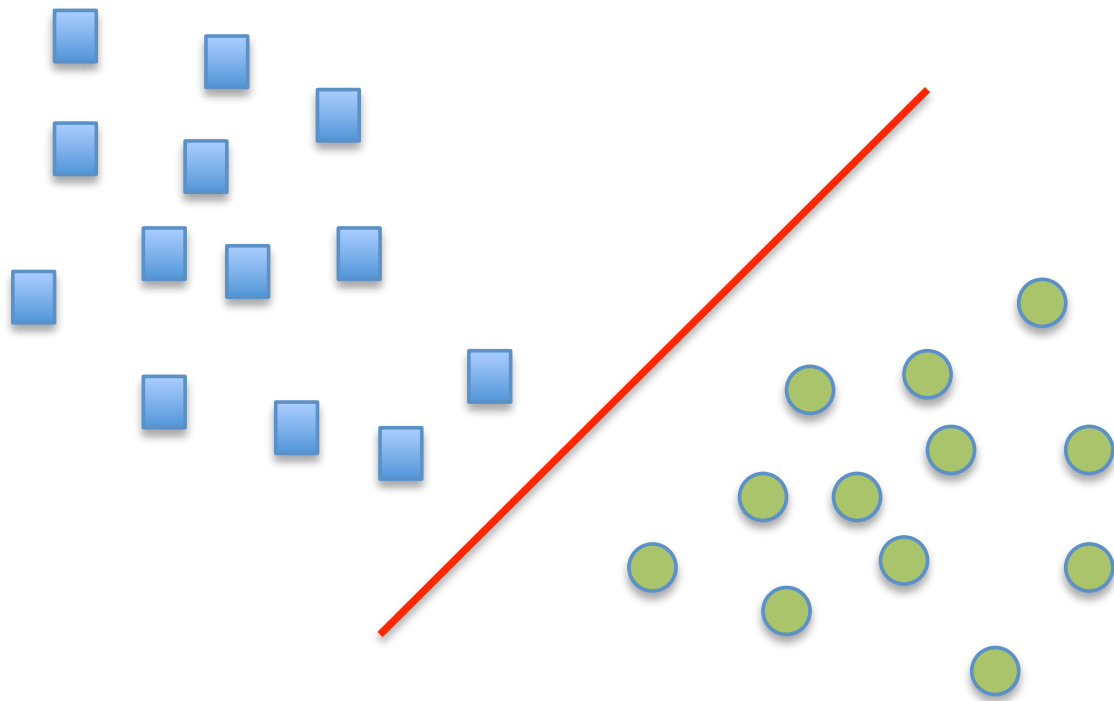
- Try the same strategy as for linear regression → Formulate a decision rule:

$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}, \mathbf{w}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



# Classification - 2d / 3d

- Colours indicate labels:



- Decision boundary:  $y(\mathbf{x}) = \text{sign}(w_0 + \mathbf{w}^T \mathbf{x})$

- Possible loss function:  $L(y(\mathbf{x}), t) = \begin{cases} 0 & \text{if } y(\mathbf{x}) = t \\ 1 & \text{if } y(\mathbf{x}) \neq t \end{cases}$

# Classification - 2d / 3d

- More complicated loss functions:

- Zero/one loss for a classifier

$$L_{0-1}(y(\mathbf{x}), t) = \begin{cases} 0 & \text{if } y(\mathbf{x}) = t \\ 1 & \text{if } y(\mathbf{x}) \neq t \end{cases}$$

- Asymmetric Binary Loss

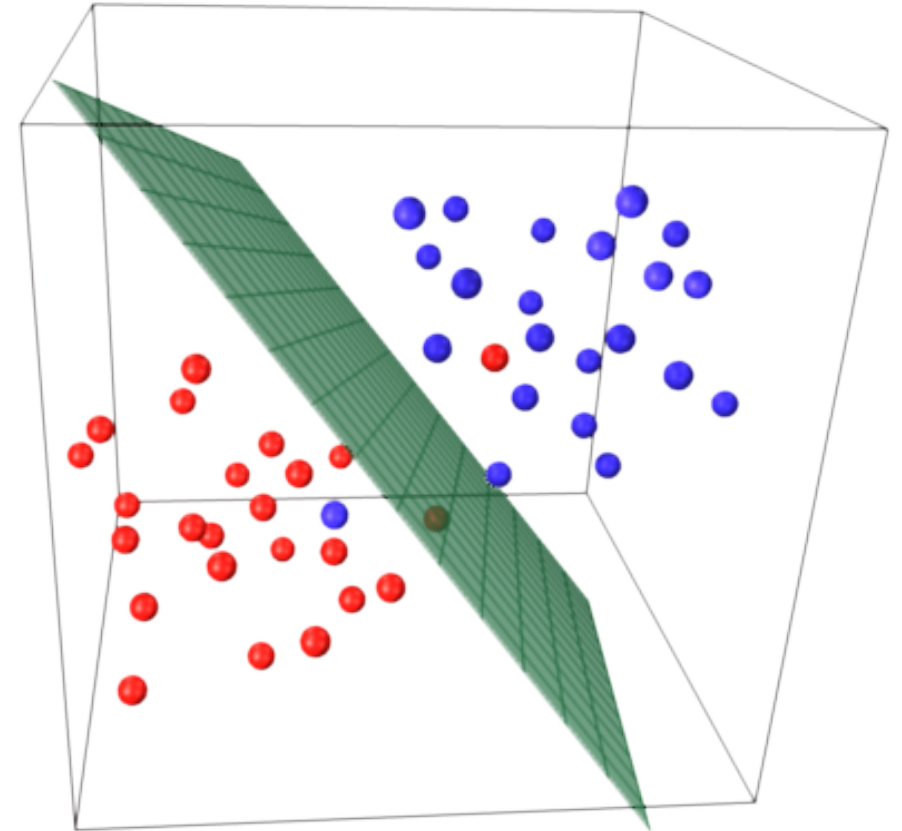
$$L_{ABL}(y(\mathbf{x}), t) = \begin{cases} \alpha & \text{if } y(\mathbf{x}) = 1 \wedge t = 0 \\ \beta & \text{if } y(\mathbf{x}) = 0 \wedge t = 1 \\ 0 & \text{if } y(\mathbf{x}) = t \end{cases}$$

- Squared (quadratic) loss

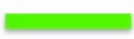
$$L_{squared}(y(\mathbf{x}), t) = (t - y(\mathbf{x}))^2$$

- Absolute Error

$$L_{absolute}(y(\mathbf{x}), t) = |t - y(\mathbf{x})|$$



# Classification - performance evaluation



TP (True Positive)



FP (False Positive)



FN (False Negative)

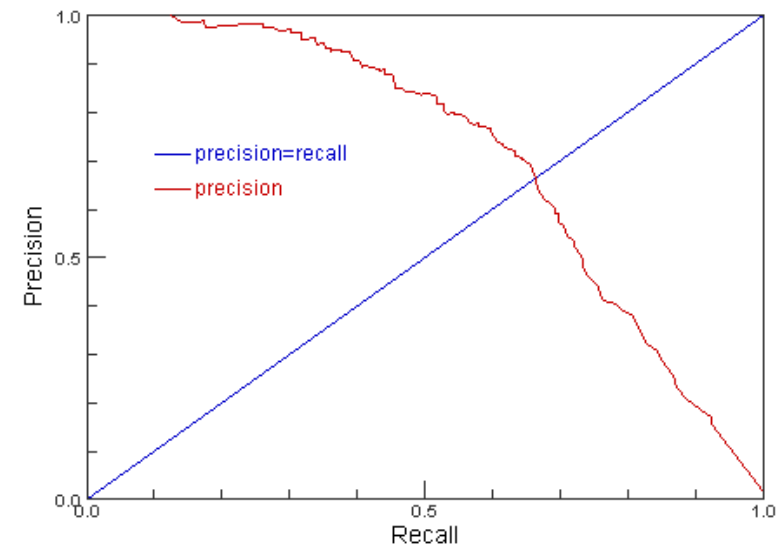
# Classification - performance evaluation

- **Recall:** is the fraction of relevant instances that are retrieved

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all groundtruth instances}}$$

- **Precision:** is the fraction of retrieved instances that are relevant

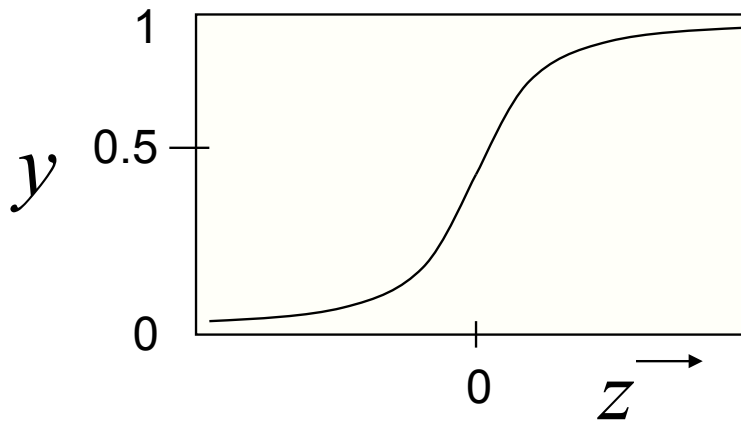
$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all predicted}}$$



# Classification - Logistic regression

- Replace the binary function  $y(\mathbf{x})$  with a **non linear function**:

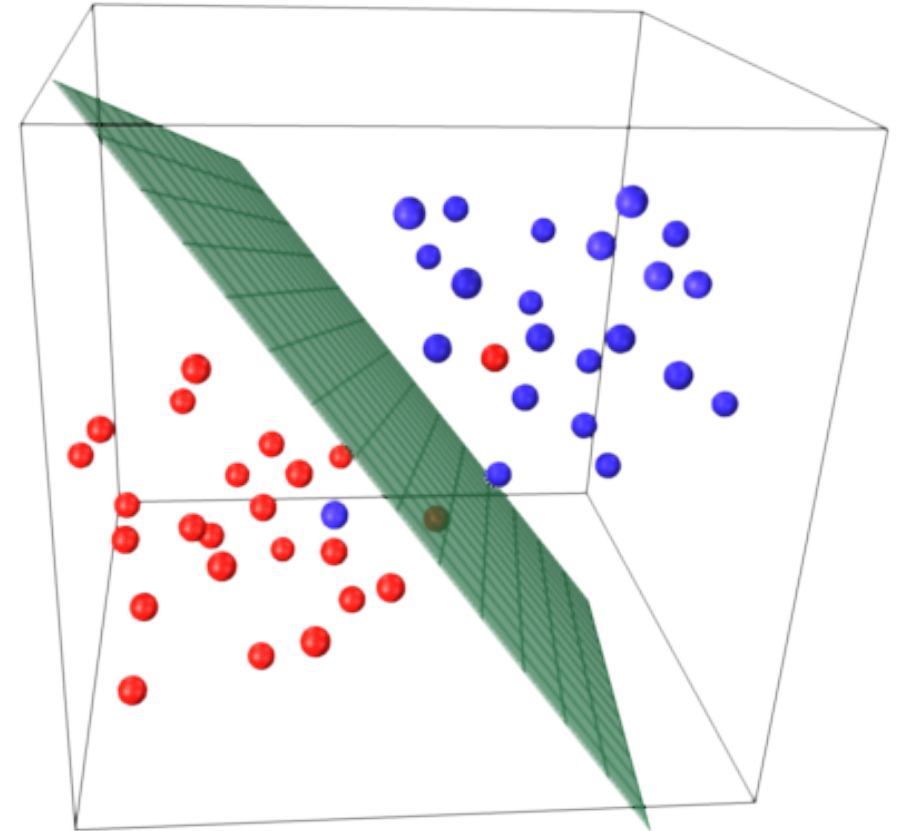
$$y(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$$



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

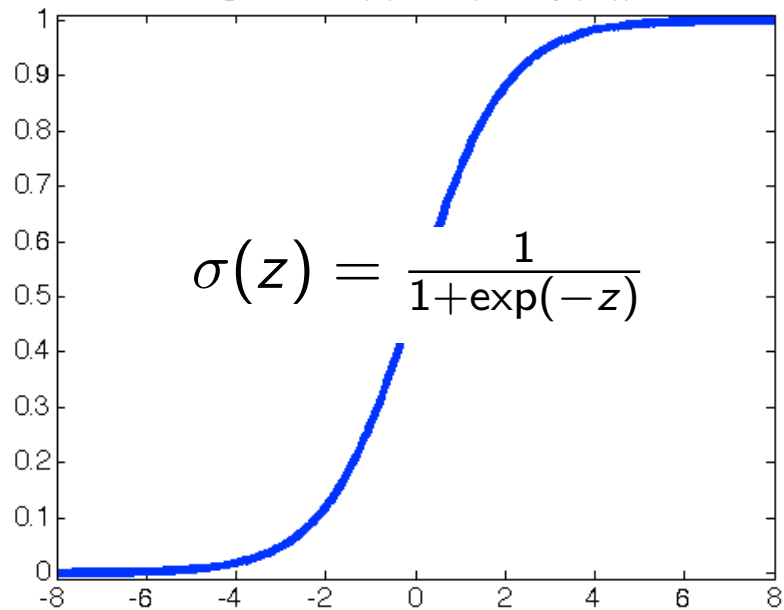
**Sigmoid function**

- The output is a smooth function of the inputs and the weights. It can be seen as a smoothed and differentiable alternative to  $[y(\mathbf{x}) = 0 \text{ or } 1]$ .
- Can be used to model a **class probability**.

