

# INTRODUCTION TO MACHINE LEARNING

JESSE KRIJTHE  
CSE2510 - MACHINE LEARNING

**LET'S PLAY**

**Human-generated or Computer-generated?**

*Easier Task?*



# *Discussion*

Suppose we want to build a system to solve this task (distinguishing human-generated from machine-generated paintings, or **distinguishing apples and pears**).

Discuss (in pairs):

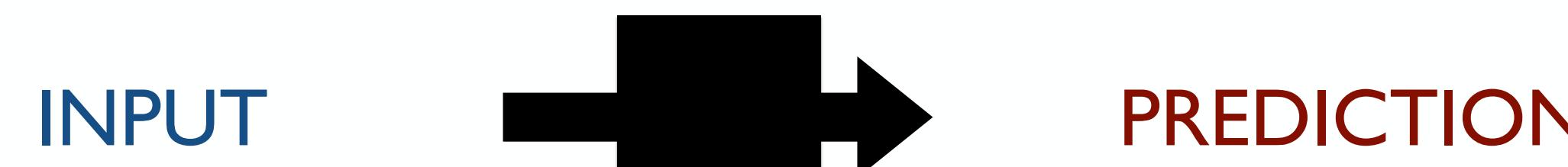
- What would you need to construct such a system?
  - What physical tools would you need?
  - What additional information do you need to gather (talk to a biologist? converse with a farmer? use a search engine?)
  - How do you implement the decision making component of the system?
  - What steps would you need to take to develop such a system?

# *Intuition ML*

- Task can seem very simple, but describing rules that a machine can use is very hard (think of all the exceptions, etc.)
- Idea: create a rule that constructs these rules by looking at many examples.
- “Finding patterns” in the data

# *Machine Learning*

Study of algorithms that use example objects (data),  
to learn mappings from objects to desired outcomes



Painting  
Fruit  
Photo  
Opinion poll  
Patient  
Facebook user  
Google search  
TikTok session  
Traffic scene  
Chess board

“MODEL”

Human vs. Machine Made  
Apple vs. Pear  
Objects in the photo  
Outcome of an election  
Diagnosis / Optimal treatment  
Advertisement  
Relevant website  
Next video  
Brake vs. Don't Brake  
Best Move

# ML is Everywhere?



POLICY TECH YOUTUBE

## YouTube's recommendation system pushed election denialist content to election deniers

Skeptics got more recommendations for fake news and fraud videos, a study found

By Nicole Wetsman | Sep 1, 2022, 11:01am EDT

f t SHARE



Sale: Get WIRED for \$29.99 \$5! [Subscribe now.](#)

= WIRED

WILL KNIGHT BUSINESS AUG 18, 2022

## Sloppy Use of Machine Learning Is Causing 'Reproducibility' Science

AI hype has researchers in medicine to sociology rush techniques that they don't understand—causing a waste of results.

f t e SHARE

In the past few years, art made by programs like Midjourney and OpenAI's DALL-E has gotten surprisingly compelling. These programs can translate a text prompt into literally (and controversially) award-winning art. As the tools get more sophisticated, those prompts have become a craft in their own right.



MIT News  
ON CAMPUS AND AROUND THE WORLD

Menu



SEARCH NEWS

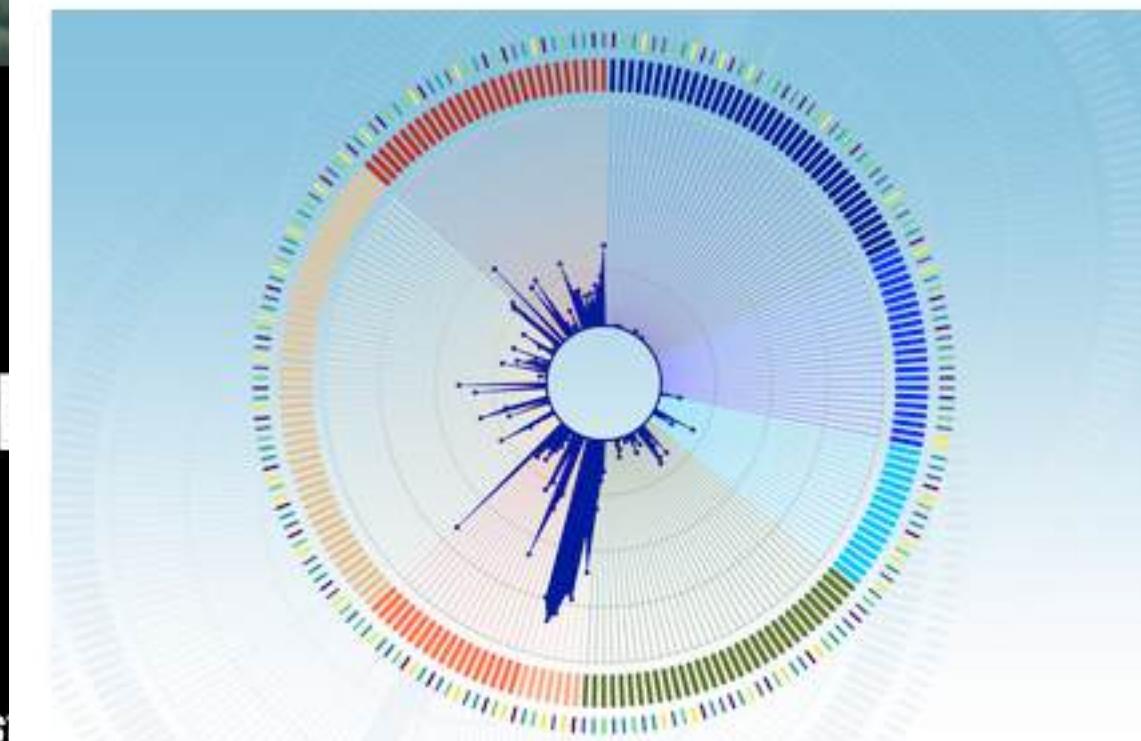


A Verge-generated application of a Midjourney prompt for "futuristic succulents."

## WEB PROFESSIONAL AI WHISPERERS HAVE LAUNCHED A MARKETPLACE FOR DALL-E PROMPTS

*AI art isn't just an experiment — it's a side hustle.*

By Adi Robertson | @thedextrarchy | Sep 2, 2022, 10:30am EDT | 1 comment



"Machine learning tools like this one could empower oncologists to choose more effective treatments and give more guidance to their patients," says Koch Institute clinical investigator Salil Garg.

The first step in choosing the appropriate treatment



DuckDuckGo

machine learning



All Images Videos News Maps Shopping

Netherlands Safe search: off Any time

Machine learning shows links between bacterial population growth and environment

Phys.org | 23 hours ago

How machine learning helps the New York Times power its payroll

VentureBeat | 2 days ago

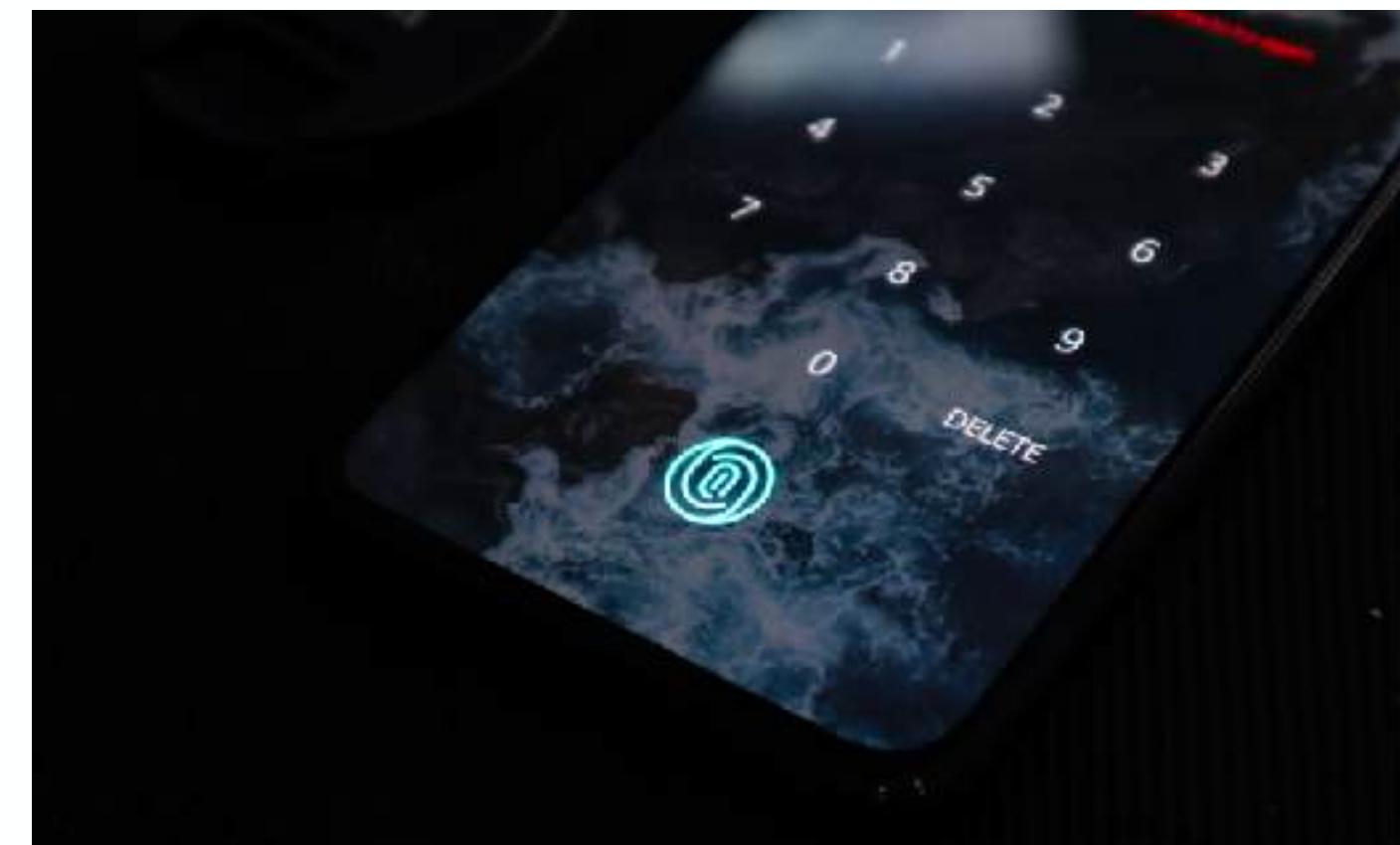
Machine learning using climatic pattern data may help predict harmful algal blooms earlier

Phys.org | 22 hours ago

Machine Learning Breakthroughs Have Sparked the AI Revolution

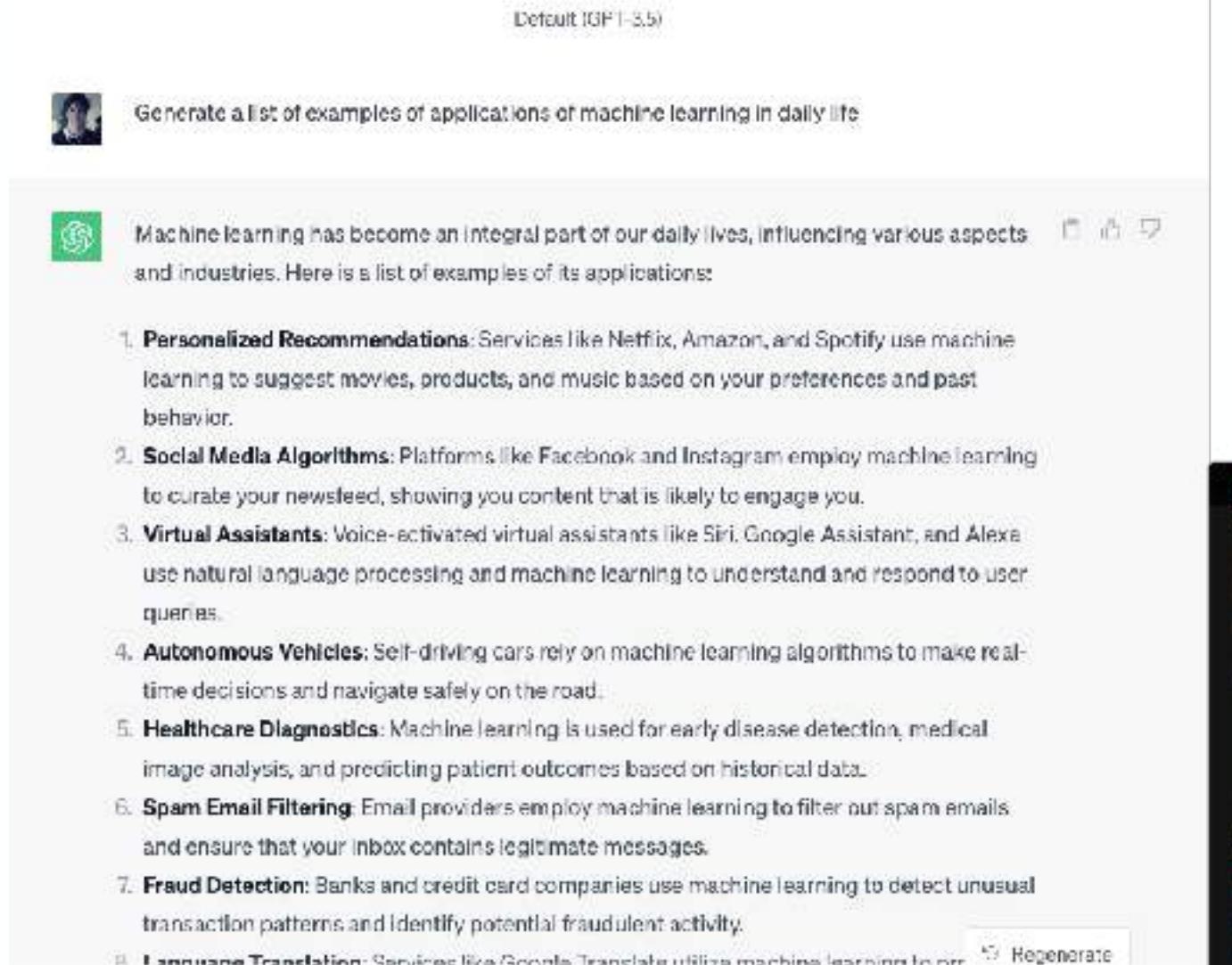
InvestorPlace on MSN.com | 19 hours ago

Machine learning at the edge: The AI chip company challenging Nvidia and Qualcomm



# Some Examples

- Siri / Alexa / Google Assistant
- Recommending movies
- Targeting advertisements
- Game playing
  - Chess, Go
  - Videogames (Atari, DotA, Starcraft, etc.)
- Protein structure prediction
- Image generation
- “Text” generation (Large Language Models)
  - Chatbots
  - Code completion
  - Etc.



# 200 Training Episodes

---

*DeepMind's Deep Q-Network used to play Atari's Breakout game*

# *Application Types*

- Artificial Intelligence / Signal Processing / Multimedia
- Data Analysis
- Scientific ML

# *Learning Goals*

After successfully completing this course, you are able to:

- explain the basic concepts and algorithms of machine learning and their underlying statistical concepts.
- implement, apply and evaluate basic ML algorithms in Python.
- explain the concept of and identify (implicit) bias in data and ML algorithms. (Focus in week 5)

# THE COURSE

# People

## Lecturers



Jesse Krijthe



Gosia Migut



David Tax

## TAs

Aditi Rawat  
Aleksander Buszydlik  
Aleksandra Andrasz  
**Aratrika Das**  
Arnav Chopra  
Bernadett Bakos

Iulia Aldea  
Kirill Zhankov  
Rares Bites  
Sagar Chethan Kumar  
Saunaq Chakrabarty  
Yousef Bakri

# *Note on Schedule*

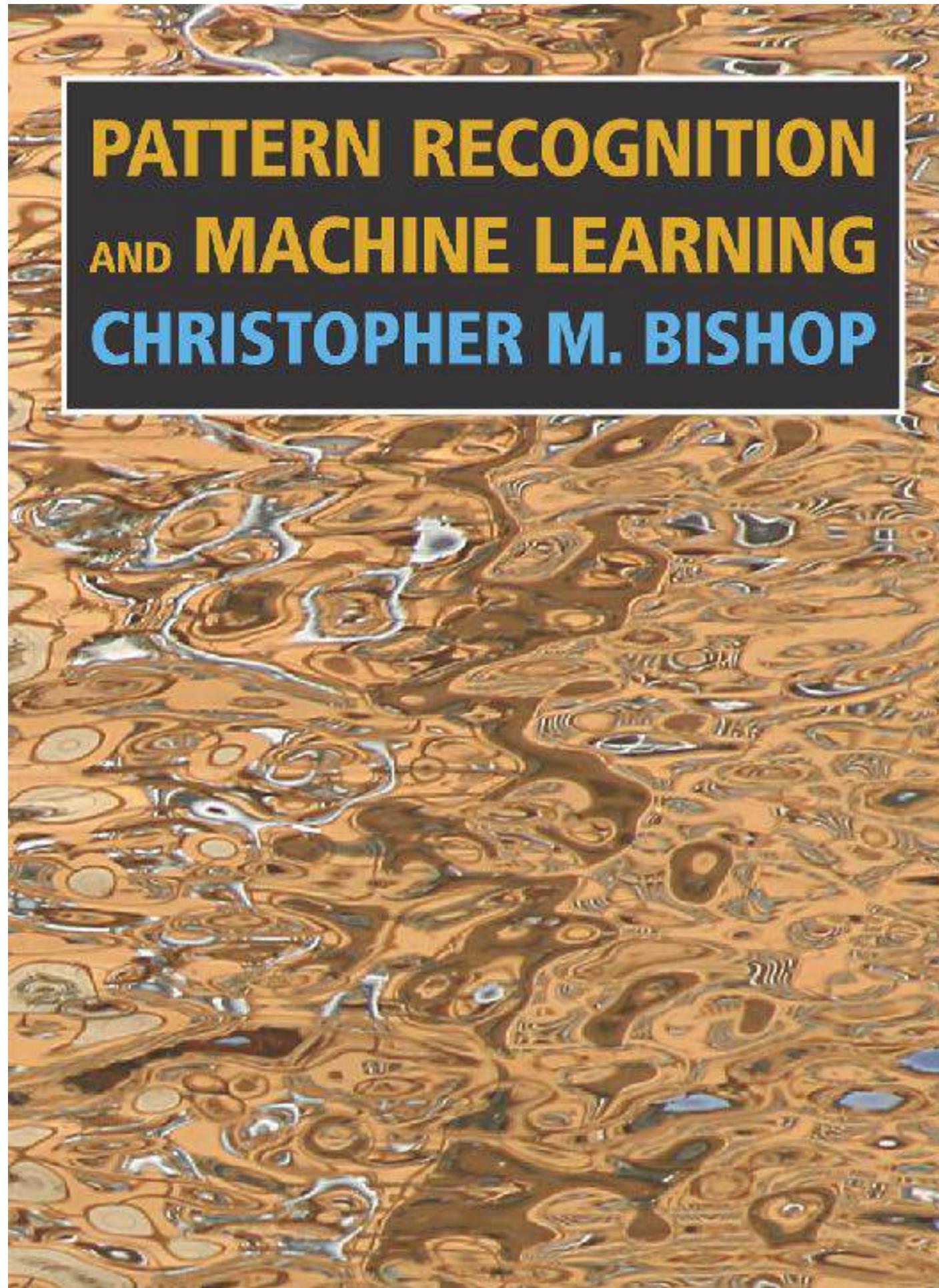
- Course consists of 8 topics / weeks
- Each week corresponds to a topic
  - Lecture 1 on Monday
  - Lecture 2 on Tuesday morning
  - Lab session on Tuesday afternoon

# *Structure*

Carefully read the full instructions on Brightspace

- **Read the reading material** (mostly based on Bishop's Pattern Recognition & Machine Learning)
- Attend lectures and revise lecture notes
- Practice weekly theory questions in Weblab (<https://weblab.tudelft.nl/cse2510/2024-2025>)
- Practice implementation questions in lab notebooks
- Engage with colleagues on <https://answers.ewi.tudelft.nl/>
- *Exam:* mix between theory and implementation questions
- To prepare for exam: Weblab practice exams

# *Book!*



Freely available from Brightspace and the author's website.

Some weeks we supplement the book with other reading materials.

The math can be challenging, but this is part of the learning goals.

**Please use the reading materials!**

# *Weekly Exercises (Weblab)*

- <https://weblab.tudelft.nl/cse2510/2024-2025/>
- Should be available after the lecture
- Highly recommended to make them
- Helps you understand and practice with the material
- Many are similar to the style of the exam questions

# Labs

- Each week, we provide an assignment in the form of a Python notebook that contains insight and implementation questions.
- Suggest you work on it in pairs
- TAs and lecturer are available to help you with questions
- We use Queue for management during the weekly scheduled lab sessions
- Answers will not be provided, so use the TAs to discuss exercises
- **One of the exam questions will be based directly on the lab assignment**
- This week's lab: focussed on familiarising yourself with (matrix computations in) Python

# *Exam*

- 3 hour Weblab exam on campus
- **100% of the final grade**
- Note: make sure you have enrolled in Weblab before the exam (and check it is the correct edition of the course)
- Mix of theory and implementation questions
- Access to Python documentation
- No access to your notes
- See Practice Exams for examples

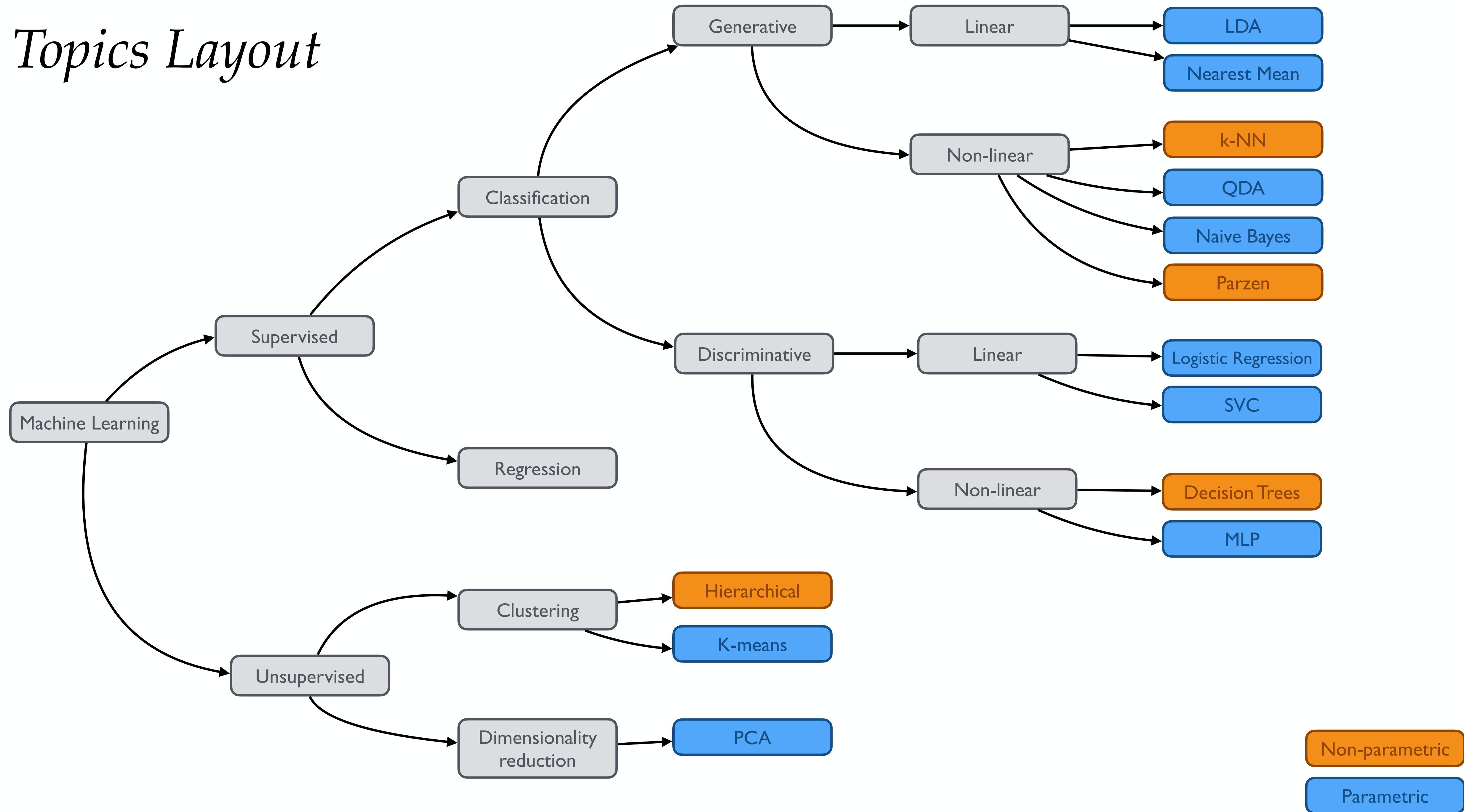
# *Questions/Help*

1. Ask a colleague
2. Ask the TAs
3. Ask on answers platform <https://answers.ewi.tudelft.nl/>
4. If this failed and the question is unique to your situation:  
email us at [ml-cs-ewi@tudelft.nl](mailto:ml-cs-ewi@tudelft.nl)

# Topics

Topic	Description	Lecturer
1	Introduction to ML	Jesse Krijthe & David Tax
2	Generative Parametric Models	David Tax
3	Non-parametric generative models	Gosia Migut & David Tax
4	Discriminative Linear Models	Jesse Krijthe
5	Responsible machine learning	Gosia Migut & Guest
6	Discriminative Non-Linear Models	Jesse Krijthe
7	Unsupervised learning	Gosia Migut
8	Guest Lectures + Q&A	Multiple

# Topics Layout



# *What we improved this year*

- Try to give more concrete examples during lectures
- Revised presentation of difficult notation for backpropagation and PCA.

# *Notes*

- Math notation may seem opaque sometimes: start practicing now when things are still simple, or we'll regret it later.
- Focus of the course is on good understanding of the fundamentals, we build on this in later courses to work on more complex models and interesting applications.
- Let us know when you think we can improve something (preferably during the course, but also after).

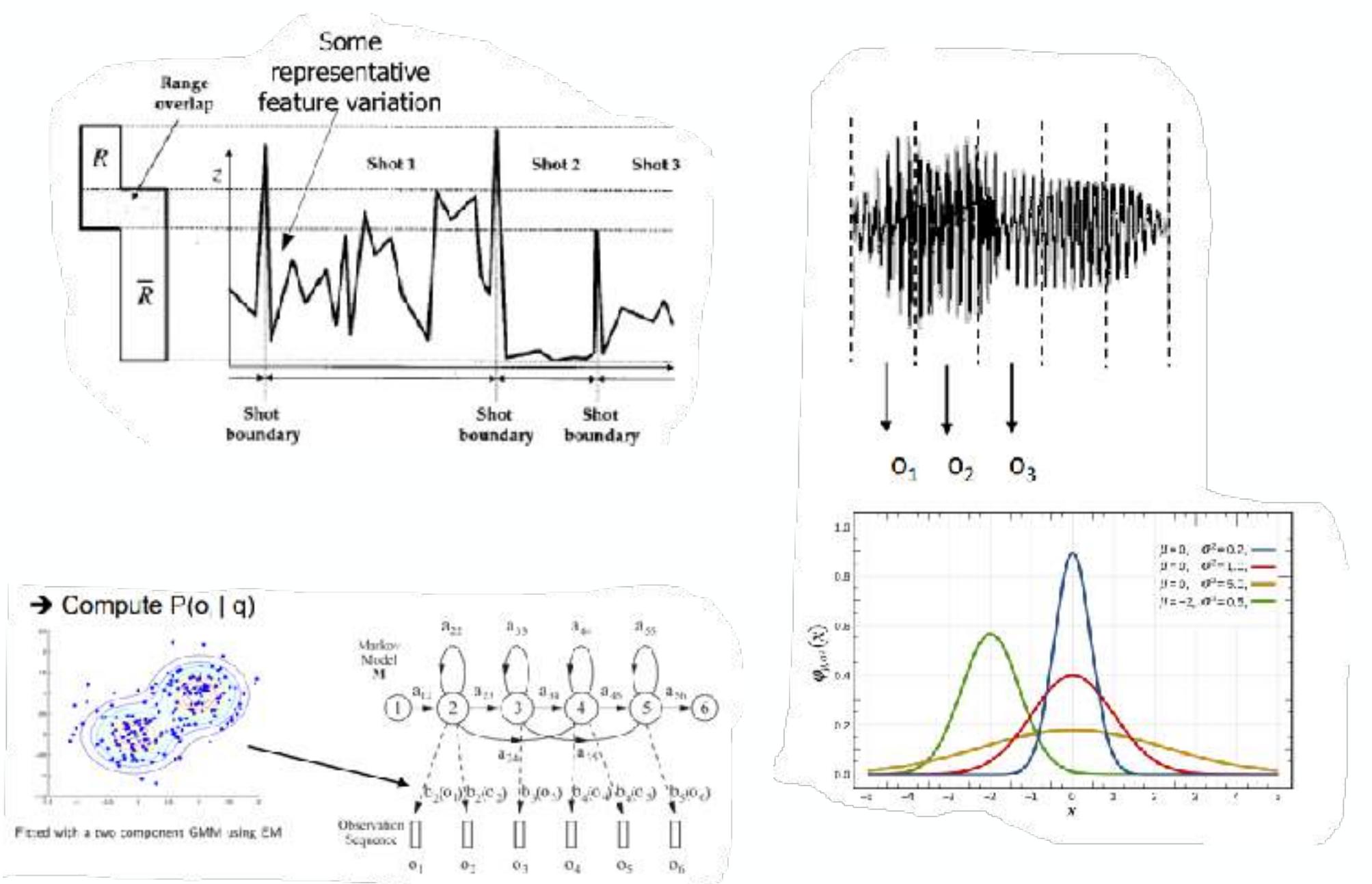
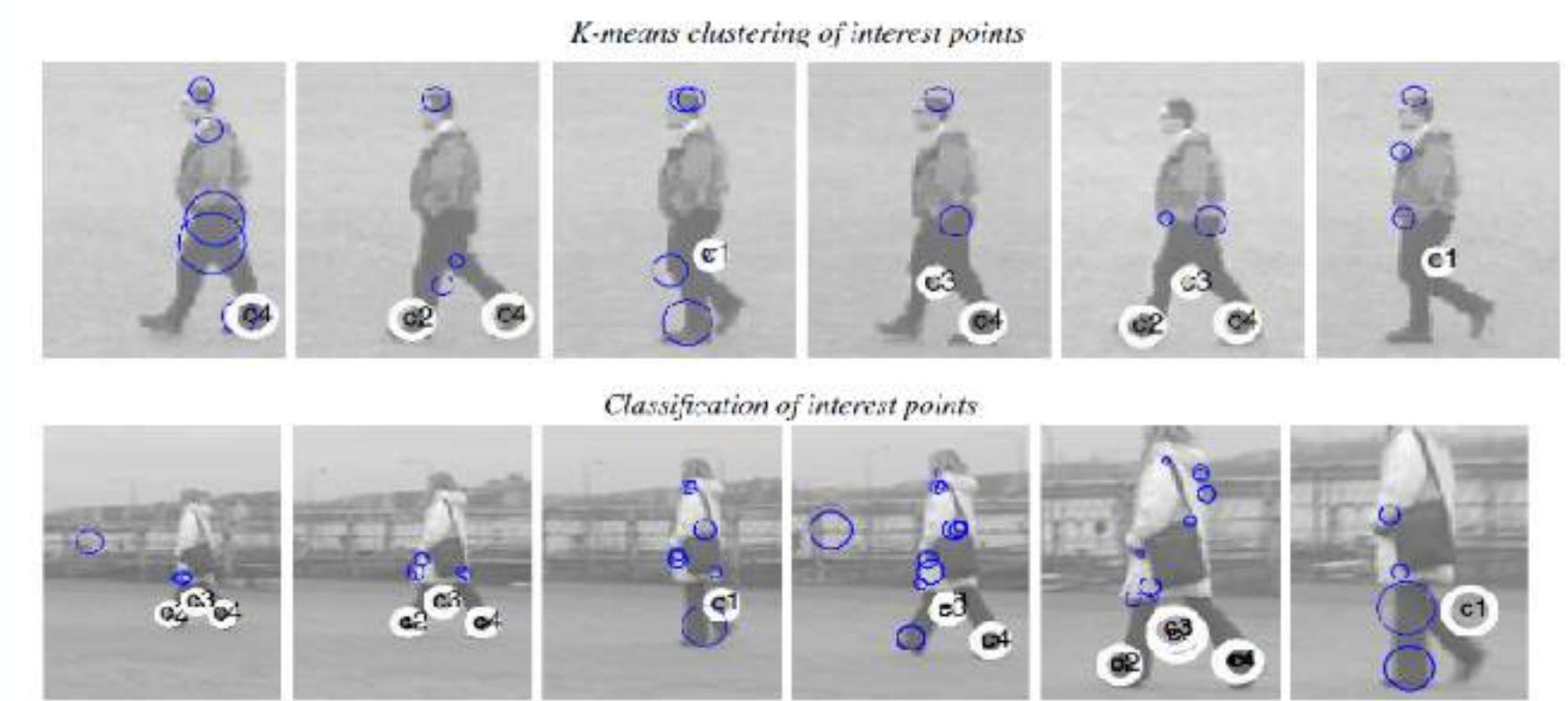
# Use of ML in CSE2230 Multimedia Analysis

Multimedia content (text, images, speech, video) is an important source of knowledge that is useful to people.

Building systems that allow people to make use of that knowledge, and evaluating such systems from both a **technological** and **human** point of view.

Machine Learning Topics covered:

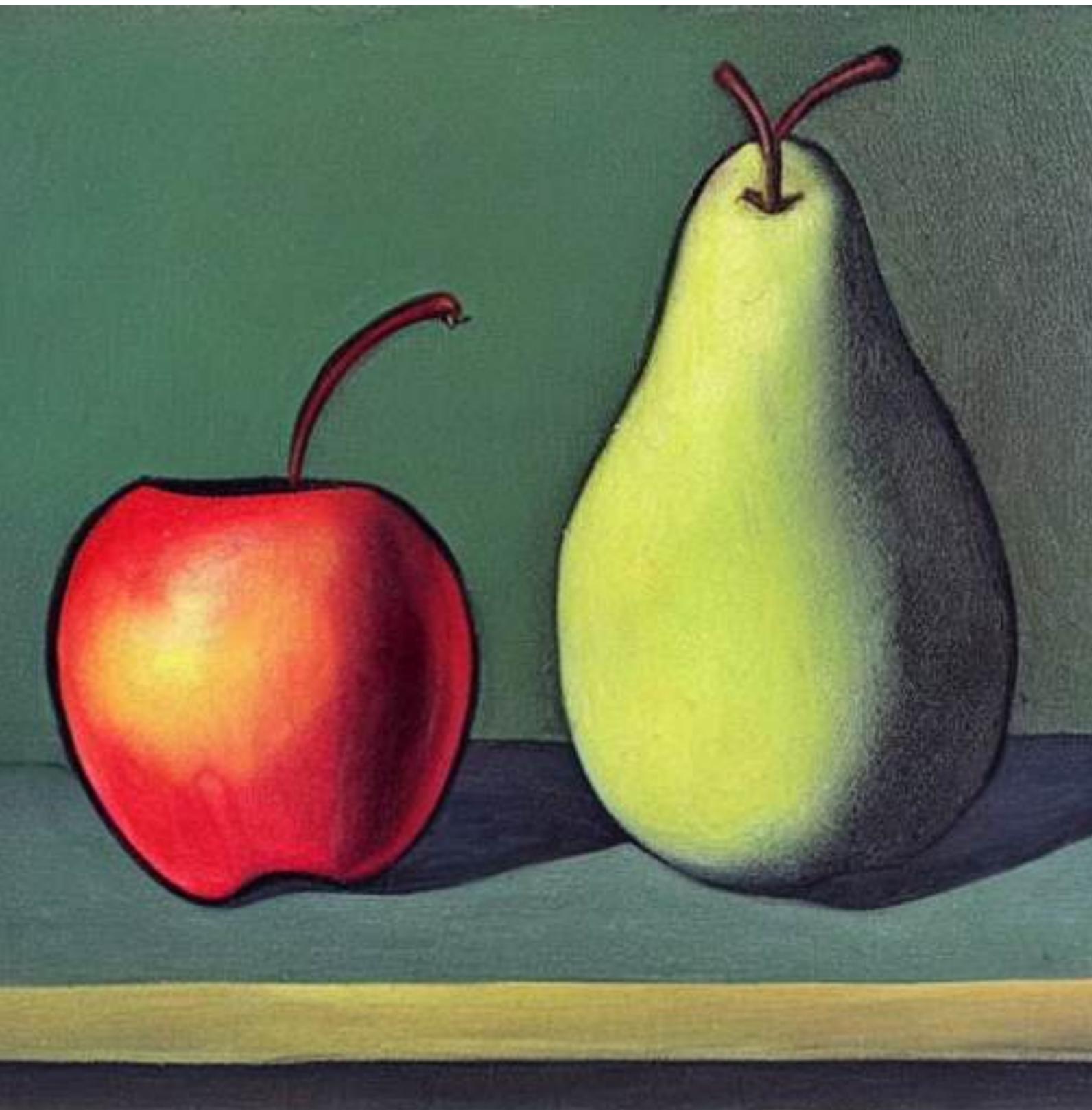
- **K-means clustering** for e.g. image geo-location, visual words representation, audio-visual video segmentation,
- Gaussian Mixture Model for e.g. Acoustic Modelling
- **Naive Bayes Classifier** and Hidden Markov Models (not covered in ML) for e.g. Automatic Speech Recognition
- **Linear classifier** for e.g. video shot change detection,
- **k-nearest neighbour classifier** for e.g. phoneme recognition from Speech
- **Parameter training** for e.g. weight training for late multi-modal data fusion
- System Evaluation e.g. **accuracy, precision, recall, f-measure, ROC, AUC...** but also metrics for information retrieval systems Mean Average Precision (not covered in ML)



QUESTIONS?

Back to Machine Learning...

# *Example*



# *What is Machine Learning?*

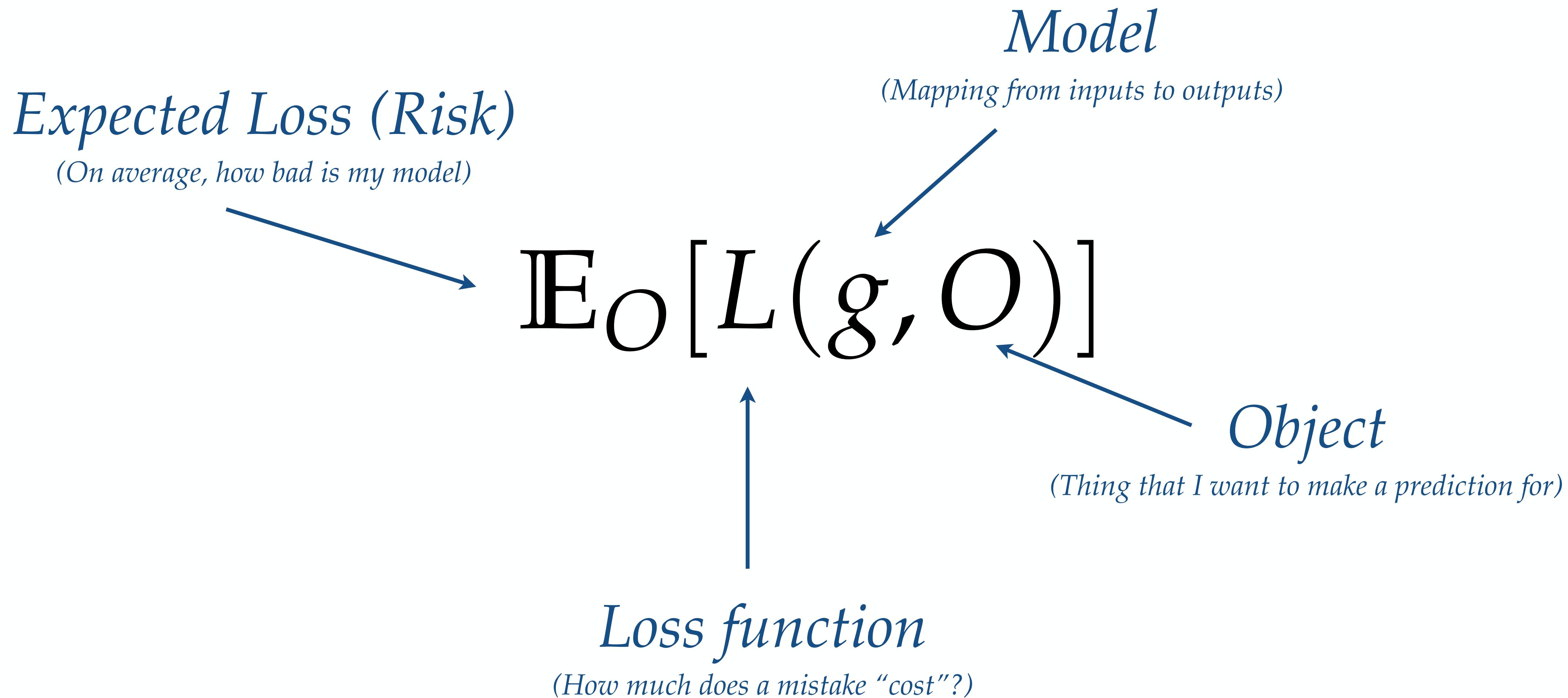
Machine Learning tries to identify regularities in the world, by **learning** from examples (“data”). These regularities should **generalise**: work beyond the specific examples the model has seen before.

# *Why Machine Learning?*

1. Many tasks are too complicated to explicitly encode by hand: learn a mapping from input to output using examples.
2. Learning about the world using data, to use objective evidence to base our decisions on.