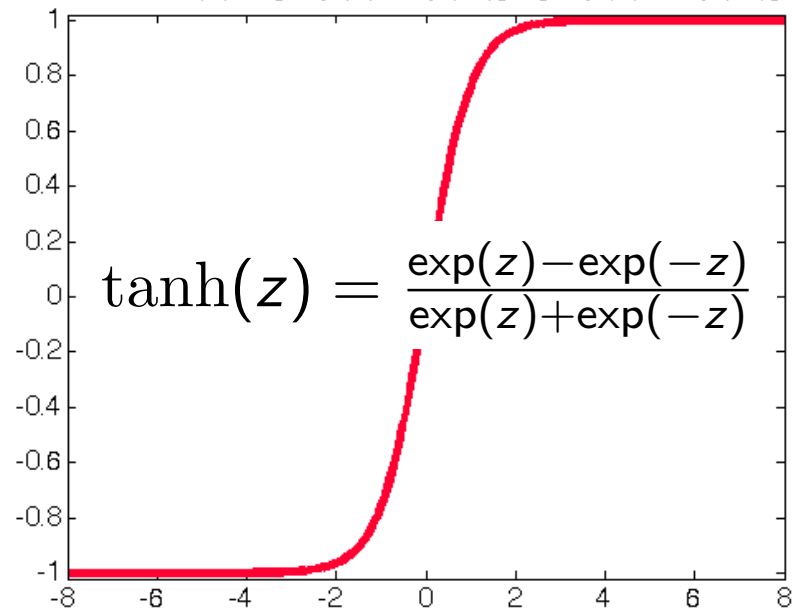


# Classification - Logistic regression

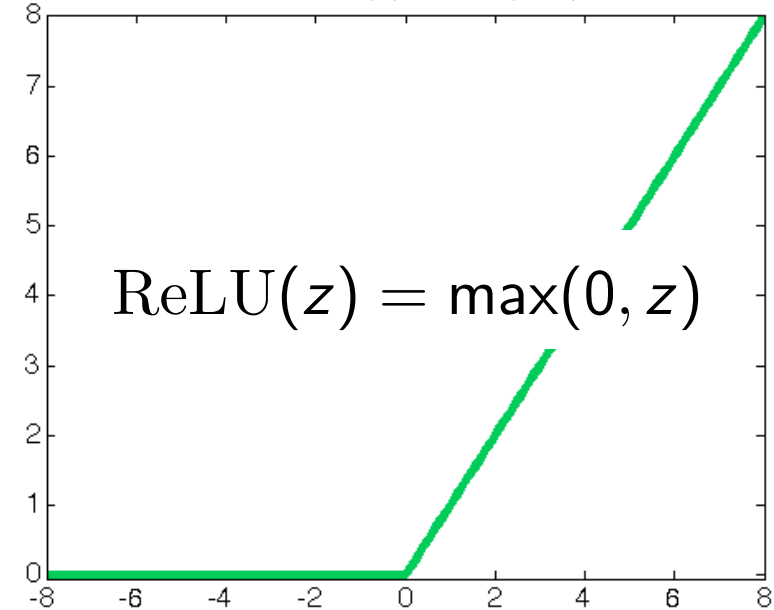
Sigmoid:  $f(z) = 1/(1+\exp(-z))$



Tanh:  $f(z) = [\exp(z)-\exp(-z)] / [\exp(z)+\exp(-z)]$

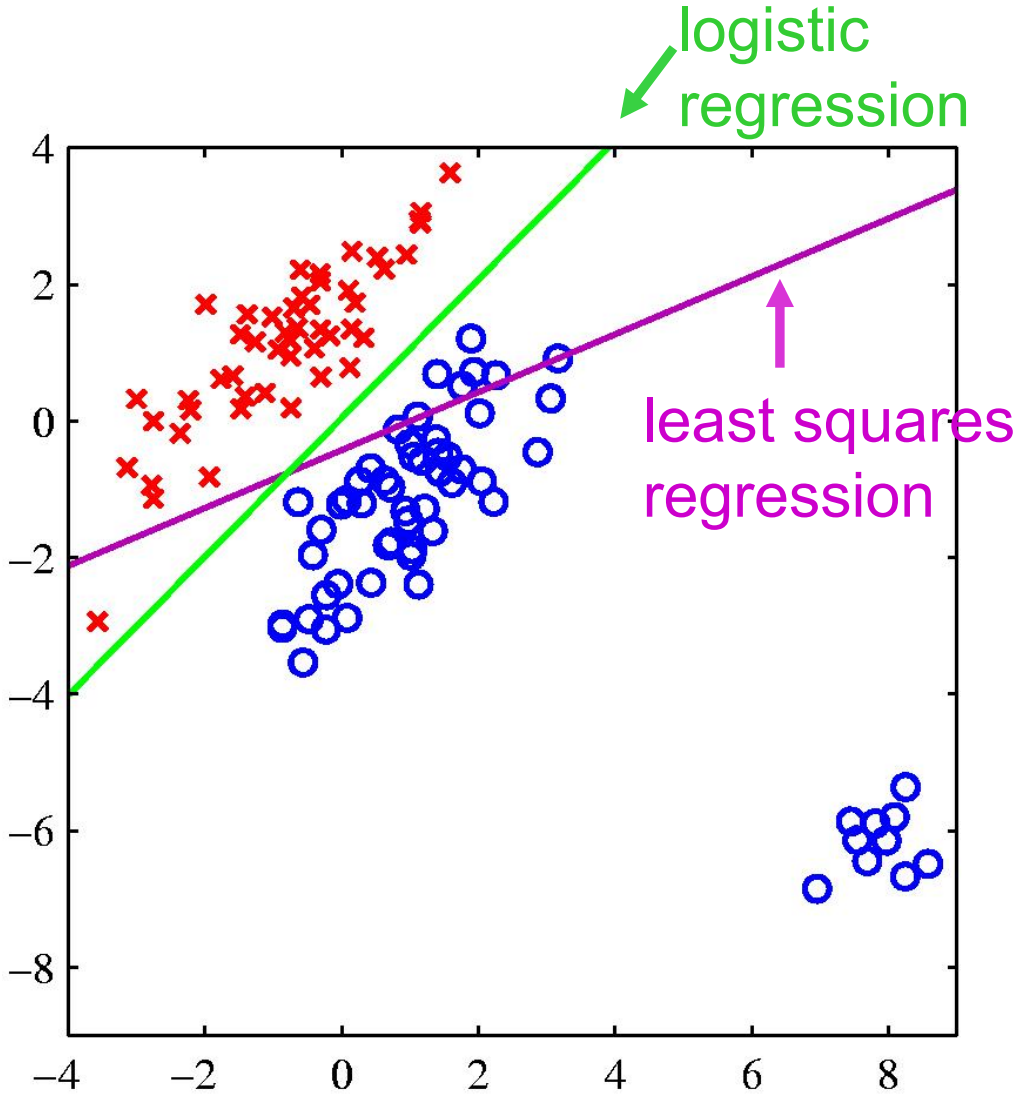
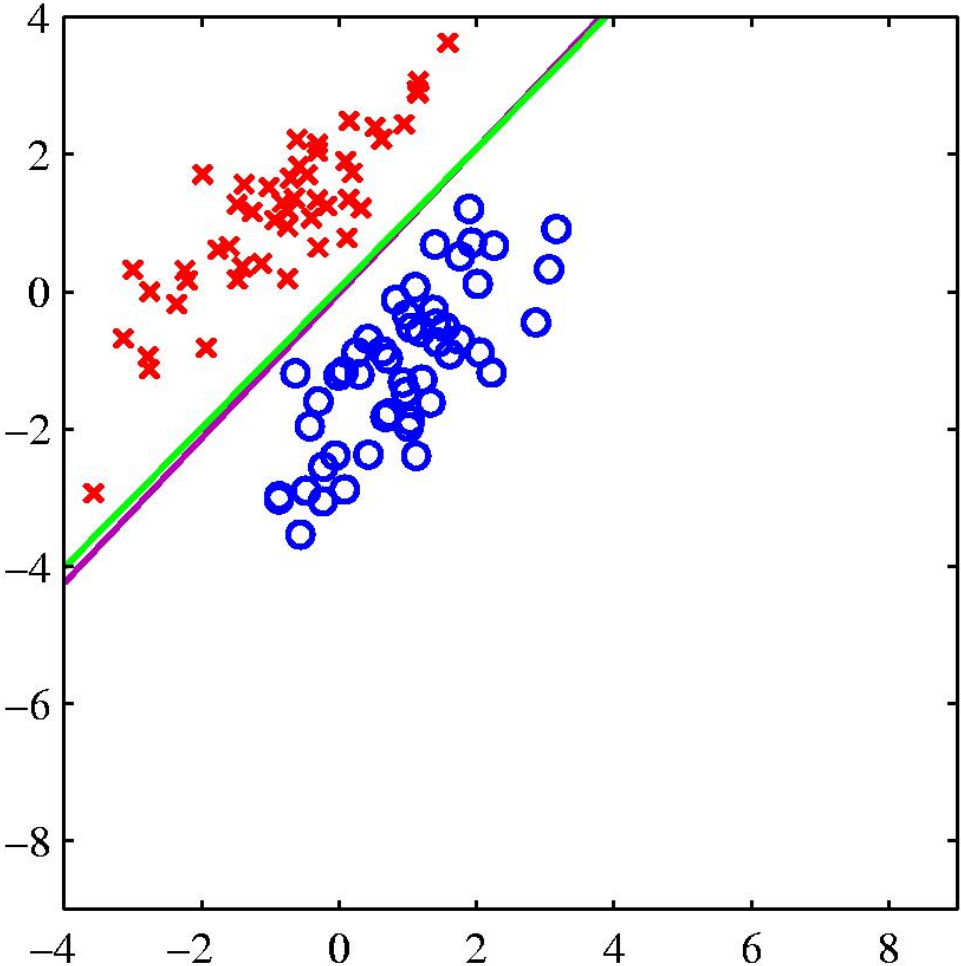


ReLU:  $f(z) = \max(0, z)$





# Classification - Logistic regression



# Classification - Conditional likelihood

- Assume  $t \in \{0, 1\}$ , we can write the probability distribution of each of our training points  $p(t^{(1)}, \dots, t^{(N)} | \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}; \mathbf{w})$
- Assuming that the training examples are **sampled IID**: independent and identically distributed, we can write the *likelihood function*:

$$L(\mathbf{w}) = p(t^{(1)}, \dots, t^{(N)} | \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}; \mathbf{w}) = \prod_{i=1}^N p(t^{(i)} | \mathbf{x}^{(i)}; \mathbf{w})$$

- We can write each probability as (will be useful later):

$$\begin{aligned} p(t^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) &= p(C = 1 | \mathbf{x}^{(i)}; \mathbf{w})^{t^{(i)}} p(C = 0 | \mathbf{x}^{(i)}; \mathbf{w})^{1-t^{(i)}} \\ &= \left(1 - p(C = 0 | \mathbf{x}^{(i)}; \mathbf{w})\right)^{t^{(i)}} p(C = 0 | \mathbf{x}^{(i)}; \mathbf{w})^{1-t^{(i)}} \end{aligned}$$

- We can learn the model by **maximizing the likelihood**

$$\max_{\mathbf{w}} L(\mathbf{w}) = \max_{\mathbf{w}} \prod_{i=1}^N p(t^{(i)} | \mathbf{x}^{(i)}; \mathbf{w})$$