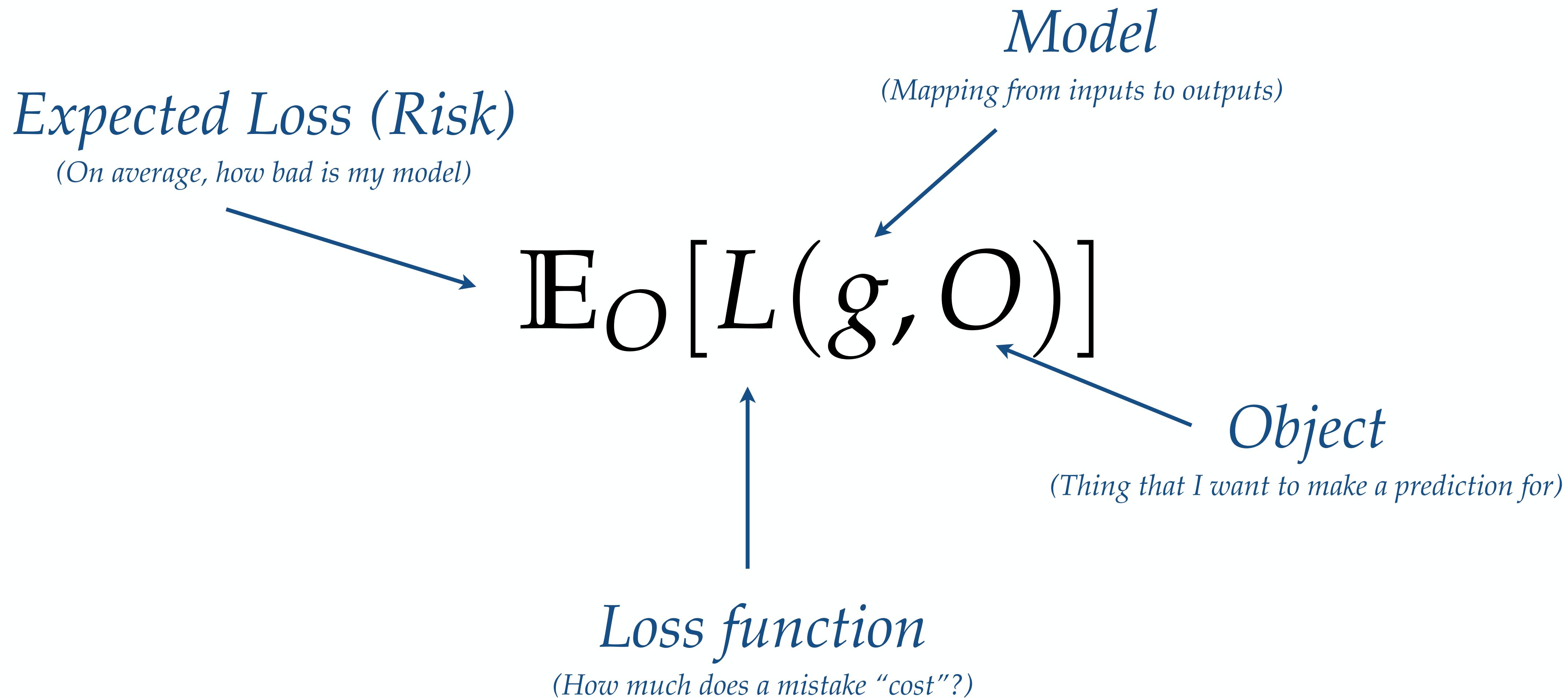
Connections to statistics, signal processing,  
optimisation

# *Goal of Machine Learning*

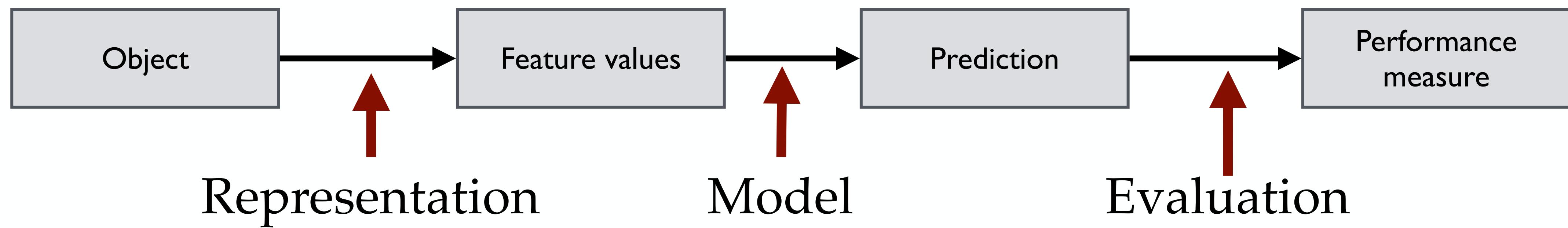




# *Goal of Machine Learning*



# *Steps involved in a Machine Learning model*

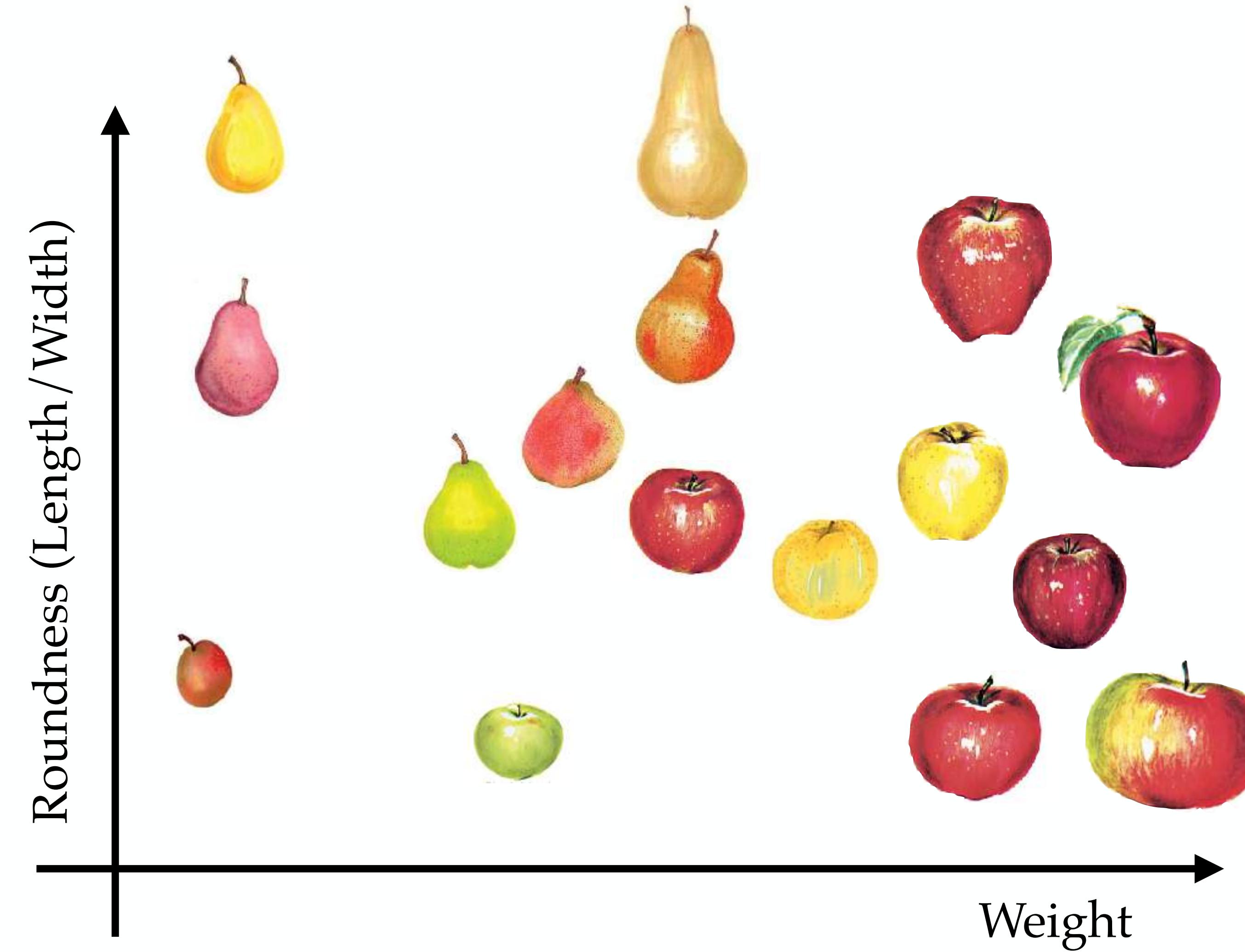


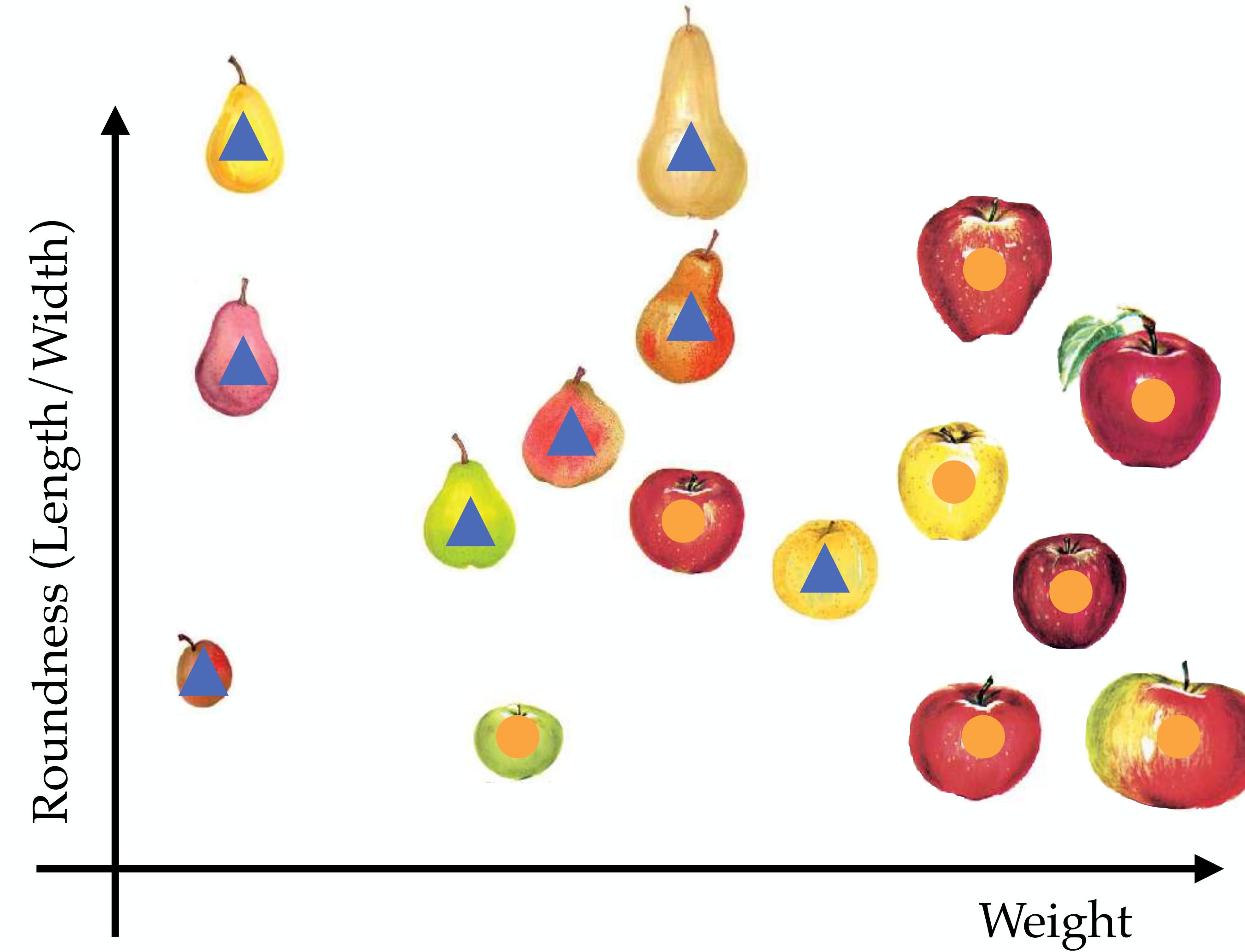
# *Representation -> Features*

- How do I represent my object to use it in a mathematical function? What measurements do I do to represent the object?
- We call these measurements the features (variables, covariates), and depict them using a vector. We call this vector space the **feature space**. For example:

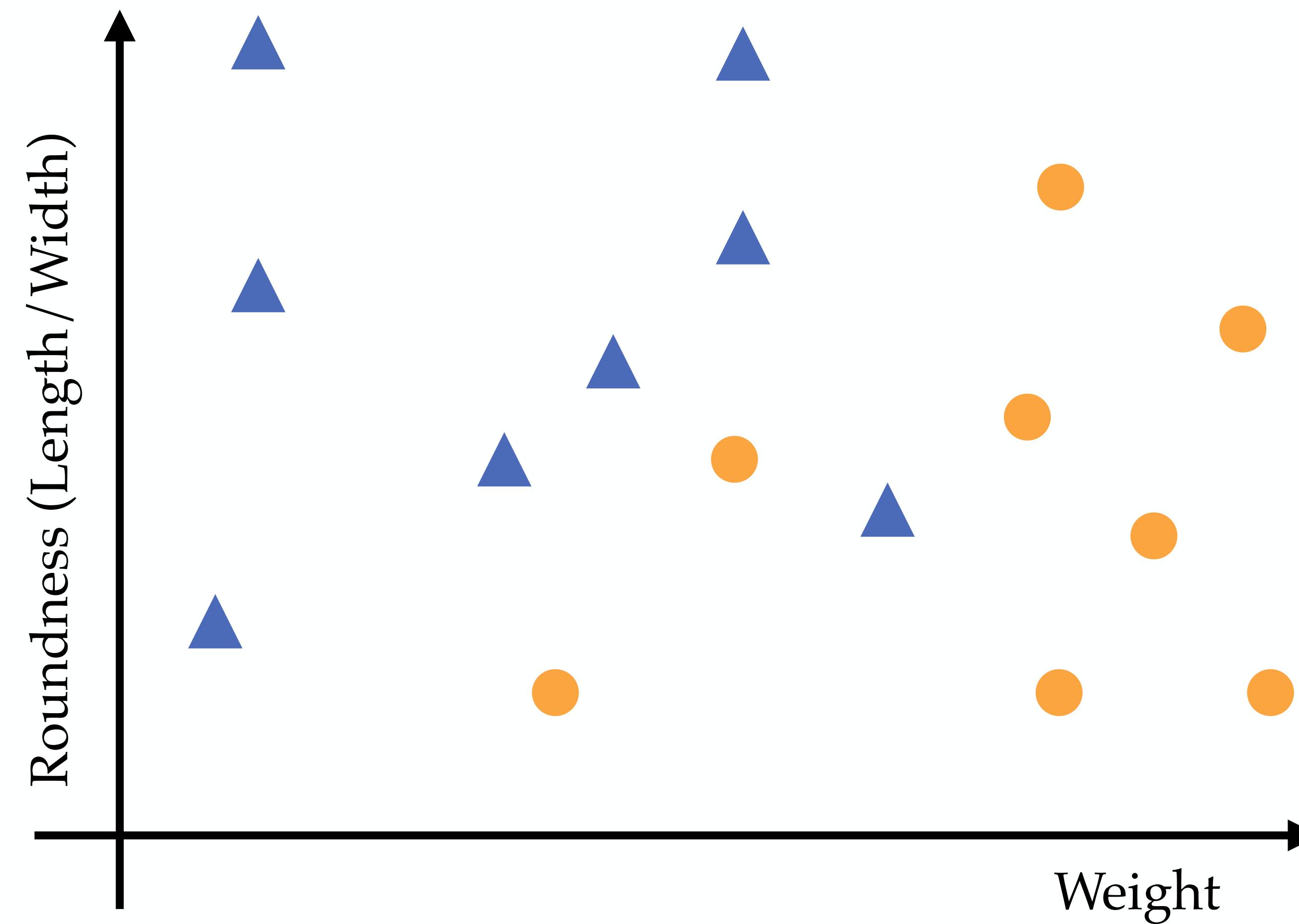
$$\mathbf{x} = \begin{bmatrix} x^1 \\ x^2 \\ \dots \\ x^M \end{bmatrix} = \begin{bmatrix} 0.1 \\ -100 \\ 25 \\ 0 \end{bmatrix}$$

What measurements can I do that will help me  
distinguish the Apple from the Pear?

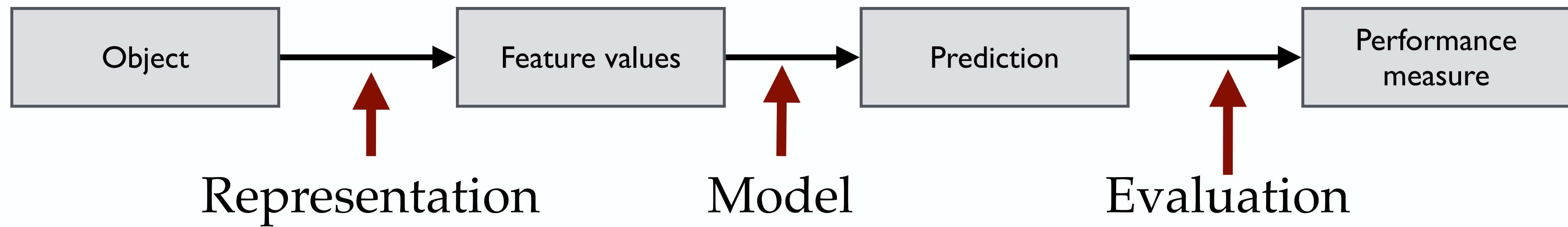




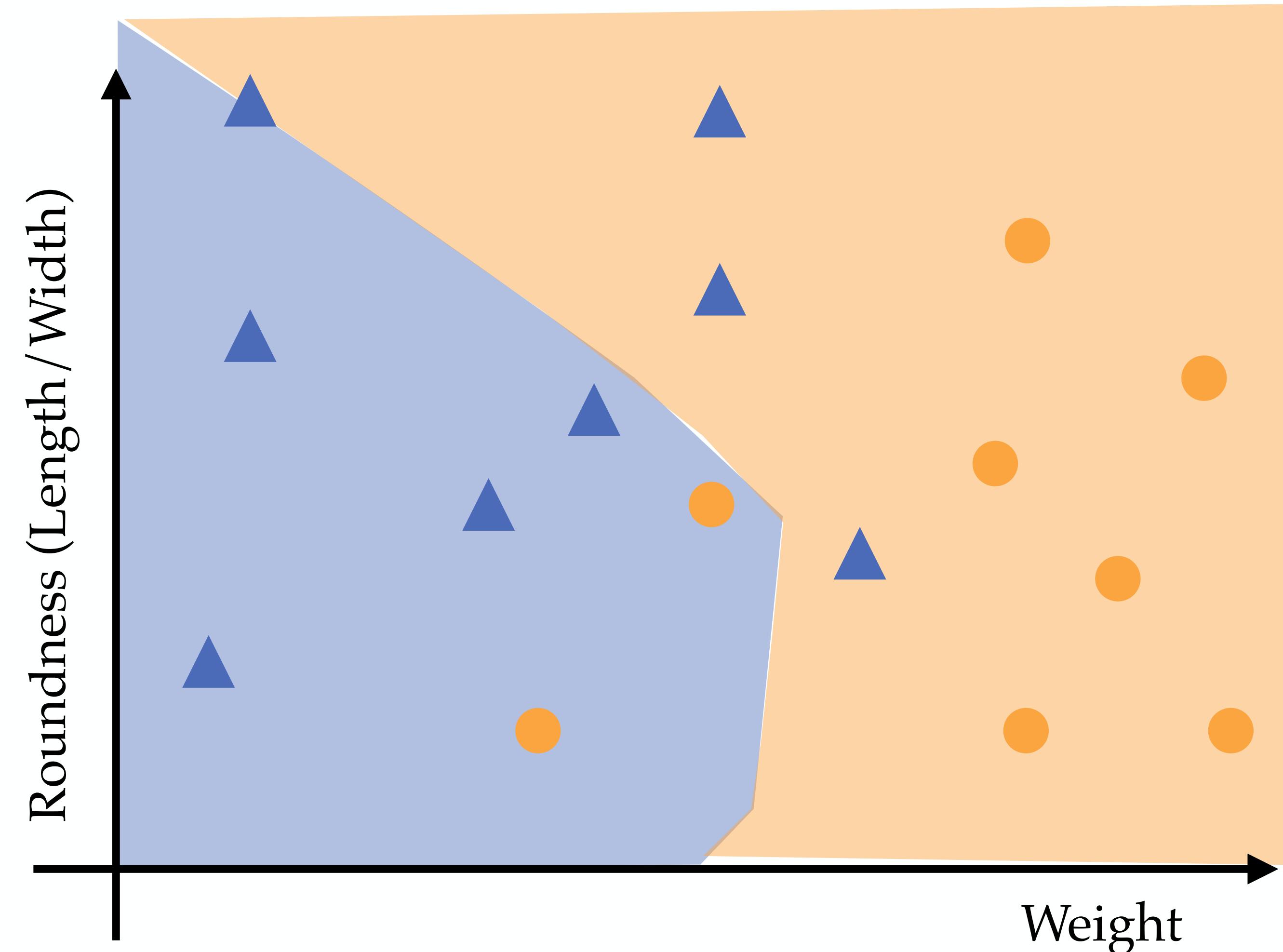
# *Classification Problem*

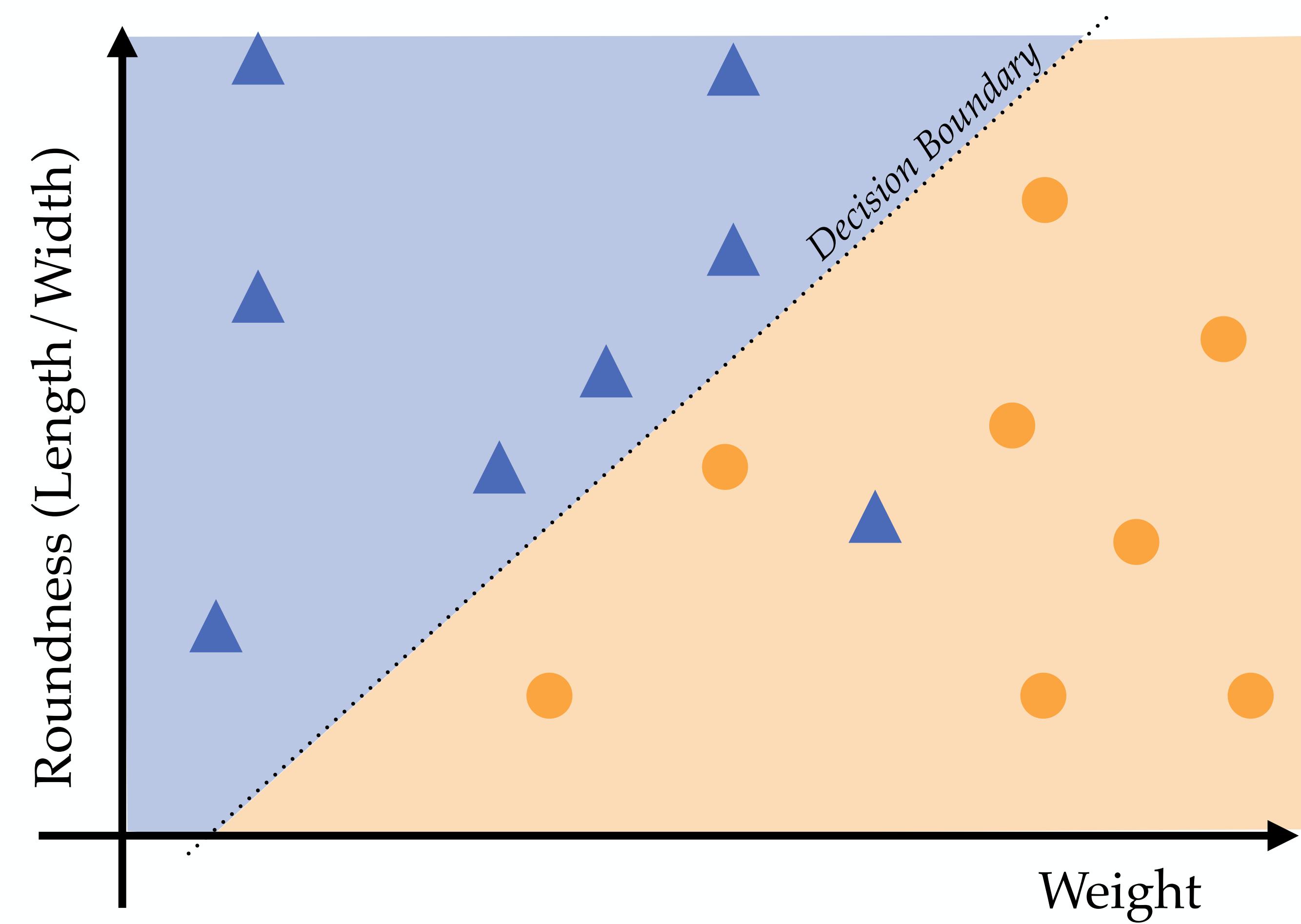


# *Steps involved in a Machine Learning model*



# *What does a mapping/classifier/model look like*

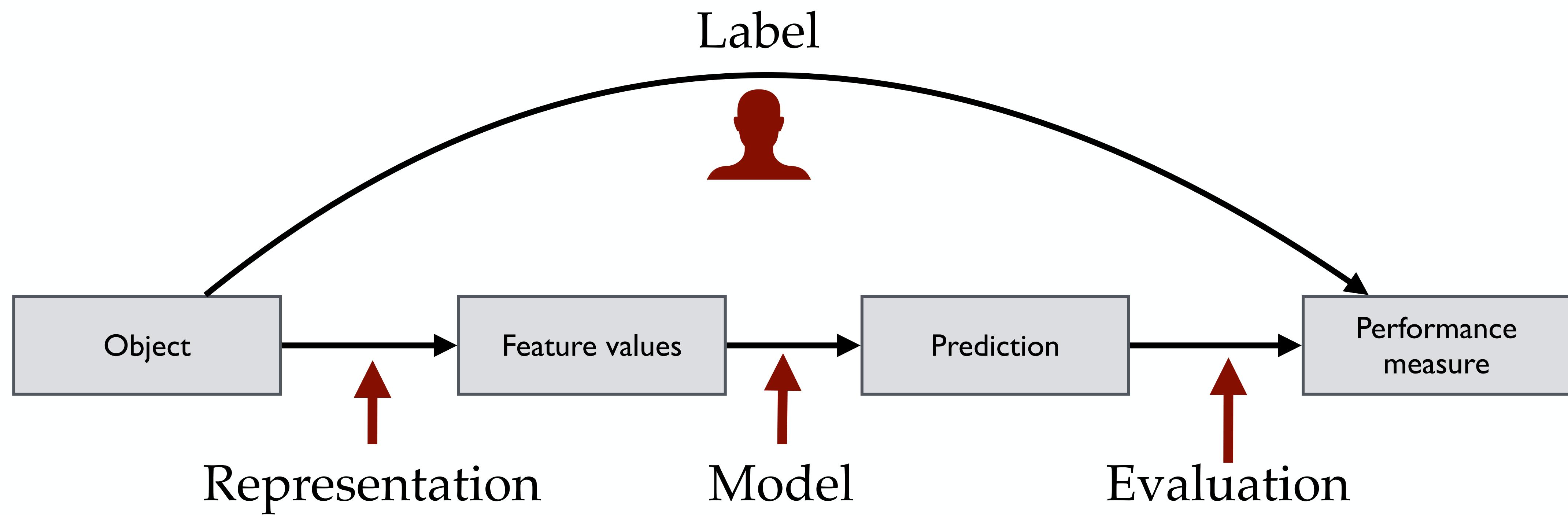




## *More on Features*

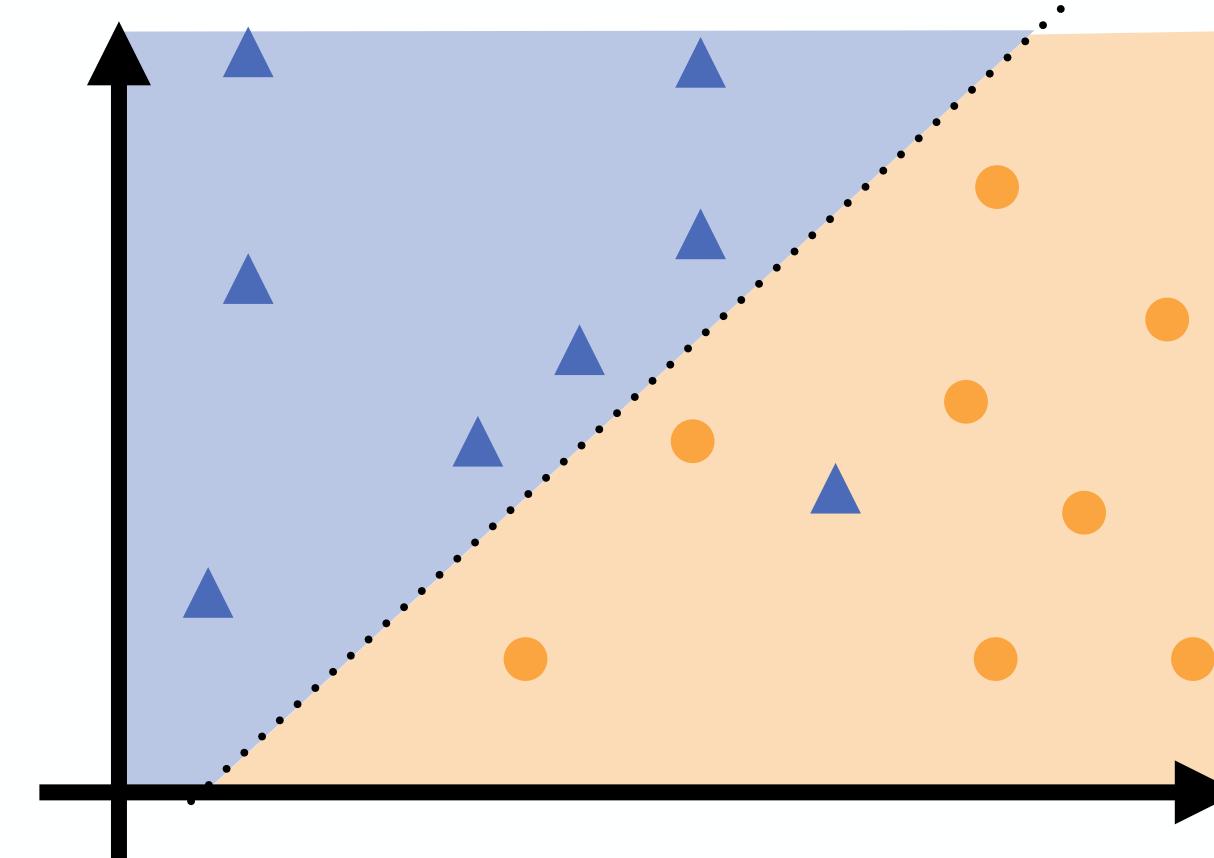
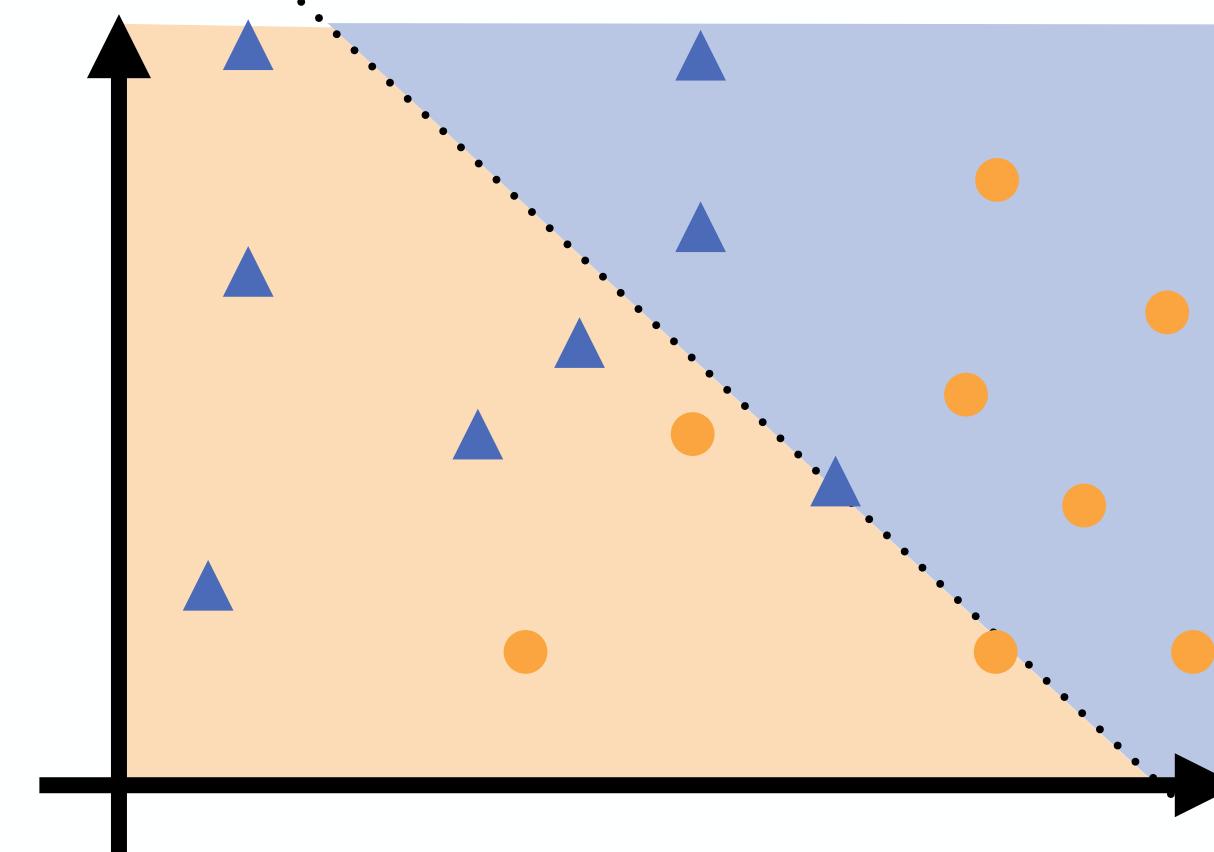
- Choosing informative features for the task makes it easy to discriminate between different classes
- If we do not measure the right things, it is impossible to have a model that performs well.
- See also: Bayes error, later this week

# *Steps involved in Machine Learning*



# *Evaluation: is this a good mapping?*

- Measure the performance: how well does the mapping solve the task?
- Measure this using a “loss function”
  - Specific to the problem we are trying to solve
  - Idea here: perhaps we should count the number of mistakes.



# Evaluation: Choosing a Performance Metric

*System 1: many small mistakes*

Predicted    Real



Only correct  
answer



**TU**Delft

*System 2: one big mistake*

Predicted    Real

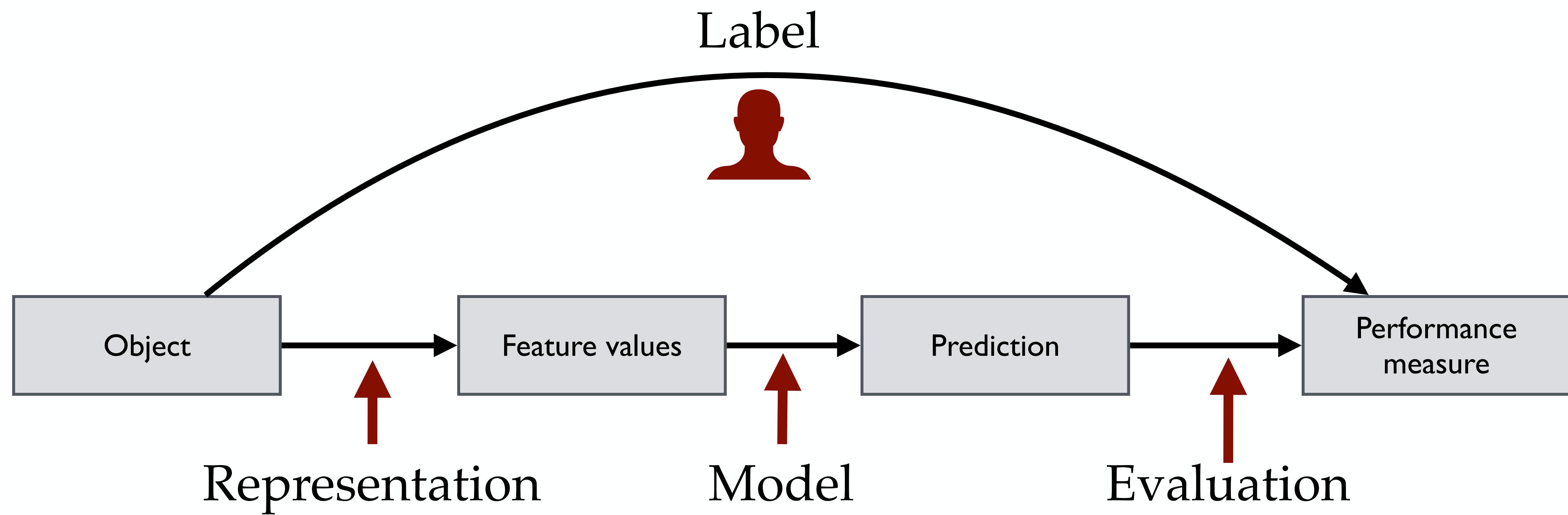


Only incorrect  
answer



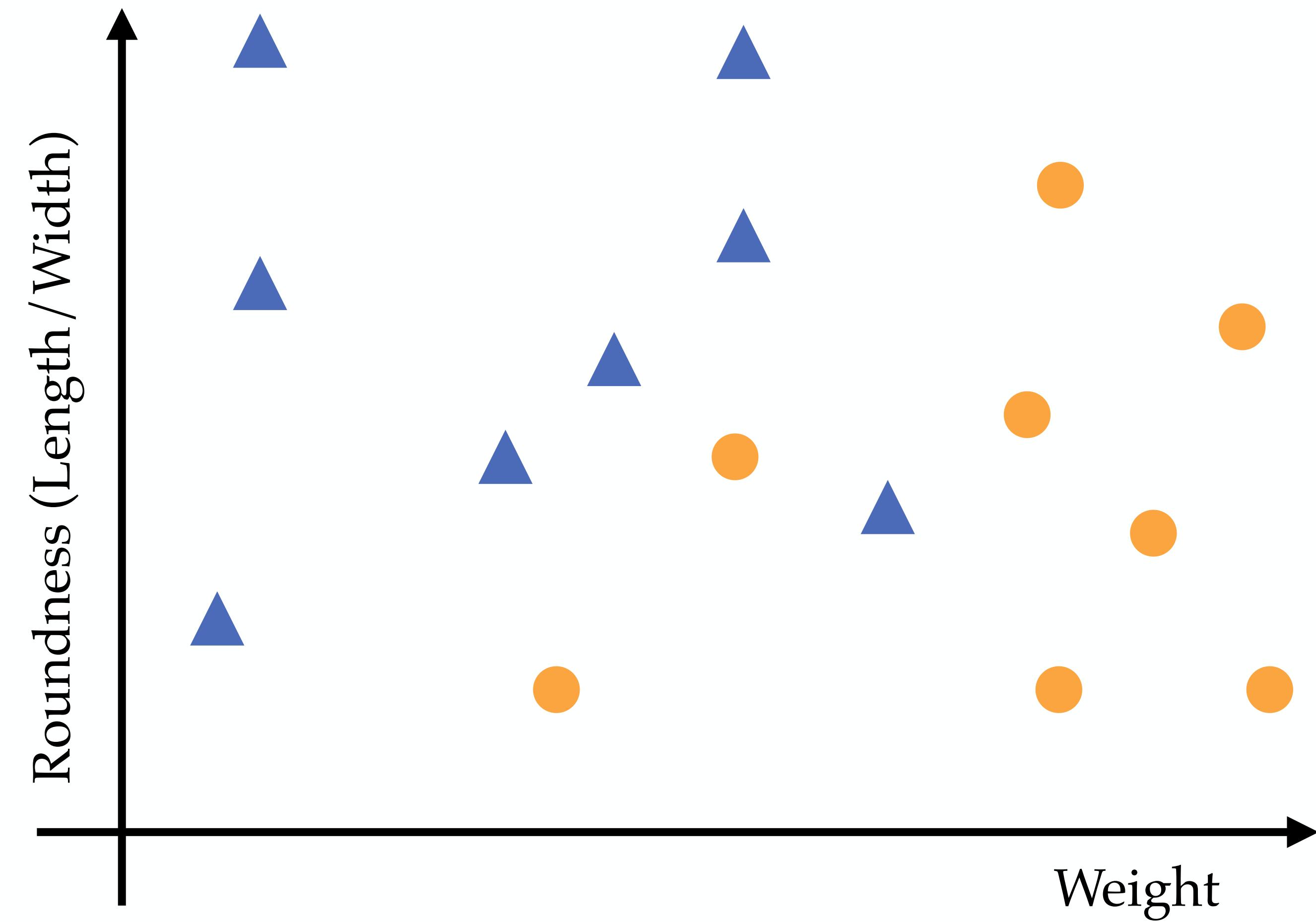
67

# *Steps involved in Machine Learning*



What about learning?

# *Learning*



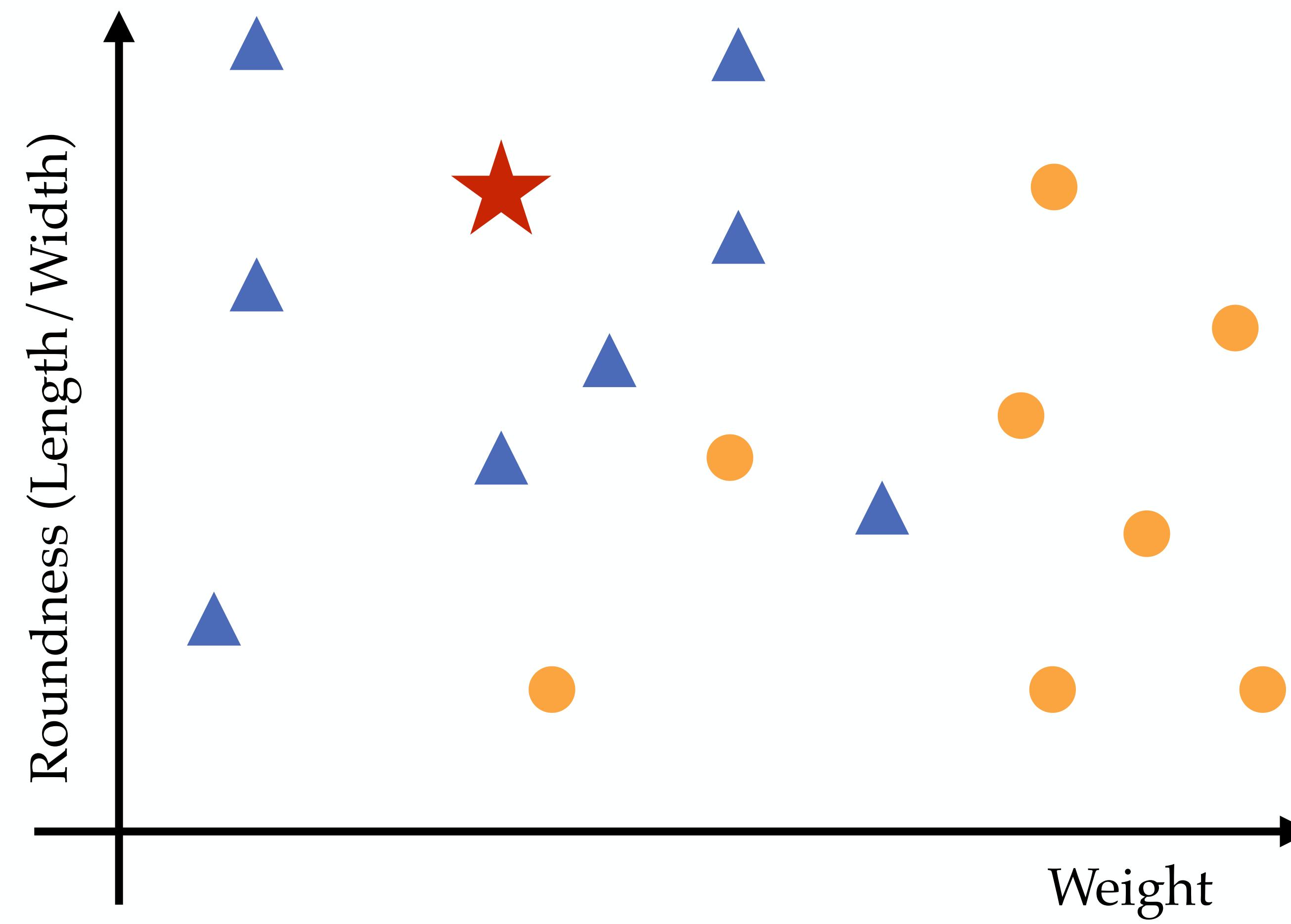
# *Learning and Generalisation*

- What about remembering?
- Going beyond just remembering each specific instance, to generalising to new instances.
- We want to learn an input-output mapping that works for previously unseen objects as well.