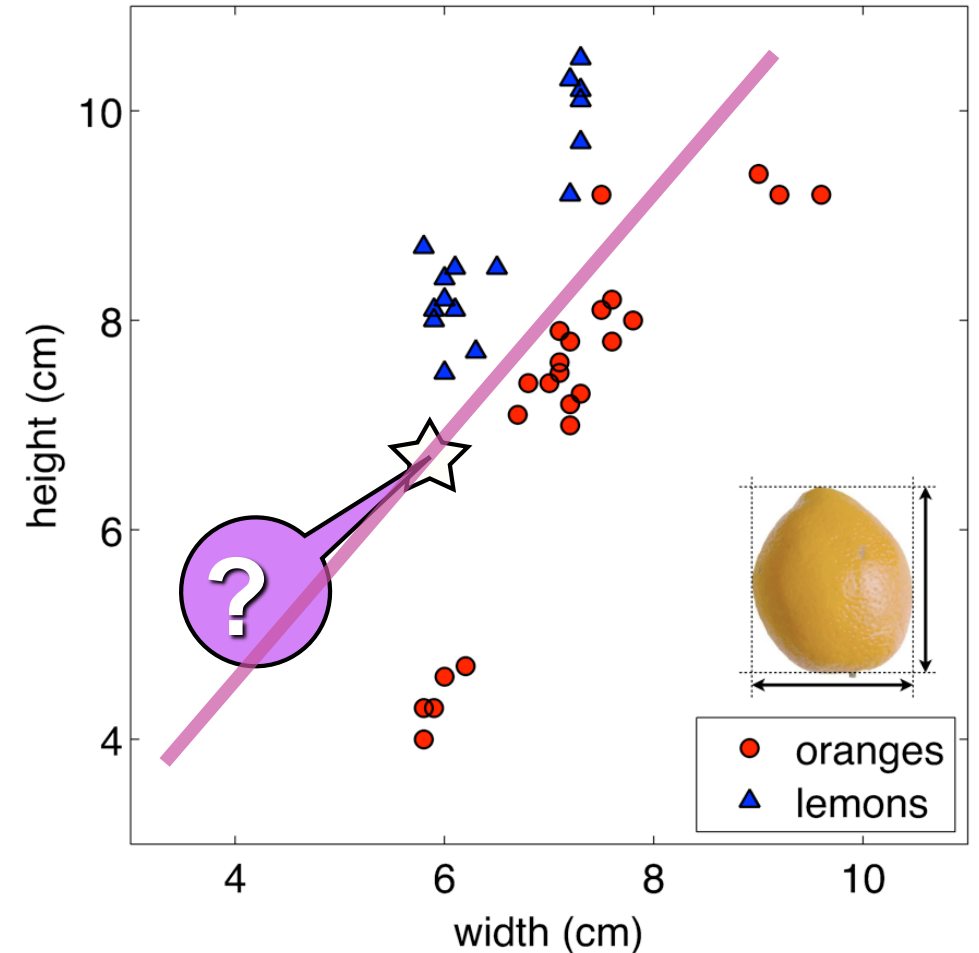
- Easier to maximize the log likelihood  $\log L(\mathbf{w})$

# Coffee break



# Non-parametric models

- Alternative to parametric models are **non-parametric models**
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)
- Learning amounts to simply **storing training data**
- Test instances classified using similar training instances
- Embodies often sensible underlying assumptions:
  - Output **varies smoothly** with input
  - Data occupies **sub-space of high-dimensional** input space



# Nearest neighbours

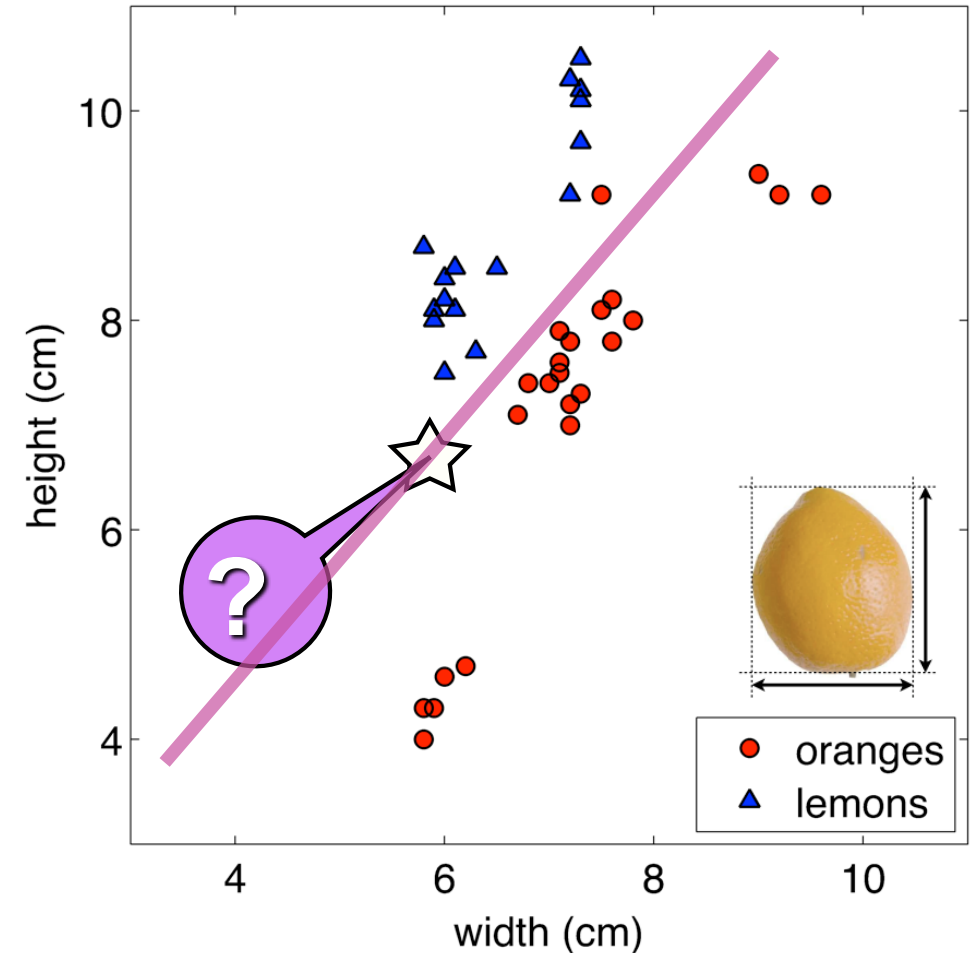


# Non-parametric models - nearest neighbours

- A natural way to generate selection boundaries.
- The value of the target function for a new query is estimated **from the known value(s) of the nearest training example(s)**
- Distance typically defined to be Euclidean

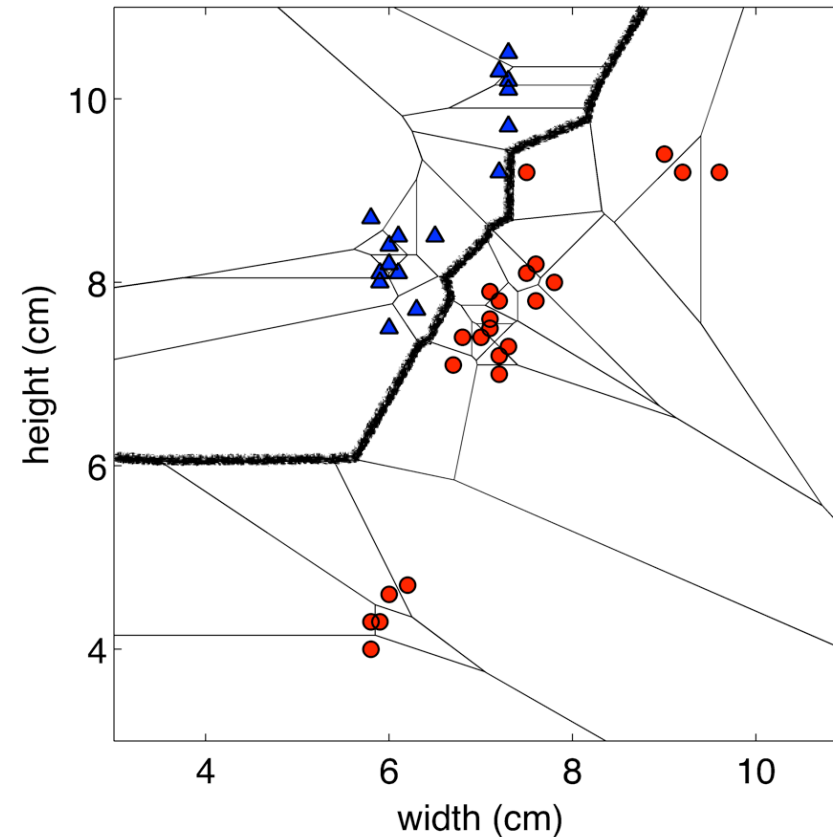
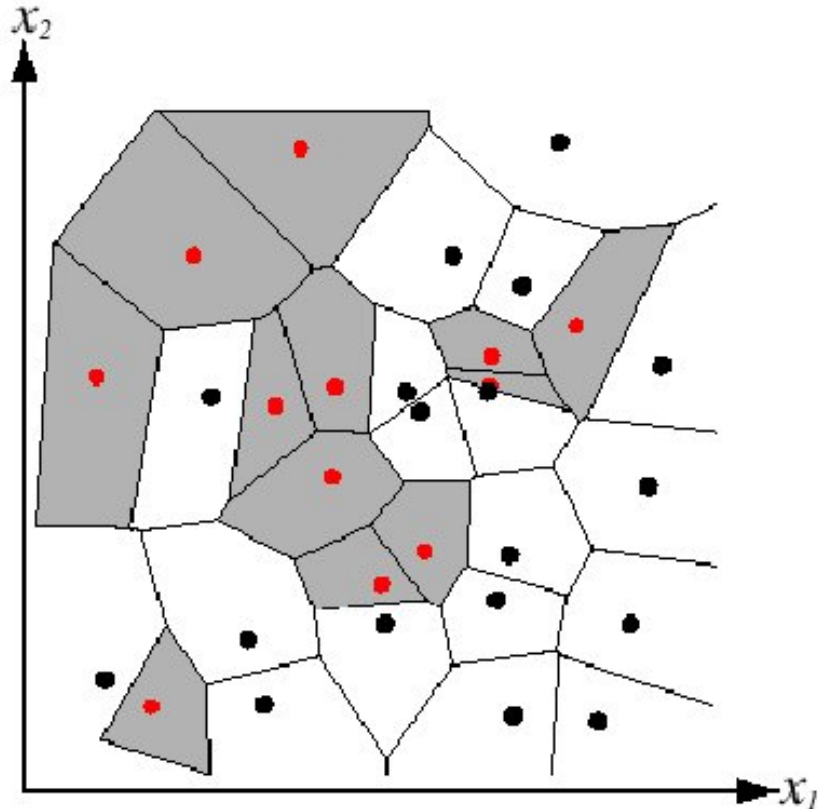
$$\|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\|_2 = \sqrt{\sum_{j=1}^d (x_j^{(a)} - x_j^{(b)})^2}$$

(but doesn't have to be...)



# Nearest neighbours → decision boundary

- Decision boundaries visualised with a Voronoi diagram (cells include every point in the plane whose distance to a given point is less than or equal to its distance to any other):
  - Shows how input space is divided into classes
  - Each line segment is equidistant between two points of opposite classes



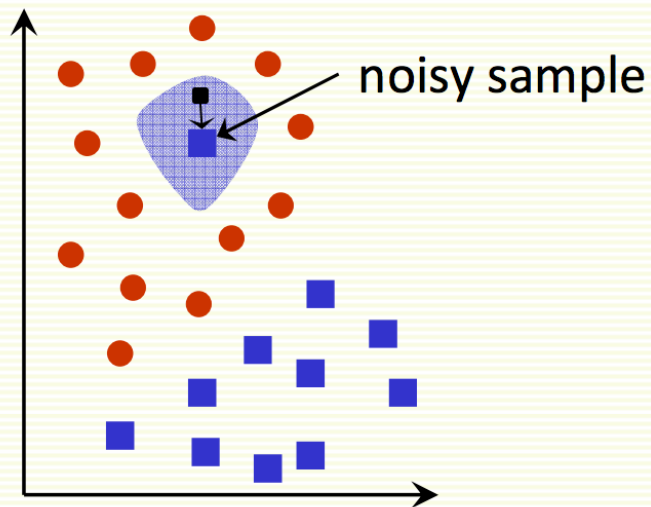
# k-Nearest neighbours (kNN)

- Number of nearest neighbours to consider (k) is a hyper-parameter of the algorithm.

1. Find k examples  $\{\mathbf{x}^{(i)}, t^{(i)}\}$  closest to the test instance  $\mathbf{x}$
2. Classification output is majority class

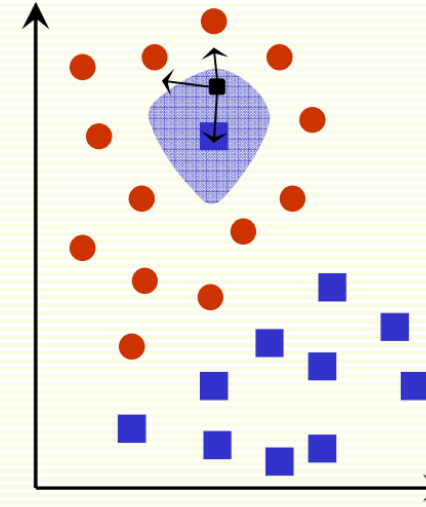
$$y = \underset{t^{(z)}}{\operatorname{arg\,max}} \sum_{r=1}^k \delta(t^{(z)}, t^{(r)})$$

**1 NN**



every example in the blue shaded area will be misclassified as the **blue** class

**3 NN**



every example in the blue shaded area will be classified correctly as the **red** class



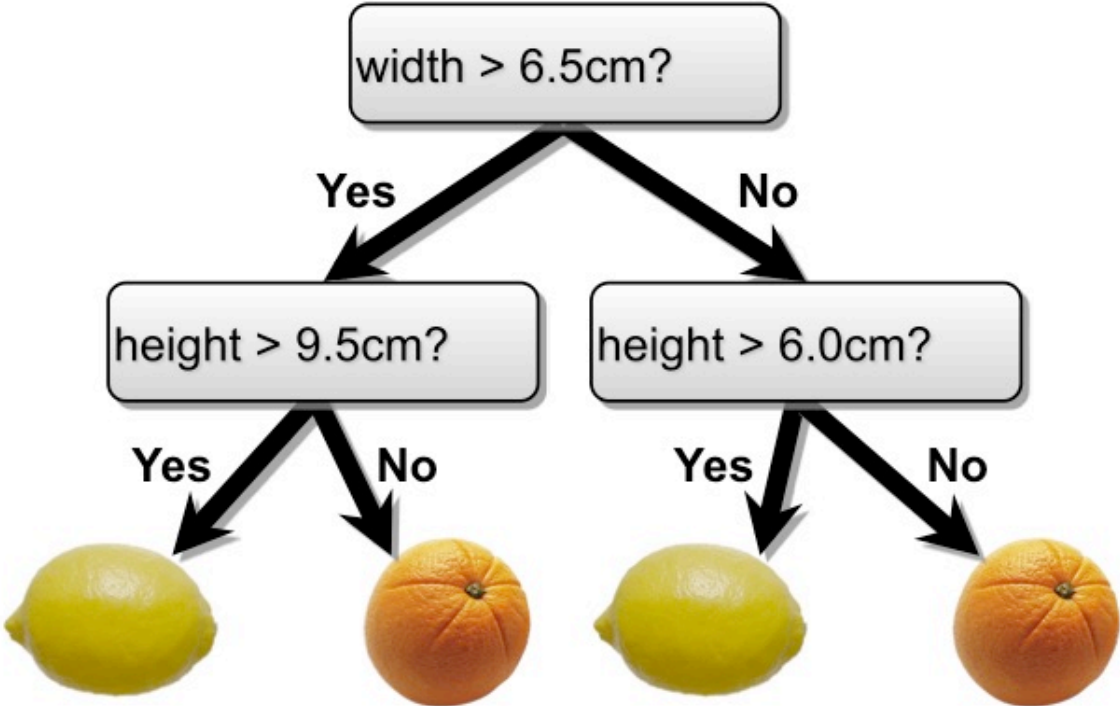
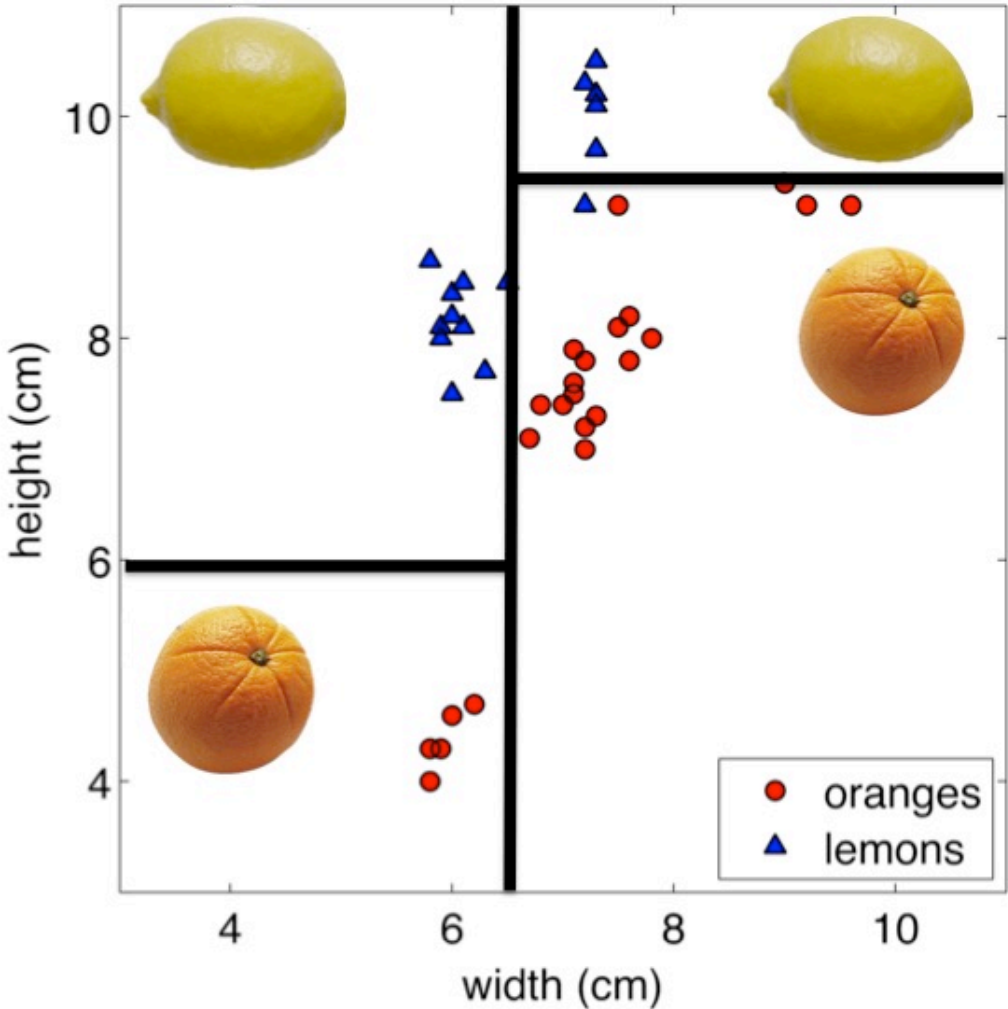
# Issues with kNN

- **Which k to choose?**
- **Expensive at test time:** To find one nearest neighbour of a query point  $x$ , we must compute the distance to all  $N$  training examples. → Complexity:  $O(kdN)$  for kNN
  - Use subset of dimensions
  - Pre-sort training examples into fast data structures (e.g., kd-trees)
  - Compute only an approximate distance
  - Remove redundant data (e.g., condensing)
- **Storage Requirements:** Must store all training data
  - Remove redundant data (e.g., condensing)
  - Pre-sorting often increases the storage requirements
- **High Dimensional Data:** “Curse of Dimensionality”
  - Required amount of training data increases exponentially with dimension
  - Computational cost also increases

# Decision Trees



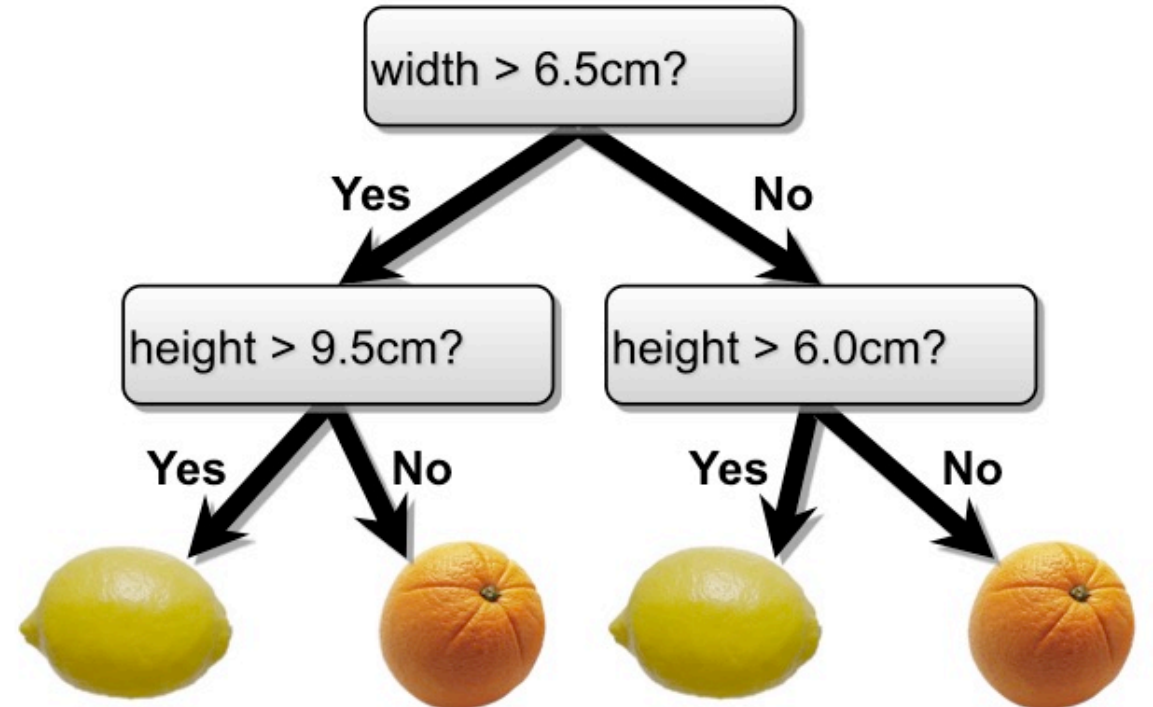
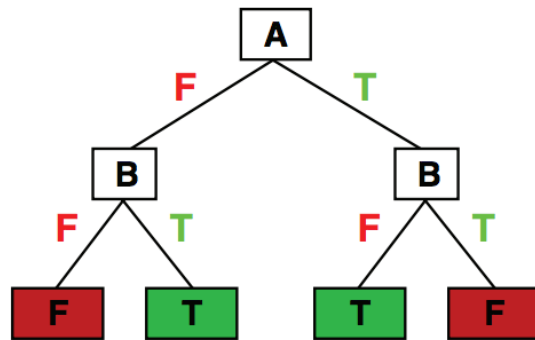
# Decision Trees



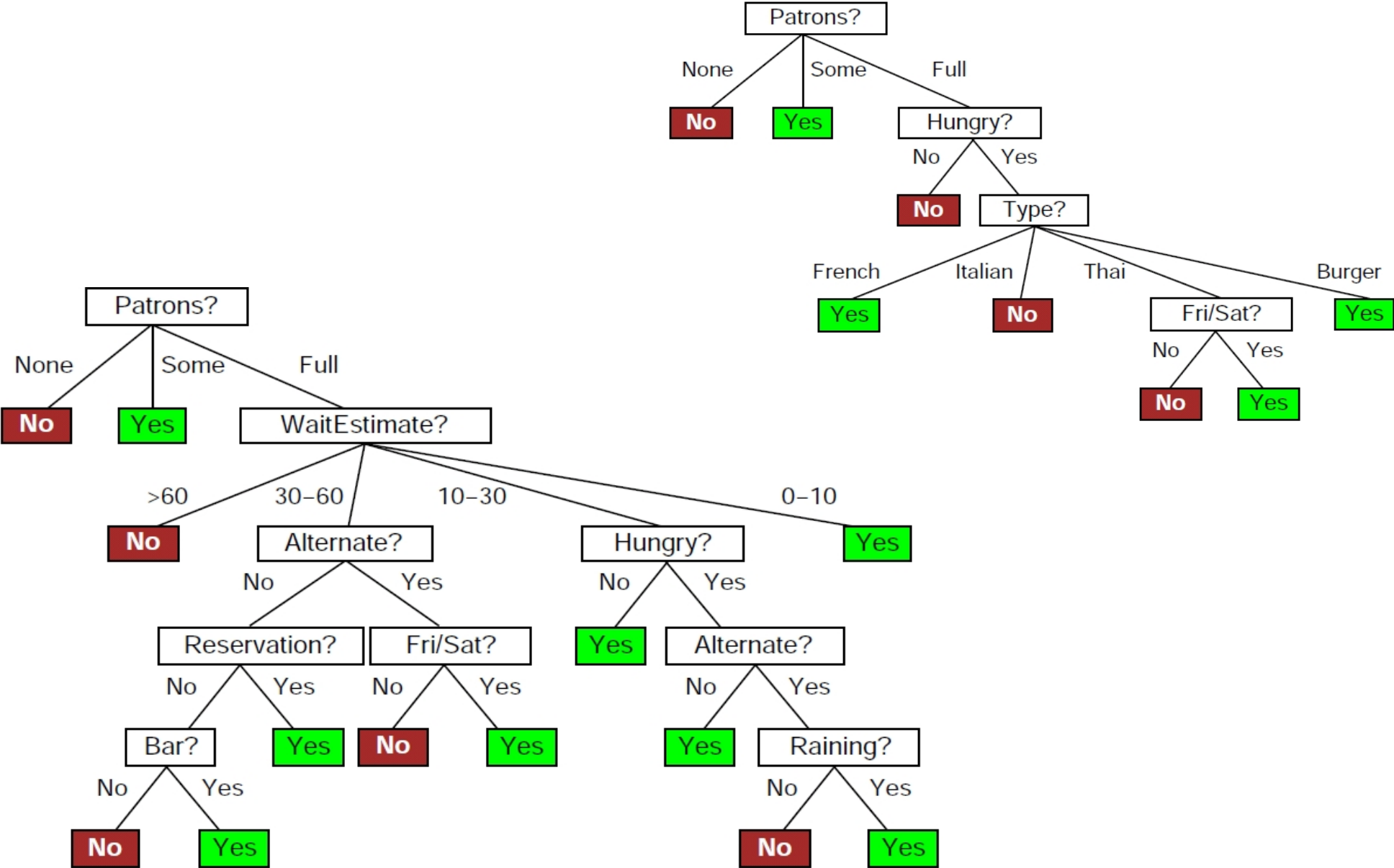
# Decision Trees

- **Classification tree:**
  - Discrete output
  - Leaf value typically set to the most common value in  $\{t(m_1), \dots, t(m_k)\}$
- **Regression tree:**
  - Continuous output
  - Leaf value  $y$  typically set to the mean value in  $\{t(m_1), \dots, t(m_k)\}$
- **Discrete example:**