“To think is to forget a difference, to generalize, to abstract.”

- Jorge Luis Borges in “*Funes the Memorious*”

# *Model Predictions*



# *Learning: how to find a mapping*

$$\mathbb{E}_O[L(g, O)]$$

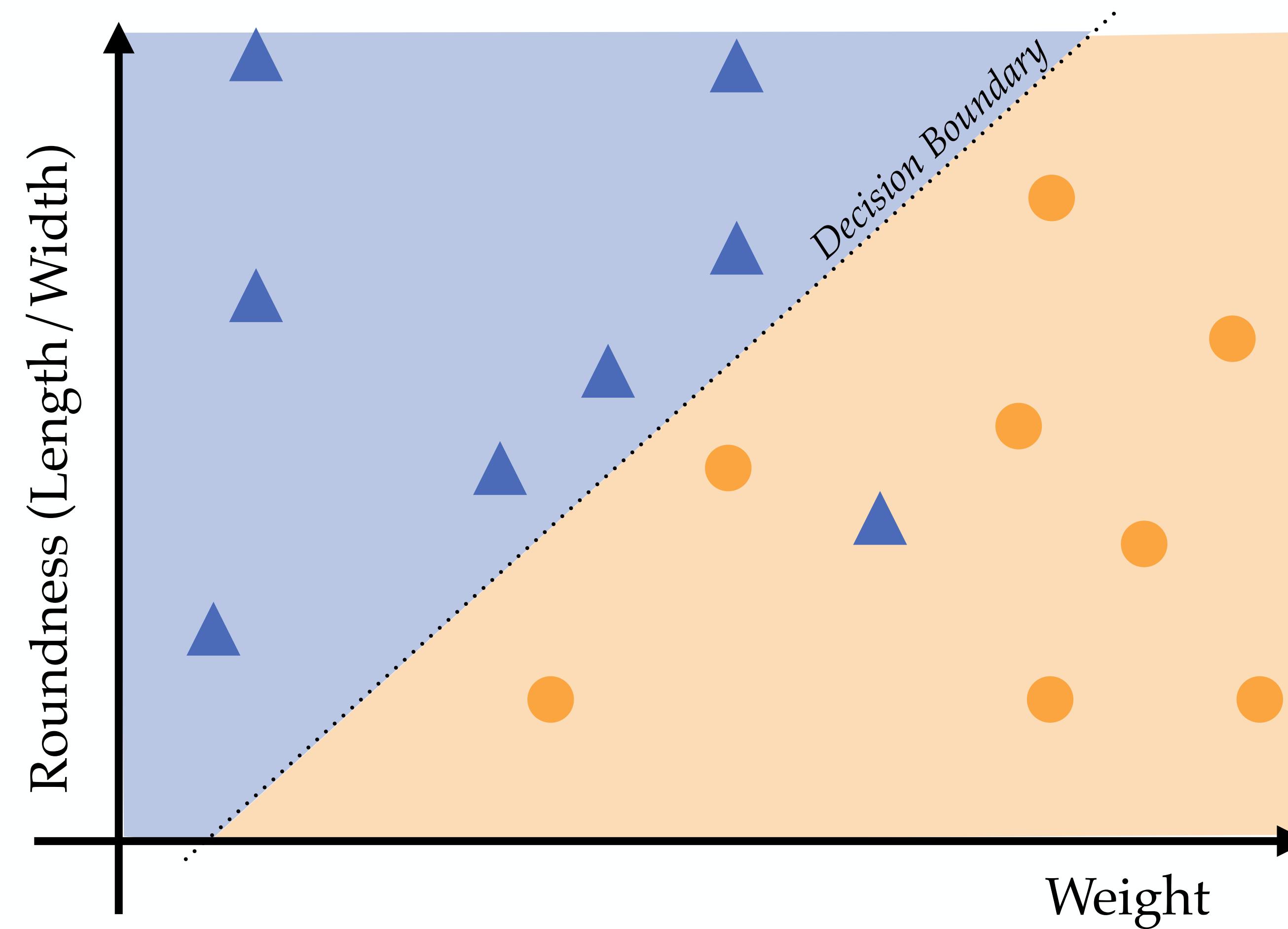
Most ML approaches: find parameters  $\mathbf{w}$  for a function  $g_{\mathbf{w}}$  that maximise the performance (using some performance metric).

**Problem 1:** what functions should we consider?

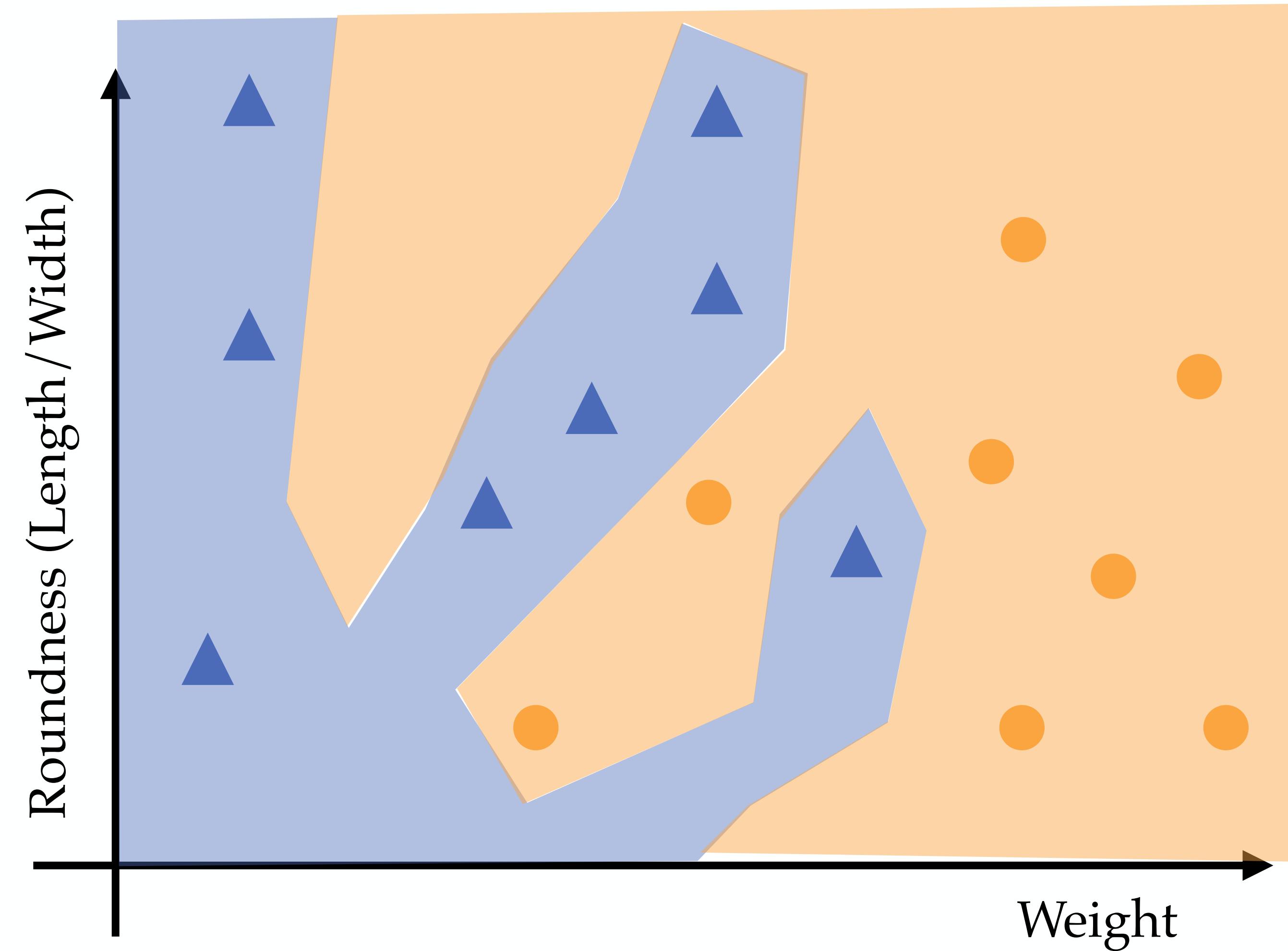
**Problem 2:** we don't have access to the true distribution

This course: cover many ideas on how to deal with these problems to construct learning algorithms.

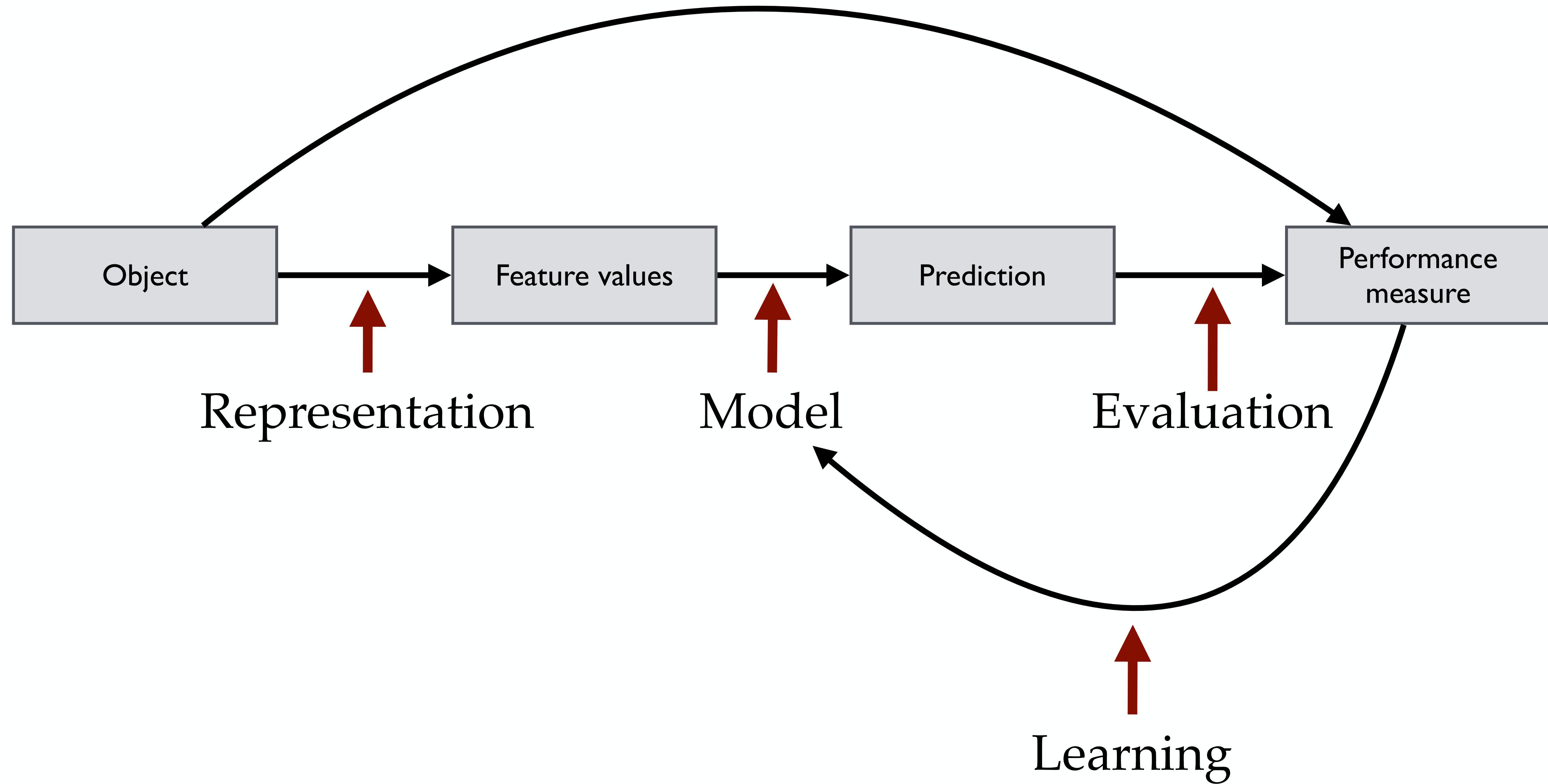
# *Linear Classifier*



# *Non-Linear Classifier (Better?)*



# *Steps involved in Machine Learning*



# Data

	Label	Features			
	Class	Weight	Length/ Width	Greenness	Density
Each object is a row →	1	Apple	150	1.05	0.2
	2	Apple	200	1.5	0.1
	3	Pear	100	1.5	0.8
	4	Apple	300	0.8	0.6

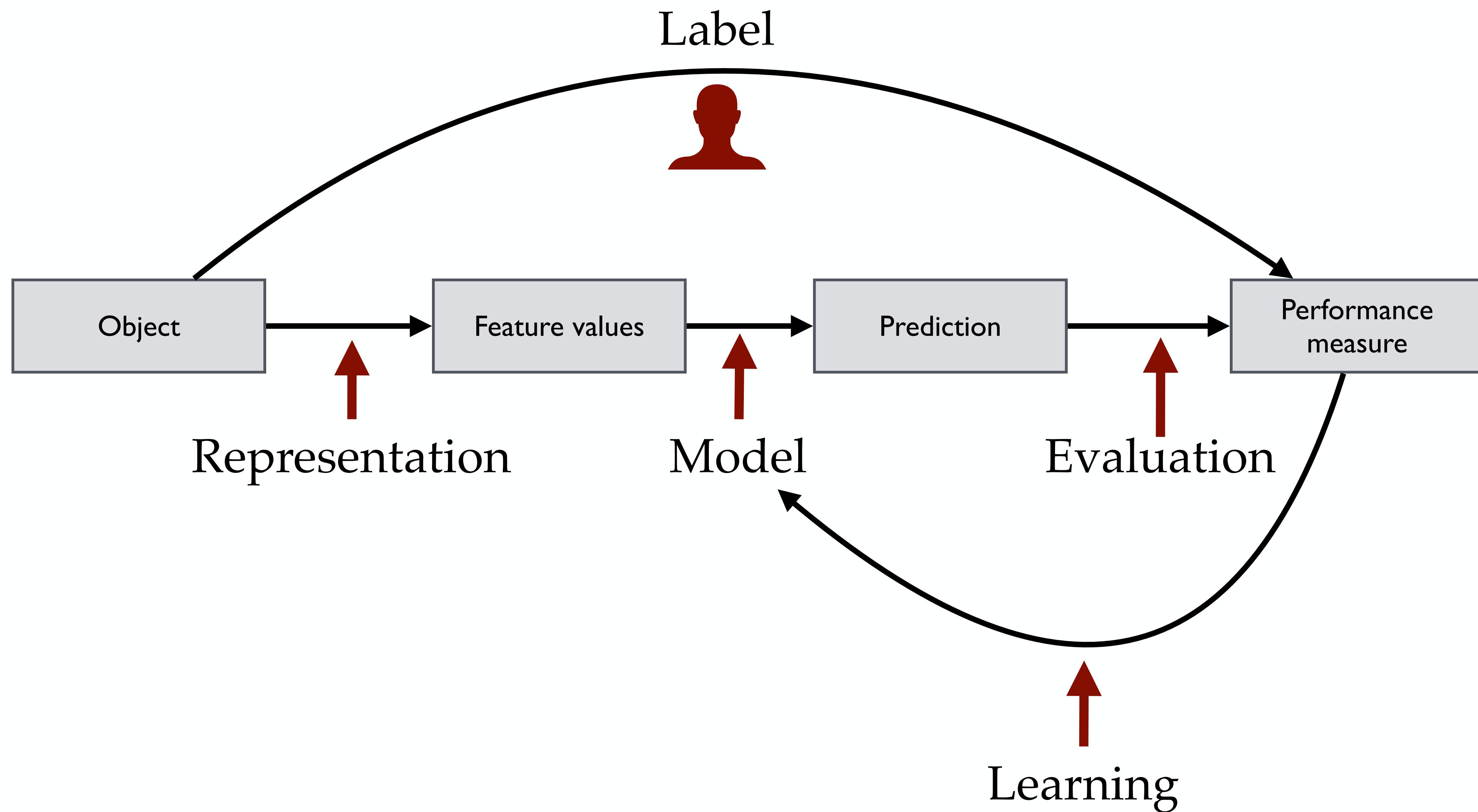
Label vector  $t$  (or  $y$ )       $N \times M$  Design matrix  $X$

Important common assumption: data sampled independently and identically distributed (i.i.d.) from an underlying problem distribution  $P(X,t)$ .

# *Data & Evaluation*

- Remember: we are also interested in the performance on new objects that we have not seen yet (it is unlikely that new fruit will have the exact same weight and dimensions as those fruits that we have observed before).
- Idea: split data into a **training set** and a **test set**
  - Use training set to fit the model
  - Evaluate performance on the test set
  - Since the test set is an i.i.d. sample from the data generating process, this can give a fair estimate of the generalisation performance
  - More on this in week 3.2

# *Steps involved in Machine Learning*



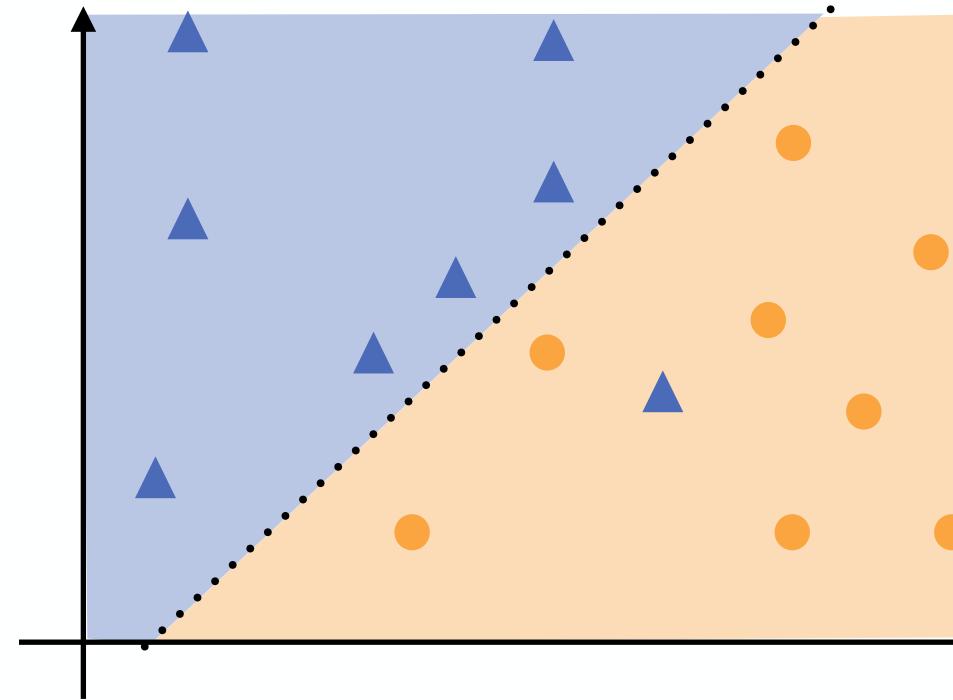
# *Types of Machine Learning*

- Supervised Learning
  - Examples: **Classification**, Regression
- Unsupervised Learning
  - Examples: Dimensionality Reduction, Clustering
- Reinforcement Learning (not this course)
  - Example: Selecting optimal actions

# *Supervised Learning*

## CLASSIFICATION

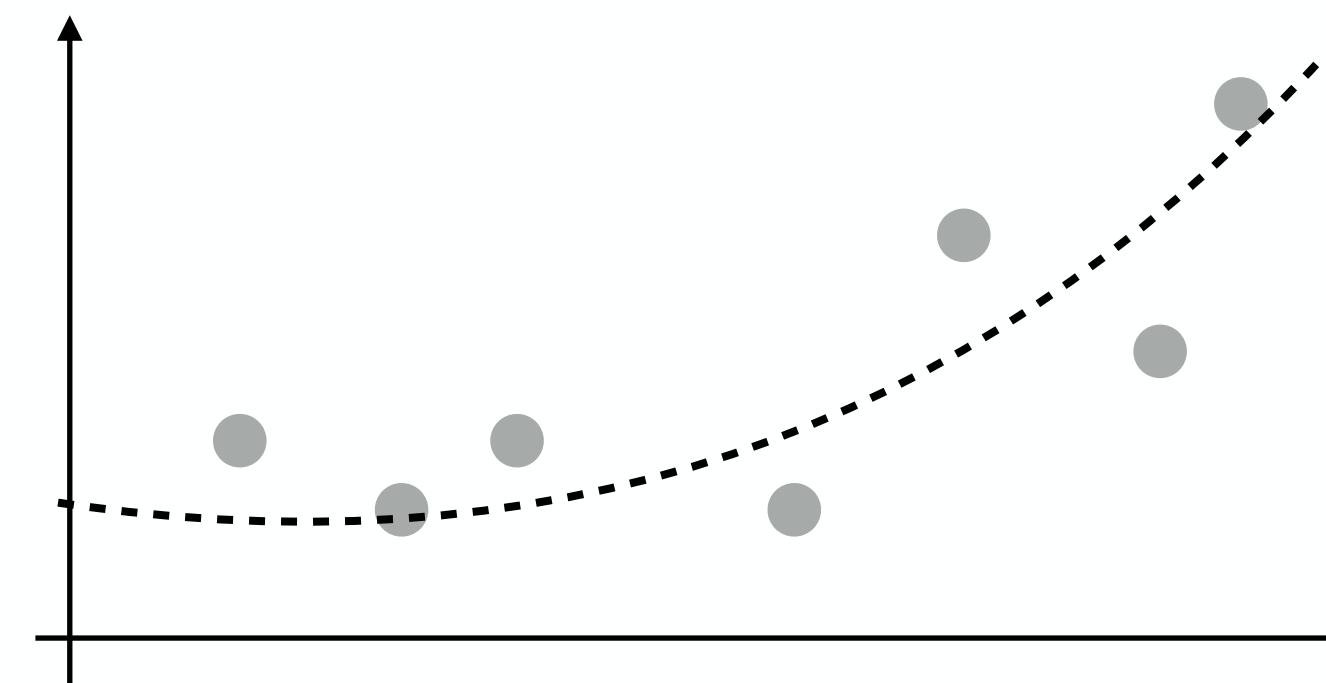
$g : \mathbb{R}^M \rightarrow S$  where  $|S|$  is small



**Examples**  
Object detection  
Click prediction

## REGRESSION

$g : \mathbb{R}^M \rightarrow \mathbb{R}$



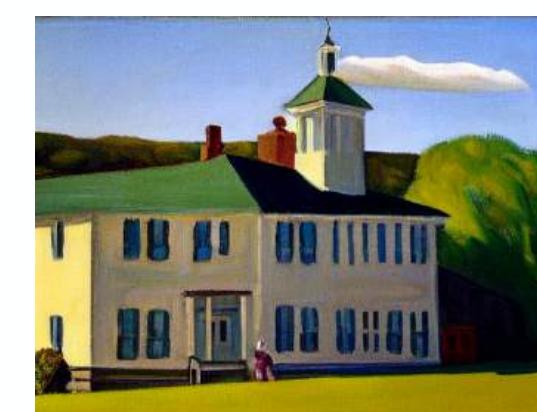
**Examples**  
Price prediction  
Temperature prediction

## “STRUCTURED OUTPUTS”

### Example

$g : \text{sentence} \rightarrow \text{image}$

“A Hopper painting of a village”

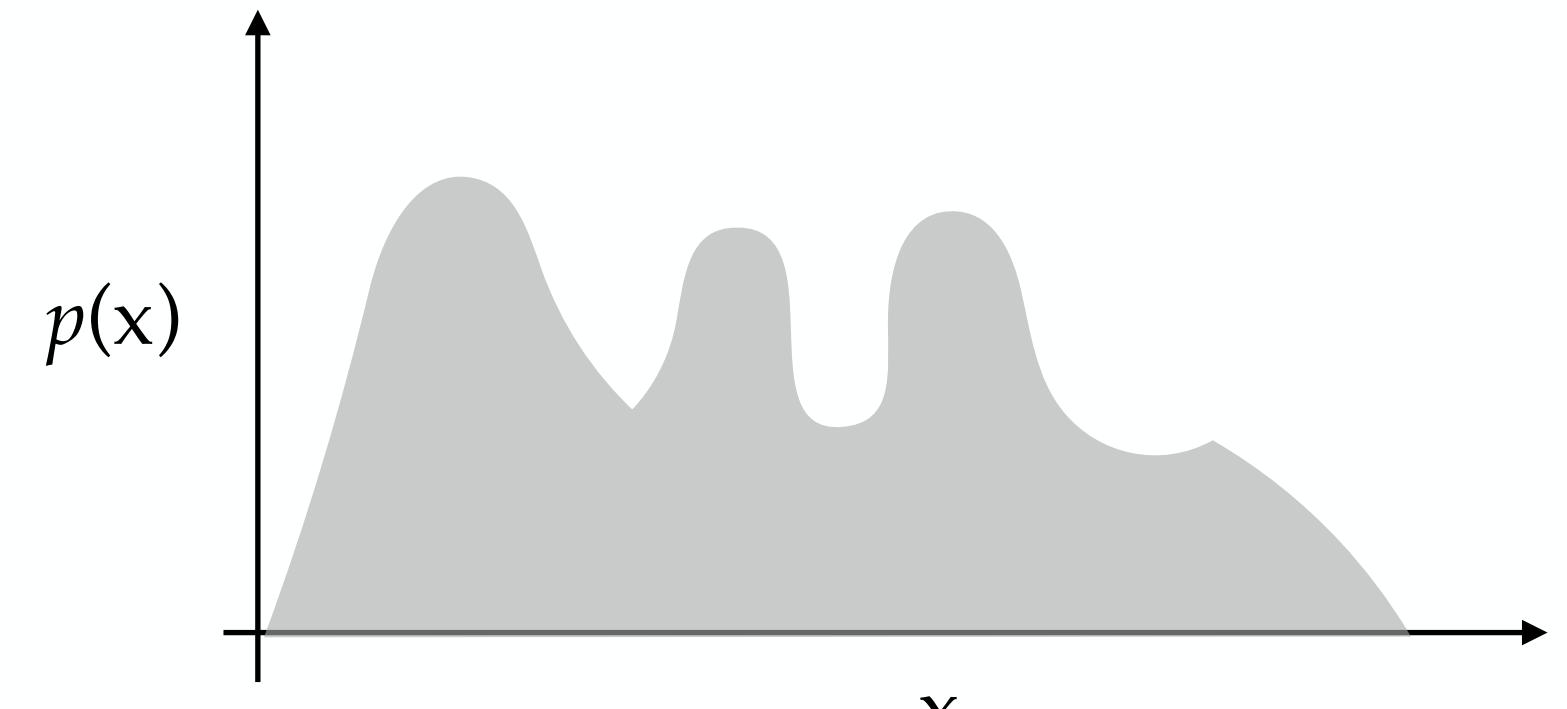


# *Unsupervised Learning*

## No output label given

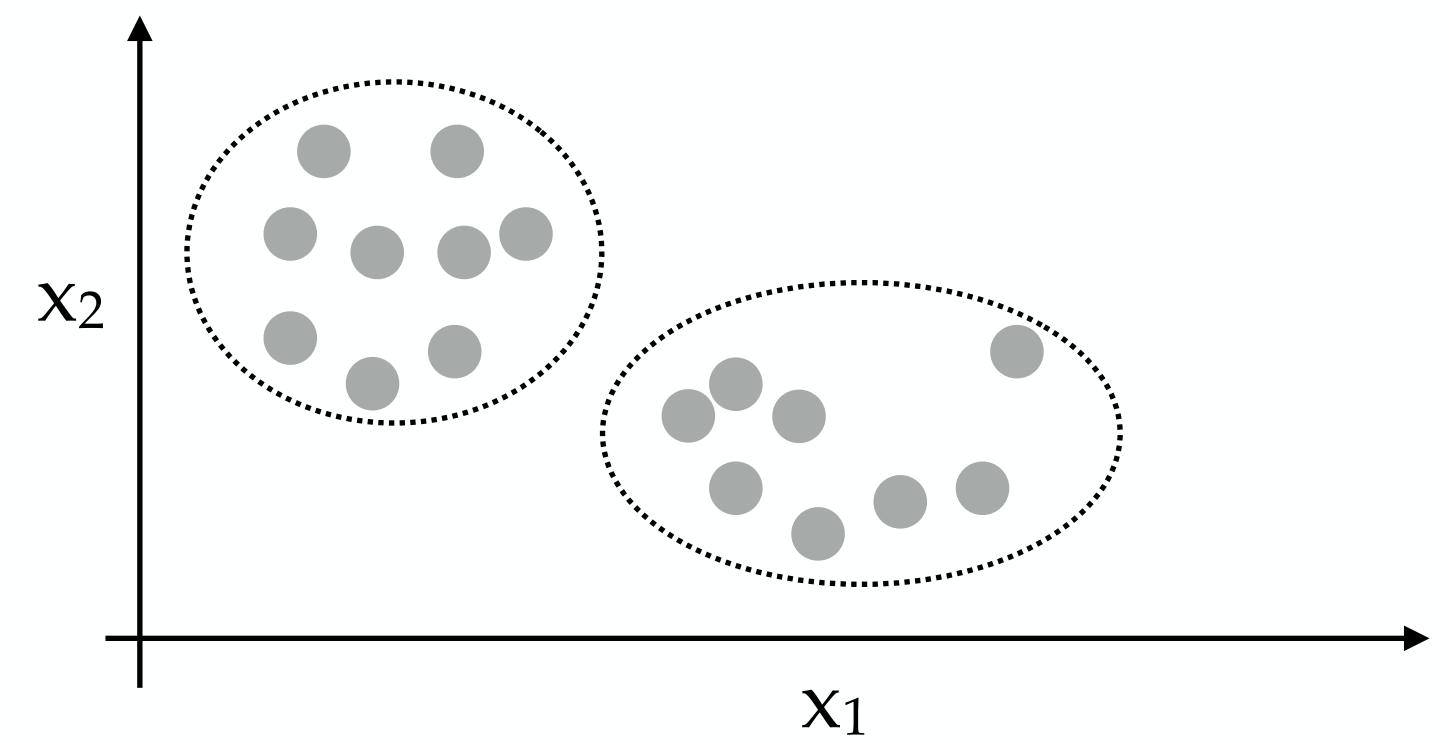
### Density estimation

- For each  $x$ , give an estimate of the probability density  $p(x)$ .



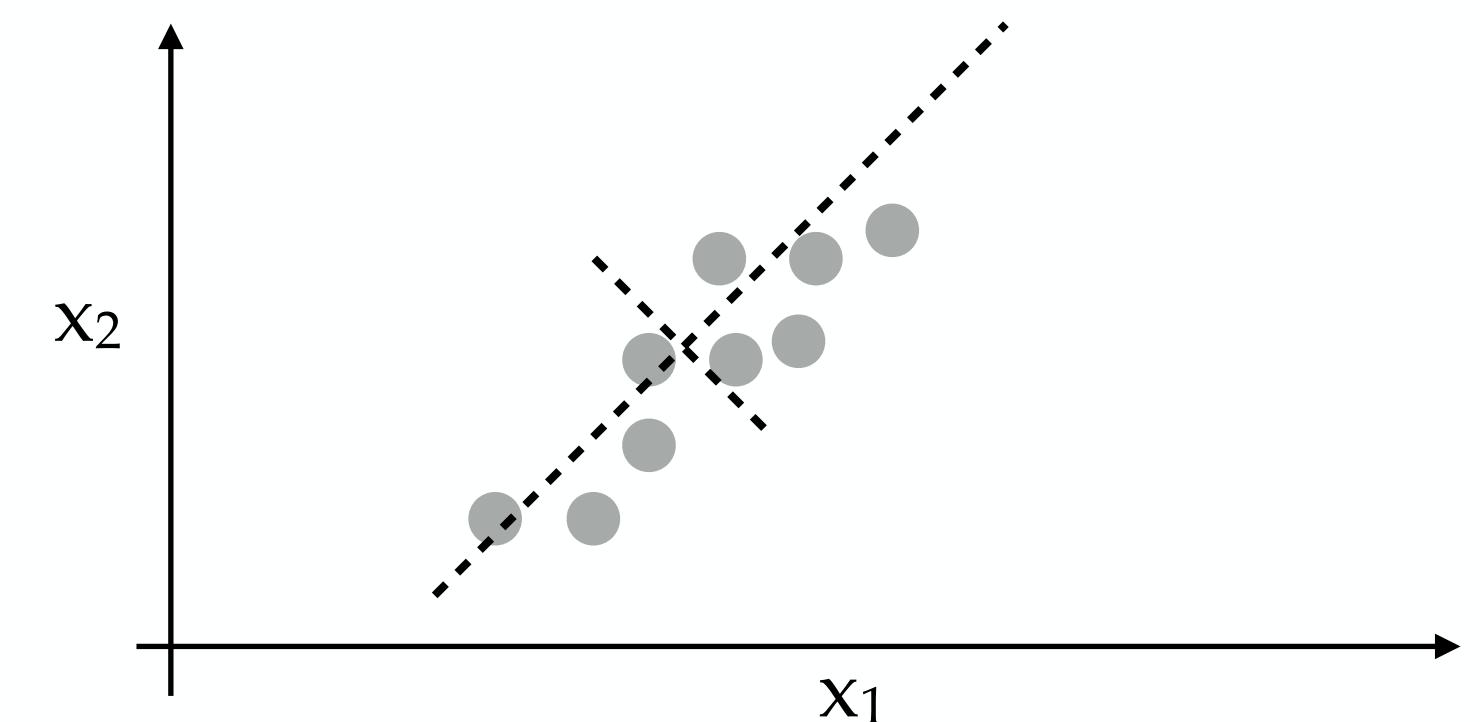
### Clustering

- Identify a small number of “groups” in the data
- For instance, groups of observations that are internally similar, but different from the other groups

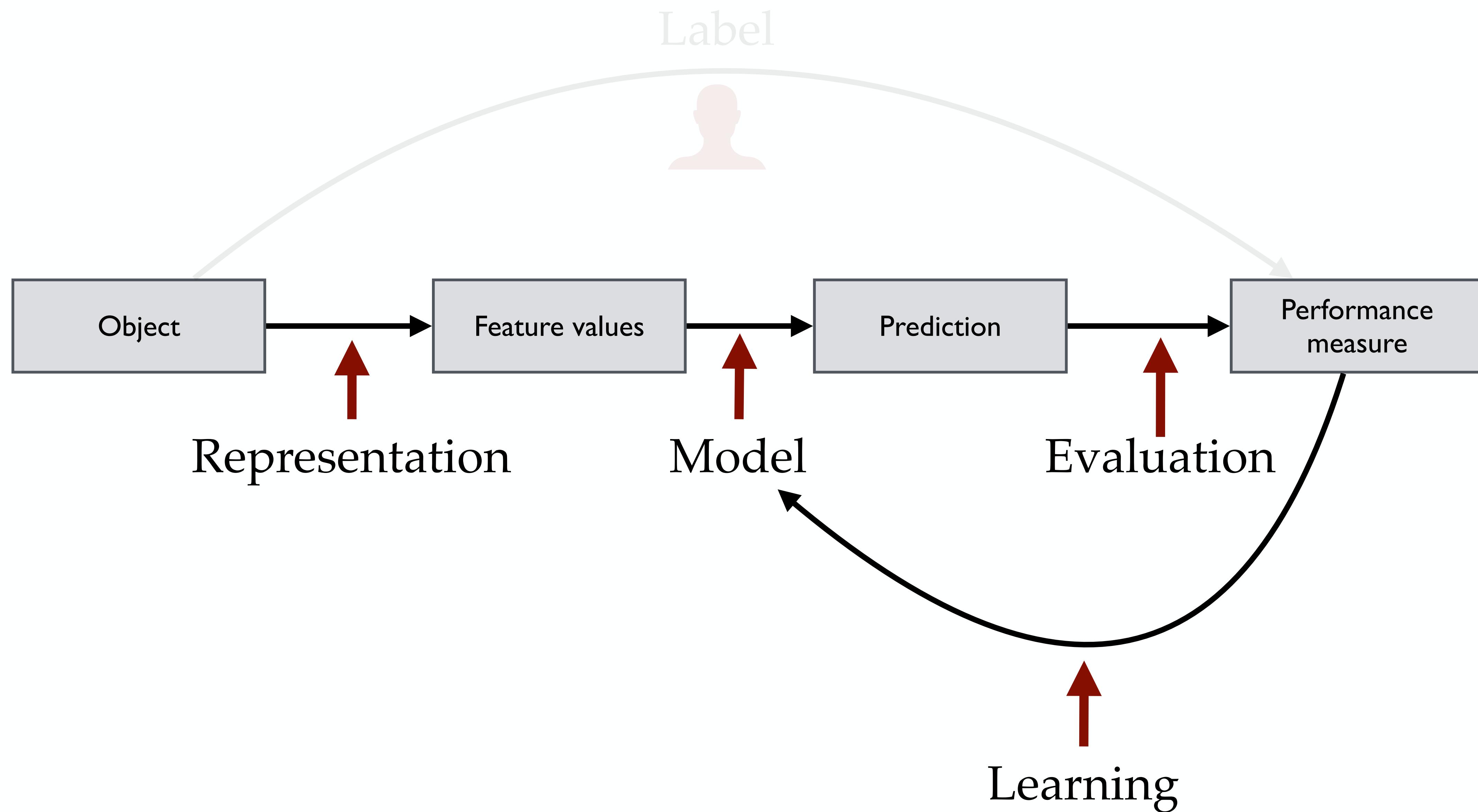


### Dimensionality reduction

- Find a lower dimensional description of a high dimensional objects
- For instance, construct new features that contain most of the information in a smaller number of features.



# *Steps involved in Machine Learning*



# *Practice: What Type of Learning?*

- Predicting customer churn (whether a customer leaves or not)
- Finding groups of customers that behave similarly
- Playing a video game
- Discovering different ways in which people play a videogame
- Identifying which photos contain waterfalls
- Predicting the price of a house
- Identifying which tax forms are more likely to be fraudulent
- Ranking webpages based on a search query
- Recommending the next movie a customer should watch
- Predicting which TikTok video the user is most likely to continue watching
- Predicting the amount of rainfall next Saturday

In the course we cover at least 10 classifiers.

Why do we need to study so many different learning approaches?

*Why not just study the best one?*

# No Free Lunch Theorems and Machine Learning

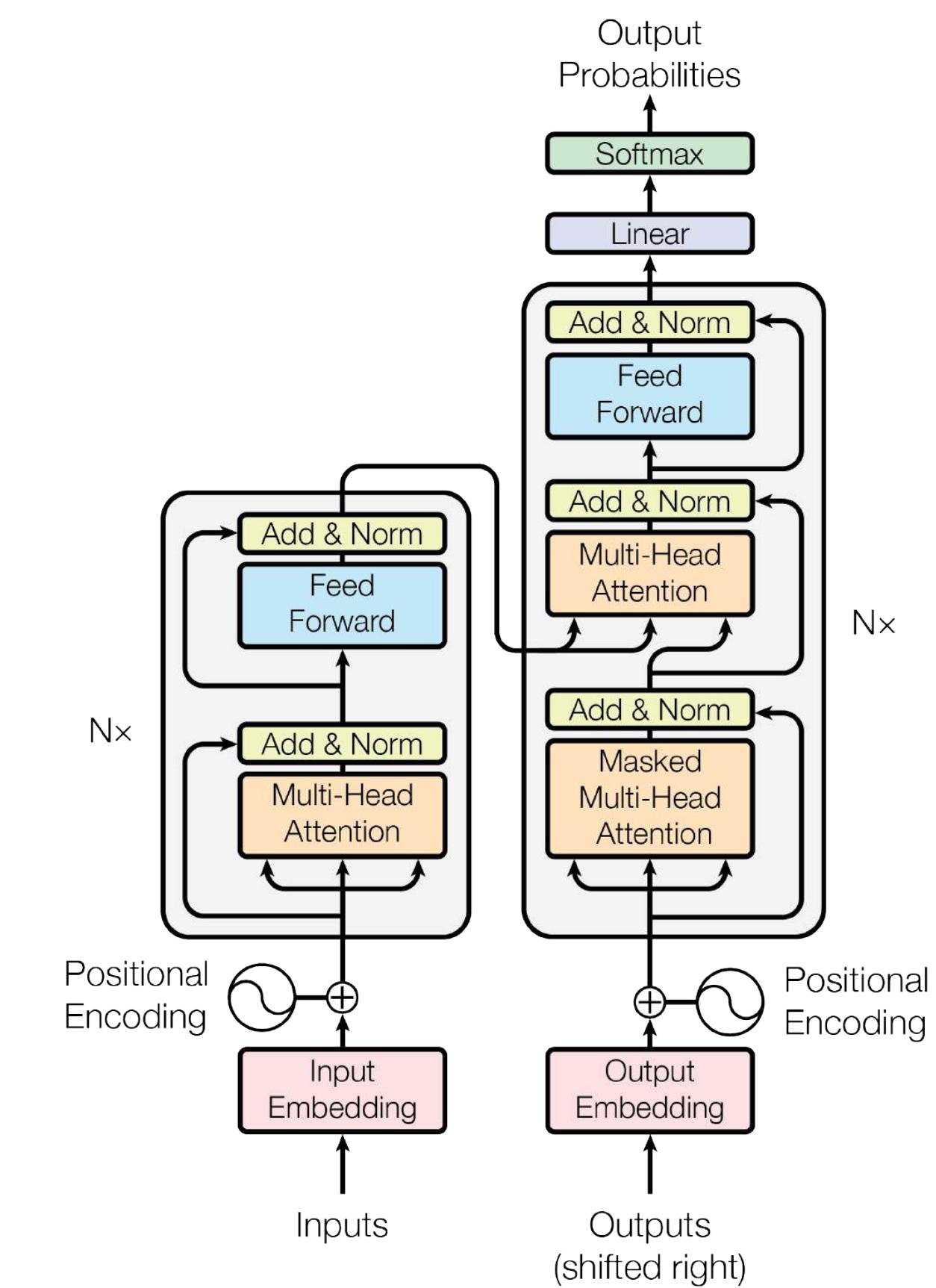
- No free lunch theorems: uniformly averaged over all data-generating processes, every classification algorithm has the same out of sample prediction error. (*Note: the actual theorems are more precise and specific*)
- *For every problem a learner works well on, there is a problem on which it does not work well*
- **Lesson:** we are not looking to find the learning algorithm that is always best, but we want to find out which approach works well for which problem, and understand why.

# Starting Simple

*Let's start with this:*

$$w_0 + \mathbf{w}^T \mathbf{x} > 0$$

*To build up to things like this:*

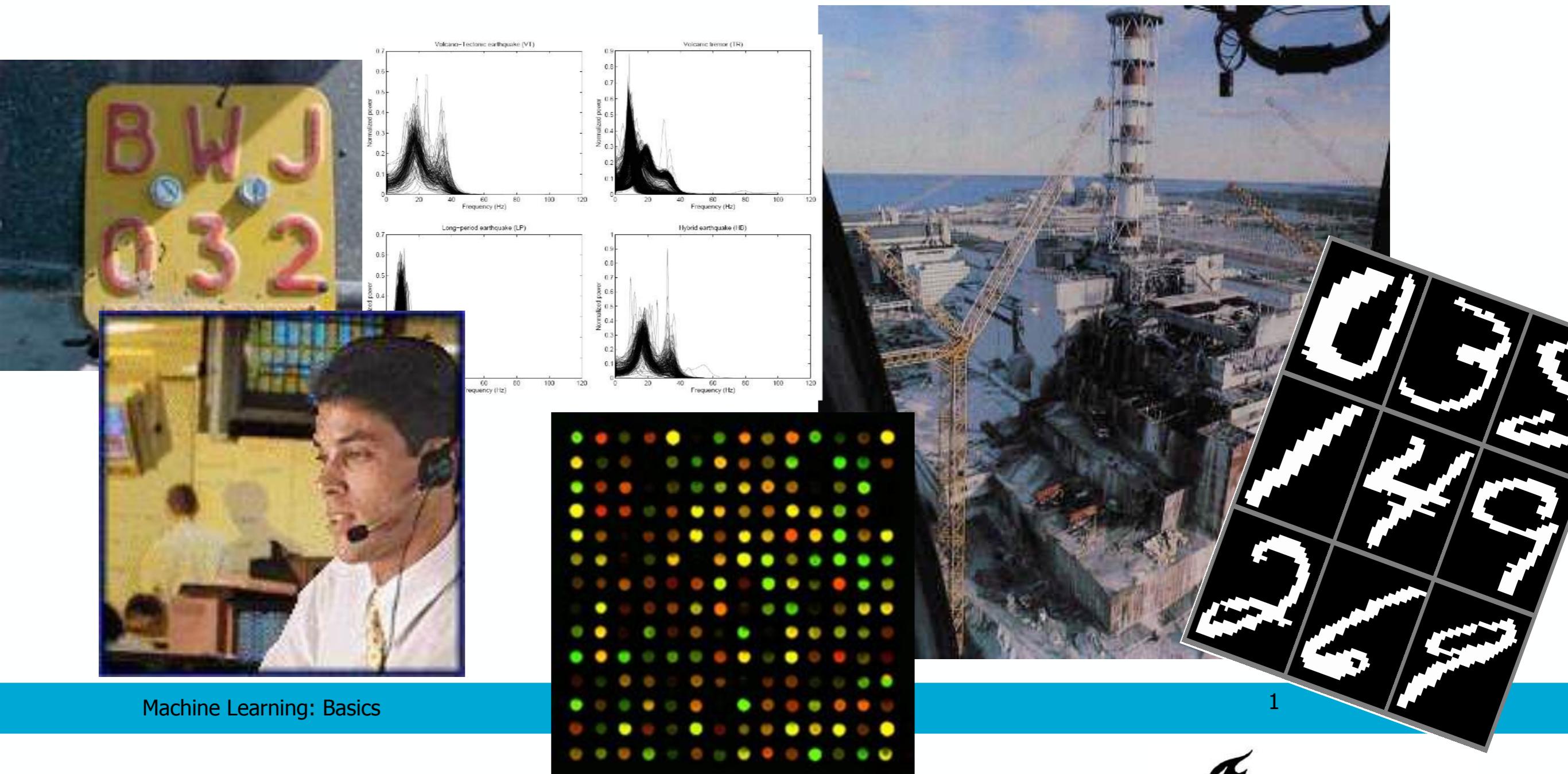


# *Take-aways*

- Machine Learning is about using examples to learn a mapping from input to a desired output, that **generalises well** to new examples.
- In this course we focus on supervised (classification & regression) and unsupervised learning problems
- We specify a learning problem using objects, features, classes (targets), data and a performance metric.
- Learning has close connections to statistics and decision theory: next lecture!

# Machine Learning: the basics

- Learning from examples



# Outline of this lecture

- Objects, features, measurements,
  - ... datasets and feature space
- Traditional pattern recognition: classification
- Class posterior probabilities and Bayes' rule
- Bayes' classifier and Bayes' error
- Misclassification costs

# Learning from Examples

- Given some **examples**, we may perform:
  - clustering
  - outlier detection
  - classification
  - regression
  - ...on new objects.



- We assume that no complete physical model is known!

# Generalization

We don't want to just describe the data...

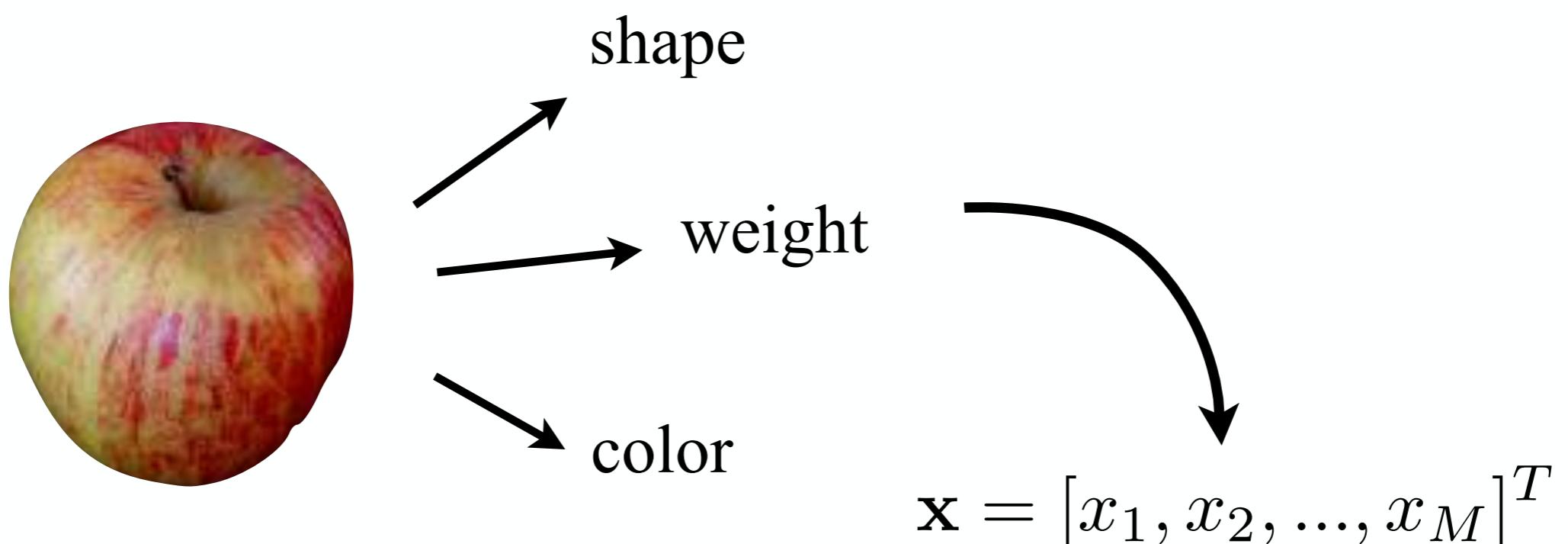
**We want to predict  
for new, unseen  
data!**

# Generalization

- **Training set:**  
All examples are labeled  
This set is used to train/develop our system
- **Test set:**  
These examples cannot be used to train our system  
The examples do not have to be labeled  
When labels are available, we can objectively evaluate our system

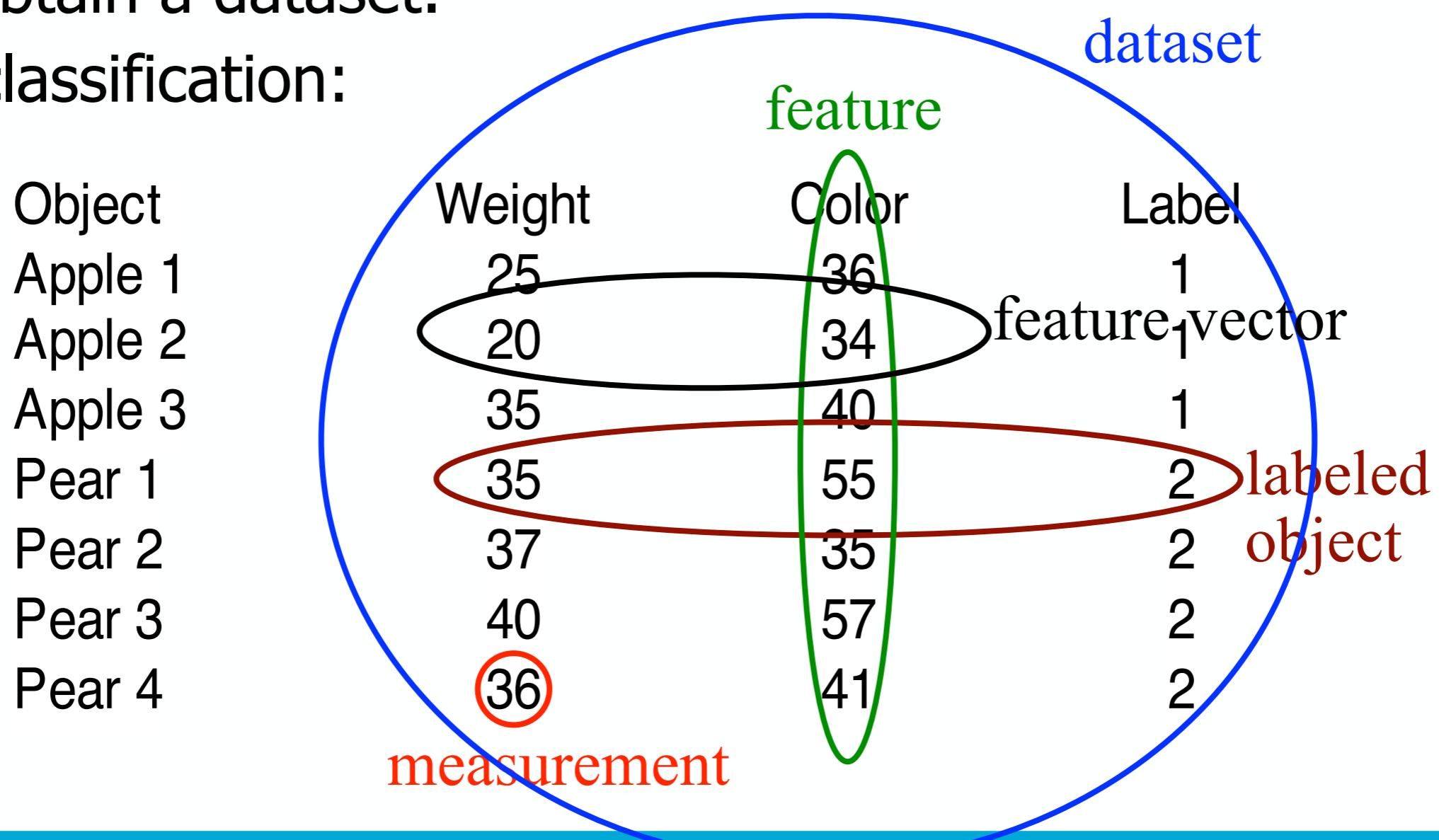
# Features

- To do these tasks automatically, we have to encode the objects.
- Objects are typically encoded by defining features:



# Datasets

- When we measure the features of many objects: we obtain a dataset.
- For classification:



# How to define features?

- Note that features reduce, and give a specific view of the objects: YOU (the user) is responsible for it
- Good features allow for pattern recognition, bad features allow for nothing
- Other (than feature vector) approaches of defining objects are:
  - Dissimilarity approach
  - Structural pattern recognition (graphs)
- Feature approach is very well developed, other approaches are still more research.

# Noise in the measurements

- The measurements will never be perfect
- Objects within a class will vary

**We need to apply some statistics to cover all the variations**

# Measurements

- Task: distinguish between 3 types of Iris flowers:

(Images from Wikipedia)



Iris Setosa

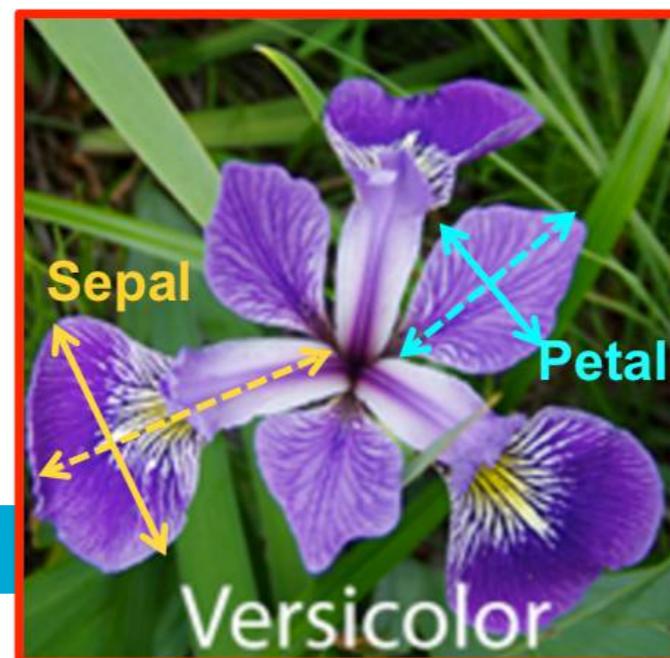


Iris Versicolor



Iris Virginica

- Measurements:  
sepal width, sepal length,  
petal width, petal length.



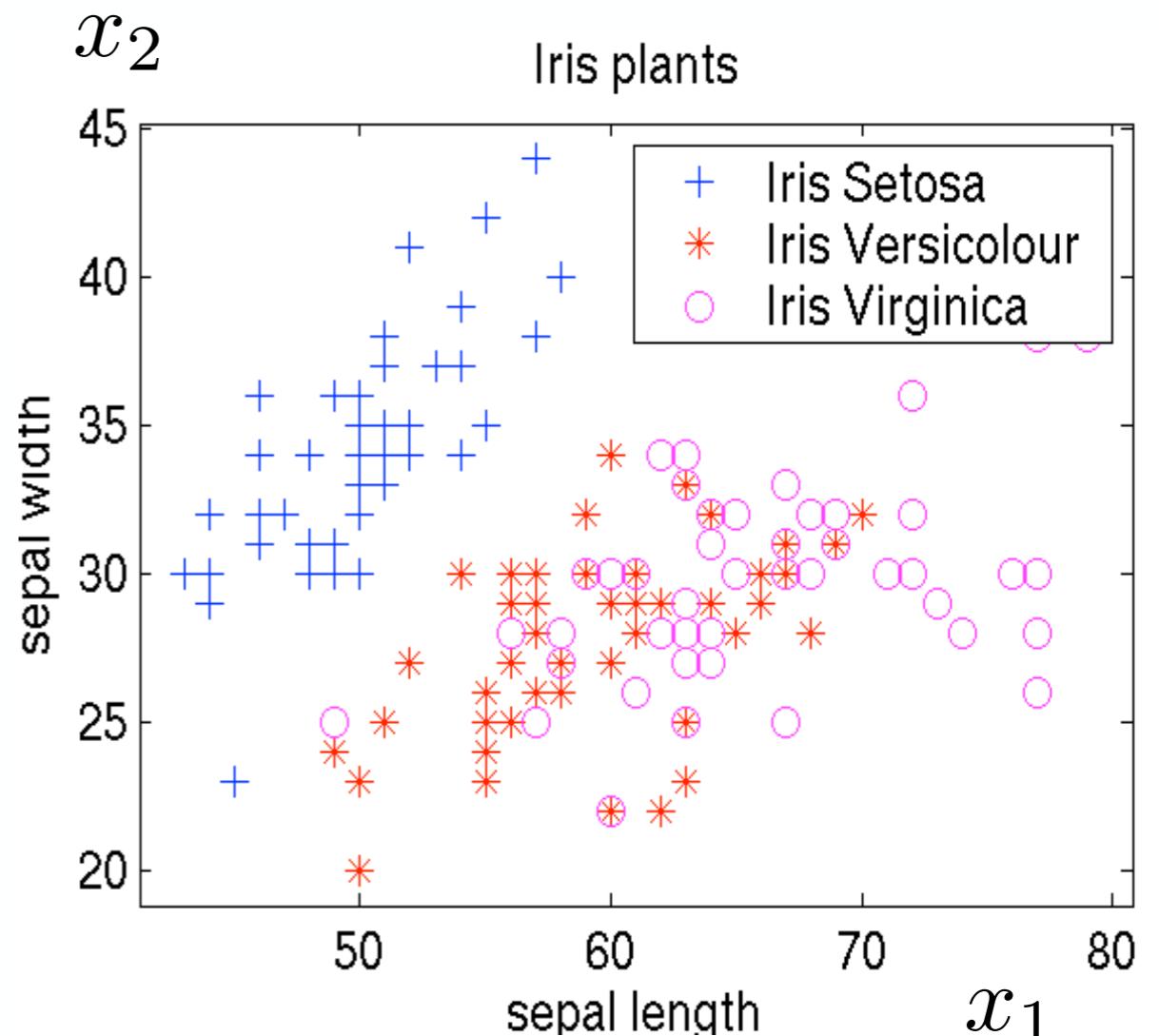
# Objects in feature space

- We can interpret the measurements as a vector in a vector space:

$$\mathbf{x} = [x_1, x_2, \dots, x_M]^T$$

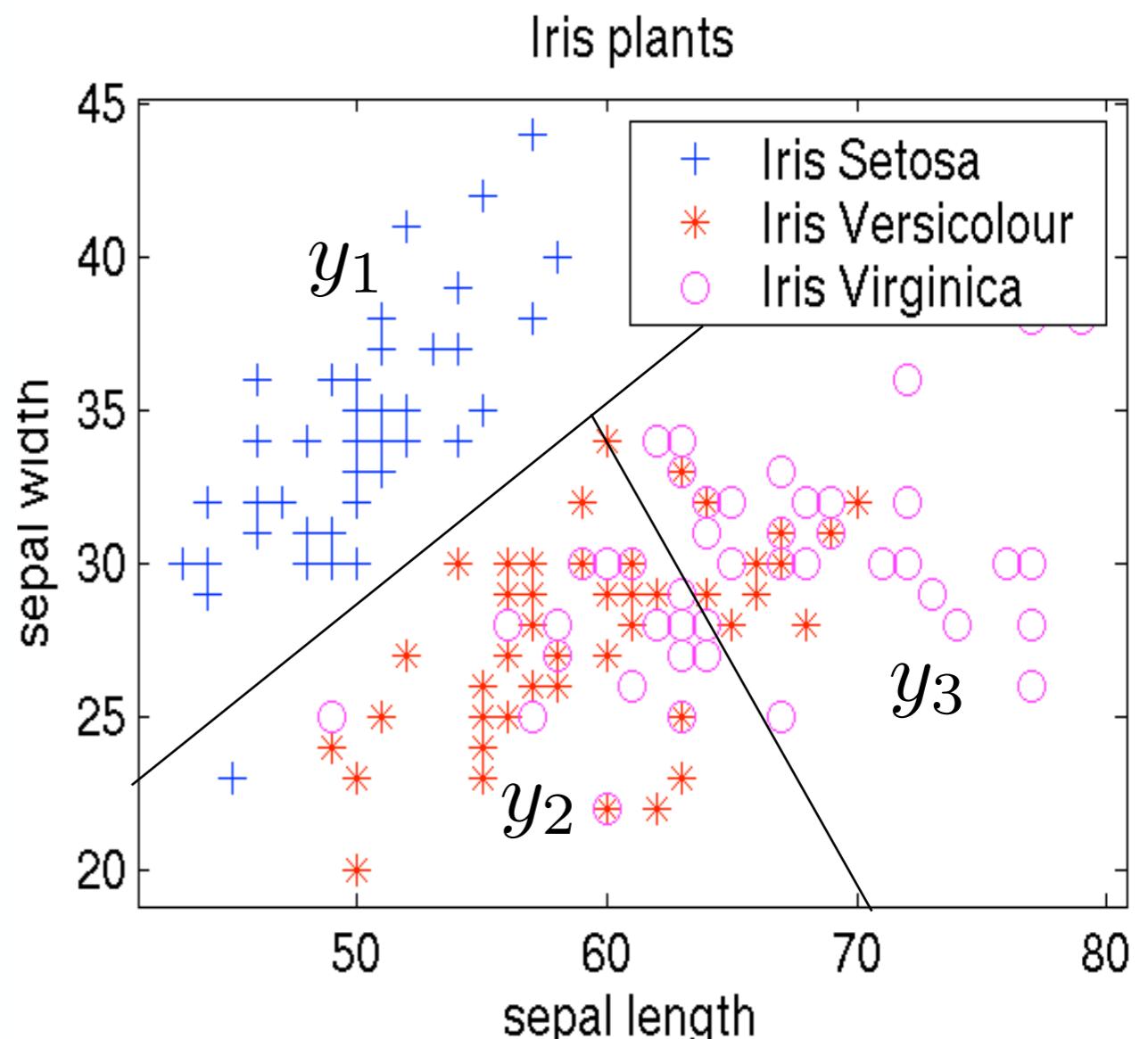
- This originates in principle from a probability density over the whole feature space

$$p(\mathbf{x}, y)$$



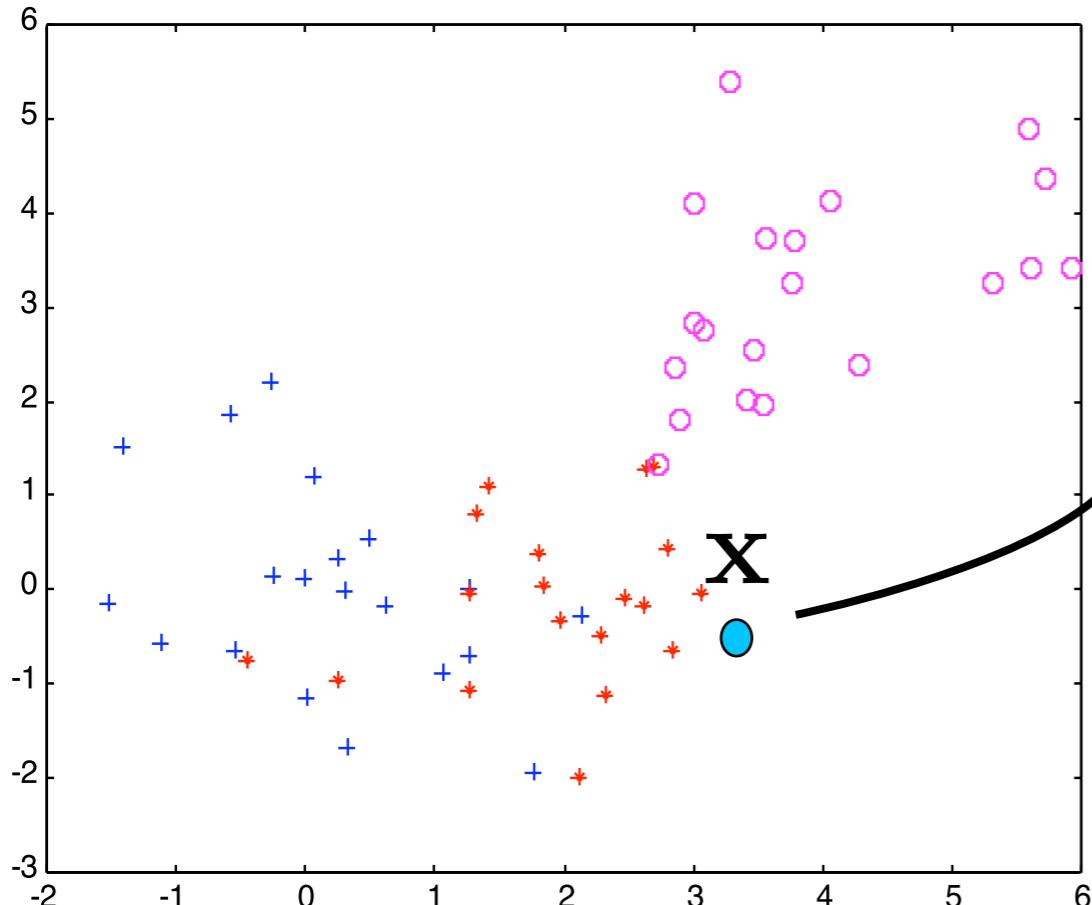
# Classification

- Given labeled data:  $\mathbf{x}$
- Assign to each object a class label  $y$
- In effect splits the feature space in separate regions



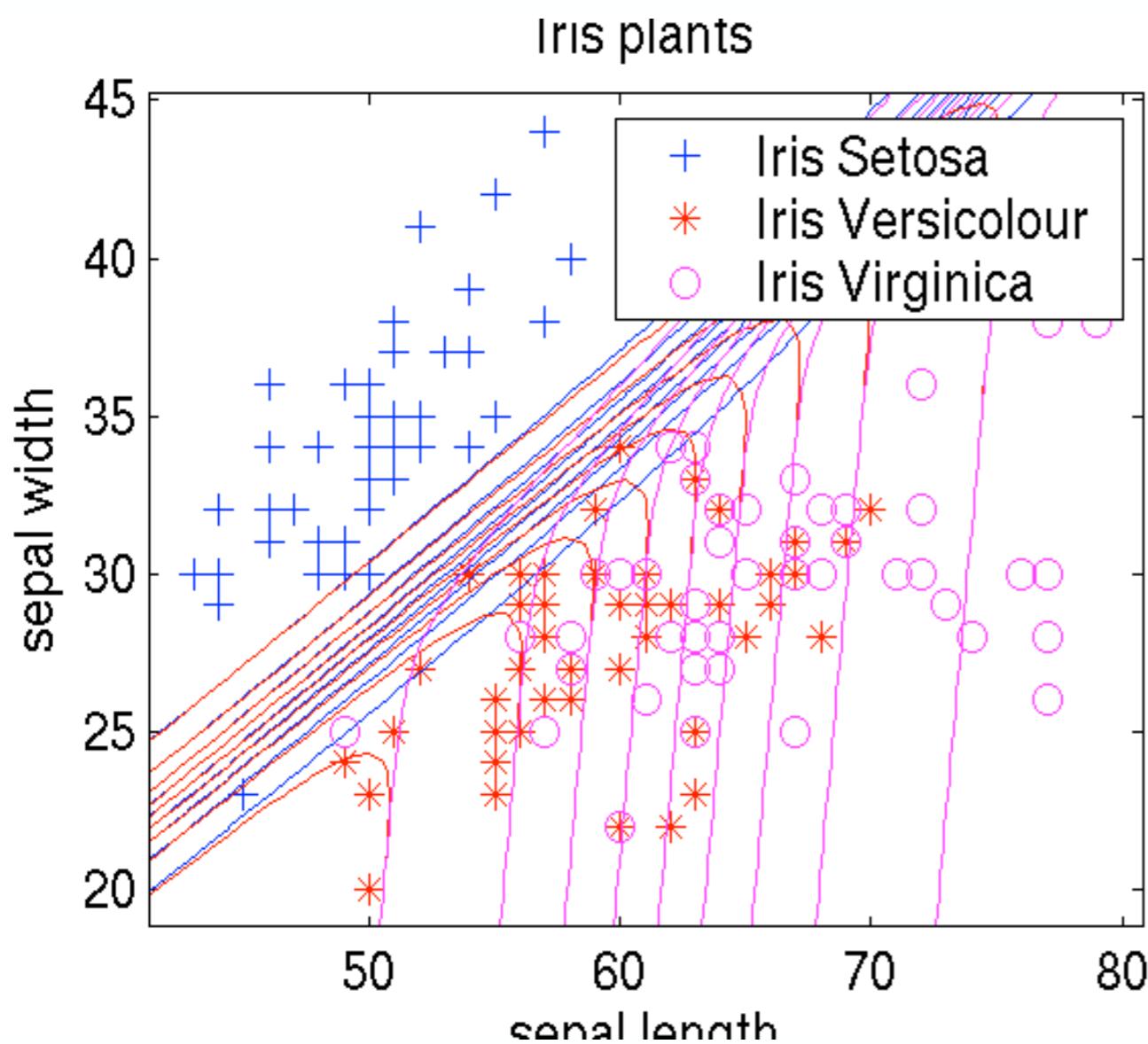
# The general model

Function  $f$  should give the predicted output.



$f(\mathbf{x}) \rightarrow y$   
and...  
model should  
be fitted to the  
data!  
 $p(y|\mathbf{x})$

# Output of the model



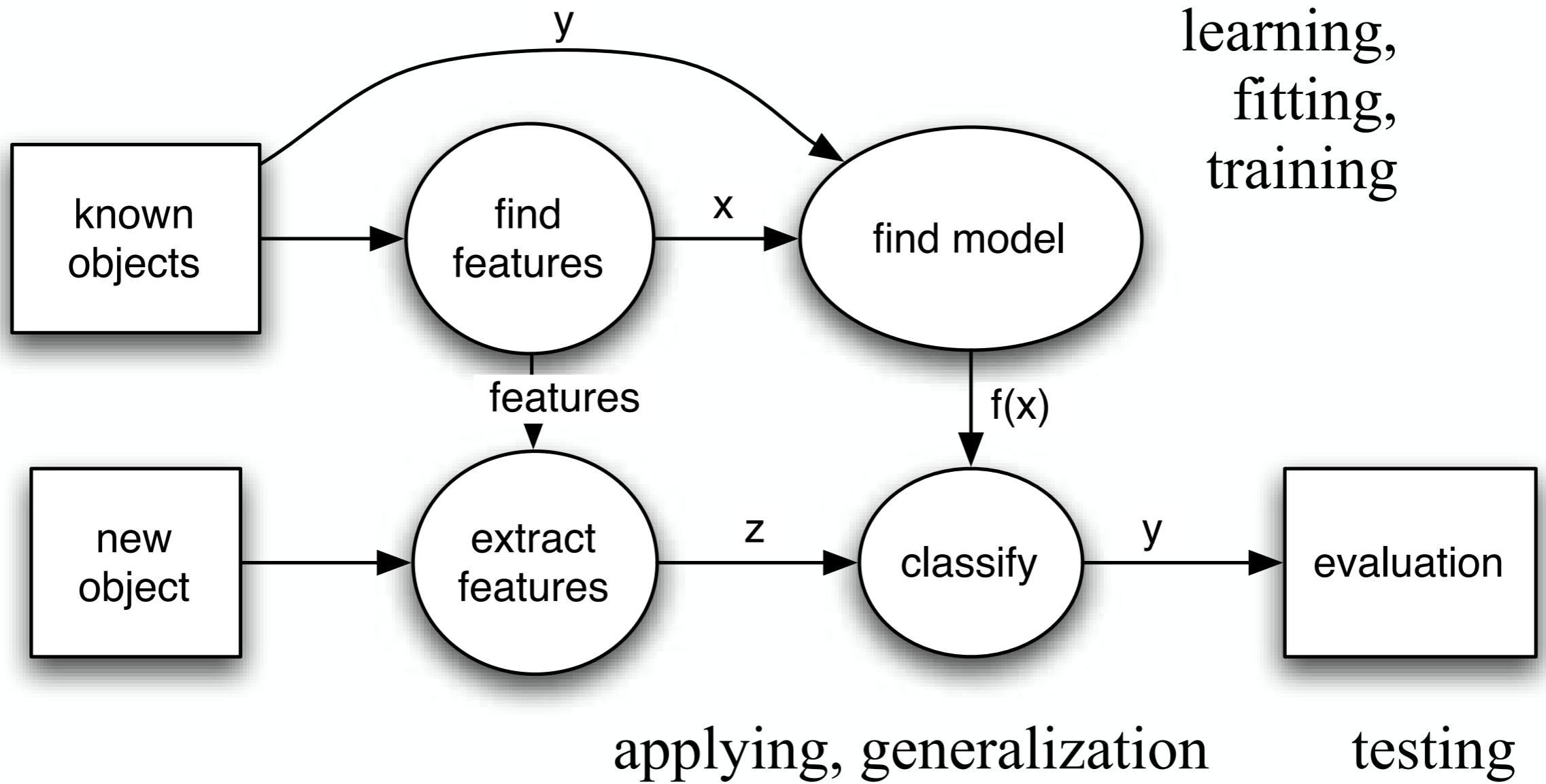
- For each object in the feature space, we should estimate:

$$p(y|\mathbf{x})$$

- In practice we fit a function:

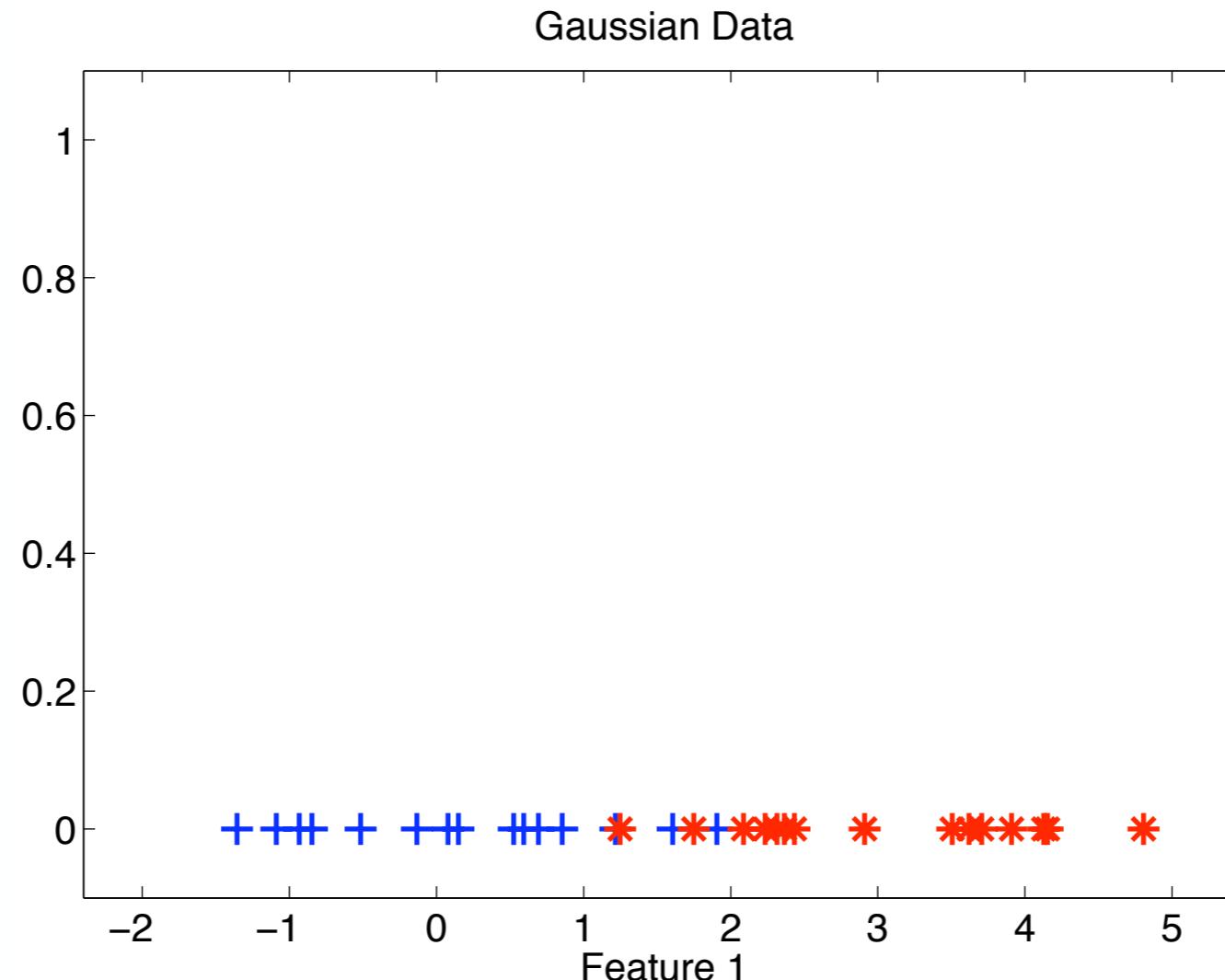
$$f(\mathbf{x})$$

# Pattern Recognition pipeline



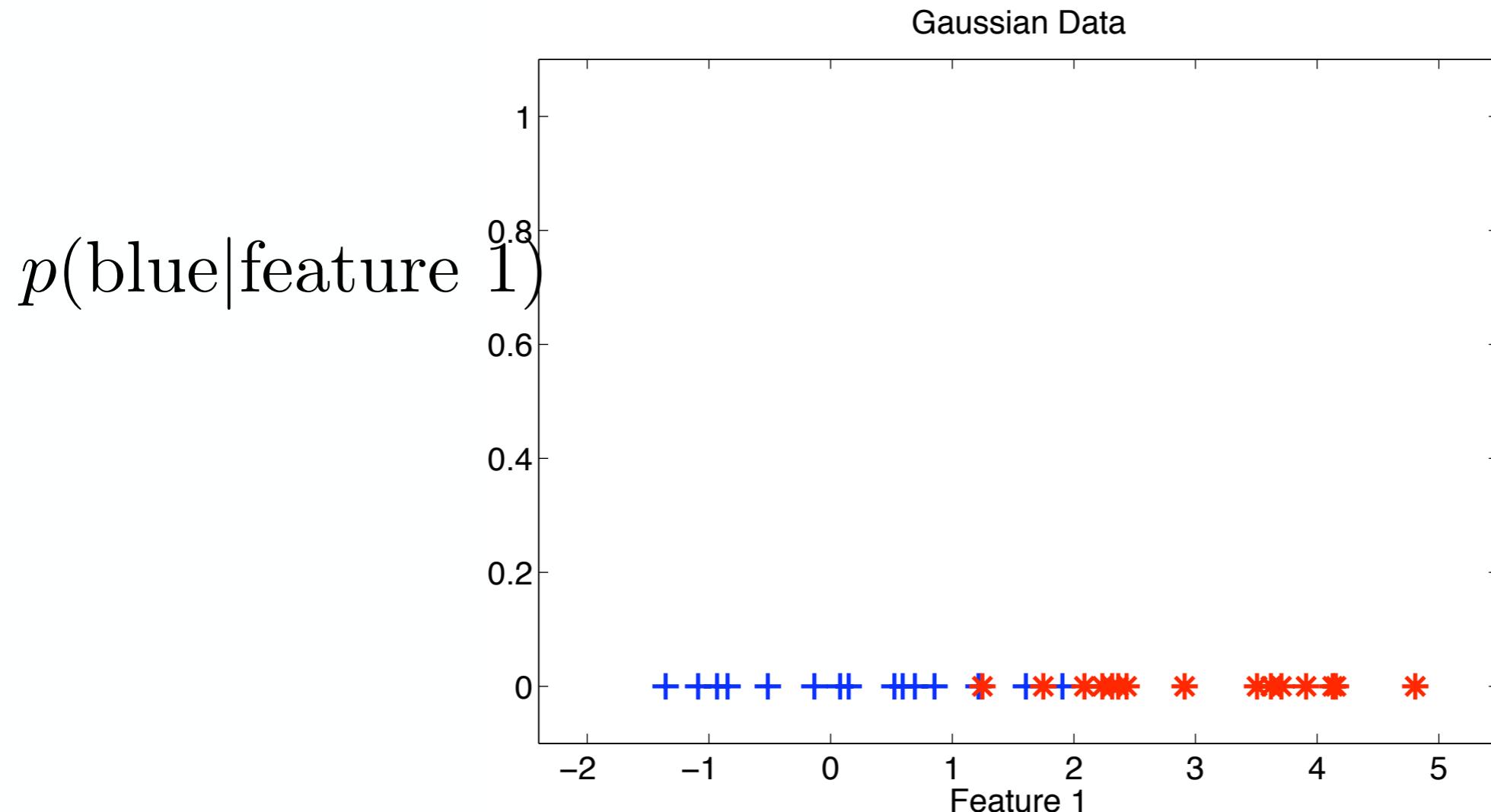
# Classification, how to do it?

- Given a feature, and a training set, where is the blue class?