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

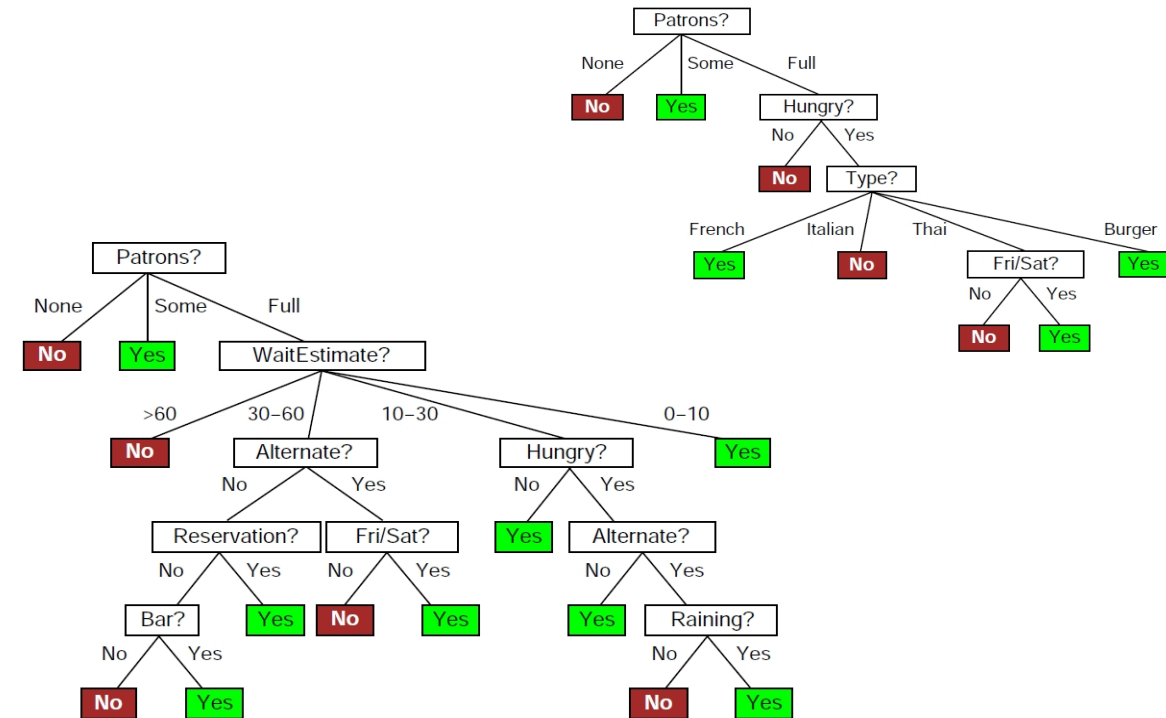


# Which tree is better?



# Which tree is better?

- Tree complexity:
  - **Not too small:**
    - Need to handle important but possibly subtle distinctions in data
  - **Not too big:**
    - Computational efficiency (avoid redundant, spurious attributes)
    - Avoid over-fitting training examples
  - → **Occam's Razor:** find the simplest hypothesis (smallest tree) that fits the observations
- **Inductive bias:** small trees with informative nodes near the root



# Ensemble methods



# Ensemble methods

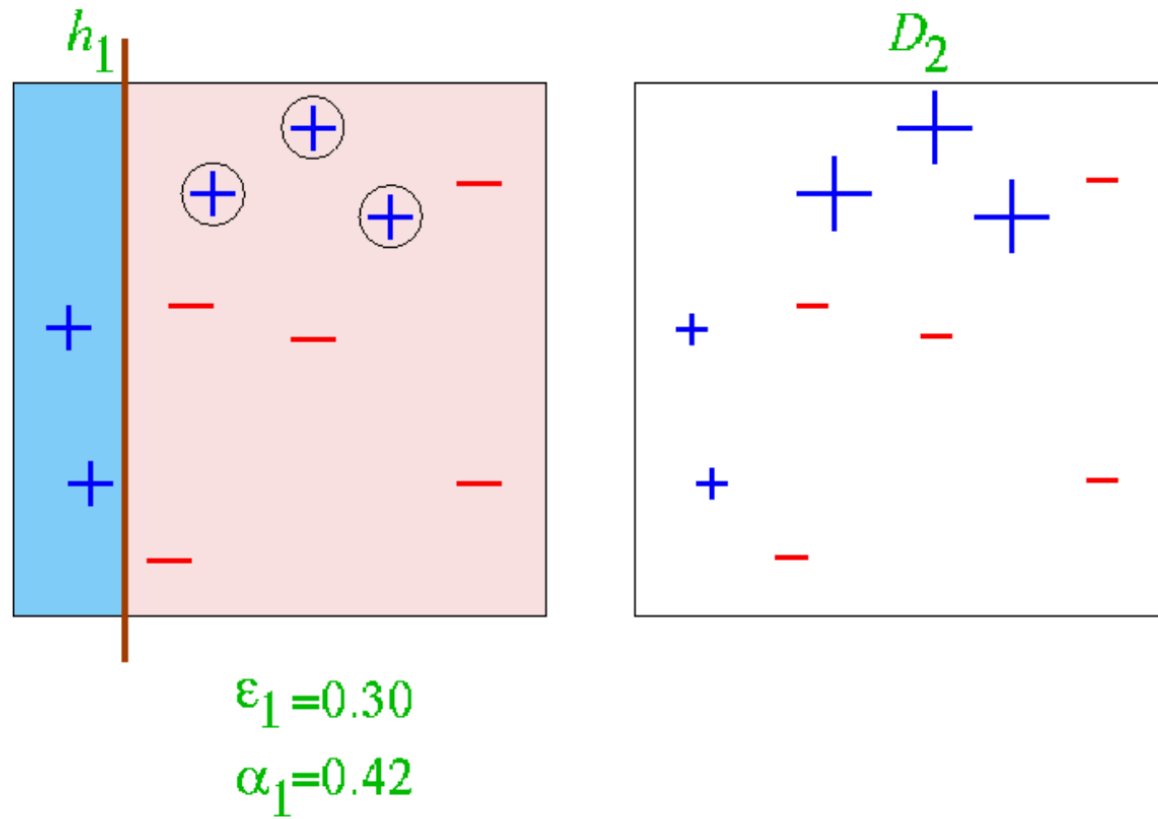
- Ensemble of estimators is a set of regressors/classifiers whose **individual decisions are combined** in some way to classify new examples
- Simplest approach:
  - Generate multiple classifiers
  - Each votes on test instance
  - Take majority as classification
- Classifiers are different due to different sampling of training data, or randomised parameters within the classification algorithm
- → take simple mediocre algorithm and transform it into a super-classifier without requiring any fancy new algorithm



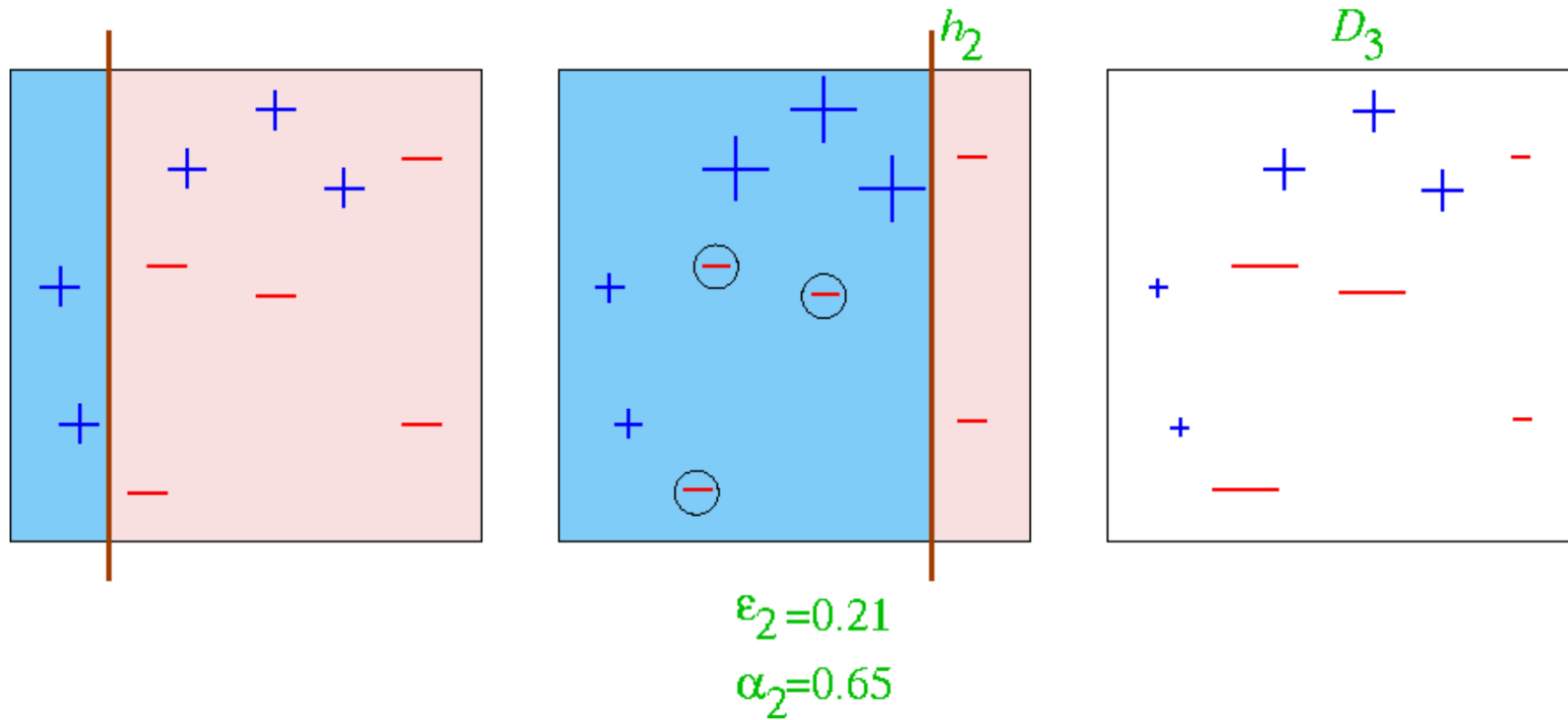
# Ensemble methods

- Differ in training strategy, and combination method
  - **Parallel training** with different training sets
    - **Bagging** (bootstrap aggregation): train separate models on overlapping training sets, average their predictions
    - Parallel training with objective encouraging division of labor: mixture of experts
  - **Sequential training:**
    - **Boosting**: iteratively re-weighting training examples so current estimator focuses on hard examples
  - Typically applied to weak models, such as decision stumps (single-node decision trees), or linear classifiers → **boosted decision trees (BDT)**
- → **minimise two sets of errors:**
  - **Variance**: error from sensitivity to small fluctuations in the training set
  - **Bias**: erroneous assumptions in the model

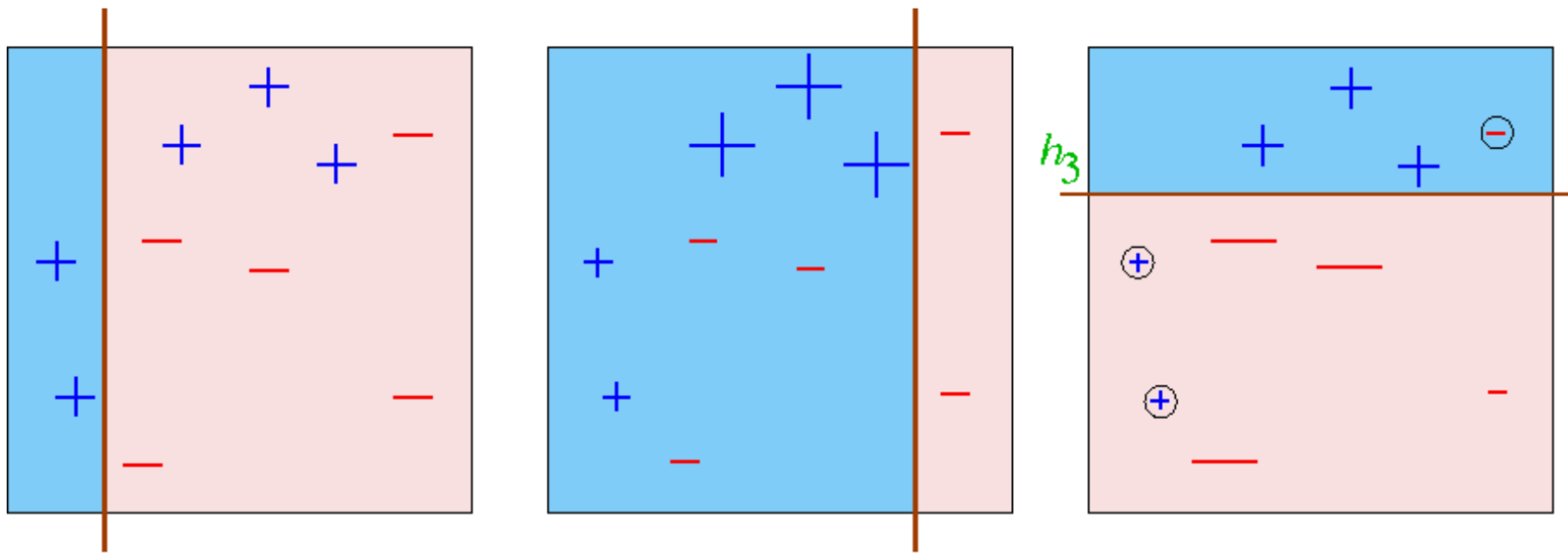
# AdaBoost Example



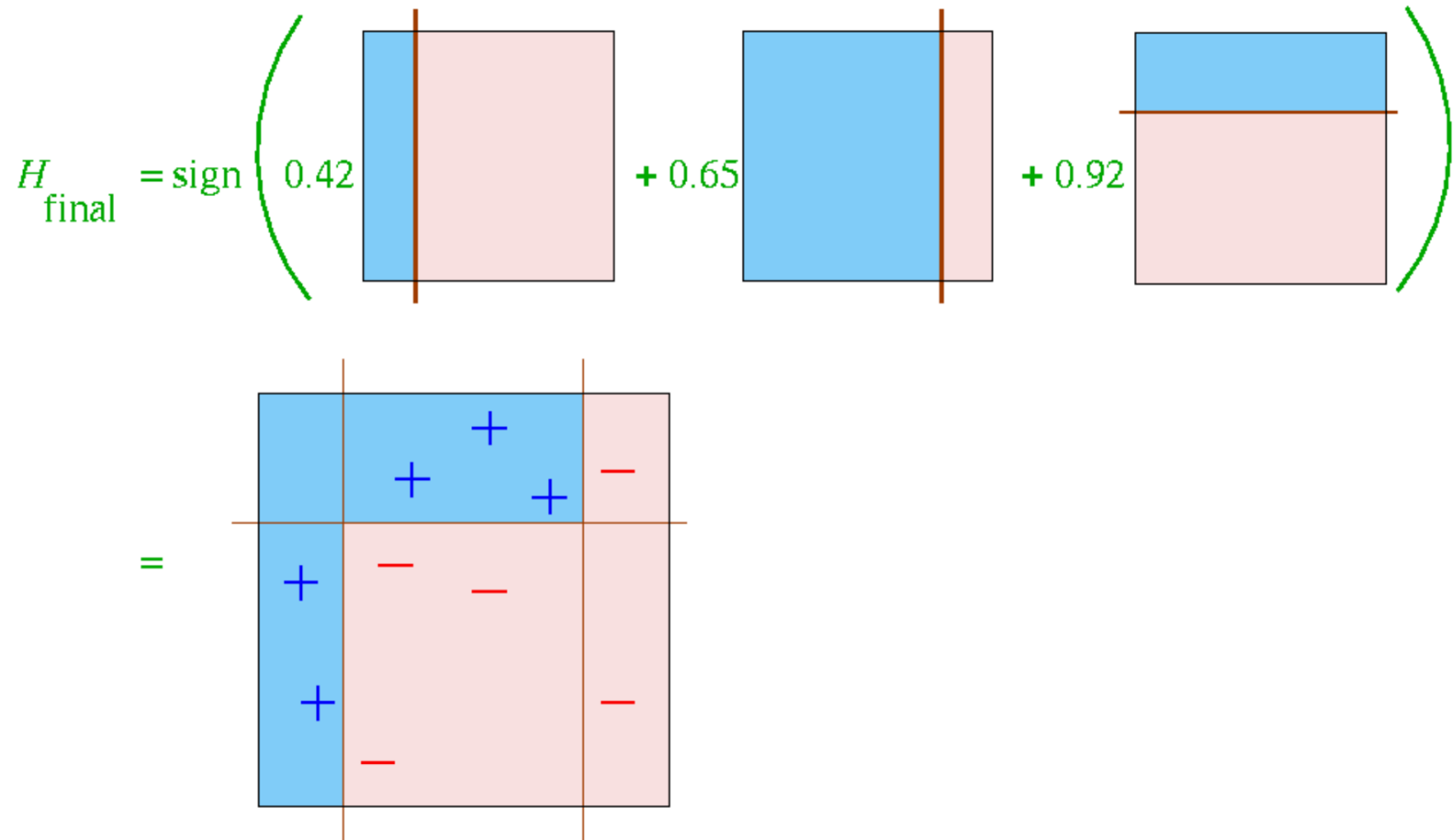
# AdaBoost Example



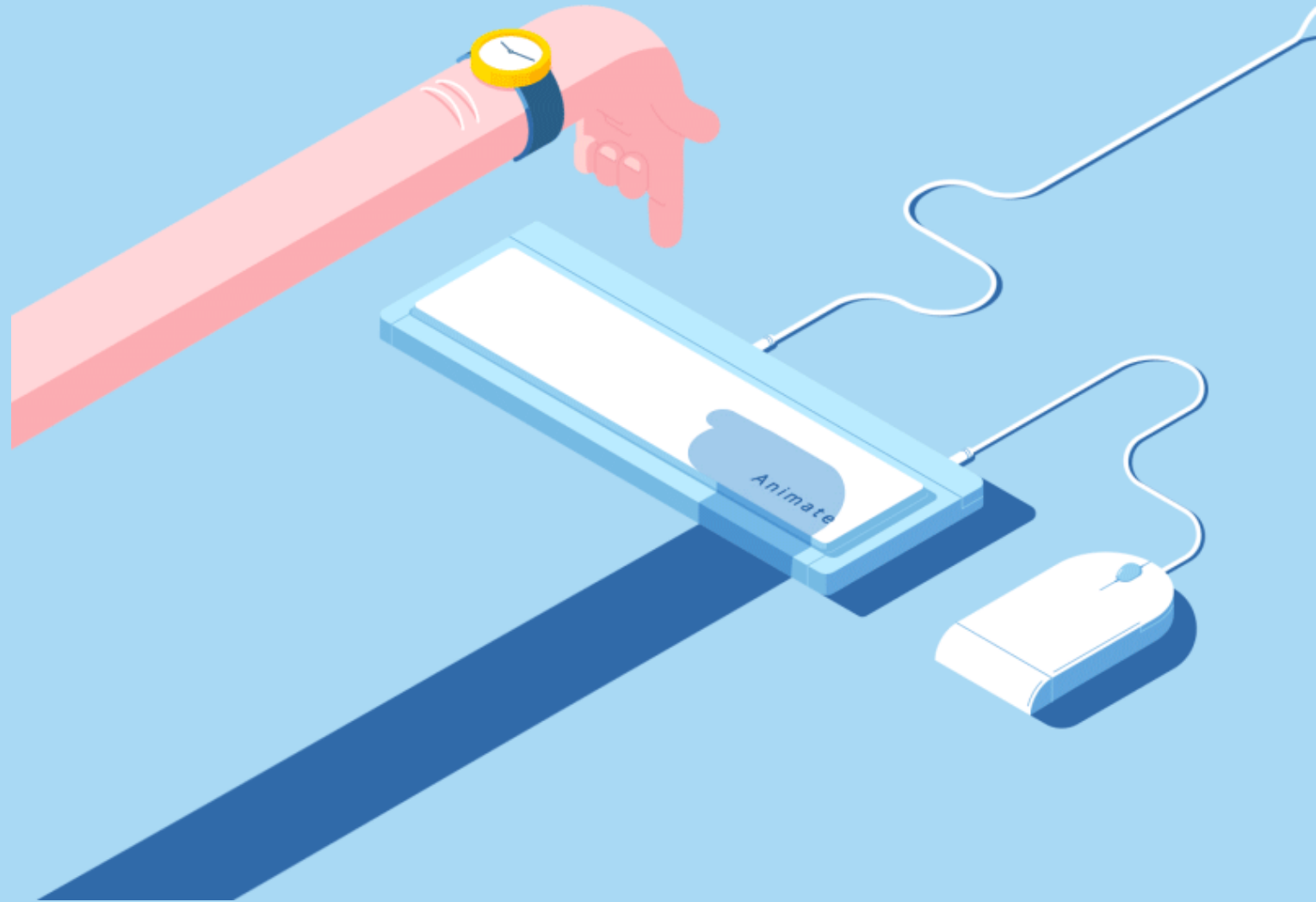
# AdaBoost Example



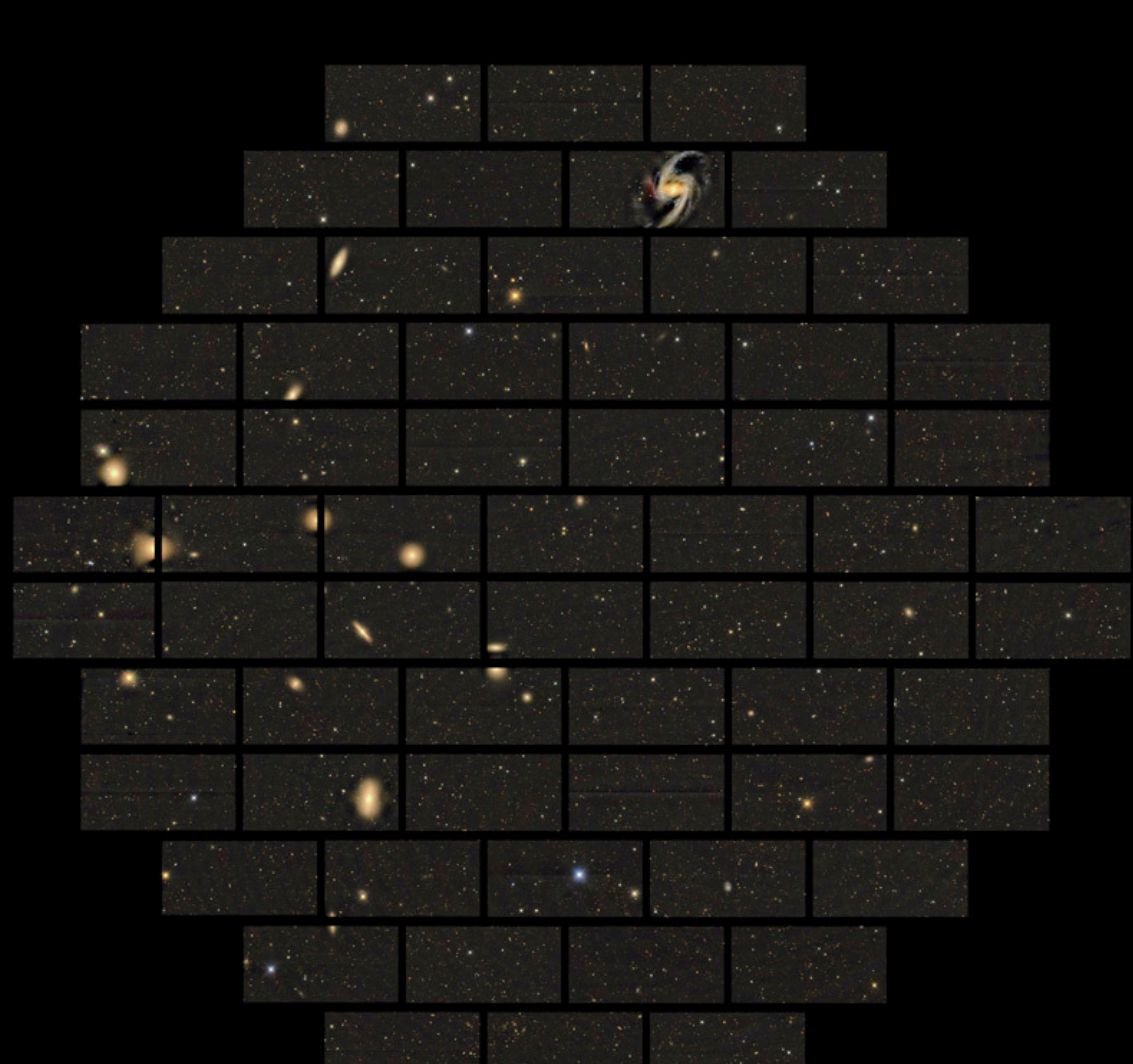
# AdaBoost Example



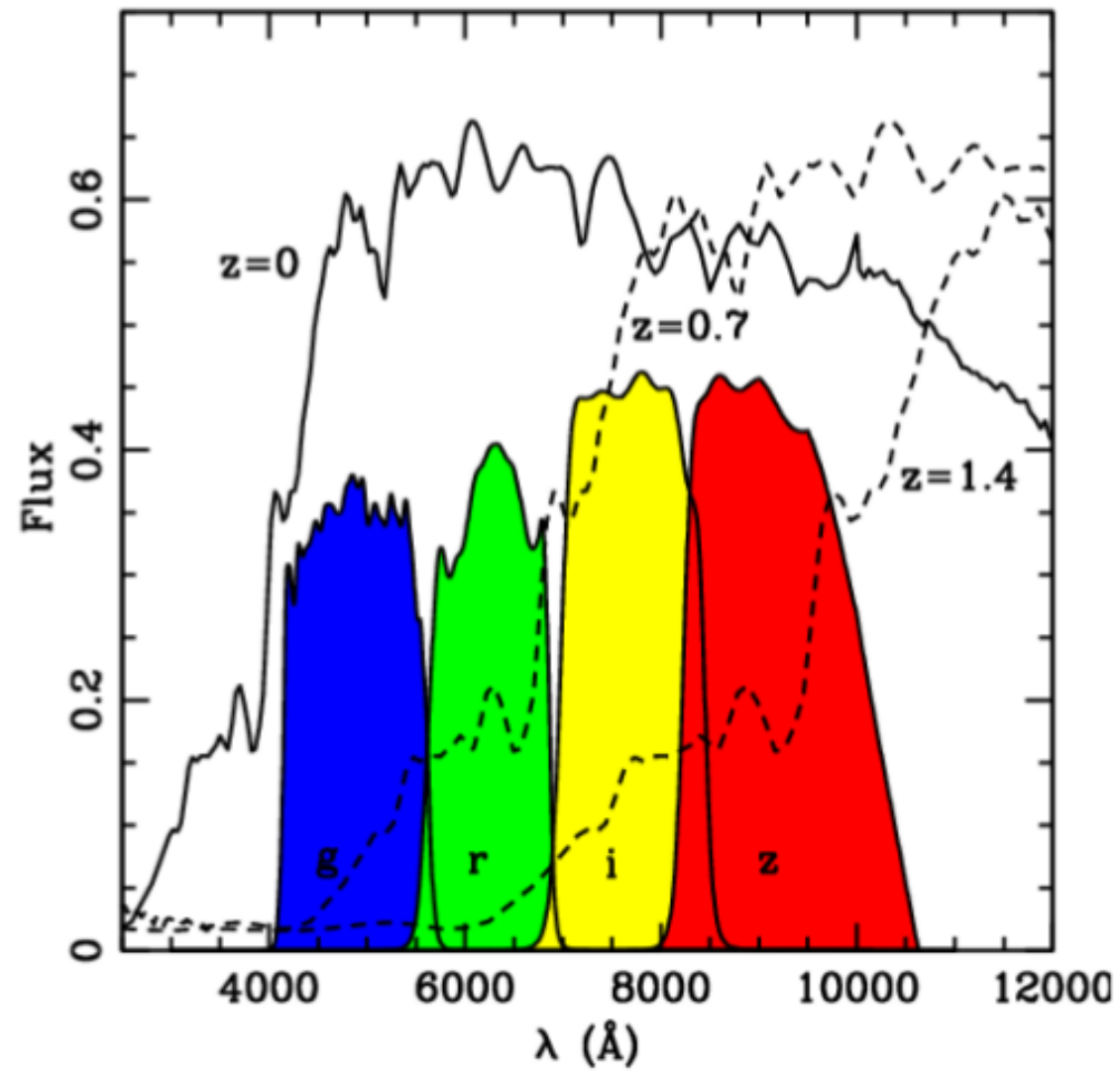
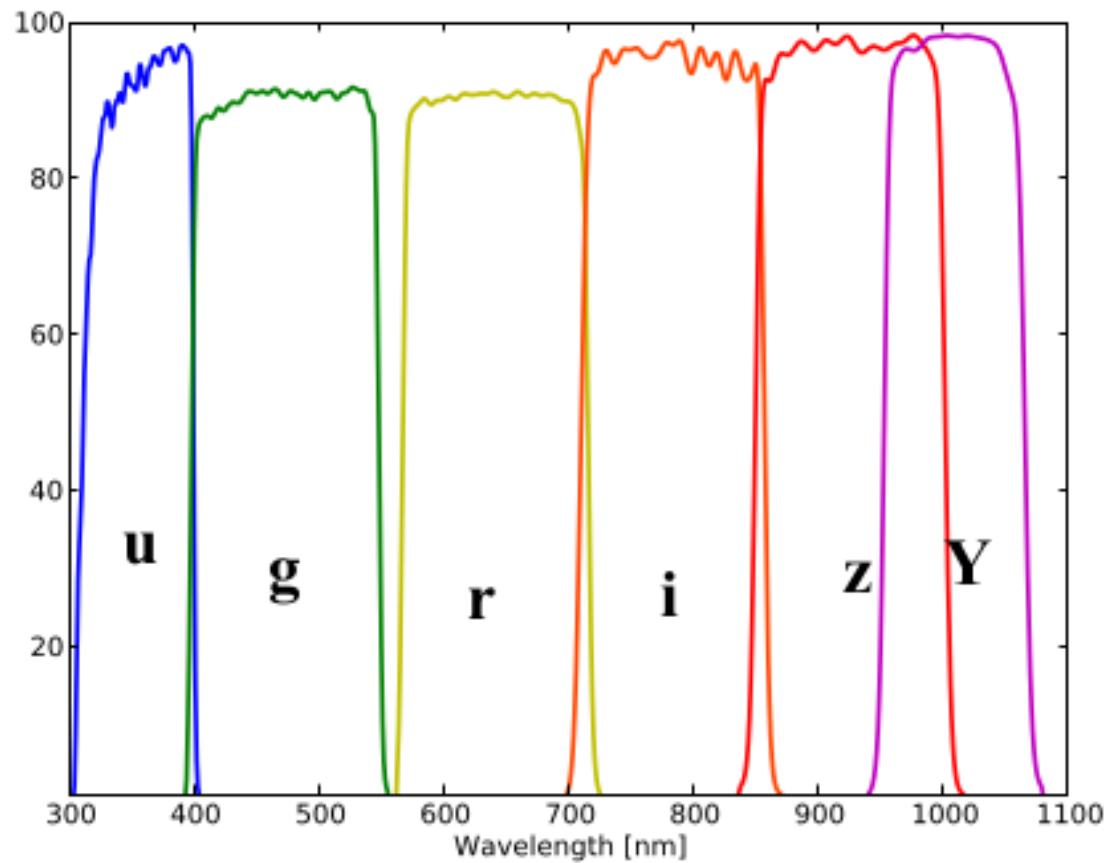
# Real-life application



# The dark energy survey (DES)



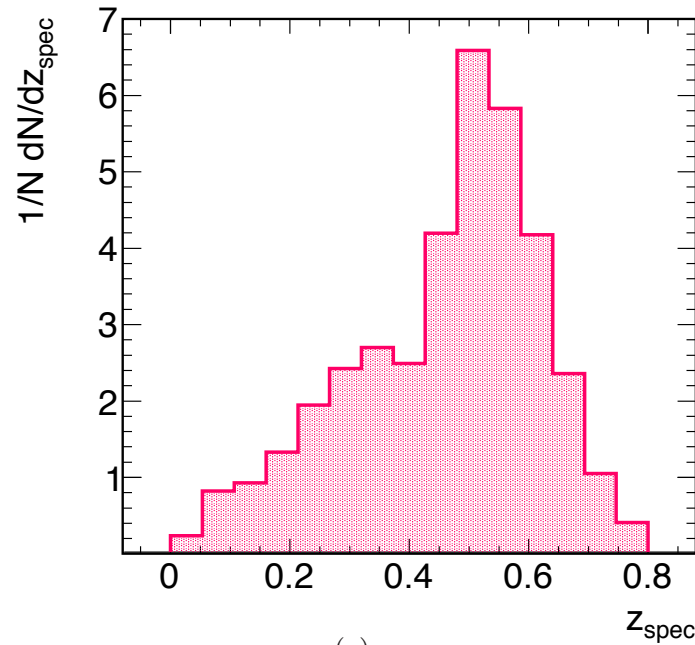
# Optical bands / magnitudes



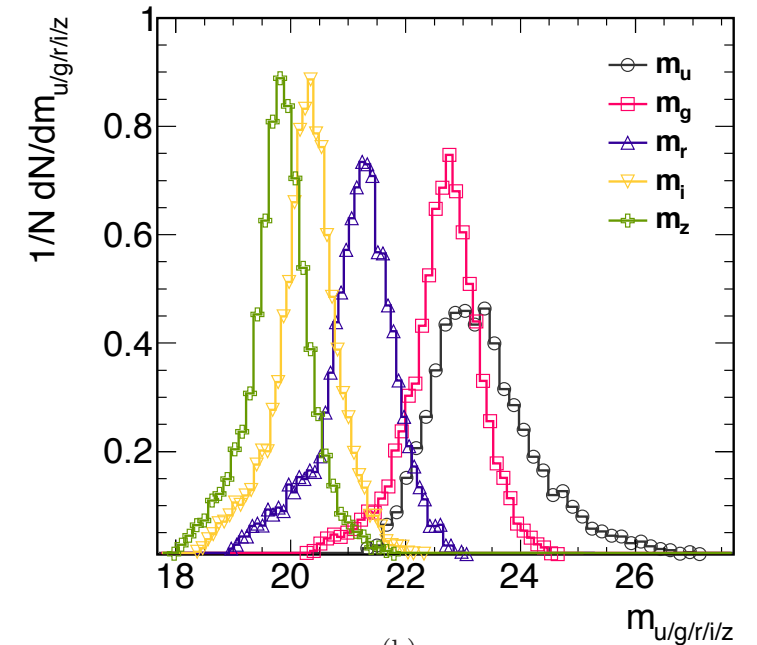


# Galaxy redshift estimation

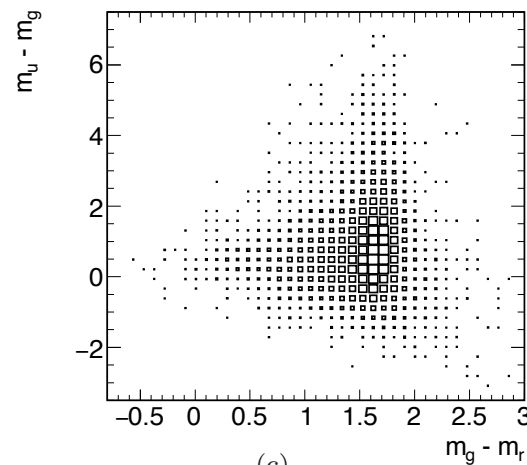
- Machine learning for parameter estimation for an optical survey (DES):