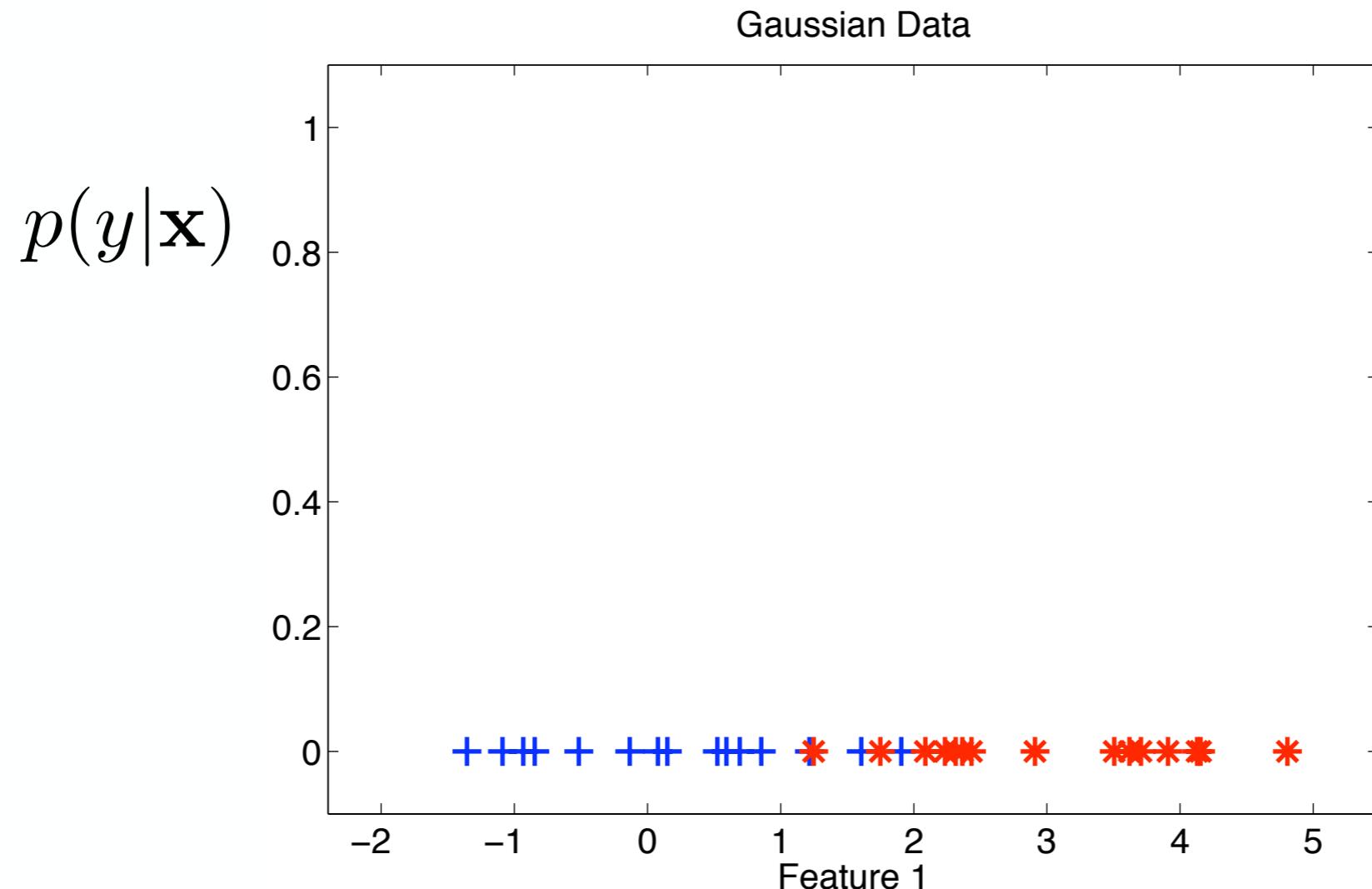
# Class posterior probability

- For each object we want to estimate  $p(\text{blue}|\text{feature } 1)$



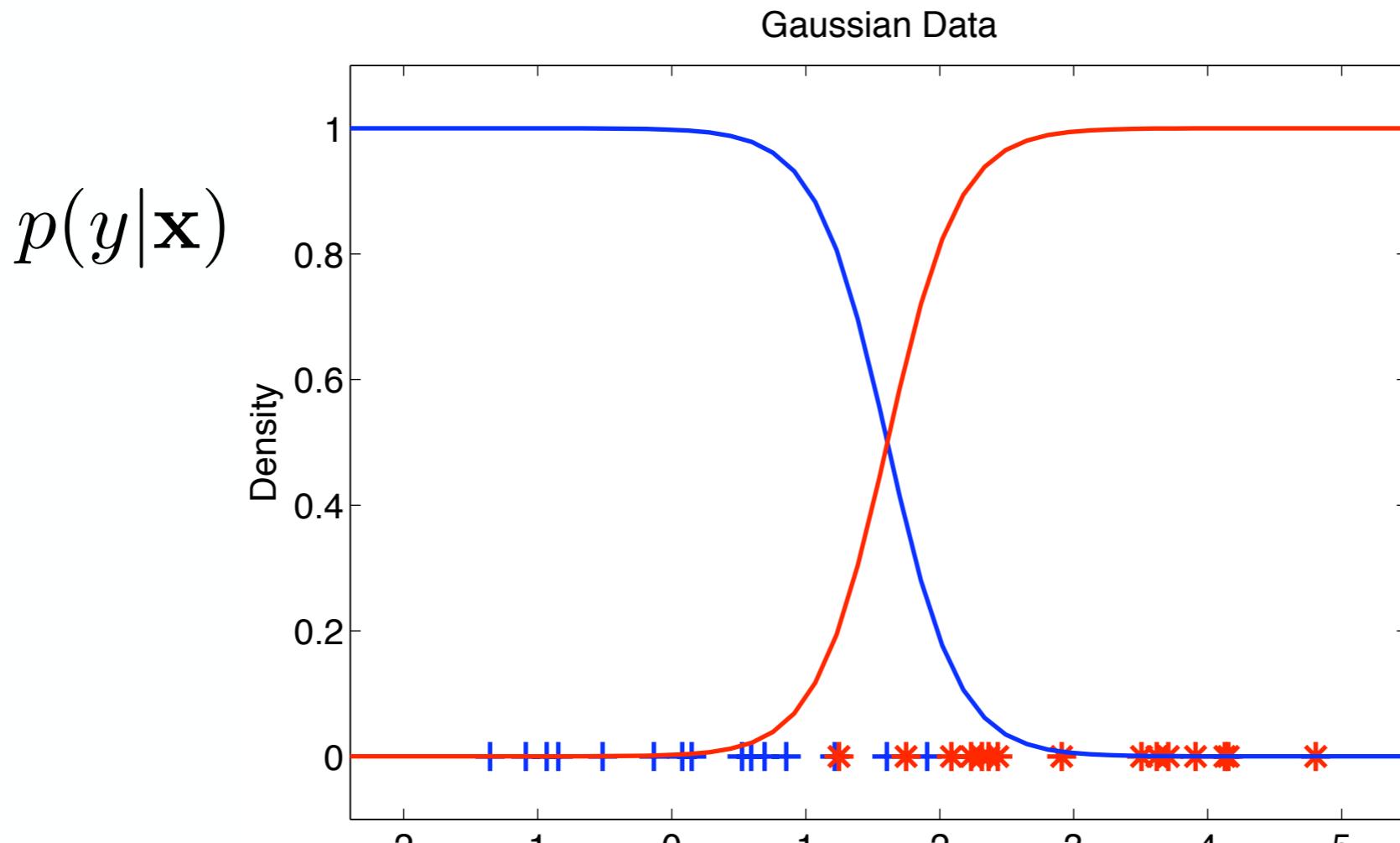
# Class posterior probability

- For each object we want to estimate  $p(y|\mathbf{x})$



# Class posterior probability

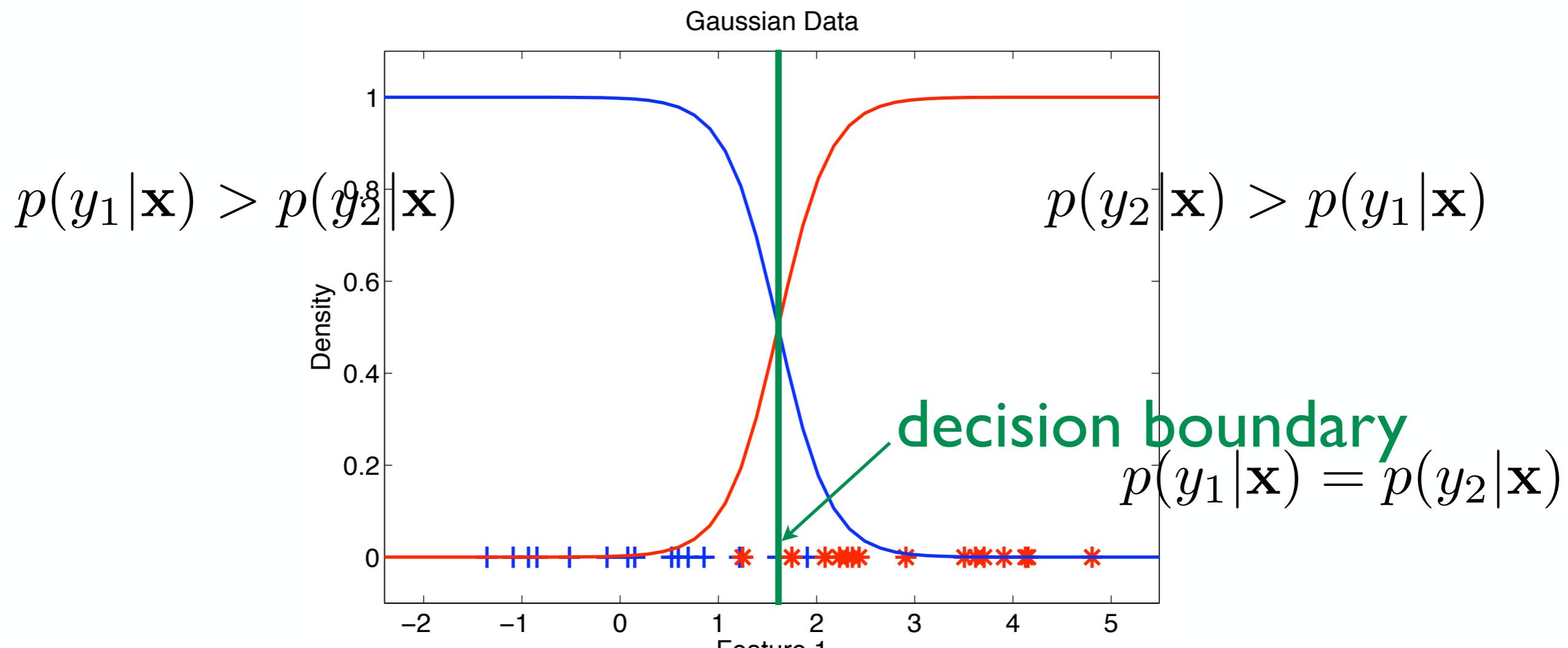
- For each object we have to estimate  $p(y|\mathbf{x})$



$$\sum_i p(y_i|\mathbf{x}) = 1$$

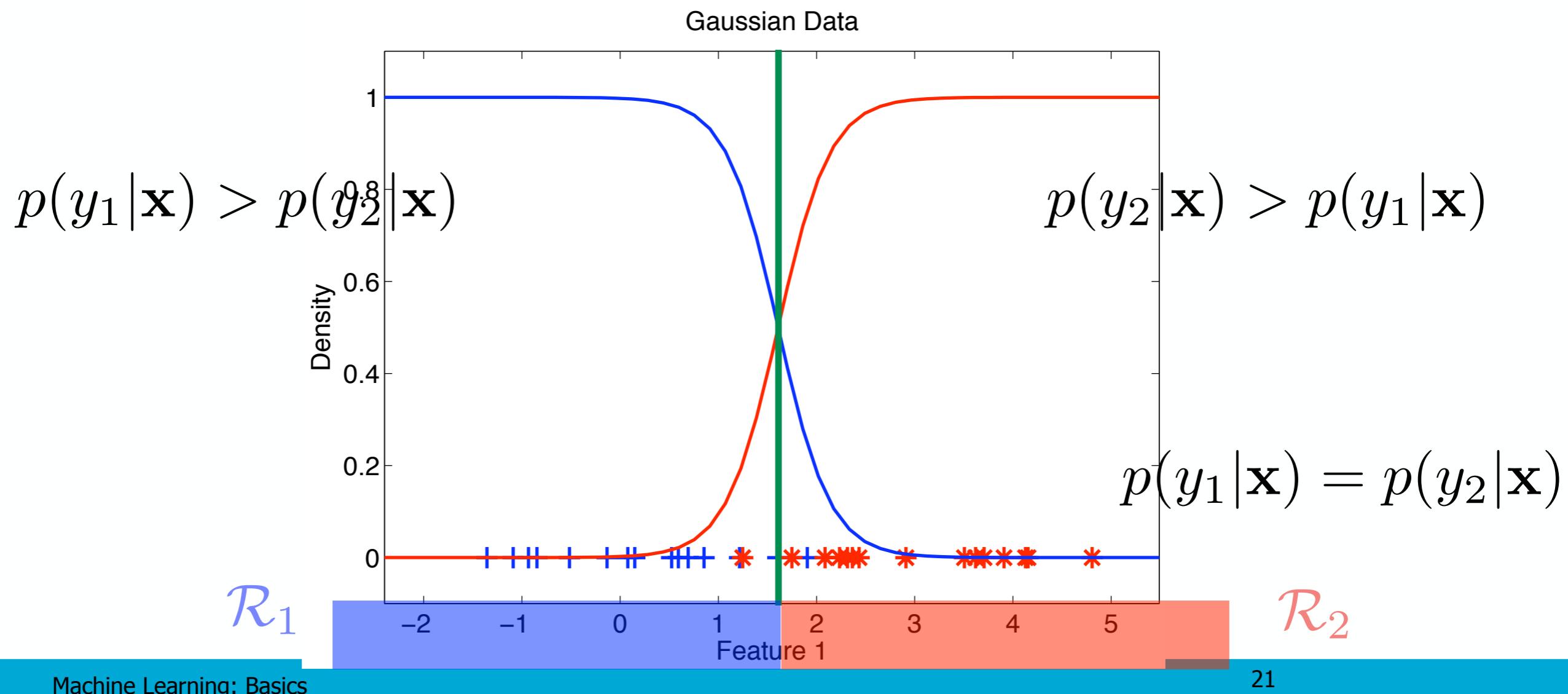
# Classify new objects

- Assign the label of the class with the largest posterior probability



# Classify new objects

- Assign the label of the class with the largest posterior probability



# Description of a classifier

There are several ways to describe a classifier:

- if  $p(y_1|\mathbf{x}) > p(y_2|\mathbf{x})$  then assign to  $y_1$   
otherwise  $y_2$
- if  $p(y_1|\mathbf{x}) - p(y_2|\mathbf{x}) > 0$  then assign to  $y_1$
- or  $\frac{p(y_1|\mathbf{x})}{p(y_2|\mathbf{x})} > 1$
- or  $\log(p(y_1|\mathbf{x})) - \log(p(y_2|\mathbf{x})) > 0$

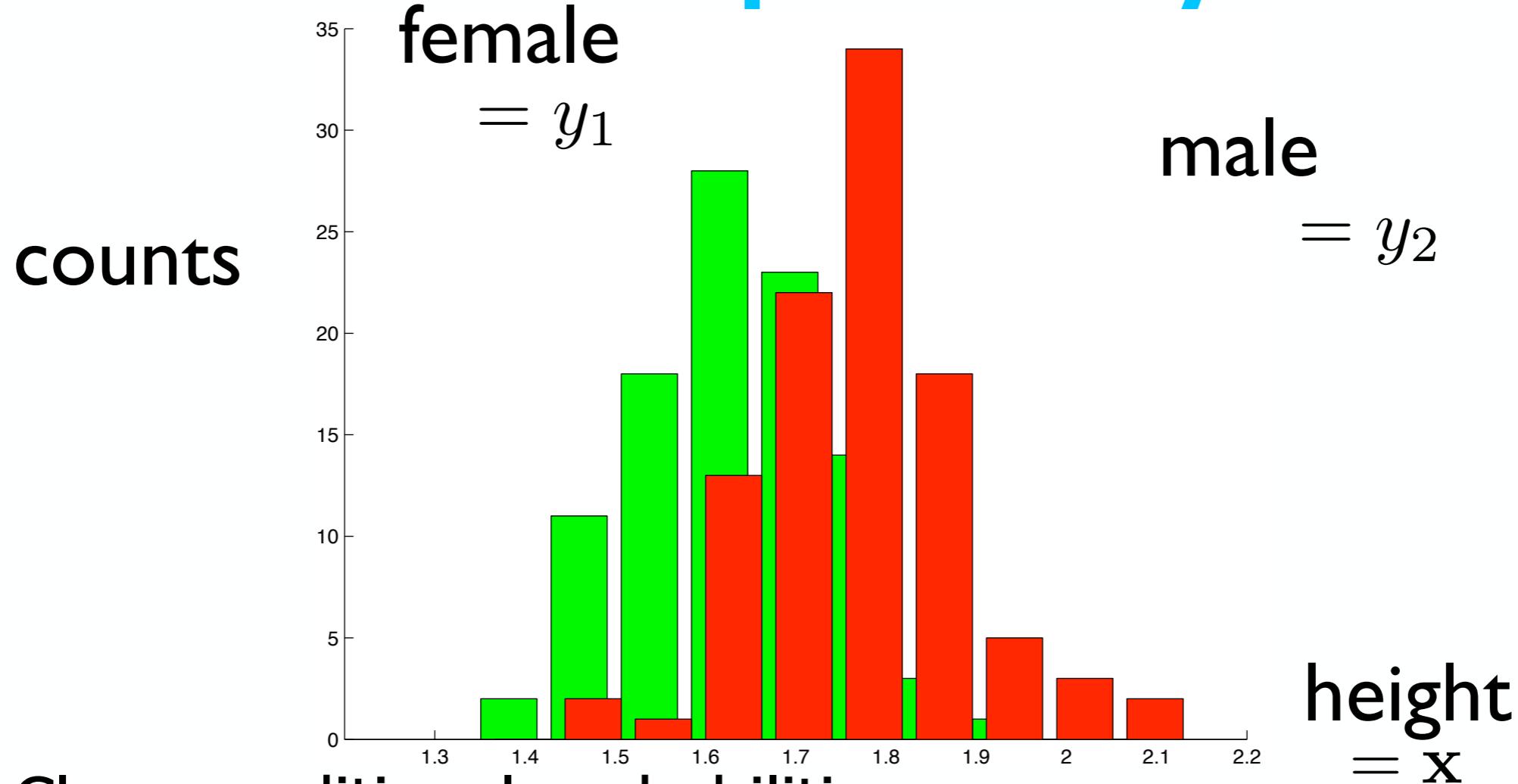
# Bayes' theorem

- In many cases the posterior is hard to estimate
- Often a functional form of the class distributions can be assumed
- Use Bayes' theorem to rewrite one into the other:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

class (conditional) distribution	$p(\mathbf{x} y)$
class prior	$p(y)$
(unconditional) data distribution	$p(\mathbf{x})$
posterior probability	$p(y \mathbf{x})$

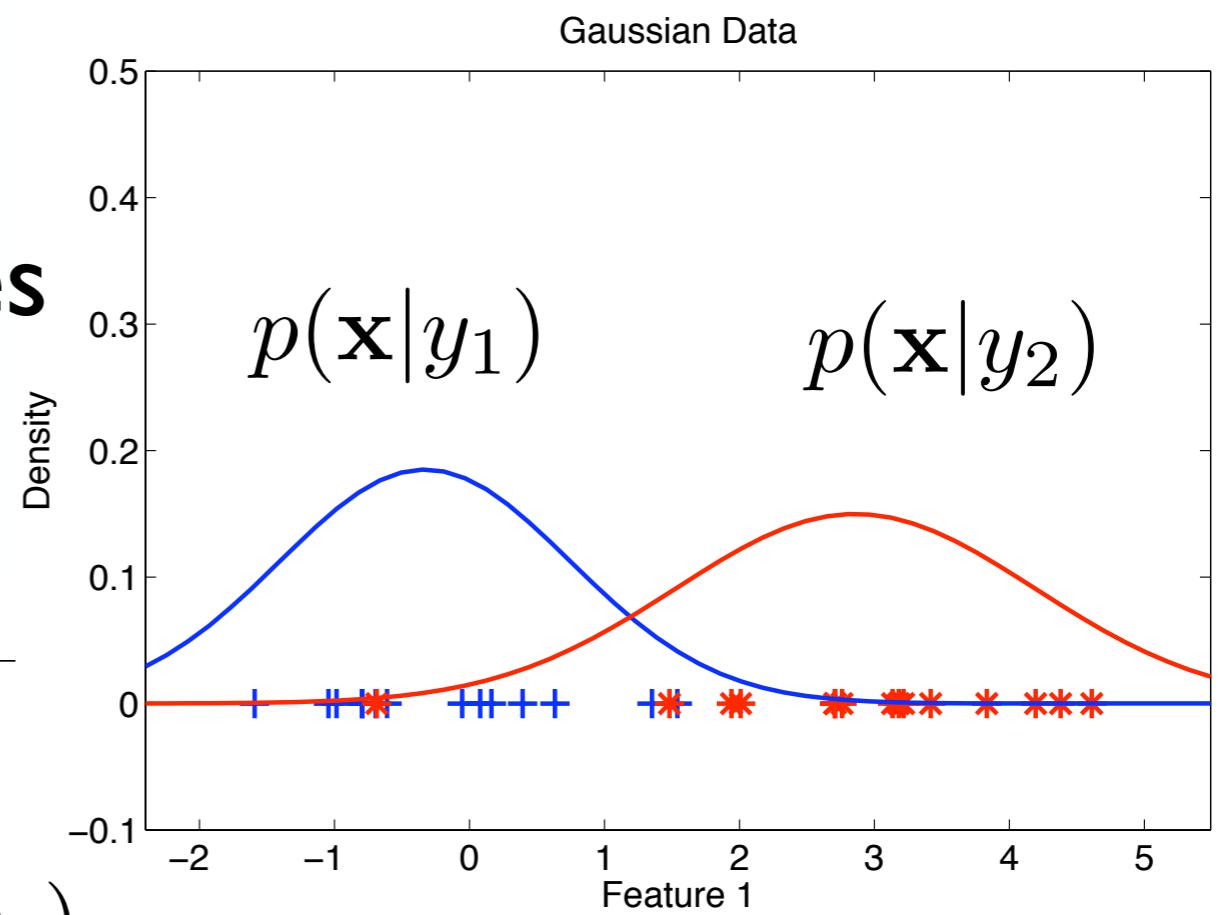
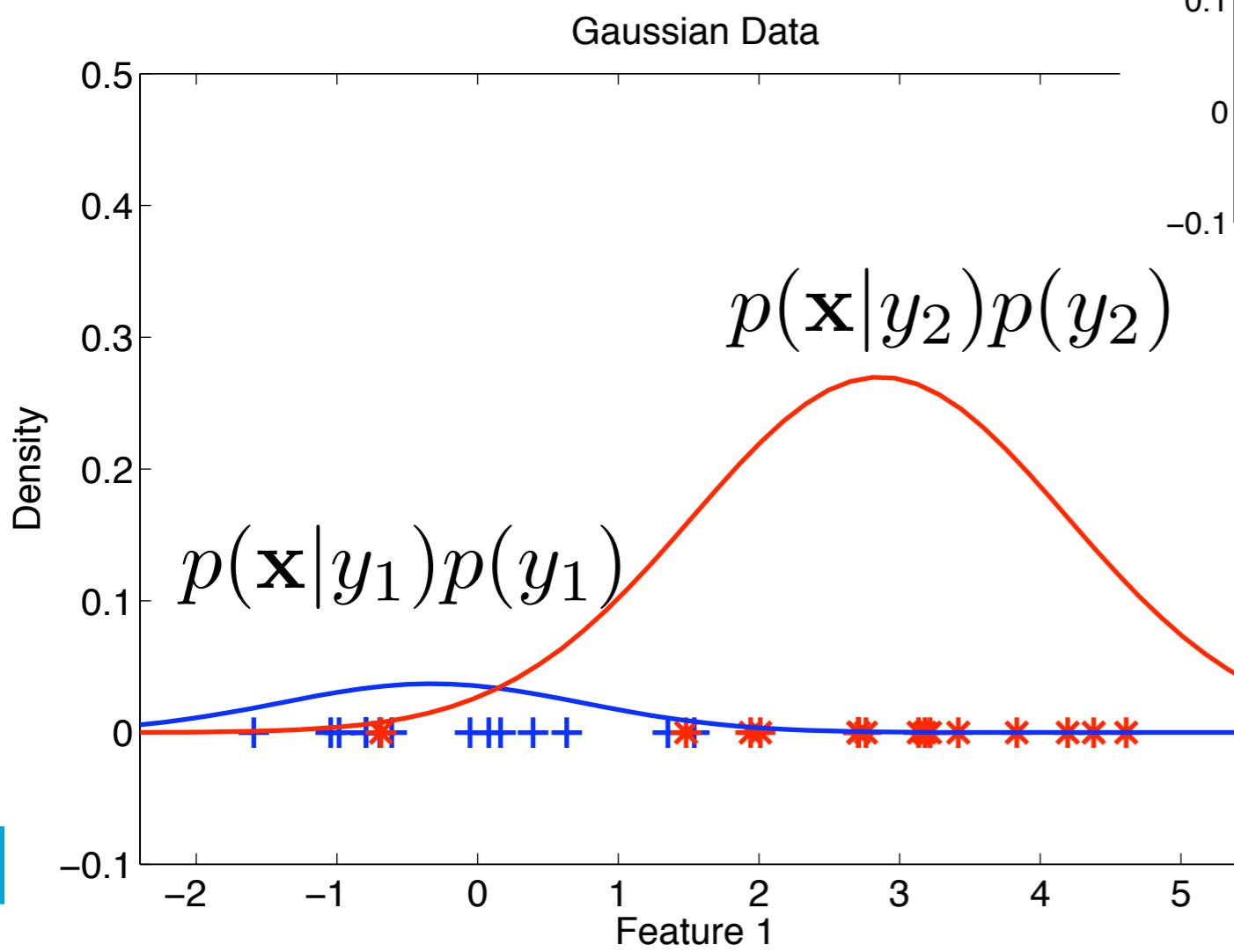
# Class conditional probability



- Class conditional probabilities:
  - The distribution of the females  $p(\mathbf{x}|y_1)$
  - The distribution of the males  $p(\mathbf{x}|y_2)$

# Bayes' rule

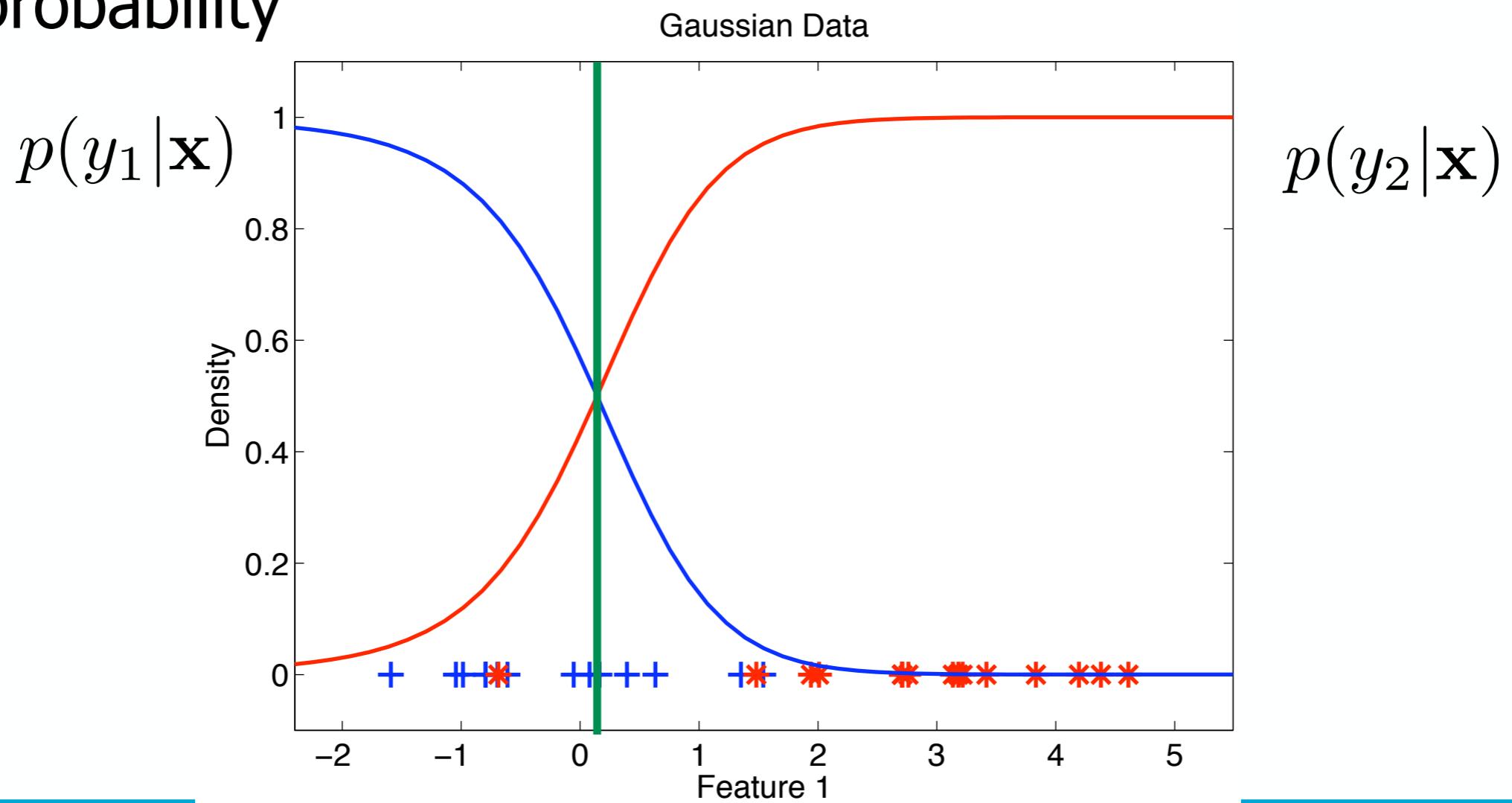
I. Estimate the class conditional probabilities



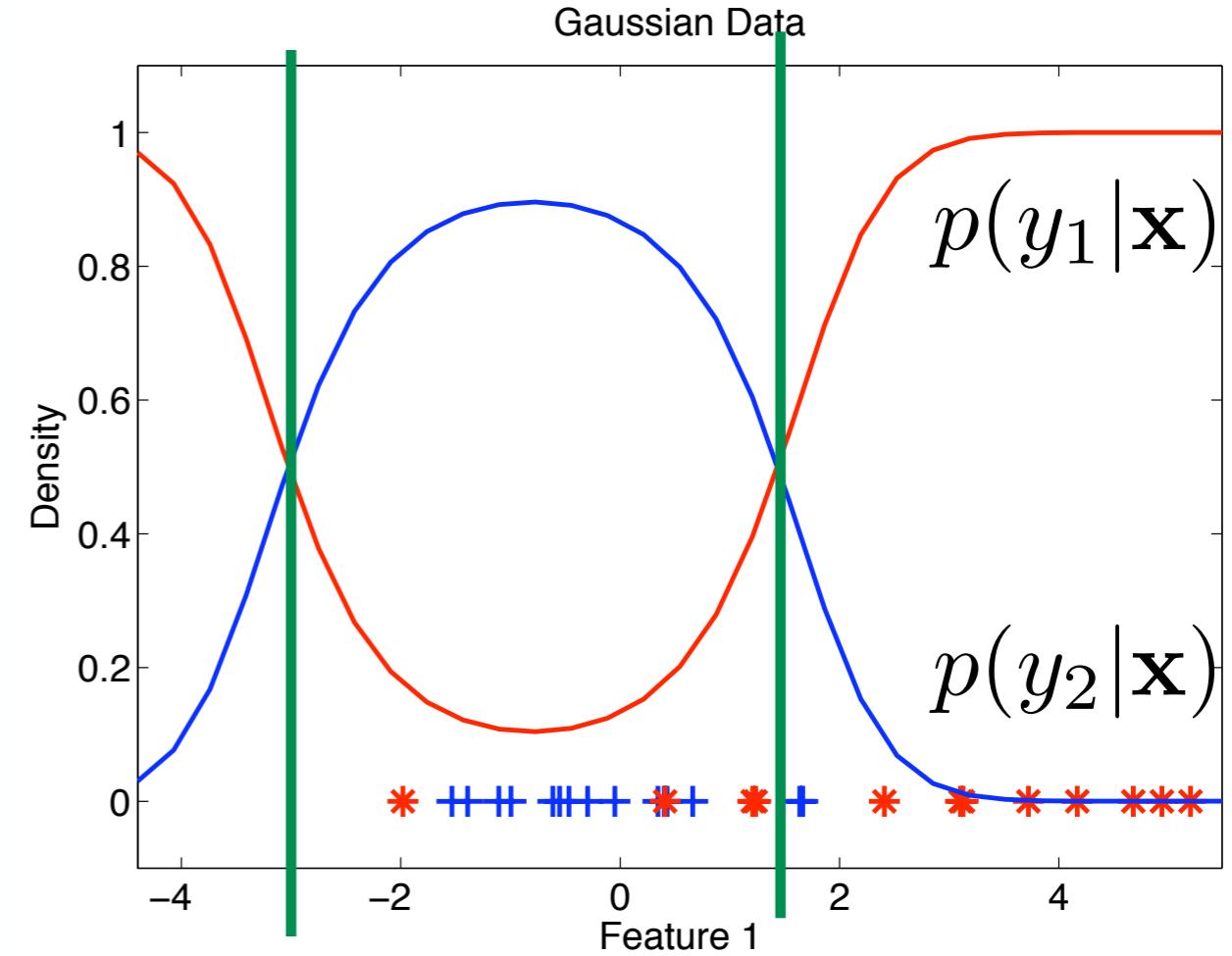
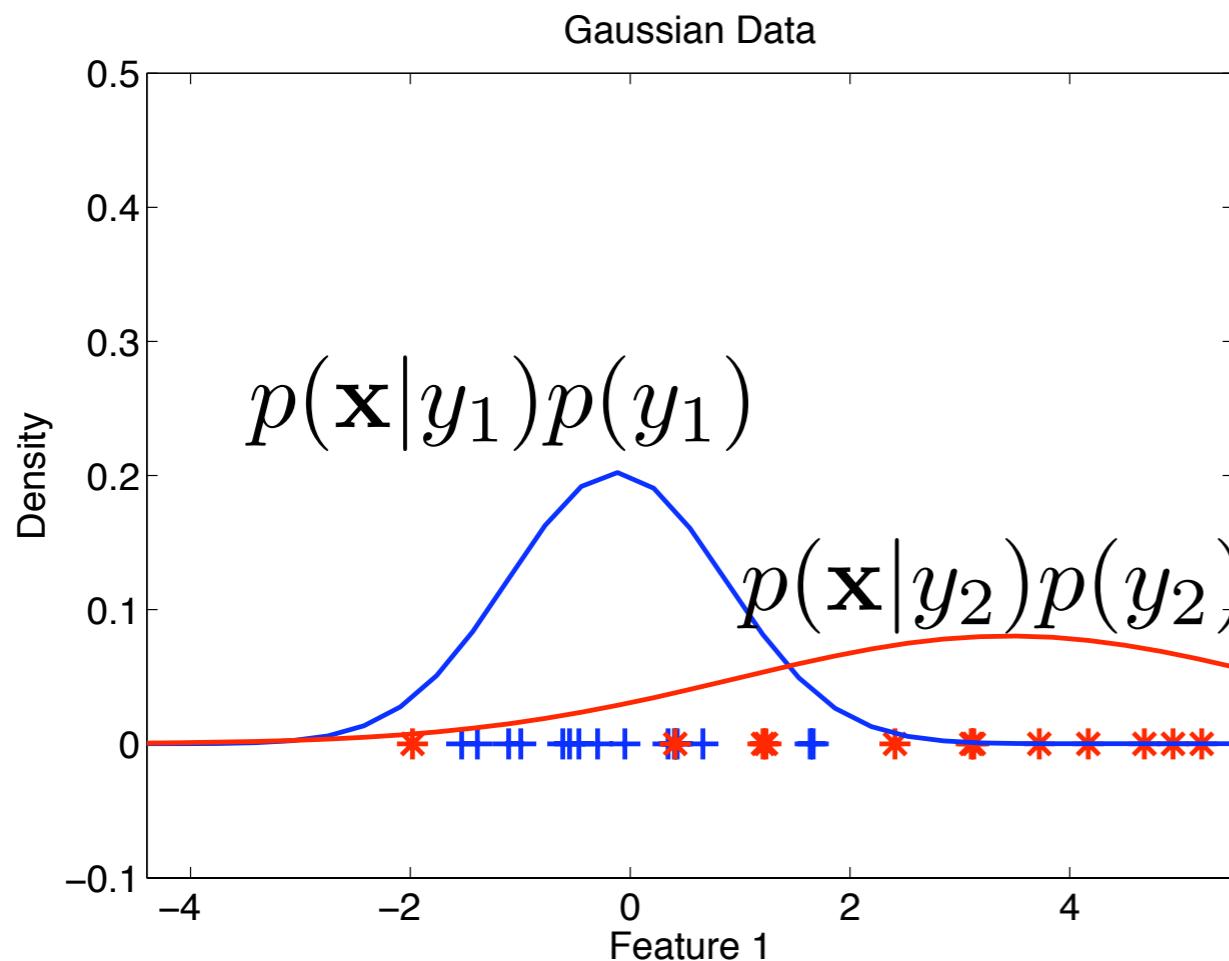
2. Multiply with the class priors

# Bayes' rule

3. Compute the class posterior probabilities
4. Assign objects to the class with the highest posterior probability

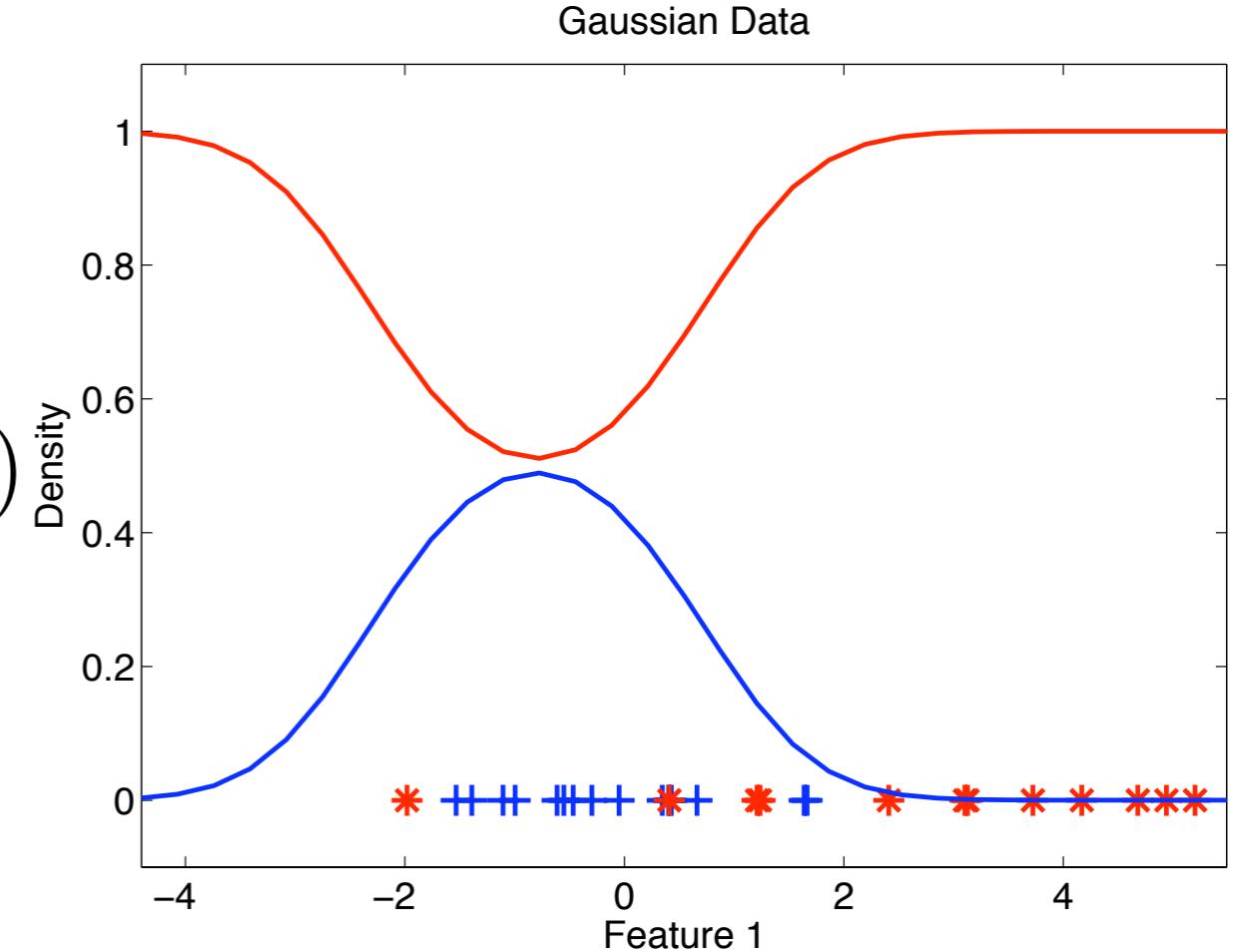
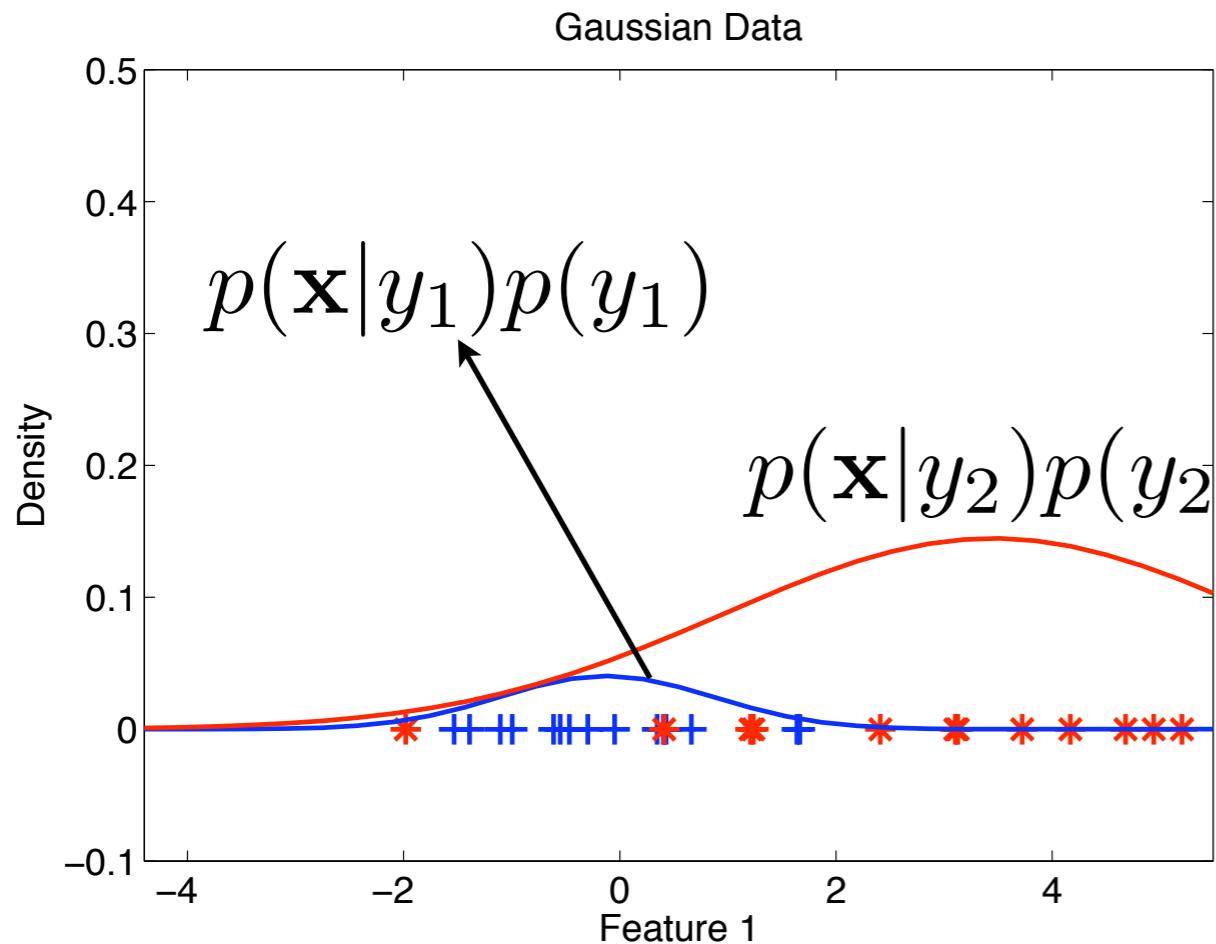