- **Regression target** is  $z$ , the redshift of a galaxy.
- **Features** are optical magnitudes,  $m$ , in 5 bands.



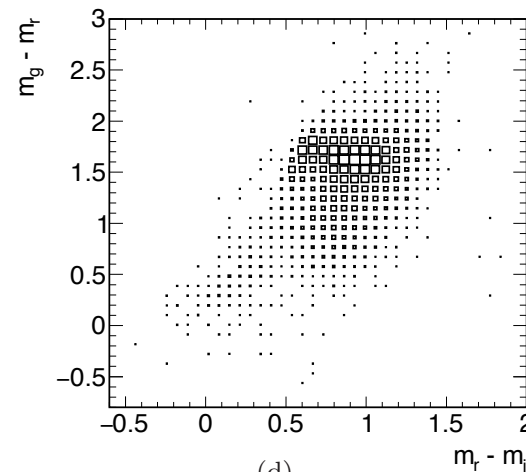
(a)



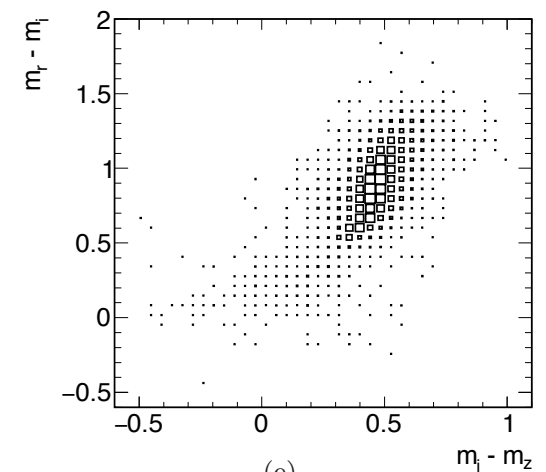
(b)



(c)



(d)

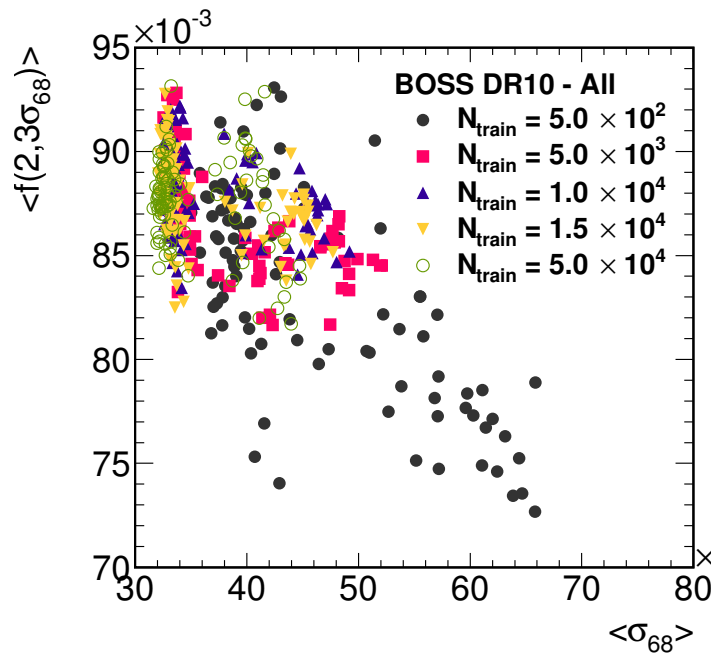


(e)

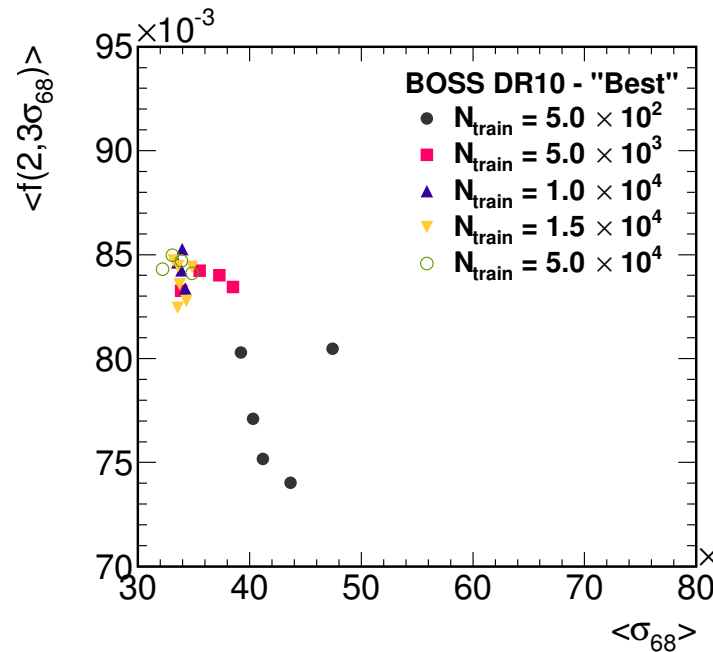
# Galaxy redshift estimation

- Train an ensemble of estimators:
  - Derive **single-best estimator** based on optimisation criteria (on bias and variance).
  - **Combine all estimators** together with uncertainties and derive a probability density function (**PDF**).

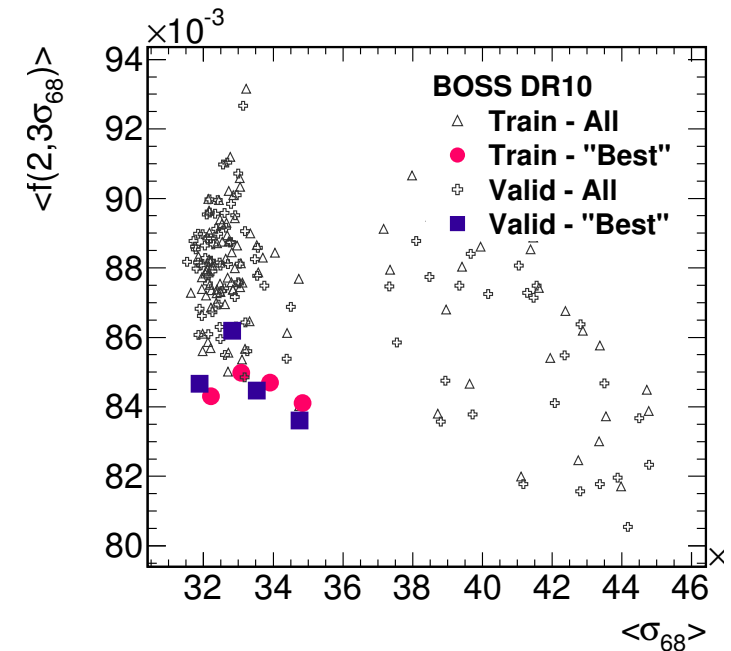
## 100 ANNs



## "Best" 5% of ANNs

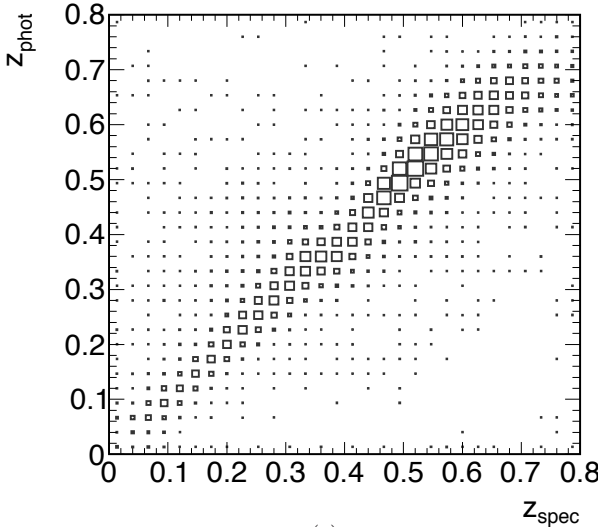


## Training/Validation

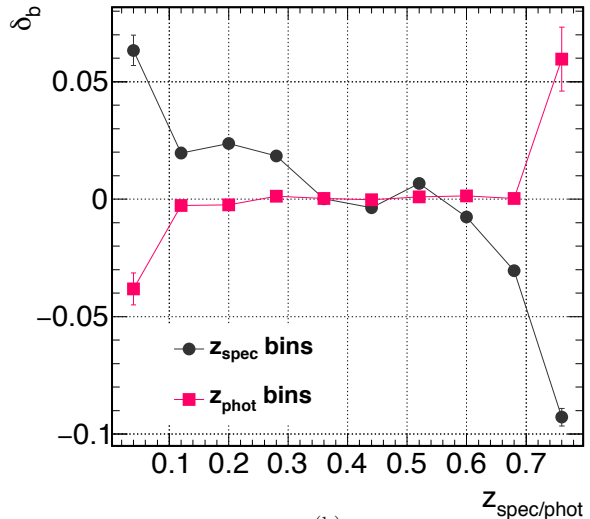


# Galaxy redshift estimation

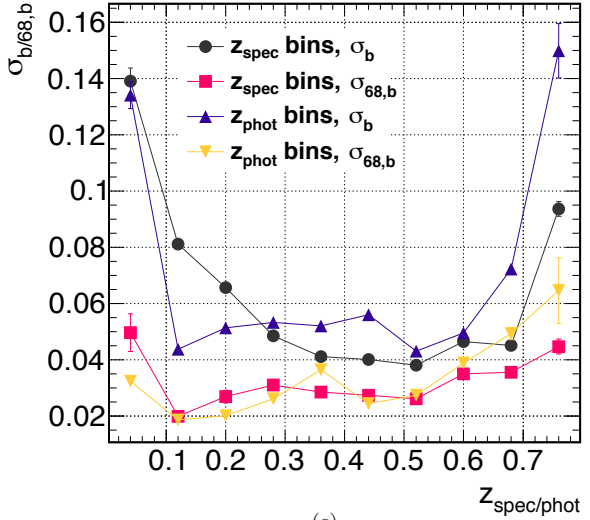
- **Evaluate** performance as a function of true redshift, bias, scatter (variance of bias), and outlier fractions.



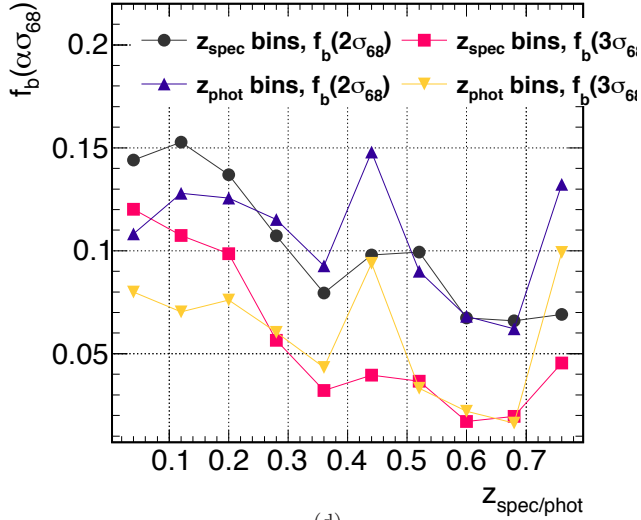
(a)



(b)



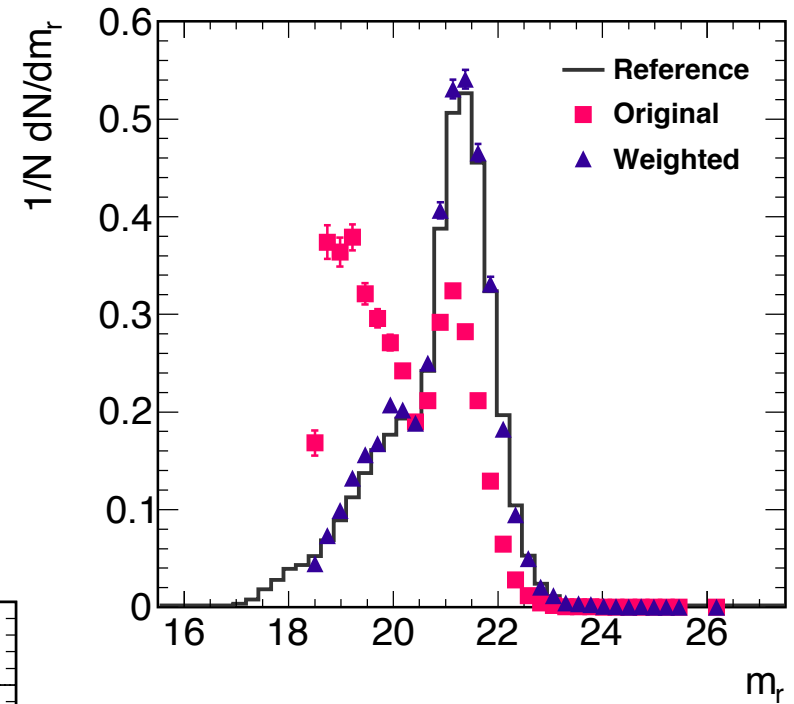
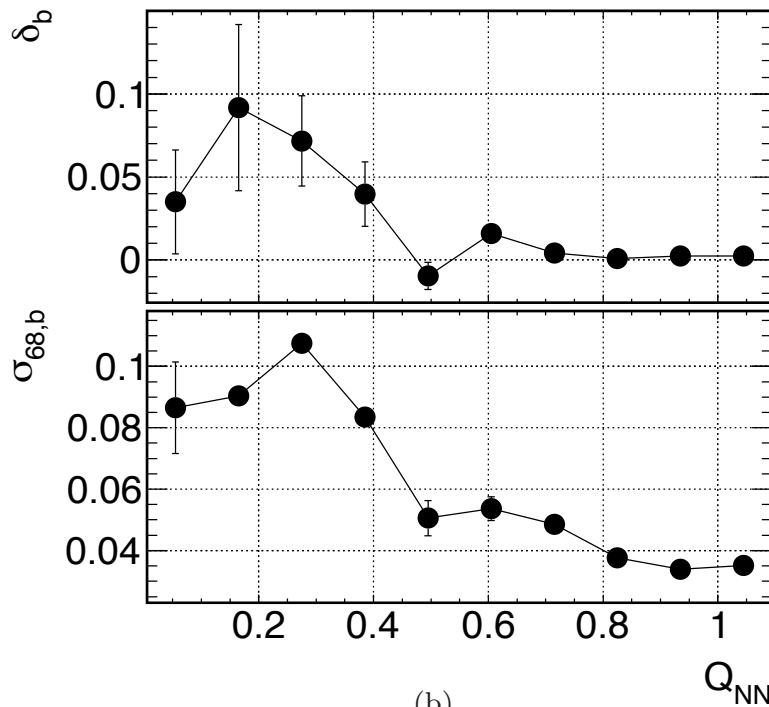
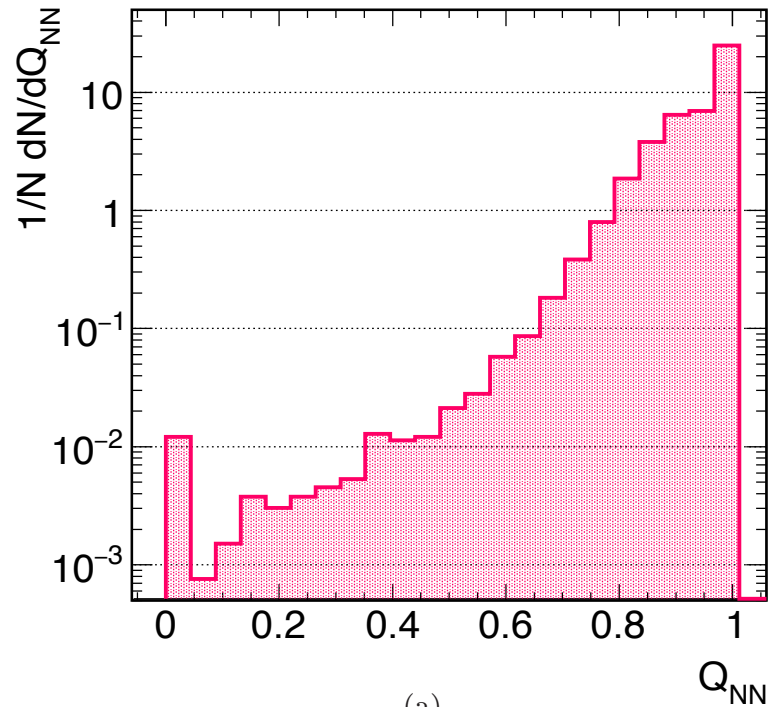
(c)



(d)

# kNN for auxiliary tasks

- **Reweighting**: make sure the training sample is representative of the properties in reality (the evaluated sample)
- **Performance evaluation**:  
 $Q_{NN}$  - the typical density of training objects with similar properties as the evaluated object



# Questions... ?