# Complicated decision boundary



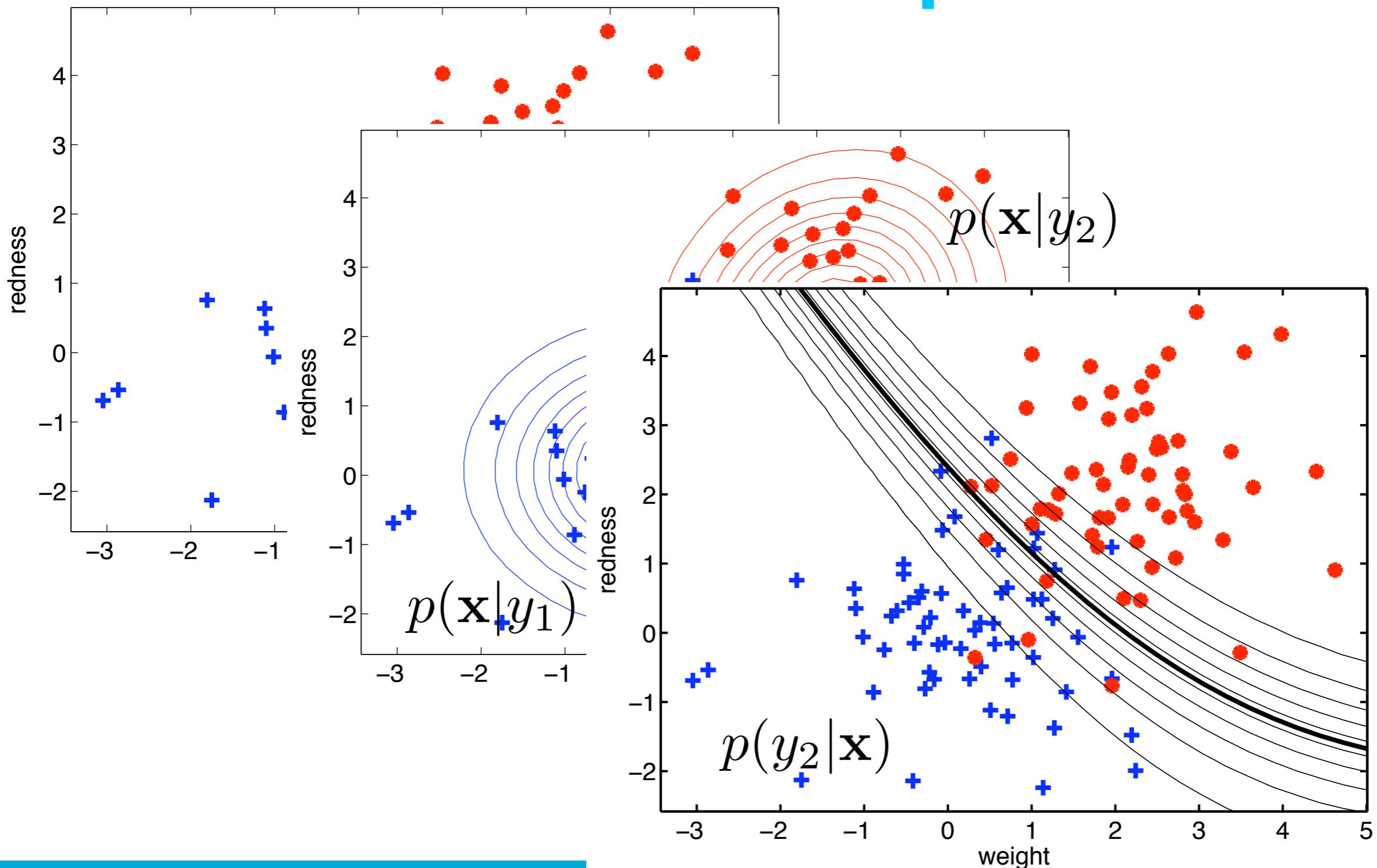
- Depending on the class-conditional probability densities, complicated decision boundaries can appear

# Missing decision boundary

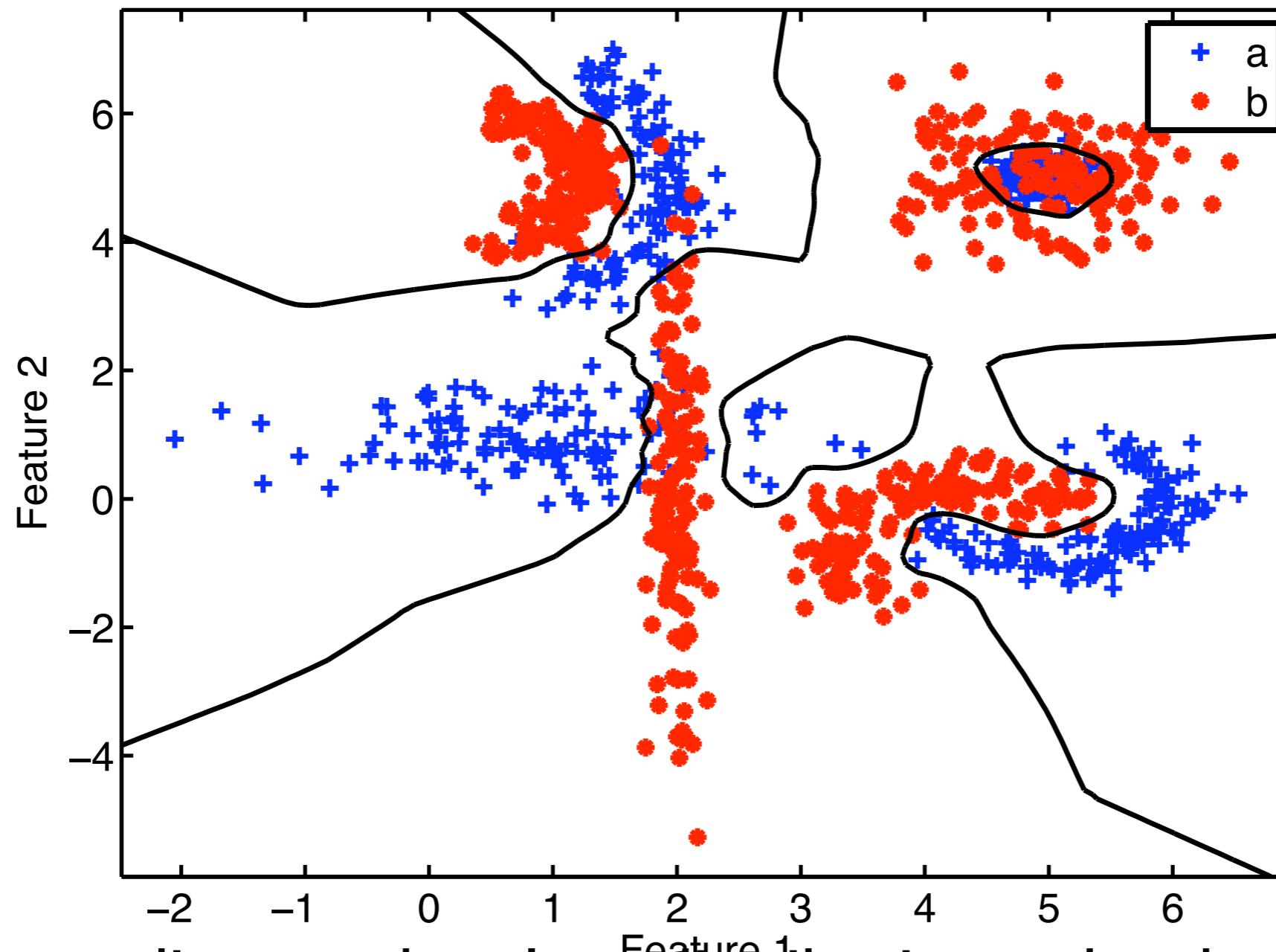


- A class can be too small (class prior is low) or too dispersed, that no objects are assigned to that class

# 2-dimensional feature space



# Multi-modal distributions



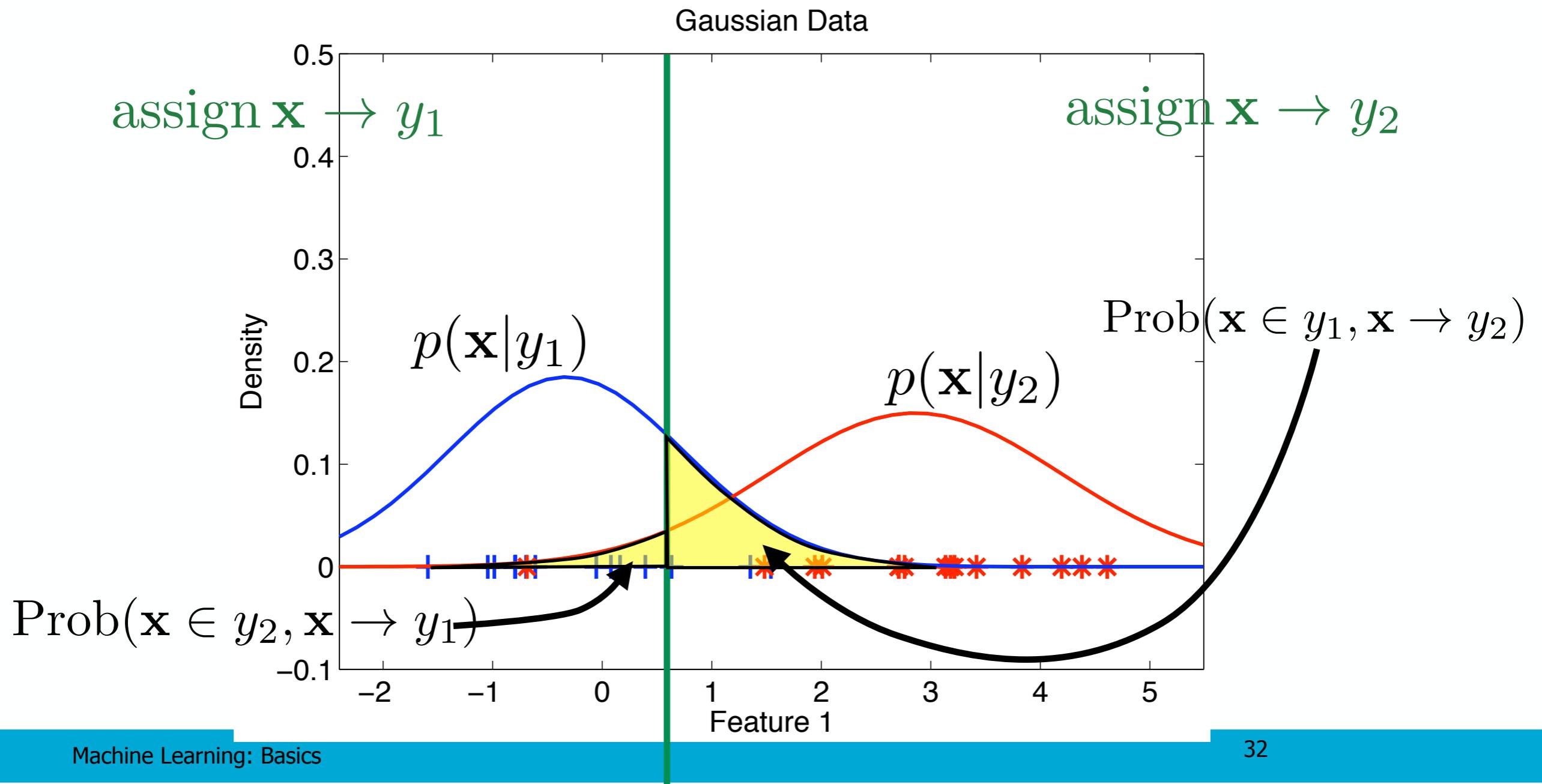
- Depending on the class distributions, the decision boundary can have arbitrary shapes

# The class conditional probabilities

- But, how do we obtain the class conditional probabilities  $p(\mathbf{x}|\omega_i)$ ?
- Typically, you need to assume a model
- Estimate the model parameters such that the example objects fit well:  
maximum likelihood estimators
- This will be the topic for the coming weeks
- Note: other approaches than Bayes' rule are possible.  
We discuss it later...

# How good is it?

- The error of the green decision boundary:



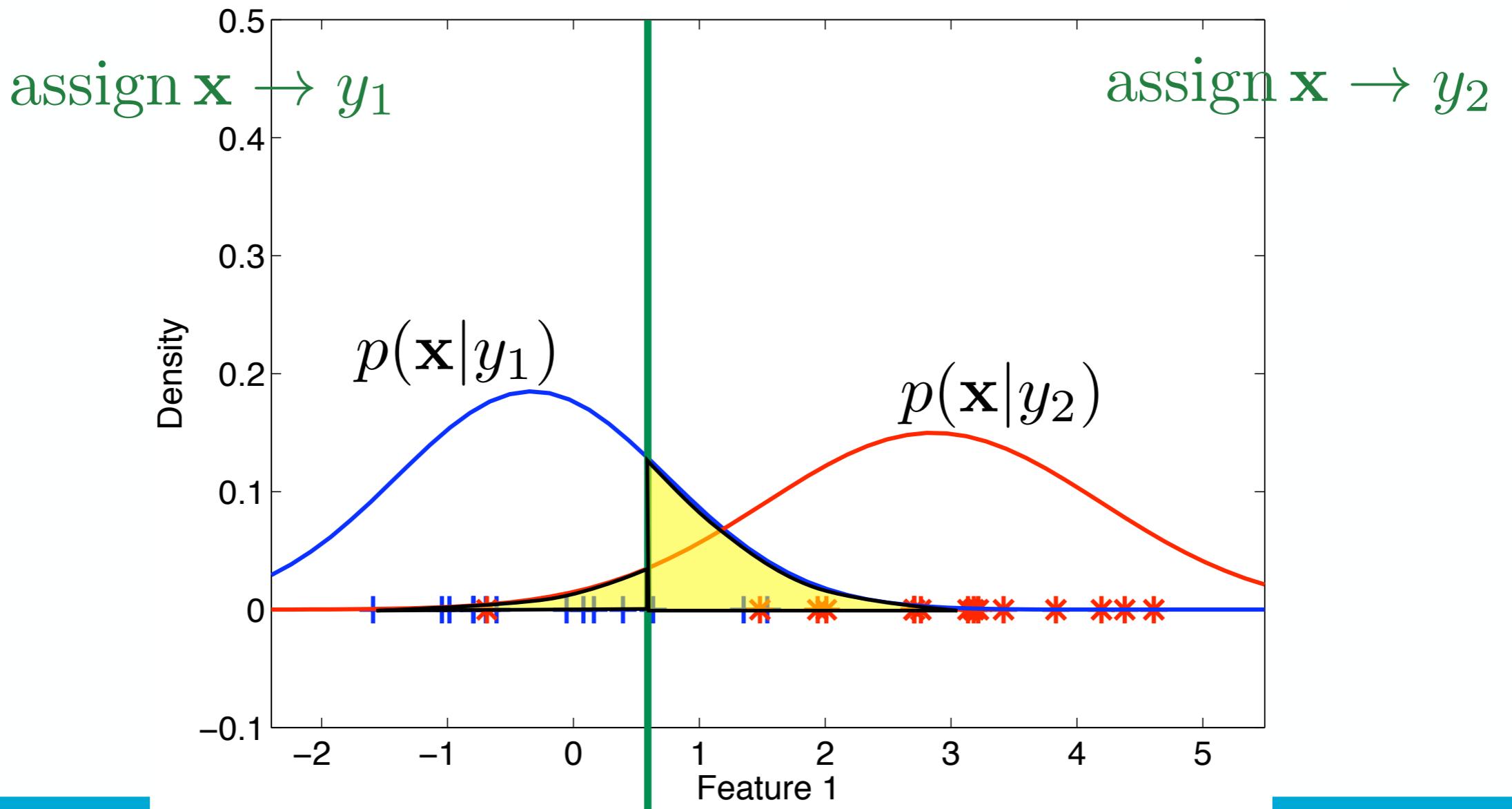
# Error of type I and II

- For a two-class classification problem, the following two errors are defined:
- Type I error:  $\varepsilon_1 = \int_{\mathcal{R}_2} p(\mathbf{x}|y_1)d\mathbf{x}$
- Type II error:  $\varepsilon_2 = \int_{\mathcal{R}_1} p(\mathbf{x}|y_2)d\mathbf{x}$
- When we call  $y_1$  the positive class, and  $y_2$  the negative class, then  $\varepsilon_1$  is the false negative fraction, and  $\varepsilon_2$  is the false positive fraction.

# Classification error

- The error:

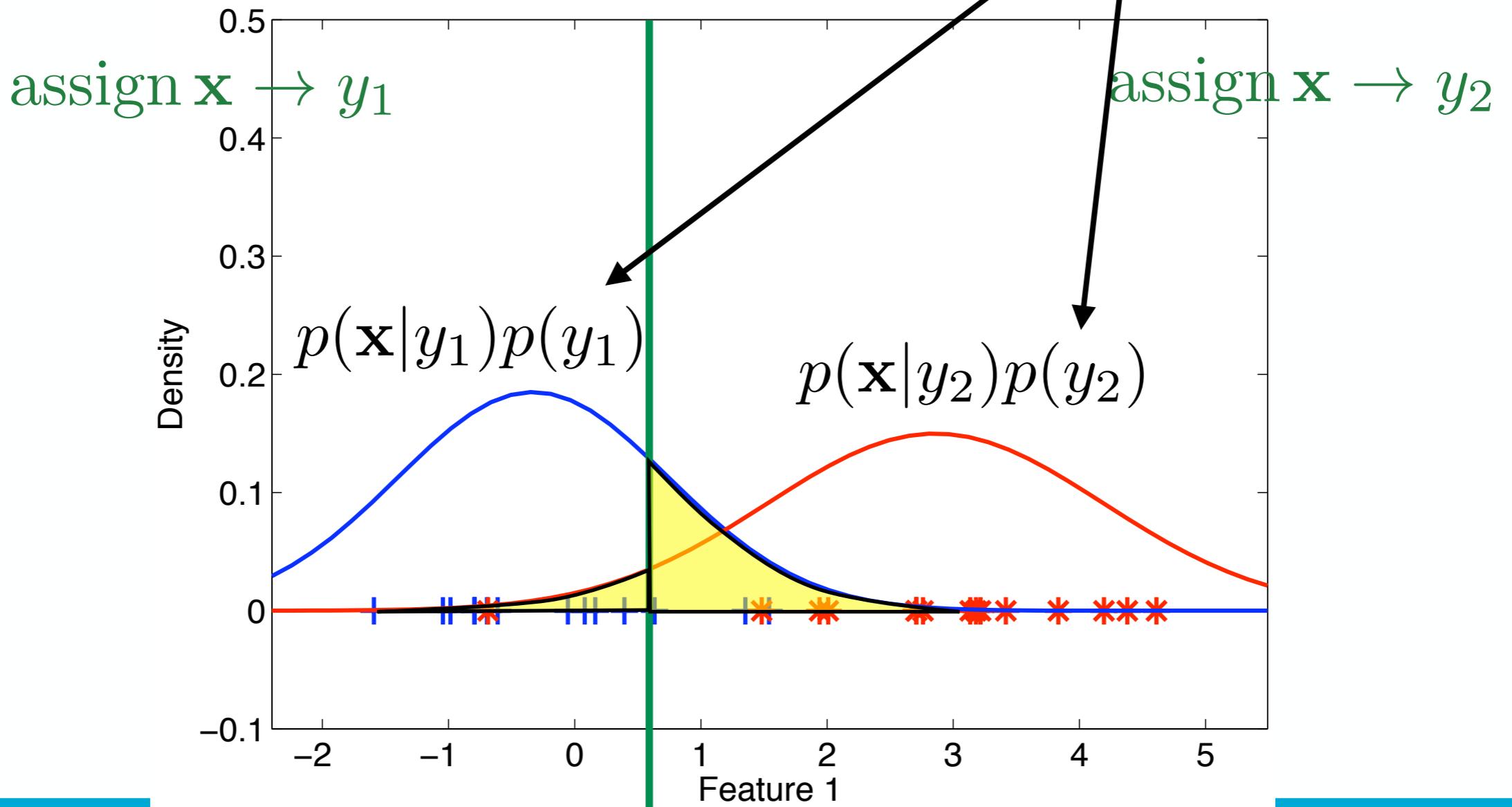
$$p(\text{error}) = \sum_{i=1}^C p(\text{error}|y_i)p(y_i)$$



# Classification error

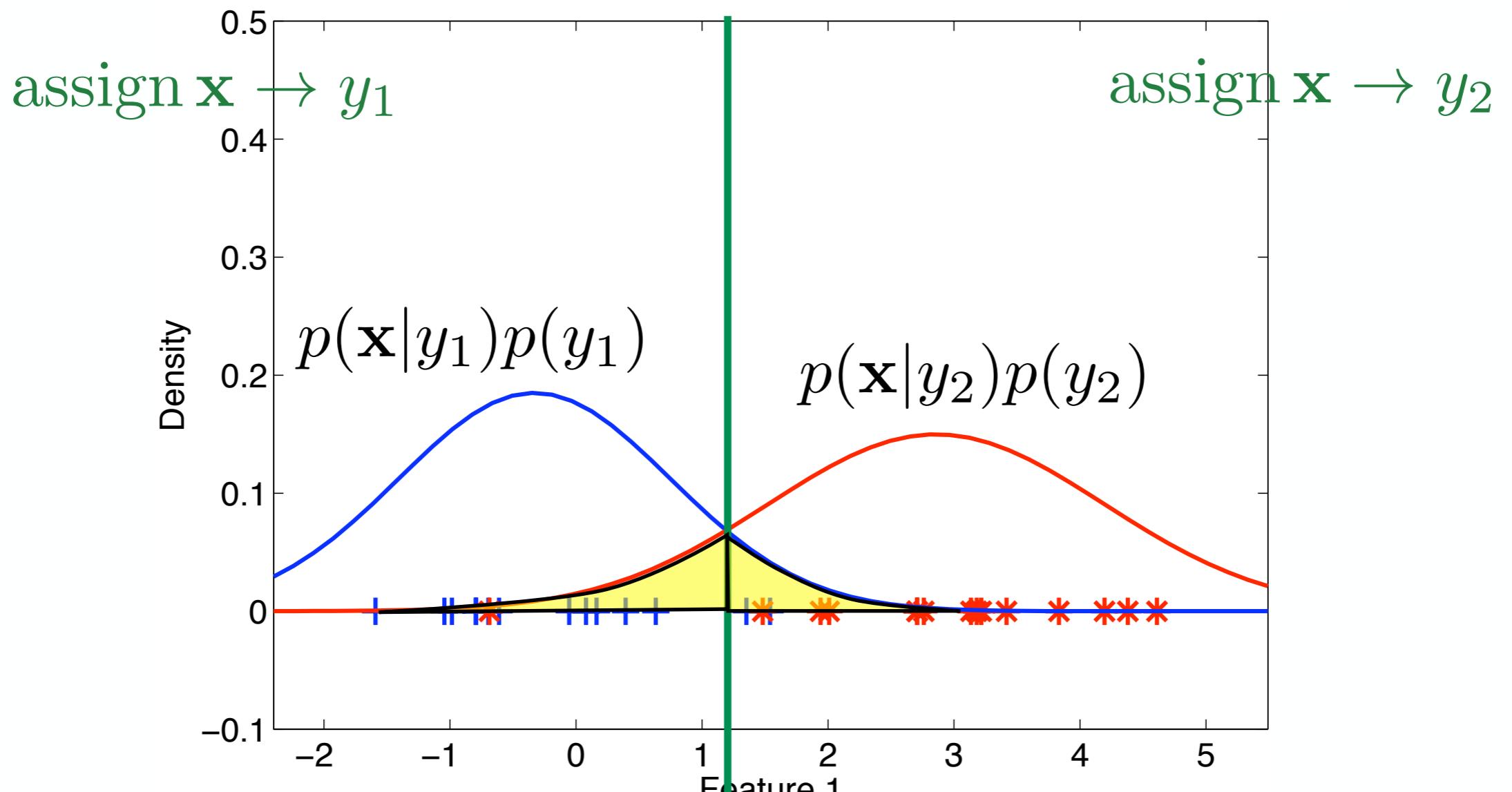
- The error:

$$p(\text{error}) = \sum_{i=1}^C p(\text{error}|y_i)p(y_i)$$



# Bayes error $\varepsilon^*$

- Bayes error is the minimum error: typically  $>0$  !!



# Bayes' Error

- Bayes' error is the **minimum** attainable error  $\varepsilon^*$
- In practice, we do not have the true distributions, and we can not obtain
- The Bayes' error does not depend on the classification rule that you apply, but on the distribution of the data
- In general you can not compute the Bayes' error:
  - you don't know the true class conditional probabilities
  - the (high) dimensional integrals are very complicated

# Misclassification Costs

- Sometimes: misclassification of class A to class B is much more dangerous than misclassification of class B to class A

misclassification:  
classify ‘healthy’ to ‘ill’



misclassification:  
classify ‘ill’ to ‘healthy’



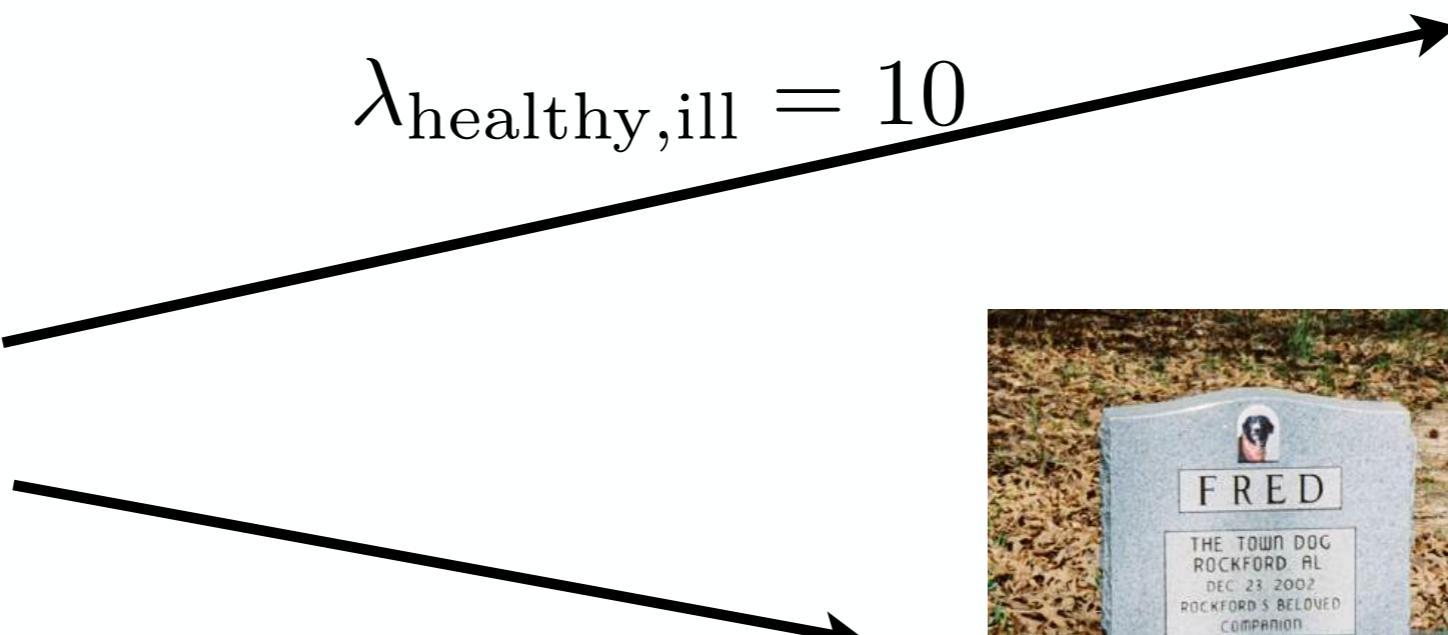
# Misclassification cost

- Introduce a loss that measures the cost of assigning an object that came from class  $y_j$  to class  $y_i$  :

$$\lambda_{ji}$$



$$\lambda_{\text{healthy}, \text{ill}} = 10$$



$$\lambda_{\text{ill}, \text{healthy}} = 100$$



# Misclassification cost of some dataset

- Assume I have a labeled dataset  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$
- Further assume that these objects are classified by a classifier, and the estimated class labels are  $\hat{y}_i$
- Then the total empirical risk on this dataset is

$$R = \frac{1}{N} \sum_{i=1}^N \lambda_{y_i, \hat{y}_i}$$

# Conditional risk, total risk

- The conditional risk of assigning object  $\mathbf{x}$  to class  $y_i$ :

$$l^i(\mathbf{x}) = \sum_{j=1}^C \lambda_{ji} p(y_j | \mathbf{x})$$

- The average risk over a region:

$$\begin{aligned} r^i &= \int_{\mathcal{R}_i} l^i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &= \int_{\mathcal{R}_i} \sum_{j=1}^C \lambda_{ji} p(y_j | \mathbf{x}) p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

- Overall risk:

$$r = \sum_{i=1}^C r^i = \sum_{i=1}^C \int_{\mathcal{R}_i} \sum_{j=1}^C \lambda_{ji} p(y_j | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

# Minimum total risk

- We minimize the risk when we define the regions  $\mathcal{R}_i$  are chosen such that each of the integrals are as small as possible:

$$r = \sum_{i=1}^C r^i = \sum_{i=1}^C \int_{\mathcal{R}_i} \sum_{j=1}^C \lambda_{ji} p(y_j | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

- So make  $\mathbf{x}$  part of  $\mathcal{R}_i$  if:

$$\sum_{j=1}^C \lambda_{ji} p(y_j | \mathbf{x}) \leq \sum_{j=1}^C \lambda_{jk} p(y_j | \mathbf{x}) \quad k = 1, \dots, C$$

# Minimum total risk: two classes

- When you predict class  $y_i$  for an object of class  $y_i$  you would typically define:

$$\lambda_{y_i, y_i} = 0$$

- For a two-class problem, you therefore have to compare:

$$\sum_{j=1}^2 \lambda_{j1} p(y_j | \mathbf{x}) = \lambda_{11} p(y_1 | \mathbf{x}) + \lambda_{21} p(y_2 | \mathbf{x}) = 0 + \lambda_{21} p(y_2 | \mathbf{x}) \\ > ?$$

$$\sum_{j=1}^2 \lambda_{j2} p(y_j | \mathbf{x}) = \lambda_{12} p(y_1 | \mathbf{x}) + \lambda_{22} p(y_2 | \mathbf{x}) = \lambda_{12} p(y_1 | \mathbf{x}) + 0$$

# Minimum total risk: two classes

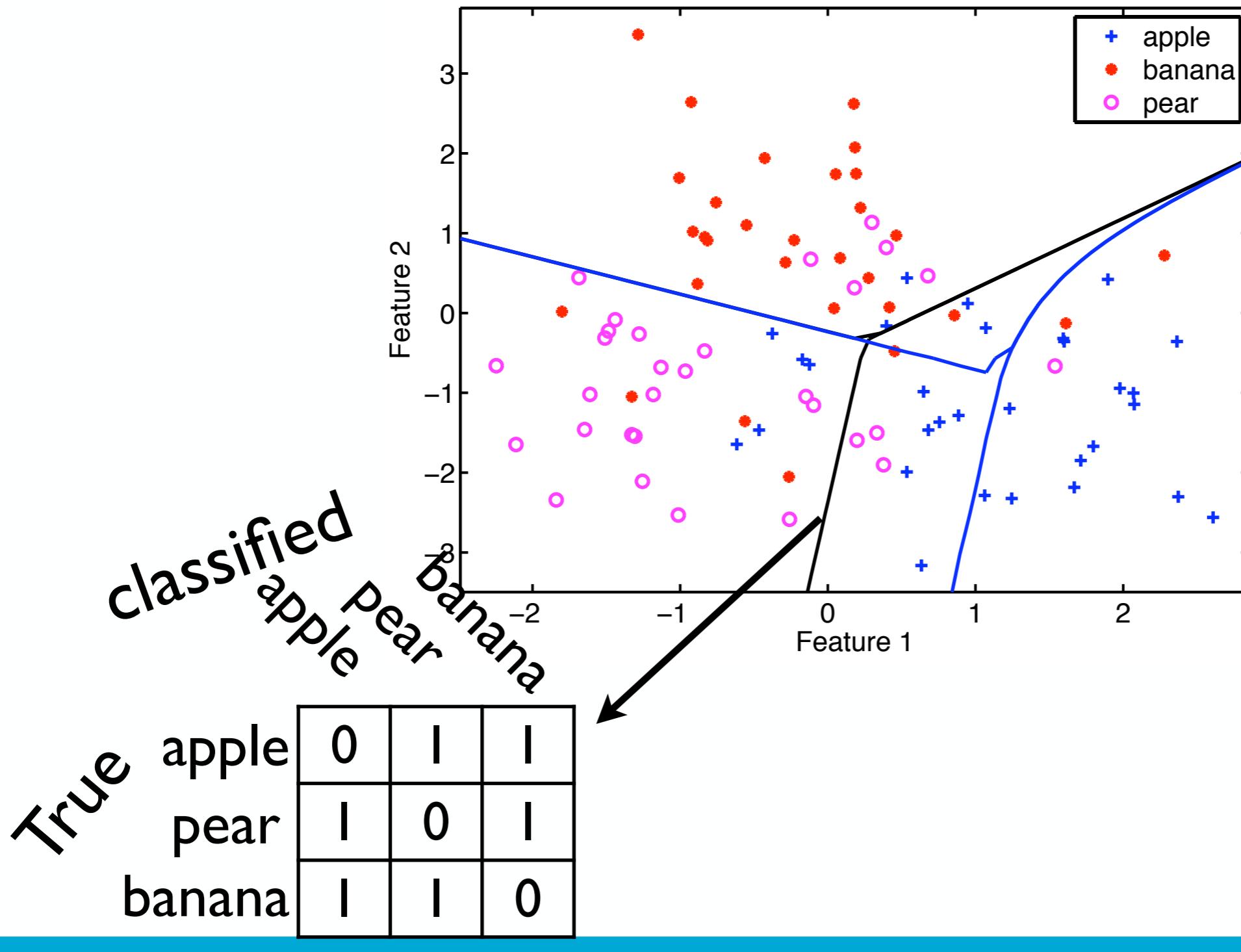
- When you predict class  $y_i$  for an object of class  $y_i$  you would typically define:

$$\lambda_{y_i, y_i} = 0$$

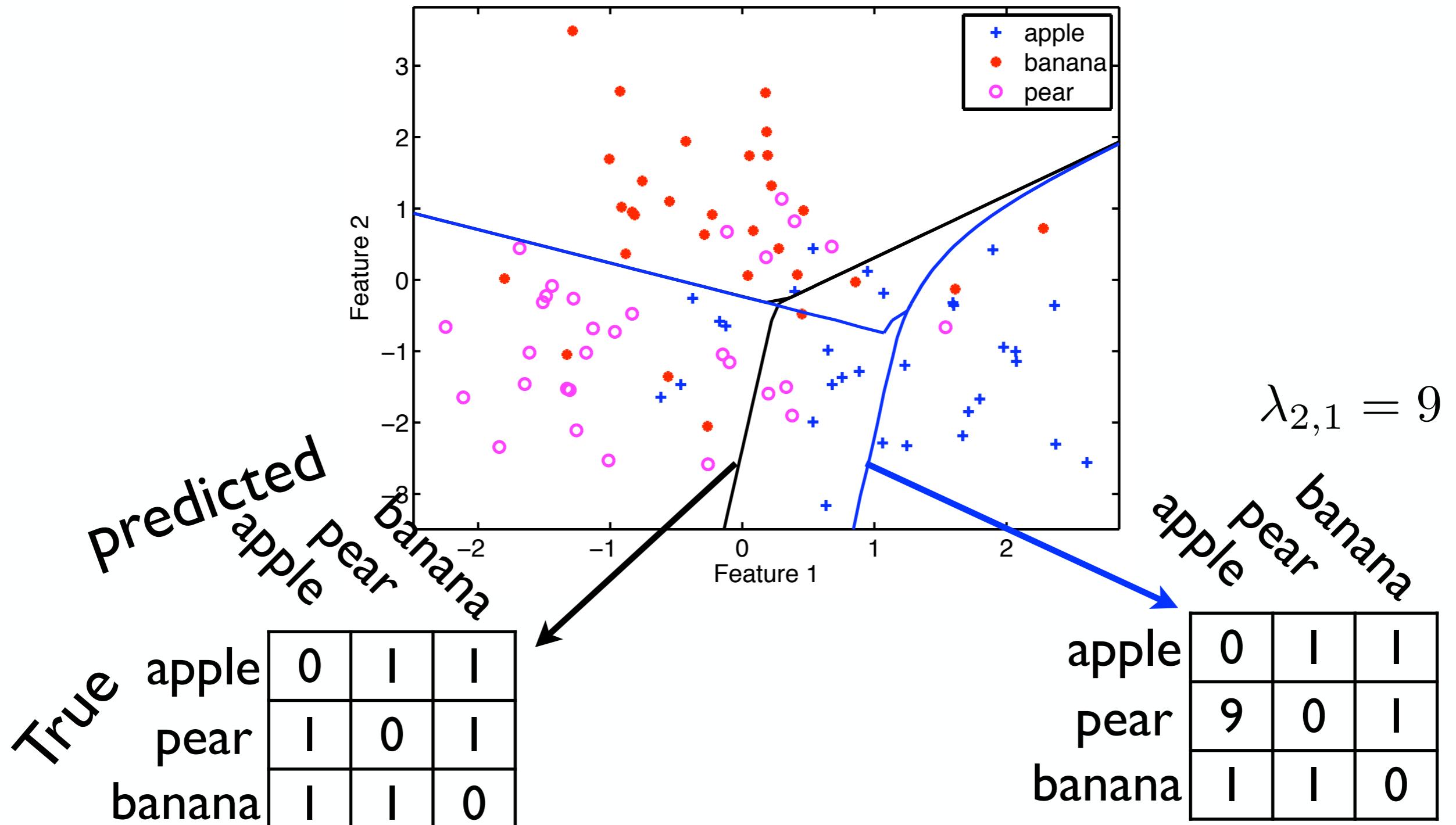
- For a two-class problem, you therefore have to compare:

$$\lambda_{21} p(y_2 | \mathbf{x}) \stackrel{?}{>} \lambda_{12} p(y_1 | \mathbf{x})$$

# Example cost



# Example cost

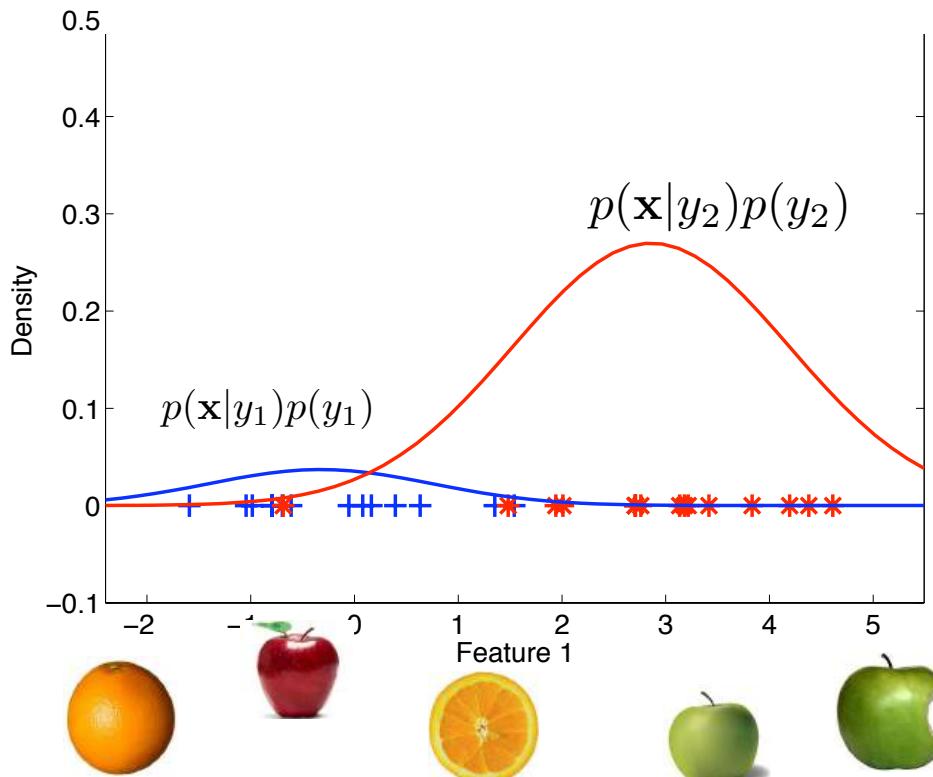


# What did we do?

- Objects, features, measurements,
  - ... datasets and feature space
- Traditional pattern recognition: classification
- Class posterior probabilities and Bayes' rule
- Bayes' classifier and Bayes' error
- Misclassification costs
- Next lecture: how do we get these class-conditional probabilities?

# Parametric Density-based Classifiers

## CSE2510 Machine Learning

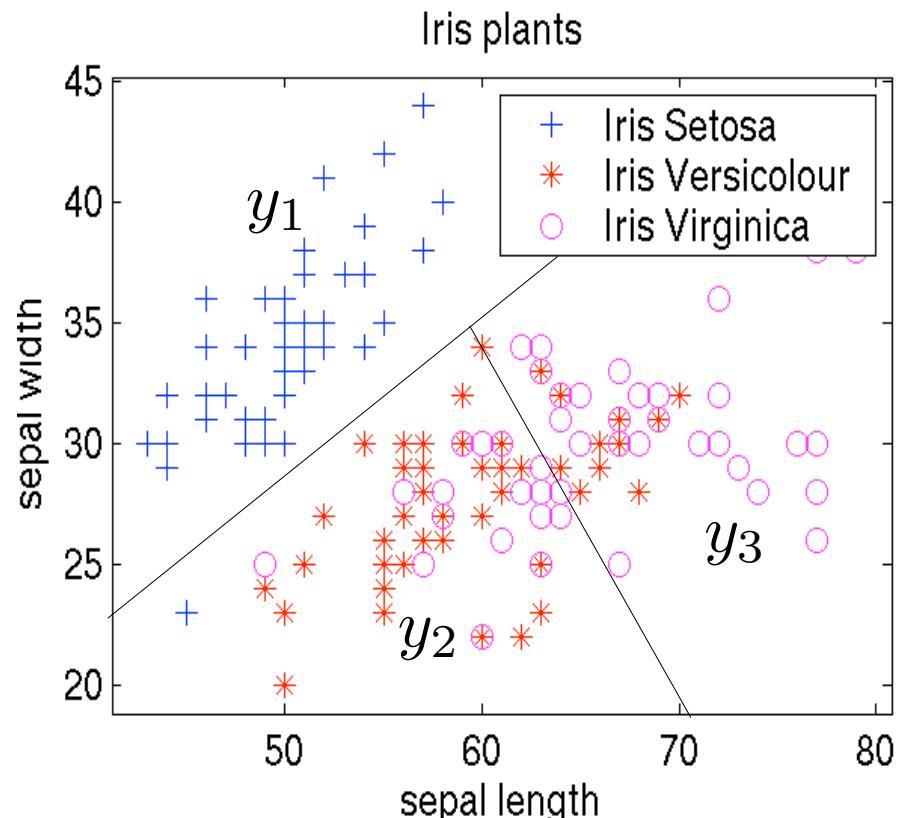


# Contents this lecture:

- Recap of last time:  
objects, feature vectors, feature space, decision boundary, class conditional probability, posterior probability, Bayes classifier, Bayes rule
- Classifiers based on probability estimates
- The curse of dimensionality
- Multivariate Gaussian distribution
- **Next lecture:** Classifiers based on Gaussian density estimates (Quadratic, linear, nearest mean)
- Scaling of features

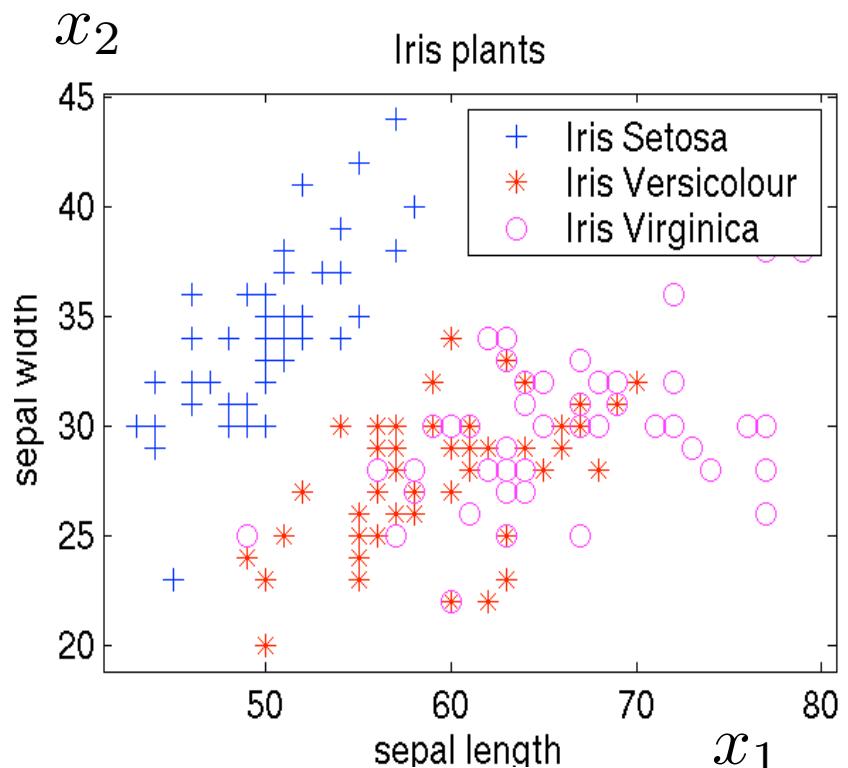
# Classification

- Given labeled data:  $\mathbf{X}$
- Assign to each object a class label  $y$
- In effect splits the feature space in separate regions



# Objects in feature space

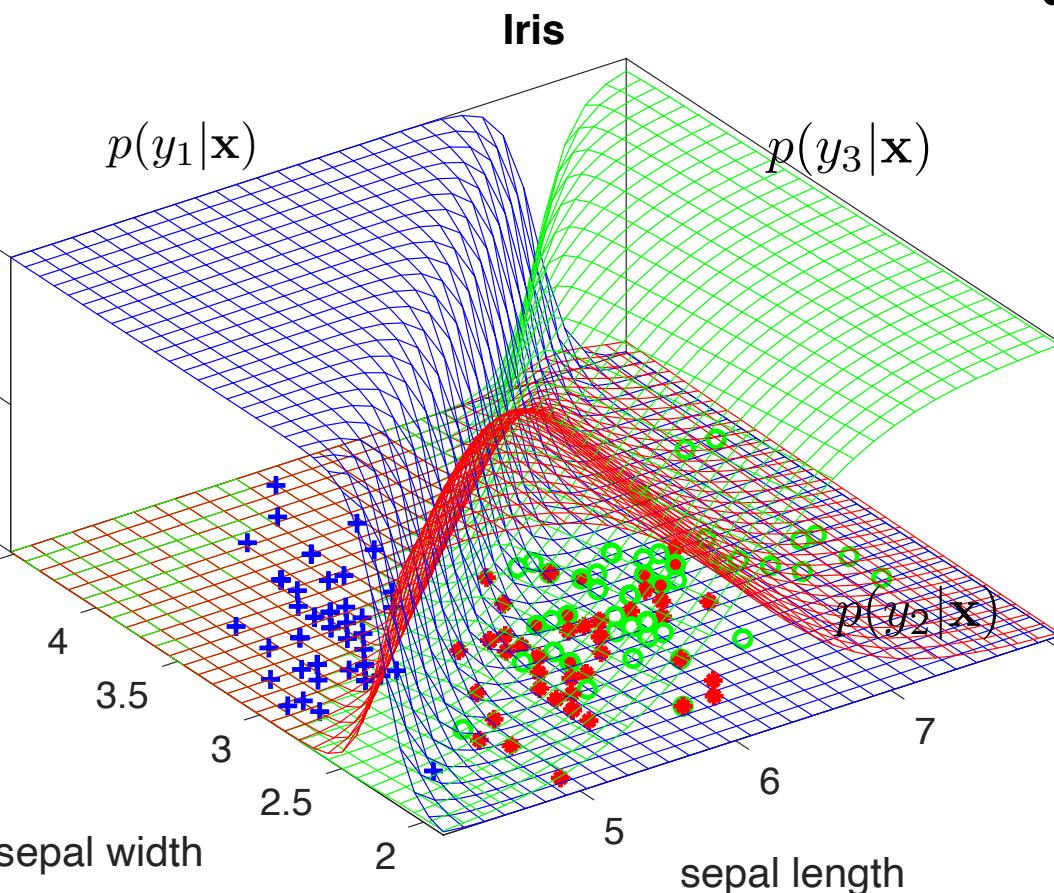
- We can interpret the measurements as a vector in a vector space:  
$$\mathbf{x} = [x_1, x_2, \dots, x_M]^T$$
- This originates, in principle, from a probability density over the whole feature space  
$$p(\mathbf{x}, y)$$



# Classification

$$p(y_1 | \mathbf{x}) > p(y_2 | \mathbf{x}) ?$$

# Output of the model



- For each object in the feature space, we should find:  
$$p(y|\mathbf{x})$$

In practice, we approximate:  
$$\hat{p}(y|\mathbf{x})$$

or we fit a function:

$$f(\mathbf{x})$$

# Bayes' theorem

- In many cases the posterior is hard to estimate
- Often a functional form of the class distributions can be assumed
- Use Bayes' theorem to rewrite one into the other:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

class (conditional) distribution	$p(\mathbf{x} y)$
class prior	$p(y)$
(unconditional) data distribution	$p(\mathbf{x})$
posterior probability	$p(y \mathbf{x})$

# A note on notation

- Mathematically speaking, there is a difference between

$$p(\mathbf{x})$$



probability density  
continuous variable

$$P(\mathbf{x})$$



probability mass  
discrete variable

# A note on notation

- Mathematically speaking, there is a difference between

$$p(\mathbf{x}) \quad \text{and} \quad P(\mathbf{x})$$

↑  
probability density  
continuous variable

$$P(\mathbf{x})$$

↑  
probability mass  
discrete variable

- (Actually, hard-core mathematicians may use

$$f_X(\mathbf{x}) \quad \text{and} \quad P_X(\mathbf{x})$$

# A note on notation

- Mathematically speaking, there is a difference between

$$p(\mathbf{x}) \quad \text{and} \quad P(\mathbf{x})$$



probability density  
continuous variable



probability mass  
discrete variable

- Machine Learners are a bit more sloppy:  
always use  $p(\mathbf{x})$

# Another note on notation

- We are working with feature **vectors**
- Column or row, that is the question
- In mathematics, typically a column:
- In data matrix, one object in a row:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_M \end{bmatrix}$$

$$X = \begin{bmatrix} -1 & -1 \\ -1 & +1 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}$$

(Two-dimensional dataset  
with four objects)

# Data distribution

- In Bayes rule you see  $p(\mathbf{x})$ . How to get it?
- You can explicitly compute it:

$$p(\mathbf{x}) = p(\mathbf{x}|y_1)p(y_1) + p(\mathbf{x}|y_2)p(y_2)$$

- But if you just find the largest posterior:

$$\frac{p(\mathbf{x}|y_1)p(y_1)}{p(\mathbf{x})} > \frac{p(\mathbf{x}|y_2)p(y_2)}{p(\mathbf{x})}$$

# Description of a classifier

There are several ways to describe a classifier:

- if  $p(y_1|\mathbf{x}) > p(y_2|\mathbf{x})$  then assign to  $y_1$   
otherwise  $y_2$
- if  $p(y_1|\mathbf{x}) - p(y_2|\mathbf{x}) > 0$  then assign to  $y_1$
- or  $\frac{p(y_1|\mathbf{x})}{p(y_2|\mathbf{x})} > 1$
- or  $\log(p(y_1|\mathbf{x})) - \log(p(y_2|\mathbf{x})) > 0$

# True and estimated probabilities

- Up to now:
  - Assume that we know the true  $p(y|x)$   
or  $p(x|y)$ ,  $p(y)$
- 
- In practice:
  - We only get a sample (training set), drawn from this distribution

# Models is What We Need!

- We don't have the true distributions, only a sampled dataset from it
- We have to approximate  $p(y|x)$  or  $p(x|y)$ ,  $p(y)$
- We will consider
  - Discriminative and generative models
  - Parametric and nonparametric models

# Generative Models

$$p(y|\mathbf{x}) \propto p(y)p(\mathbf{x}|y)$$

- When we know the prior and conditional densities, we know everything about the data for classification
- Density has to be estimated
- Given examples from different classes, 'standard' density estimation is sufficient
- It is possible to 'generate' (sample) from the classes

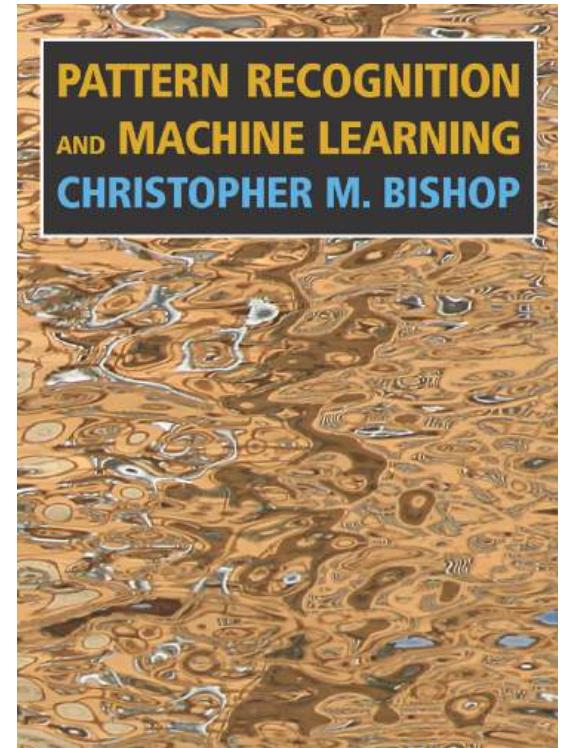
# Discriminative Models

$$\hat{p}(y|\mathbf{x})$$

- When we don't know the class conditional probabilities, and class priors, directly approximate posterior probability?
- Hard problem : given measurements, e.g. person's height, how can we estimate  $p(\text{woman}|\text{height})$ ?
- Strong assumptions are needed, or sloppy approximations are taken

# Literature

- Sections 1.2 (up to 1.2.5) and 1.4 from this book:
  - Pattern recognition and Machine Learning, by C.M.Bishop (2006)
- 
- At: <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>



# Parametric Modeling & Estimation

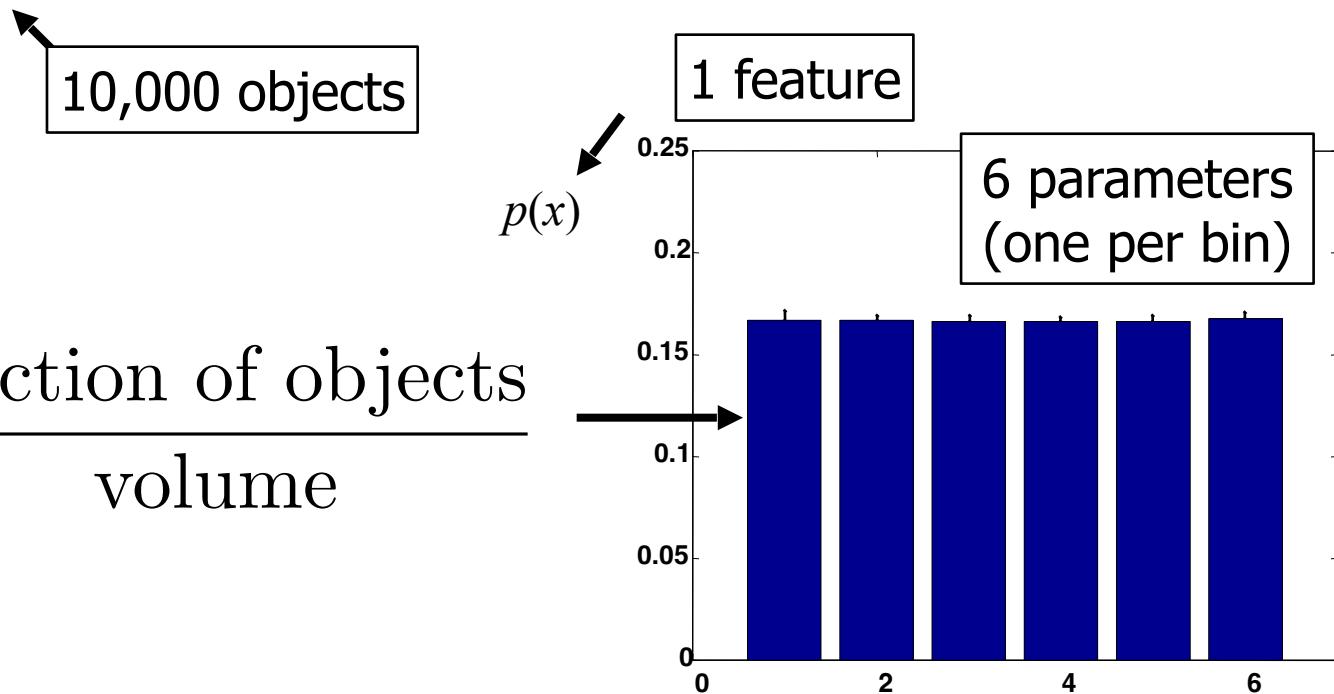
- Density estimation
  - Simple nonparametric approach
  - Curse of dimensionality
  - Parametric models
- Some related topics
  - Spherering
  - Properties of Gaussian
  - Mixture modeling



# Histogram-based Density Estimation

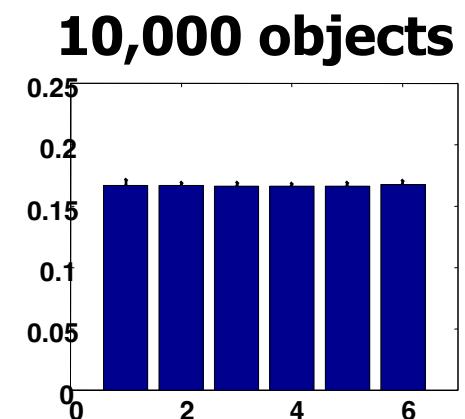
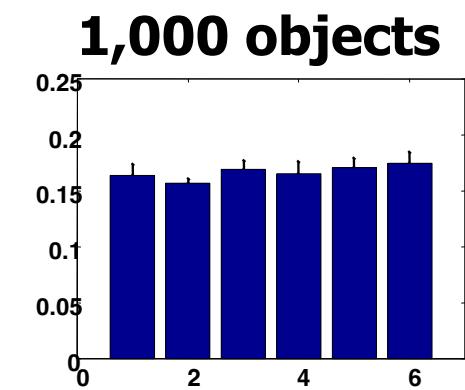
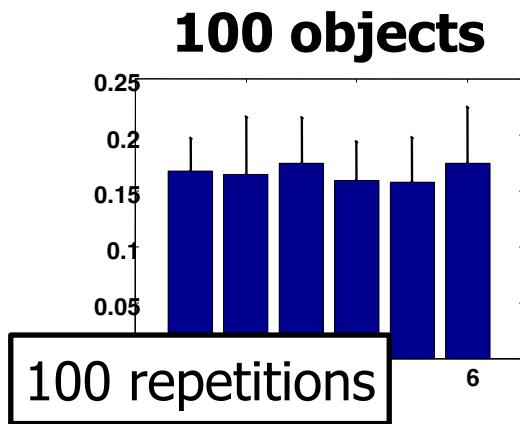
- Relatively simple approach: approximate density by histogram  
E.g. 10,000 throws of a dice

$$\hat{p}(x) = \frac{\text{fraction of objects}}{\text{volume}}$$



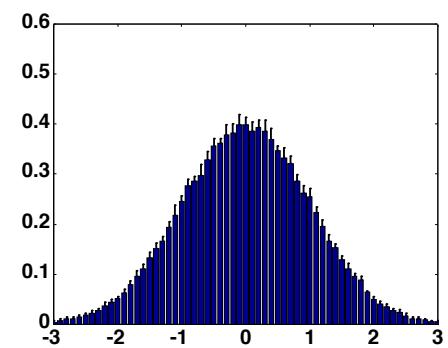
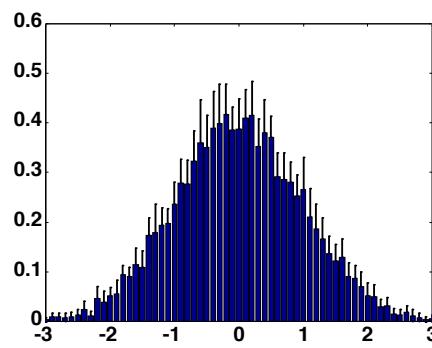
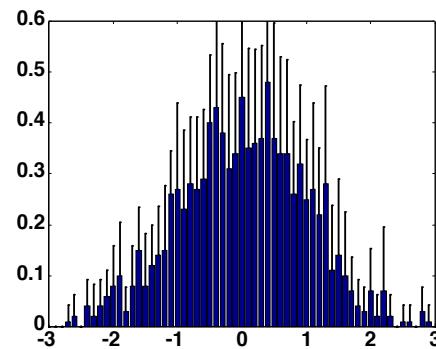
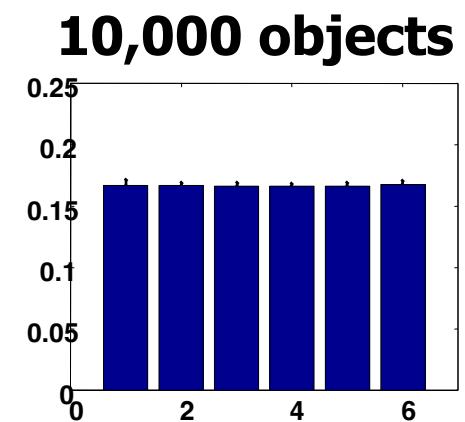
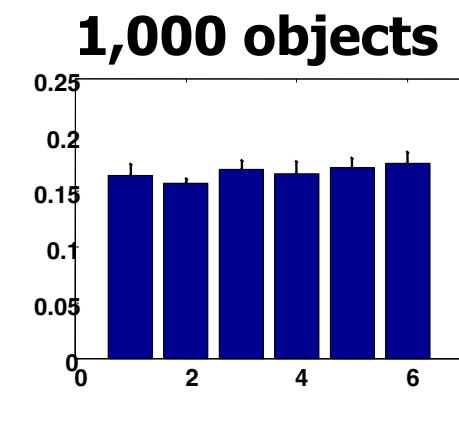
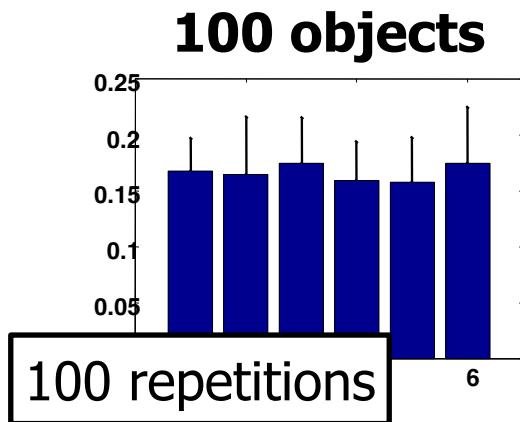
# Histogram-based Density Estimation

- Problem: accuracy



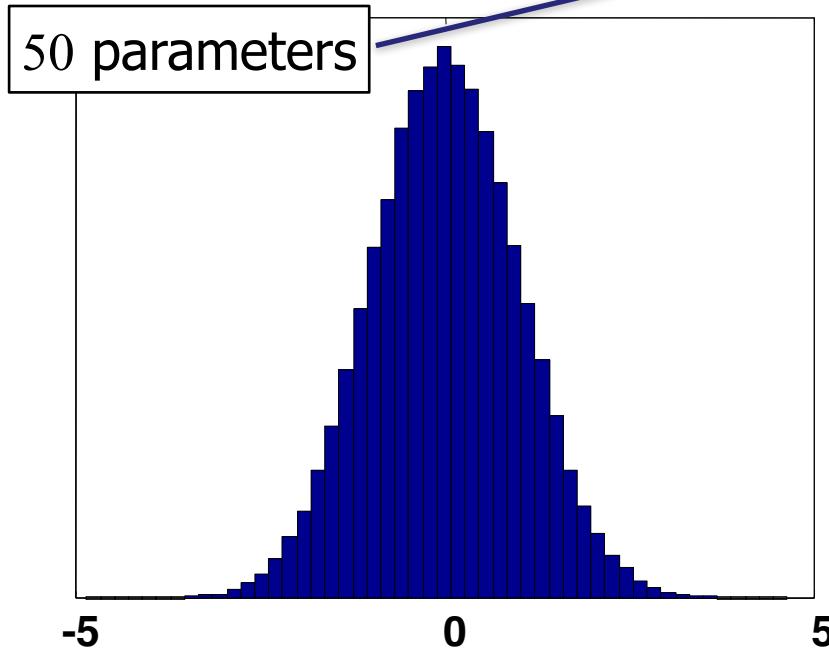
# Histogram-based Density Estimation

- Problem: accuracy



# Histogram-based Density Estimation

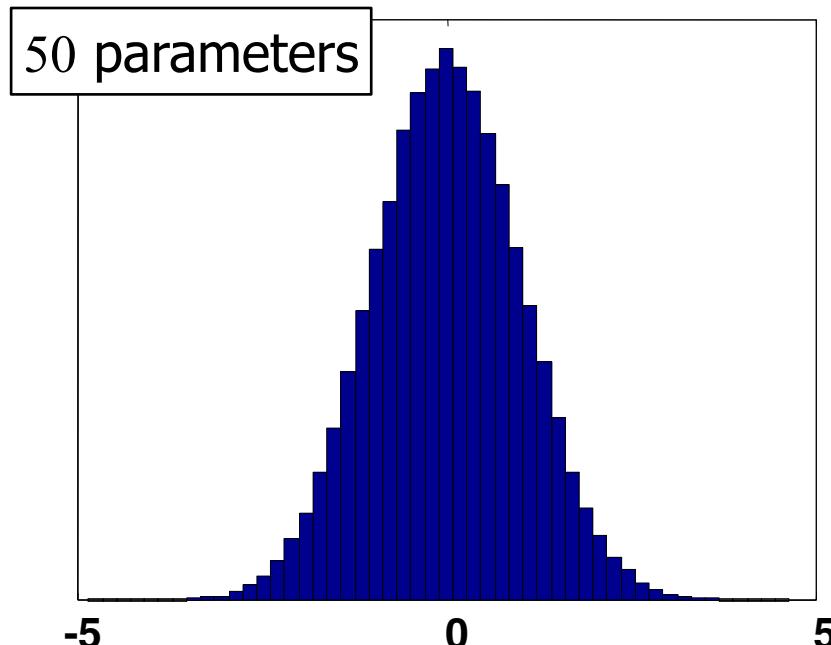
For 1-dimensional data,  
 $\pm 1000$  objects needed



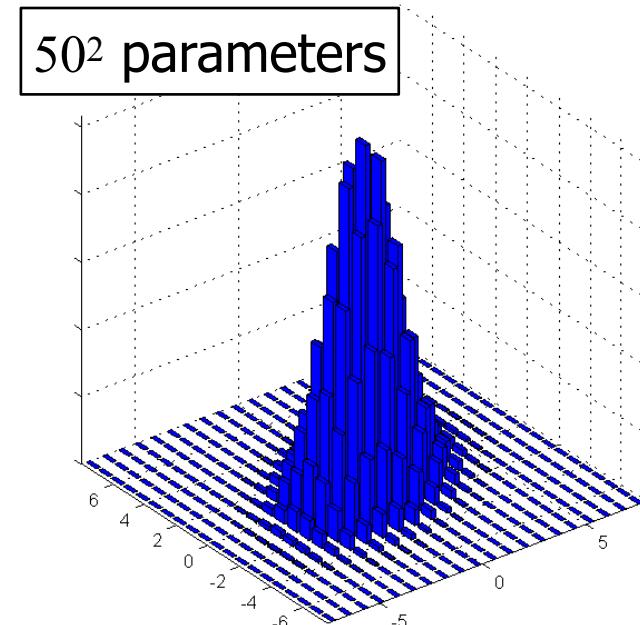
For each bin  
we estimate one  
value:  
with 50 bins  
we have 50  
parameters

# Histogram-based Density Estimation

For 1-dimensional data,  
 $\pm 1000$  objects needed



For  $M$ -dimensional data,  
 $\pm 1000^M$  objects needed

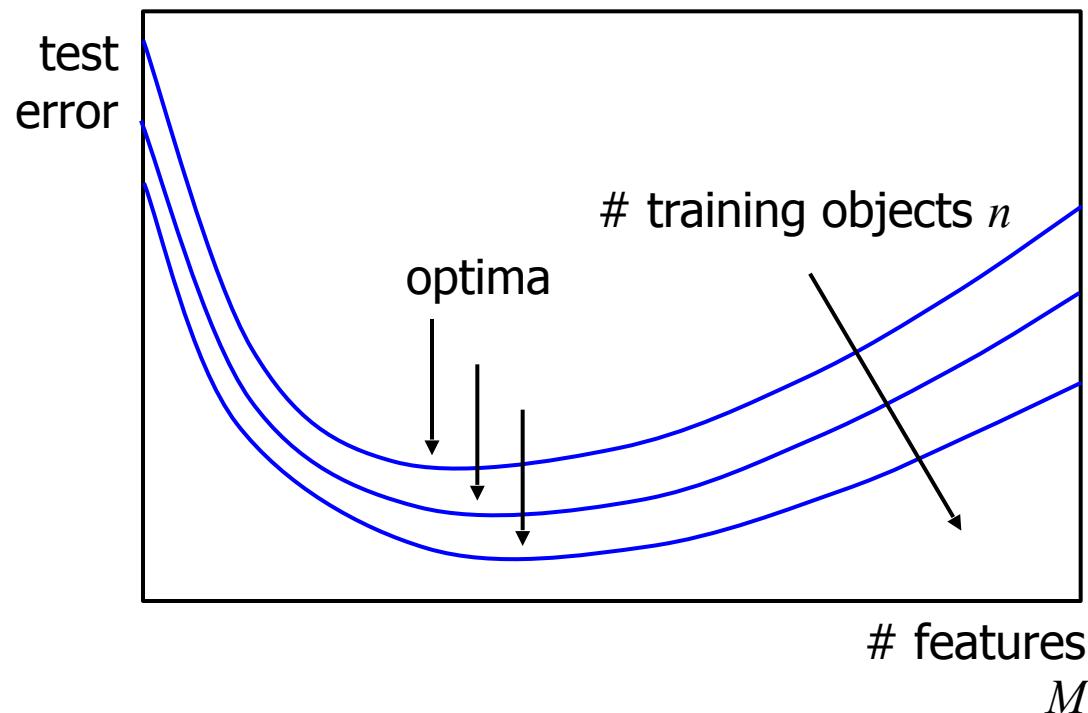


- Unworkable for  $M > 2$  features

# Curse of Dimensionality

- Intuitively, using more features [e.g. width, height, color etc.] should give us more information about the outcome to predict
- But never know densities, so have to estimate
- Number of parameters [e.g. histogram bins] to estimate increases with the number of features
- To estimate these well, you need more objects
  
- Consequence: there is an optimal number of features to use

# Curse of Dimensionality

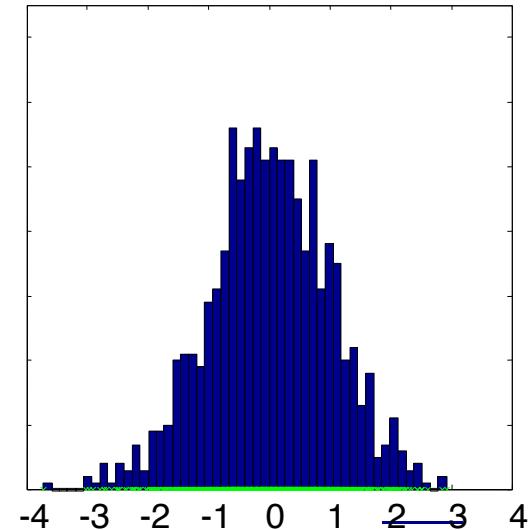
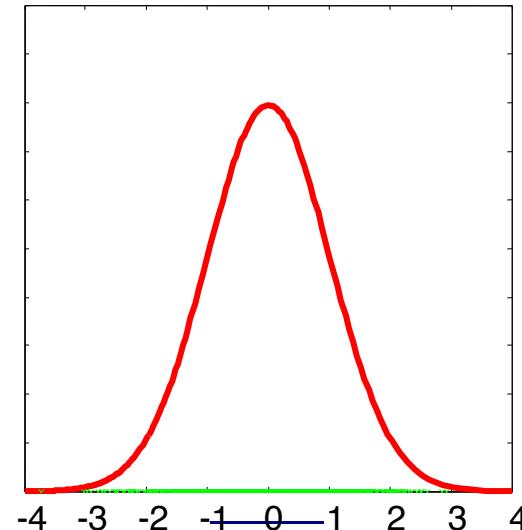
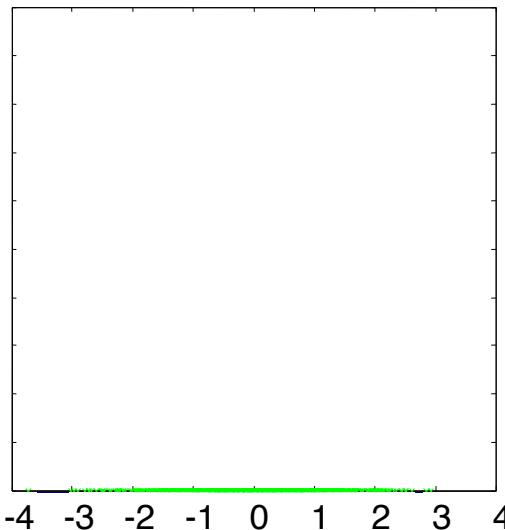




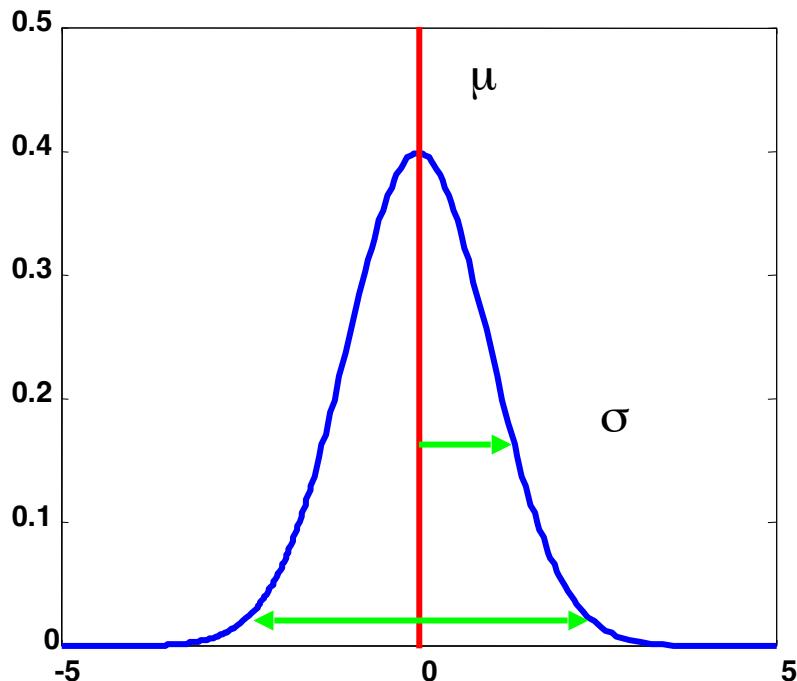
GL 890

# Density Estimation Approaches

- Parametric: need only a few parameters  
assume simple global model, e.g. Gaussian,
- Non-parametric: depends on training data,  
assume simple local model, e.g. uniform, Gaussian



# Gaussian Distribution



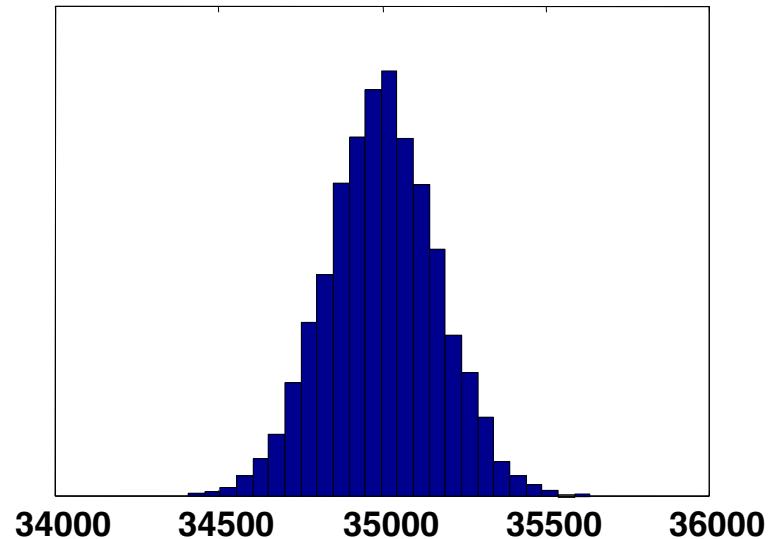
- Normal distribution = Gaussian distribution
- Standard normal distribution:  
 $\mu = 0, \sigma^2 = 1$
- 95% of data between  $[\mu - 2\sigma, \mu + 2\sigma]$  [in 1D!]

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

# Gaussian Distribution

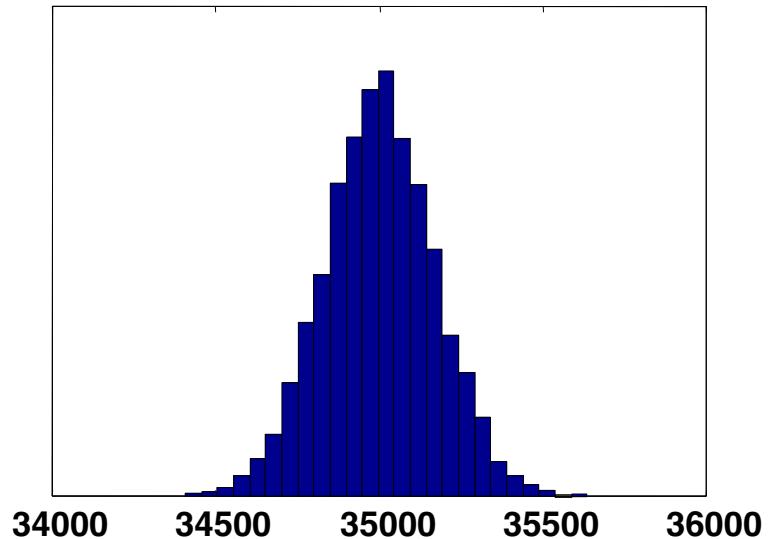
- Why Gaussians?
  - Special distribution : Central limit theorem says that sums of large numbers of i.i.d. [independent, identically distributed] random variables will have a Gaussian distribution
  - May actually occur in real life [approximately]

E.g. sum of eyes of  
10000 dice throws



# Gaussian Distribution

- Why Gaussians [continued]?
  - Simple, few parameters
  - Easy to estimate these parameters when using maximum likelihood!



# Multivariate Gaussians

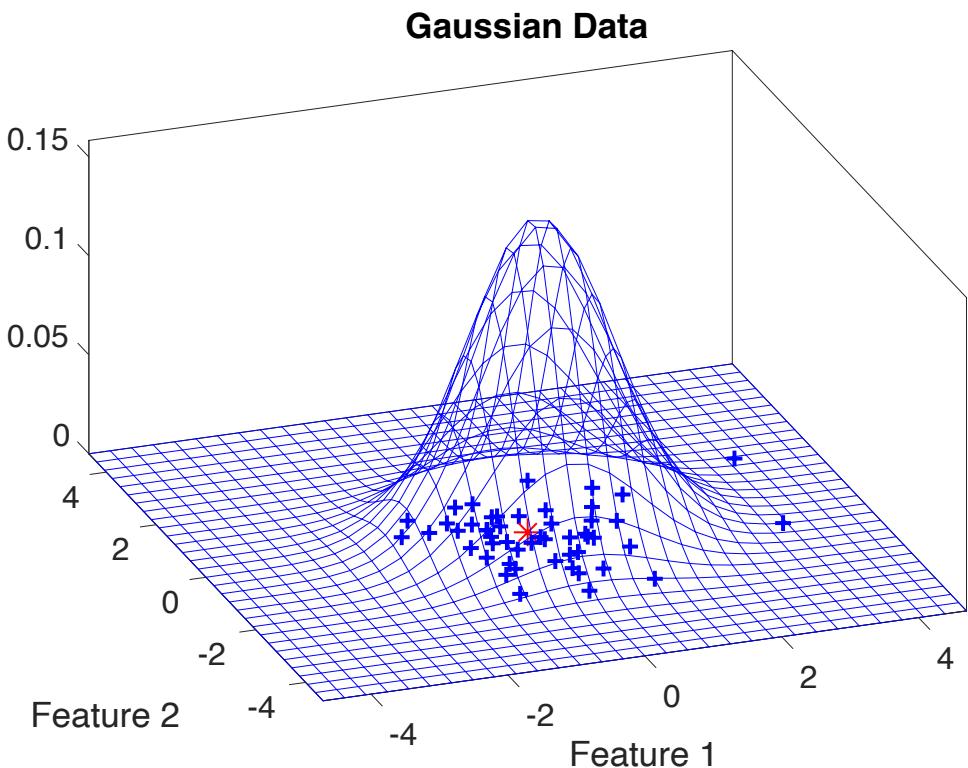
- M - dimensional density:

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

- Sometimes also denoted as:

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

# Gaussian Distribution 2D



Mean:

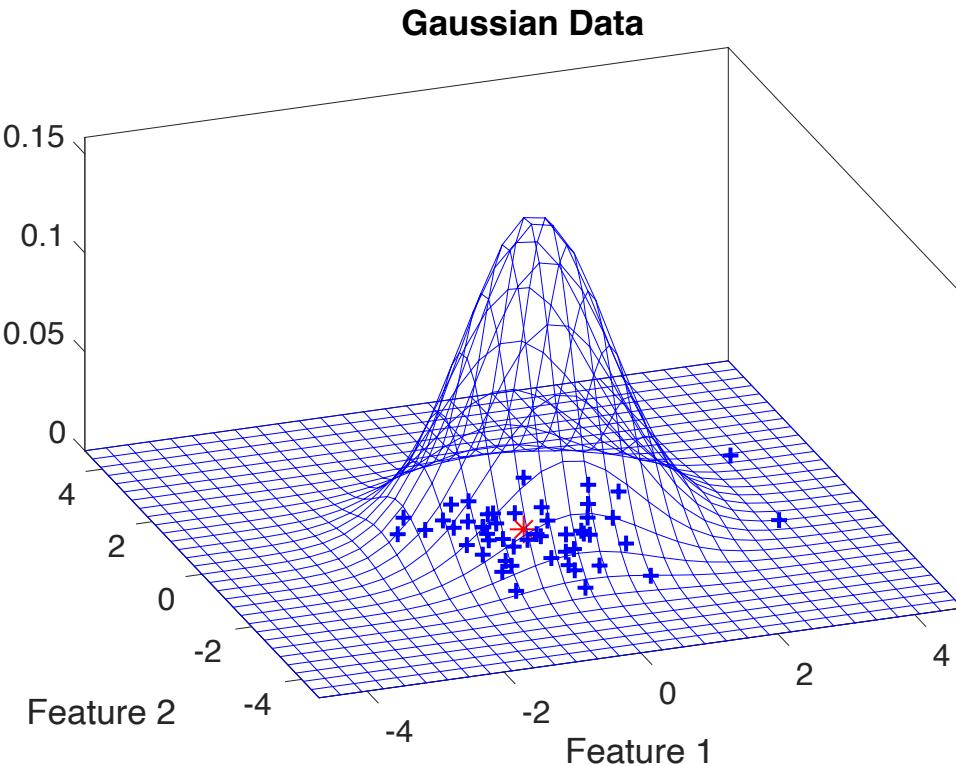
$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

Covariance matrix:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

# Gaussian Distribution 2D

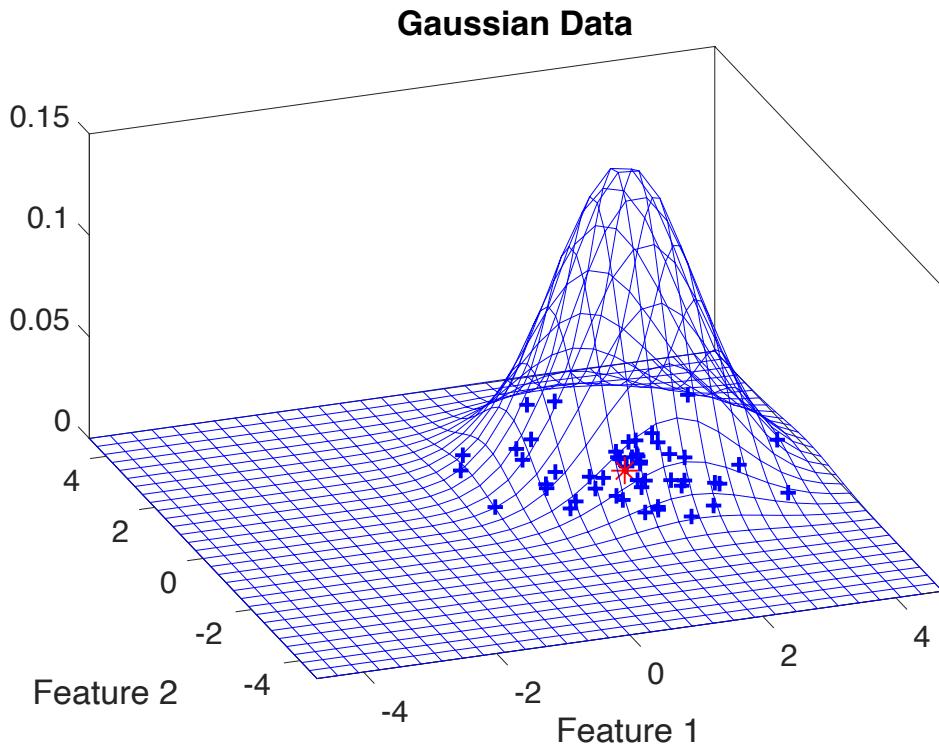


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

# Gaussian Distribution 2D

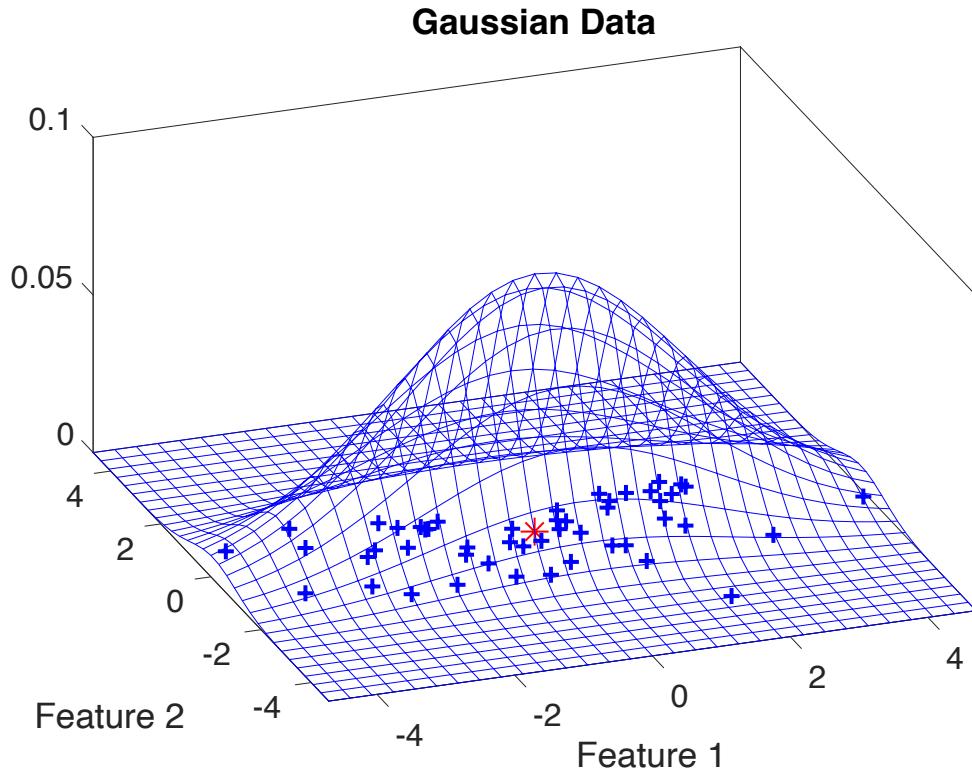


$$\mu = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

# Gaussian Distribution 2D

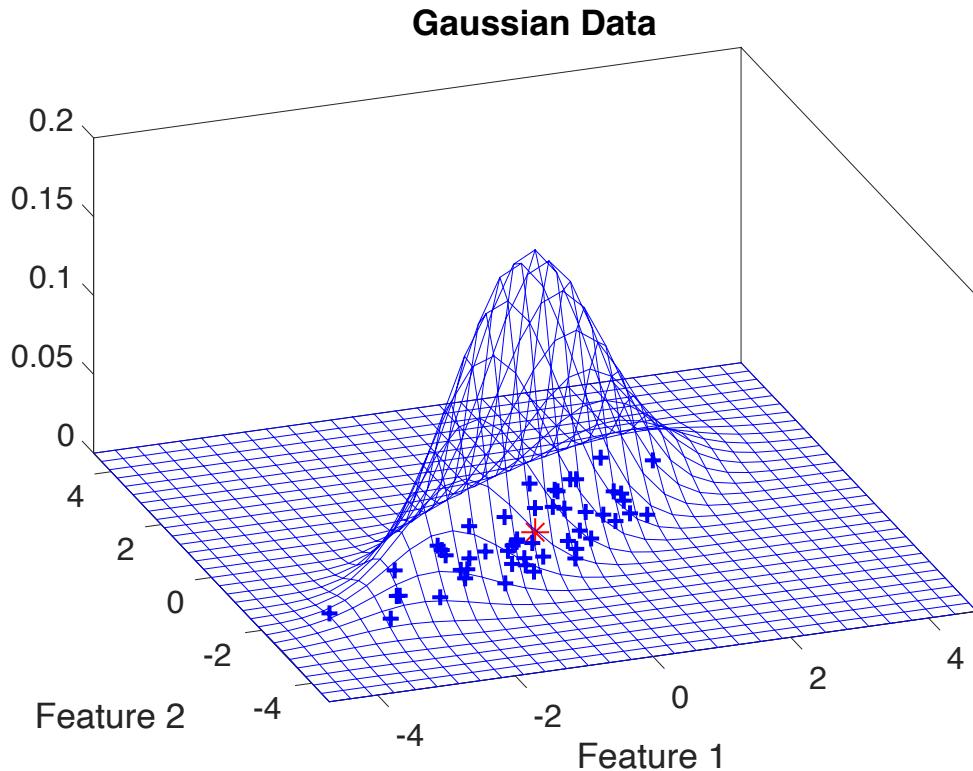


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

# Gaussian Distribution 2D



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

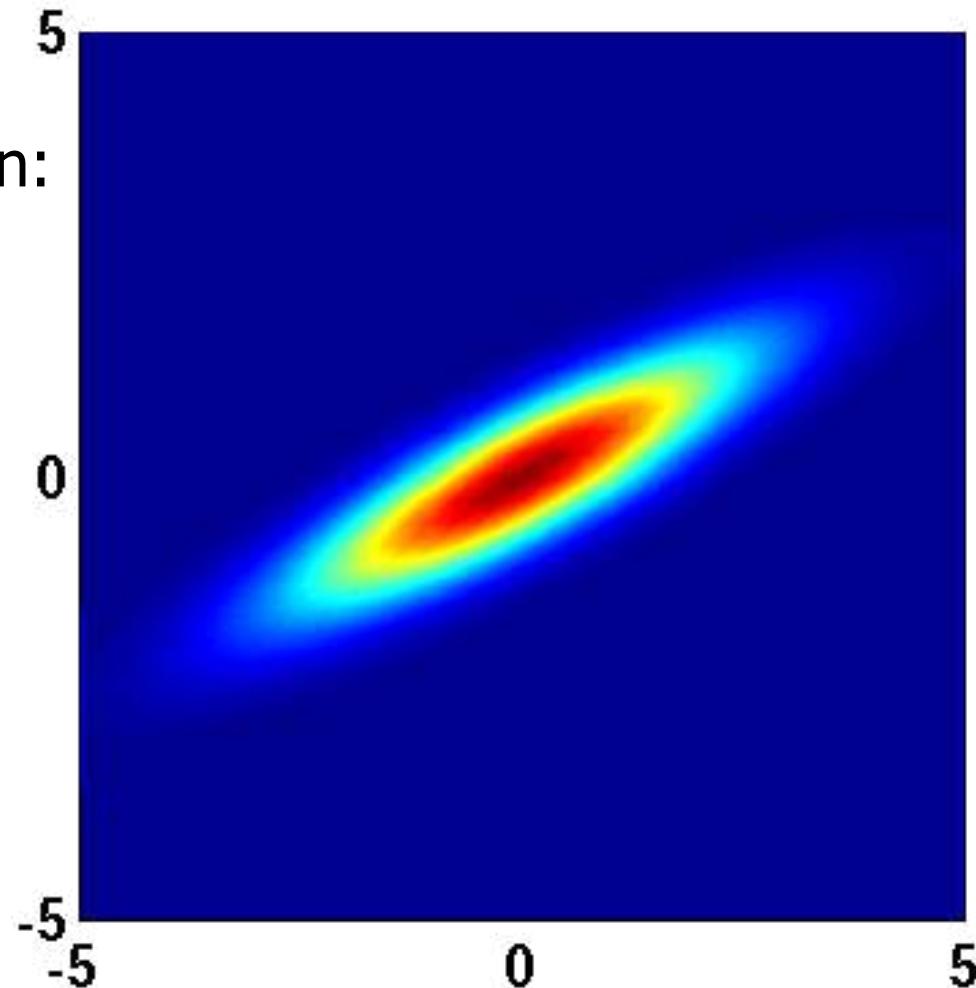
$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

# Example 2D Gaussian

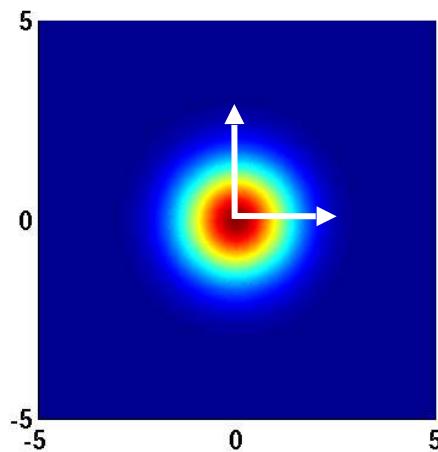
- Top view on an example 2D Gaussian:

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

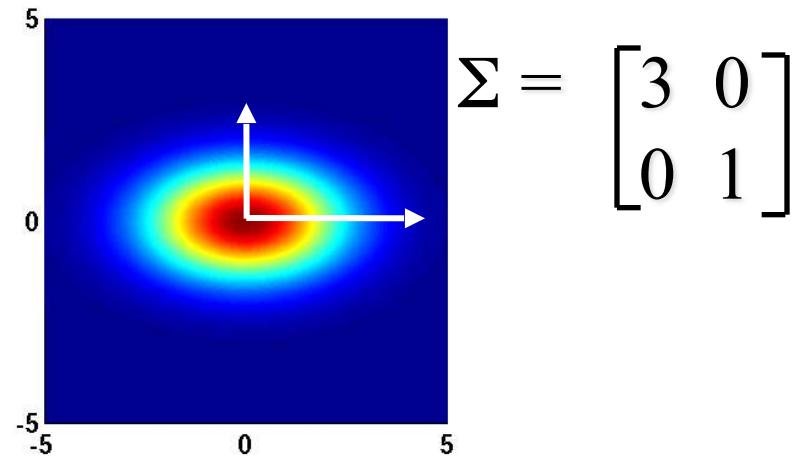
$$\Sigma = \begin{bmatrix} 3 & 1.5 \\ 1.5 & 2 \end{bmatrix}$$



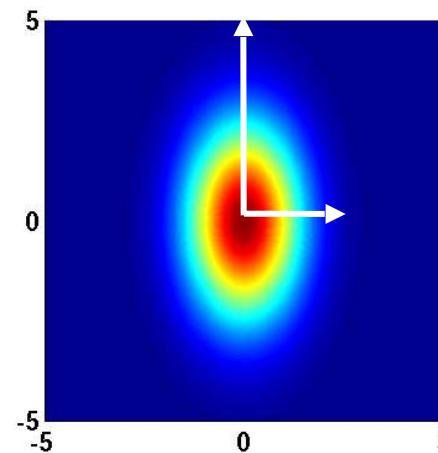
# More examples of 2D Gaussians



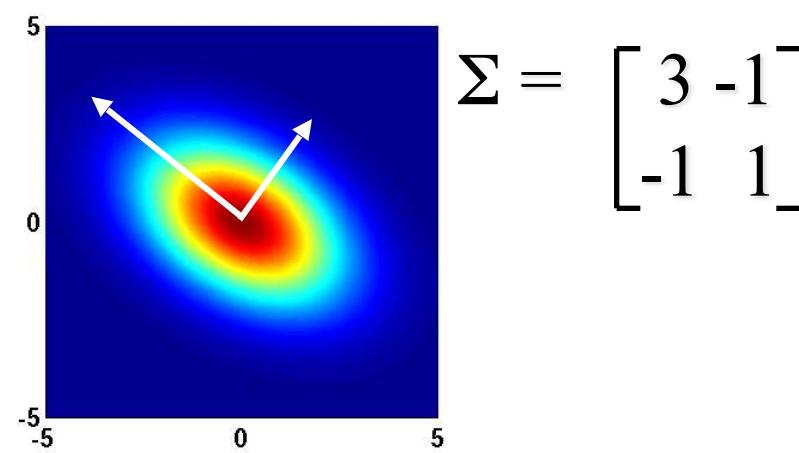
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 3 & -1 \\ -1 & 1 \end{bmatrix}$$

# Maximum likelihood estimates?

- What are the maximum likelihood estimators for  
the mean  
the covariance matrix?

# Maximum likelihood estimates

- What are the maximum likelihood estimators for the mean and the covariance matrix?

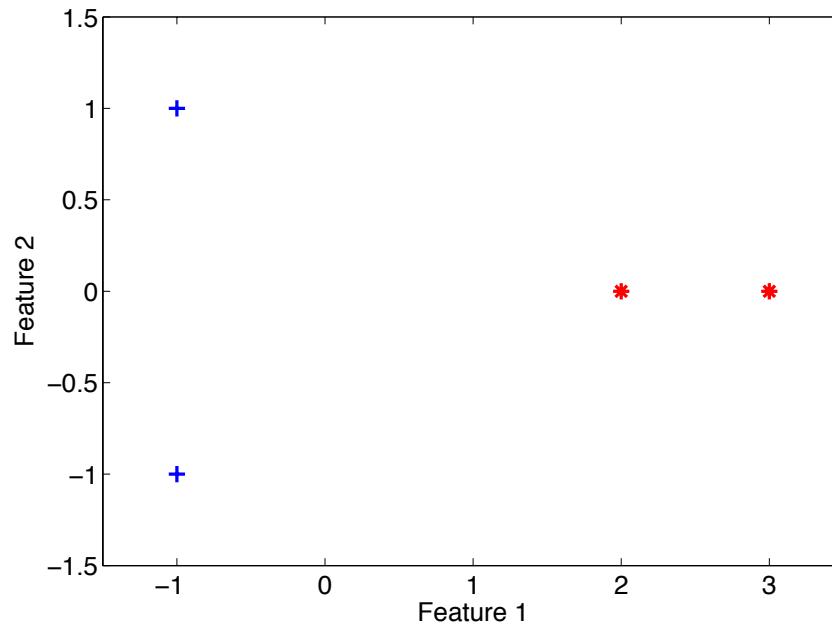
$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$$

# Estimating the covariance matrix

$$X = \begin{bmatrix} -1 & -1 \\ -1 & +1 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



- Mean of class +1?
- Covariance matrix of class +1? And its inverse?

# Estimating the mean

- Objects of class +1: stored in rows:

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$\mathbf{x}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

- Then:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i = \frac{1}{2} \left( \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 2.5 \\ 0 \end{bmatrix}$$

# Estimating the covariance matrix

- Make new, centered, dataset:

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T = \frac{1}{2} \sum_{i=1}^n \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T$$

- Then:

$$\tilde{\mathbf{x}}_1 = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \quad \tilde{\mathbf{x}}_2 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

- And:

$$\tilde{\mathbf{x}}_1 \tilde{\mathbf{x}}_1^T = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}$$

# Estimating the covariance matrix

- The same for

$$\tilde{\mathbf{x}}_2 \tilde{\mathbf{x}}_2^T = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}$$

- So in total:

$$\begin{aligned}\hat{\Sigma} &= \frac{1}{2} \sum_{i=1}^n \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T = \frac{1}{2} \left( \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}\end{aligned}$$

# Inverse of the covariance matrix?

- Inverse of?:

$$\hat{\Sigma} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}$$

- It should hold that:

$$\hat{\Sigma}\hat{\Sigma}^{-1} = \mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Can you find a,b,c, such that?

$$\begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

# Inverse of the covariance matrix?

- Inverse of?:

$$\hat{\Sigma} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}$$

- It should hold that:

$$\hat{\Sigma}\hat{\Sigma}^{-1} = \mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Can you find a,b,c, such that?

$$\begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

NO, not possible  
Not invertible

# Inverse of the covariance matrix?

- To solve

$$\begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

you need to get some 'c' for which:

$$0 \times c = 1$$

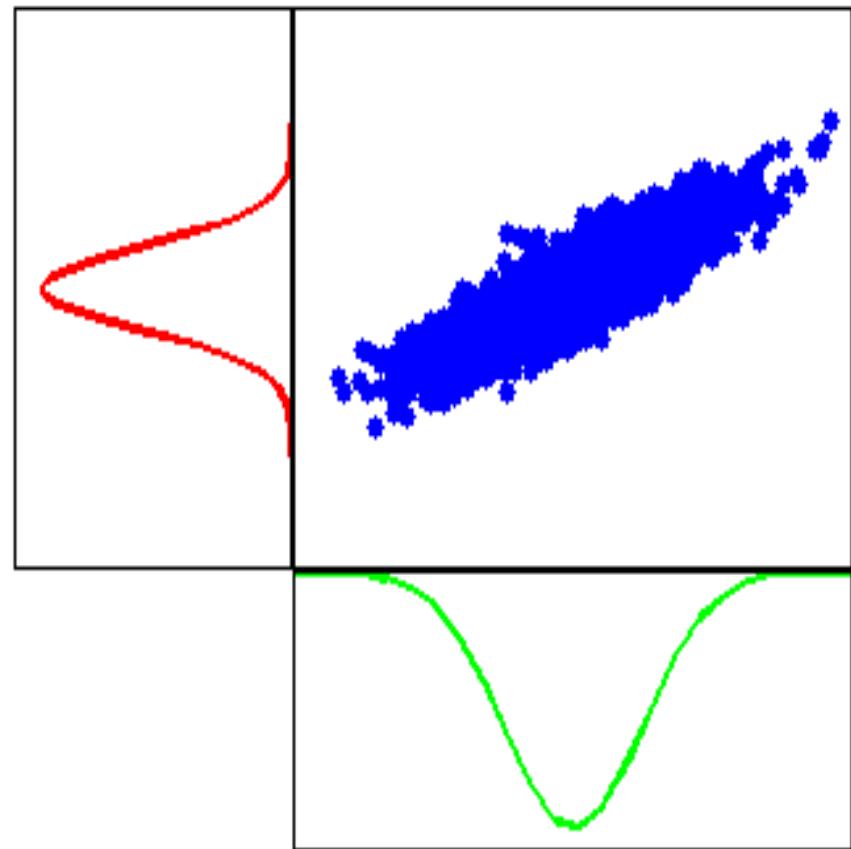
- Not possible!
- Number of objects is insufficient to find the inverse
- Two objects in a 2-dimensional feature space:  
a degenerate Gaussian distribution

# Parametric Estimation

- Sounds simple, but for  $M$ -dimensional data set :
  - $\mu$  : vector with  $M$  elements
  - $\Sigma$  : matrix with  $0.5 M(M+1)$  elements
- Number of parameters increases quadratically with  $M$  : might still need lots of data for high-dimensional data

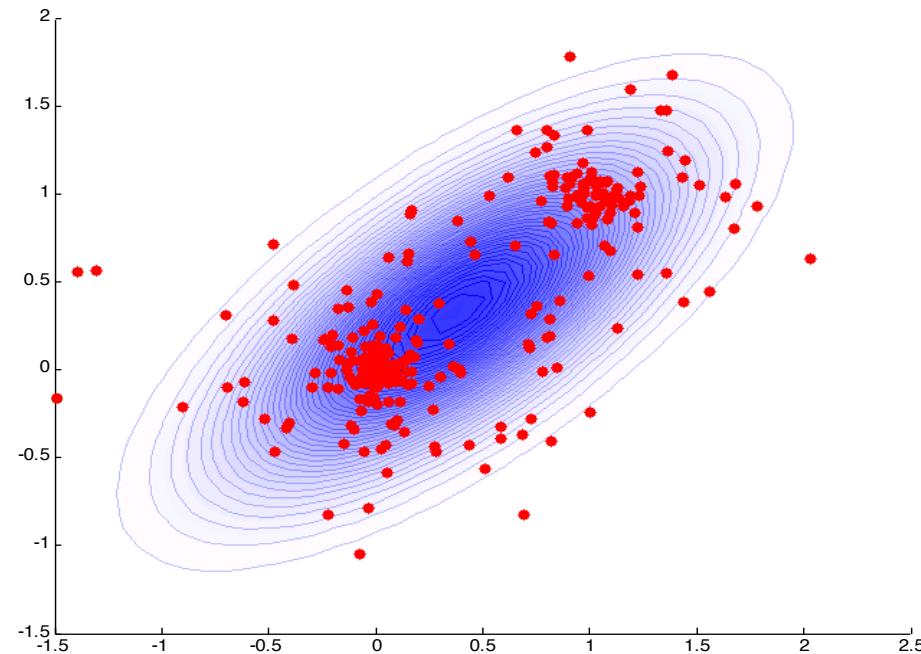
# Special Properties

- Any projection of a high-dimensional Gaussian is itself again Gaussian



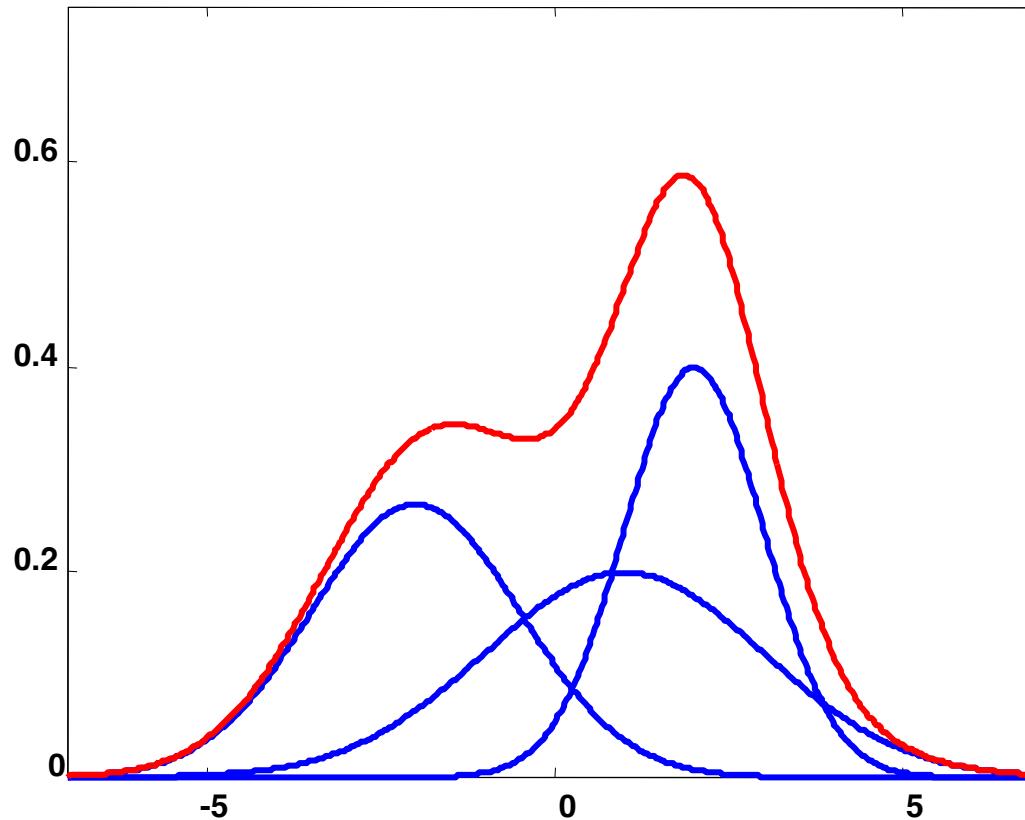
# Parametric Estimation

- Assume model, e.g. Gaussian
- Estimate mean  $\mu$  and covariance  $\Sigma$  from data



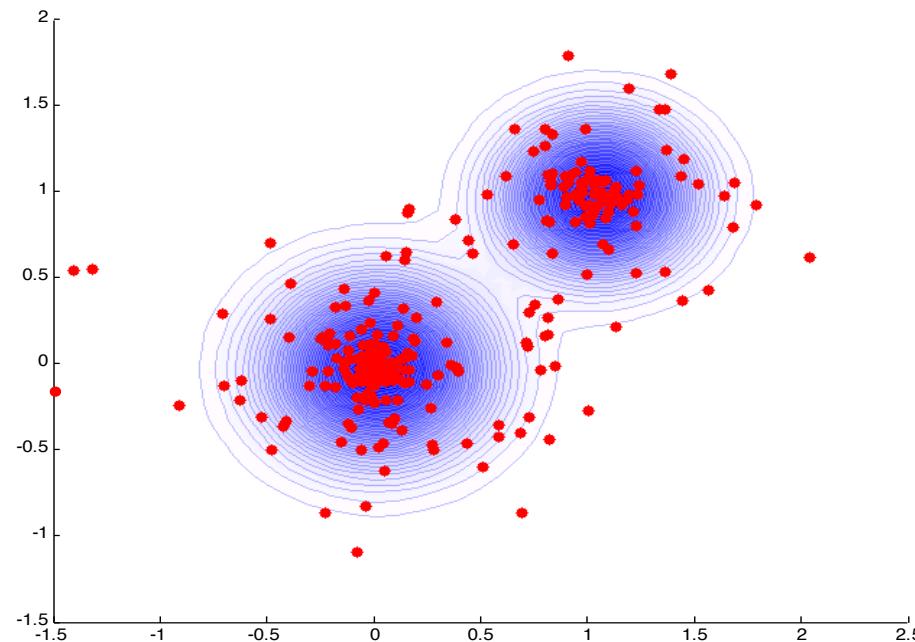
# Gaussian Distribution

- Not necessarily too restrictive: mixture models



# Mixture models

- Model : mixture of 2 Gaussians
- More parameters:  $\pi_i$ ,  $\mu_i$  and  $\Sigma_i$ ,  $i = 1, 2$



# Mixture models

- Estimating parameters not straightforward:

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- If you know memberships of objects to clusters, you can estimate means+cov.'s
- You only know the memberships if you have the means+cov.'s
- EM-algorithm: see Master course

# One may Wonder...

- In parametric case, why only consider multivariate Gaussian?
- When using histograms, what seems one of the underlying assumptions?
- With increasing dimensions / # measurements, does Gaussian or histogram need more data?
- When does one have enough data?
- What if one class has multiple modes, and the other not?

1 1 1 1 1 1 1 1 1 1 1 1  
7 7 7 7 7 7 7 7 7 7 7 7

# Recapitulation

- Up to now:
  - Without models no generalization
  - Density estimation: core of statistical learning, really hard problem
  - Curse of dimensionality: adding measurements does not always help
  - Parametric density estimation: Gaussian
- Next:
  - Classification using Gaussian densities

# After this lecture you should be able to:

- explain how you obtain a classifier using a Gaussian (multivariate) distribution for each class
- implement a simple univariate classifier in Python
- explain what the 'curse of dimensionality' is

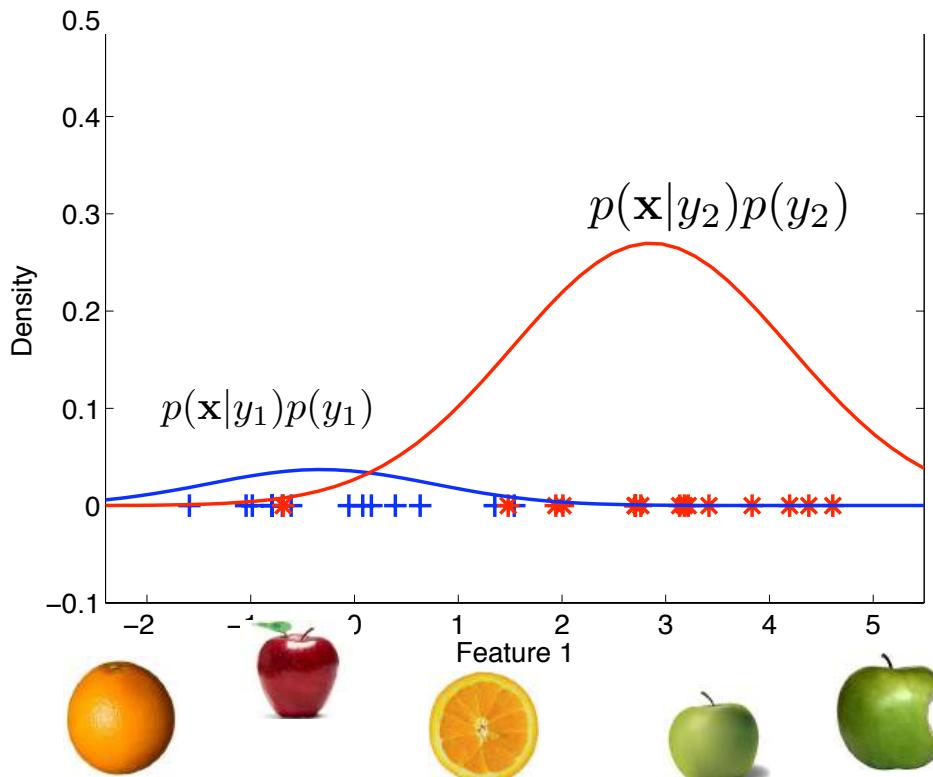
## Next lecture:

- explain (the advantages and disadvantages) of the Quadratic classifier, the LDA and the nearest mean classifier
- identify when scaling of the features is important and how to cope with feature scaling

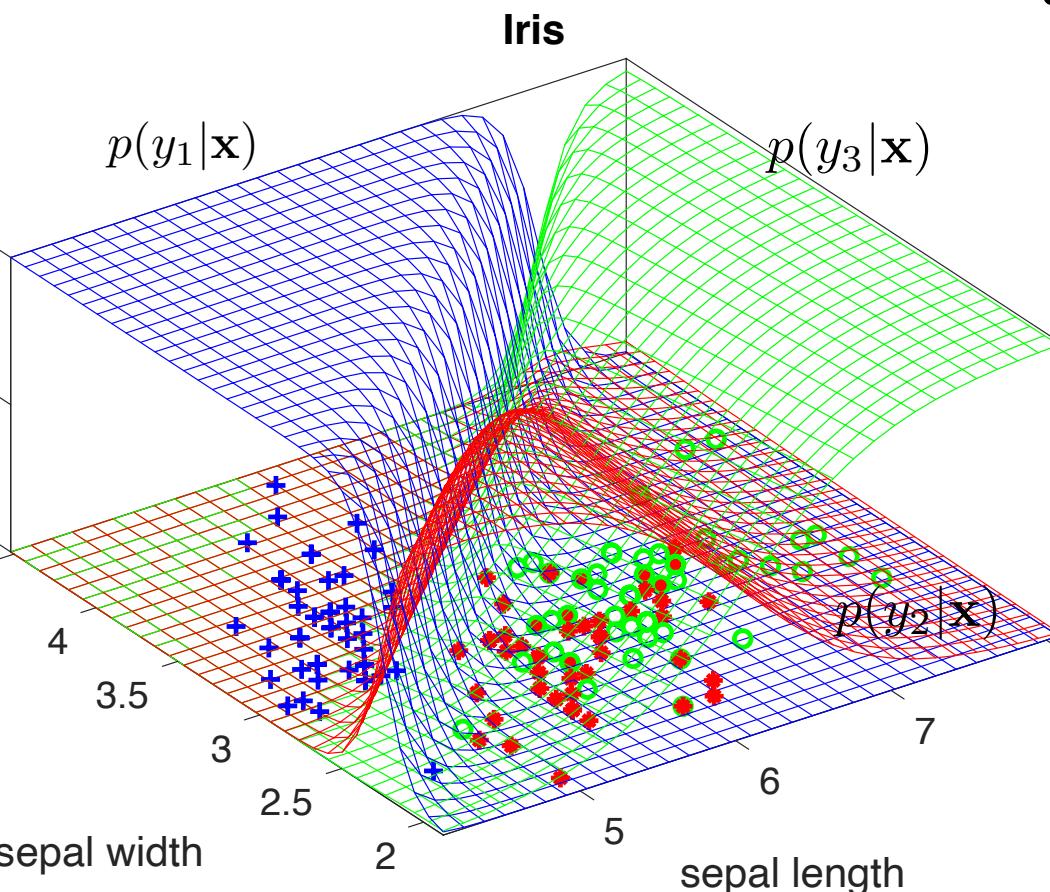


# Gaussian Density-based Classifiers

## CSE2510 Machine Learning



# Modeling the posterior probability



- For each object in the feature space, we should find:  
$$p(y|\mathbf{x})$$

In practice, we approximate:  
$$\hat{p}(y|\mathbf{x})$$

or we fit a function:

$$f(\mathbf{x})$$

# Recapitulation

- Up to now:
  - Without models no generalization
  - Density estimation: core of statistical learning, really hard problem
  - Curse of dimensionality: adding features does not always help
  - Parametric density estimation: Gaussian
- Now:
  - Classification using Gaussian densities:
  - Quadratic classifier
  - Linear classifier
  - Nearest mean classifier

(proportional to)


$$p(y|\mathbf{x}) \propto p(y)p(\mathbf{x}|y)$$

$$p(y_1|\mathbf{x}) > p(y_2|\mathbf{x})$$

# Plug-in Bayes Rule

- Classify to class 1 when :  $p(y_1|\mathbf{x}) > p(y_2|\mathbf{x})$
- May be easier to estimate the class-conditional probability density :  $\hat{p}(\mathbf{x}|y_1), \hat{p}(\mathbf{x}|y_2)$
- Use Bayes' rule to transform one into the other :

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

- Putting just  $\hat{p}(\mathbf{x}|y)$  in results in plug-in Bayes rule

# Plug-in Bayes Rule

- So, for Bayes rule  $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$   
we need to estimate:
  - (1) the class priors  $\hat{p}(y)$
  - (2) the unconditional probabilities  $\hat{p}(\mathbf{x})$
  - (3) the class-conditional probabilities  $\hat{p}(\mathbf{x}|y)$

# How to estimate the class priors?

- Given a training set, how can you estimate  $\hat{p}(y)$  ?
- The classes are discrete,  $\hat{p}(y)$  is a true probability
- Often they are known/assumed
- Otherwise, just count!

$$\hat{p}(y_1) = \frac{N_1}{N}$$

$$\hat{p}(y_2) = \frac{N_2}{N}$$

# How to estimate the unconditional pr.

- How to estimate the (unconditional) data distribution  $\hat{p}(\mathbf{x})$ ?

- You can explicitly compute it:

$$p(\mathbf{x}) = p(\mathbf{x}|y_1)p(y_1) + p(\mathbf{x}|y_2)p(y_2)$$

- But if you just find the largest posterior, not needed:

$$\frac{p(\mathbf{x}|y_1)p(y_1)}{p(\mathbf{x})} > \frac{p(\mathbf{x}|y_2)p(y_2)}{p(\mathbf{x})}$$

# Estimate the class conditional prob.

- The model for the class conditional probability is now the crucial choice
- Very common: Gaussian distribution

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

- It models a 'blob'-like distribution, in a M-dimensional feature space
- $\boldsymbol{\mu}$  is the mean of the distribution
- $\boldsymbol{\Sigma}$  is the (elliptical) shape of the distribution

# Plug-in Gaussian Distribution

- For each class we have a Gaussian distribution

$$\hat{p}(\mathbf{x}|y) = \frac{1}{\sqrt{(2\pi)^M \det(\hat{\Sigma}_y)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \hat{\mu}_y)^T \hat{\Sigma}_y^{-1} (\mathbf{x} - \hat{\mu}_y)\right)$$

- We have to estimate the parameters, e.g. ML

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$$

# Plug-in Gaussian Distribution

- For each class we have a Gaussian distribution

$$\hat{p}(\mathbf{x}|y) = \frac{1}{\sqrt{(2\pi)^M \det(\hat{\Sigma}_y)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \hat{\mu}_y)^T \hat{\Sigma}_y^{-1} (\mathbf{x} - \hat{\mu}_y) \right)$$

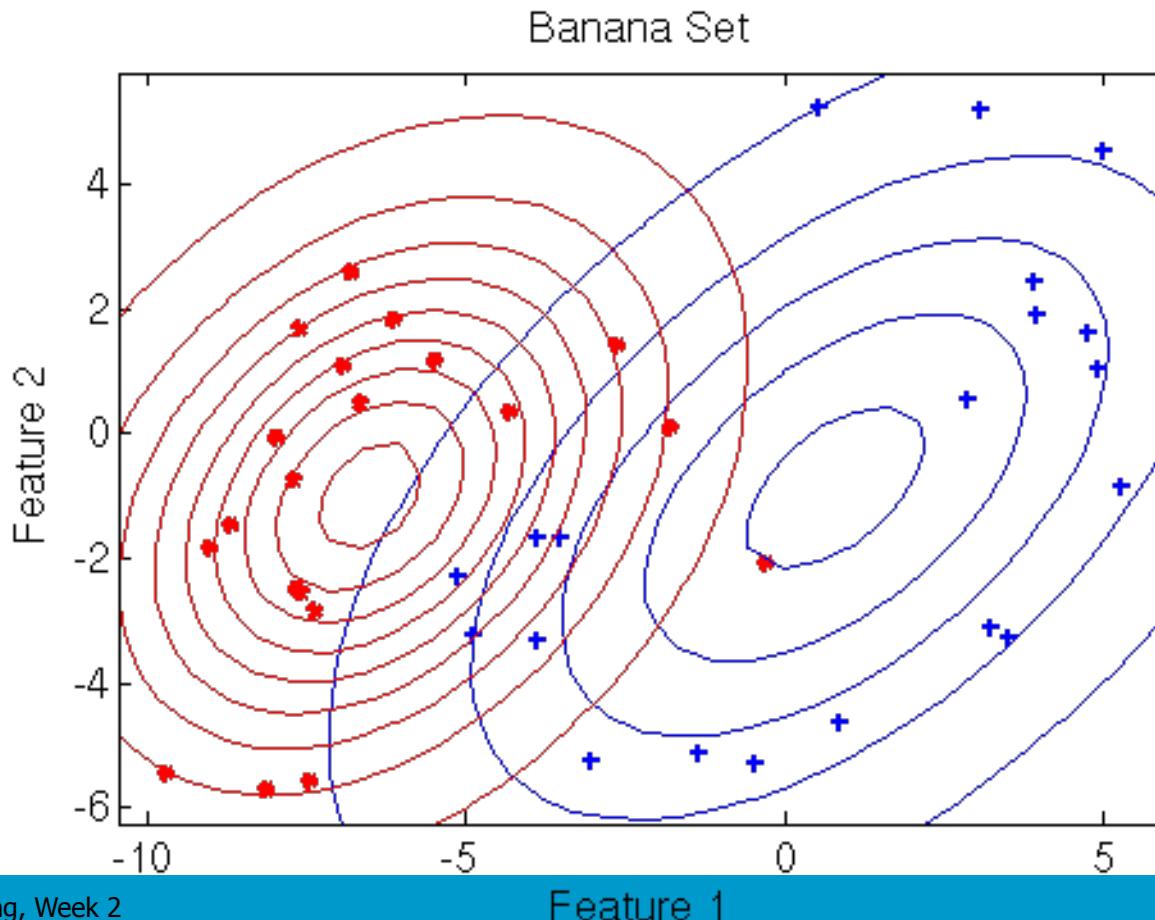
- We have to estimate the parameters, e.g. ML

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad \hat{\Sigma} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$$

unbiased estimator

# Example on Banana Data

- A single Gaussian distribution on each class:



# The Two-Class Case

- Define the discriminant

Note: in book,  
No need to  
Remember everythi

$$f(\mathbf{x}) = \log p(y_1|\mathbf{x}) - \log p(y_2|\mathbf{x})$$

- Rewriting this, it appears that one gets

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{w}^T \mathbf{x} + w_0$$

- This is called a quadratic classifier because the decision boundary is a quadratic function of  $\mathbf{x}$
- How does  $\mathbf{W}$  and  $\mathbf{w}$  look like?

# Class Posterior Probability Gaussian

- Combining

$$\hat{p}(\mathbf{x}|y) = \frac{1}{\sqrt{(2\pi)^M \det(\hat{\Sigma}_y)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \hat{\mu}_y)^T \hat{\Sigma}_y^{-1} (\mathbf{x} - \hat{\mu}_y)\right)$$

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

We can derive for  $\log(p(y_i|\mathbf{x}))$ :

$$\log(\hat{p}(y_i|\mathbf{x})) = -\frac{M}{2} \log(2\pi) - \frac{1}{2} \log(\det \hat{\Sigma}_i)$$

$$-\frac{1}{2}(\mathbf{x} - \hat{\mu}_i)^T \hat{\Sigma}_i^{-1} (\mathbf{x} - \hat{\mu}_i) + \log p(y_i) - \log p(\mathbf{x})$$

# Normal-based Classifier

- Dropping constant terms (independent of  $i$ ):

$$g_i(\mathbf{x}) = -\frac{1}{2} \log(\det \hat{\Sigma}_i) - \frac{1}{2} (\mathbf{x} - \hat{\mu}_i)^T \hat{\Sigma}_i^{-1} (\mathbf{x} - \hat{\mu}_i) + \log p(y_i)$$

- You can work out the square term
- Classifier becomes :

Assign  $\mathbf{x}$  to class  $y_i$  when for all  $i \neq j$  :

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) > 0$$

# Quadratic discriminant

- General form for a two-class classifier:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{w}^T \mathbf{x} + w_0$$

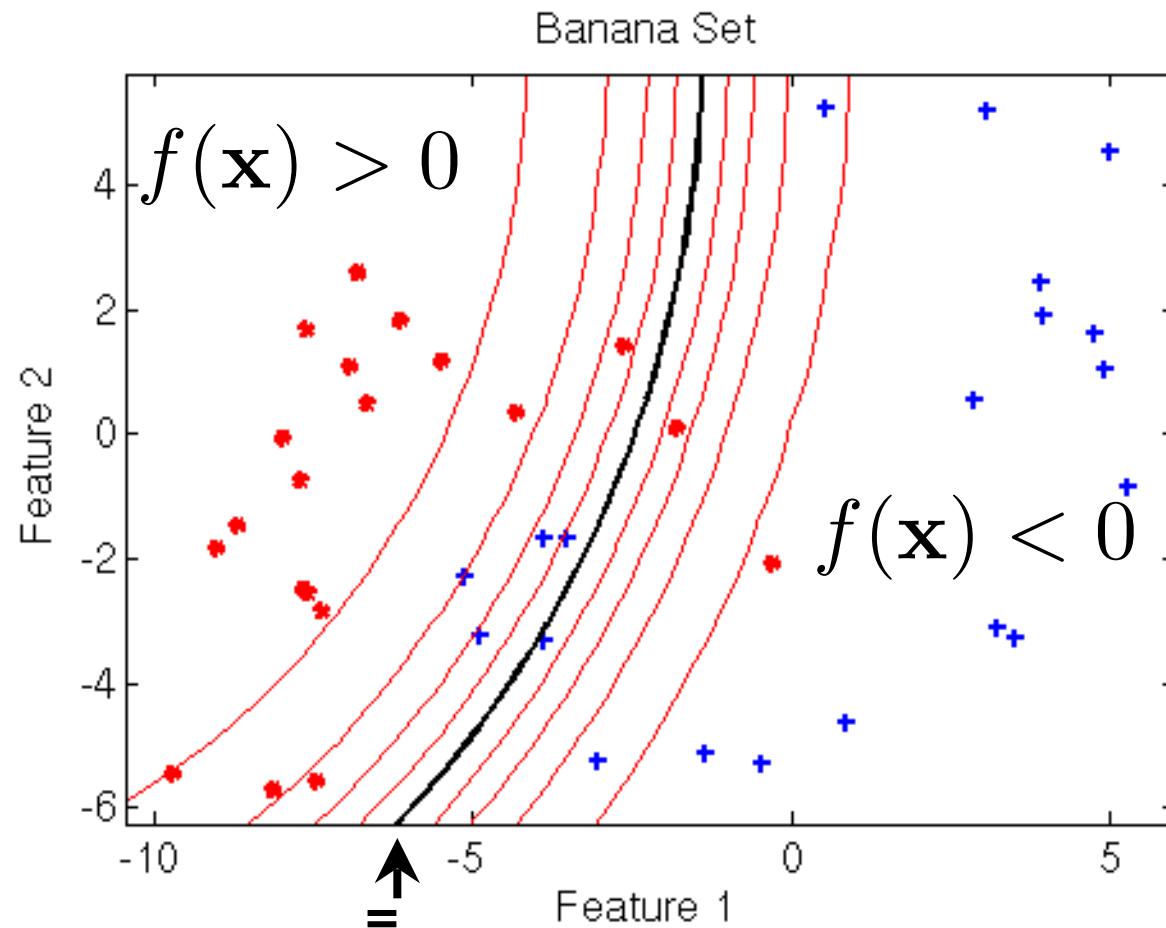
- with

$$\mathbf{W} = \frac{1}{2} \left( \hat{\Sigma}_2^{-1} - \hat{\Sigma}_1^{-1} \right)$$

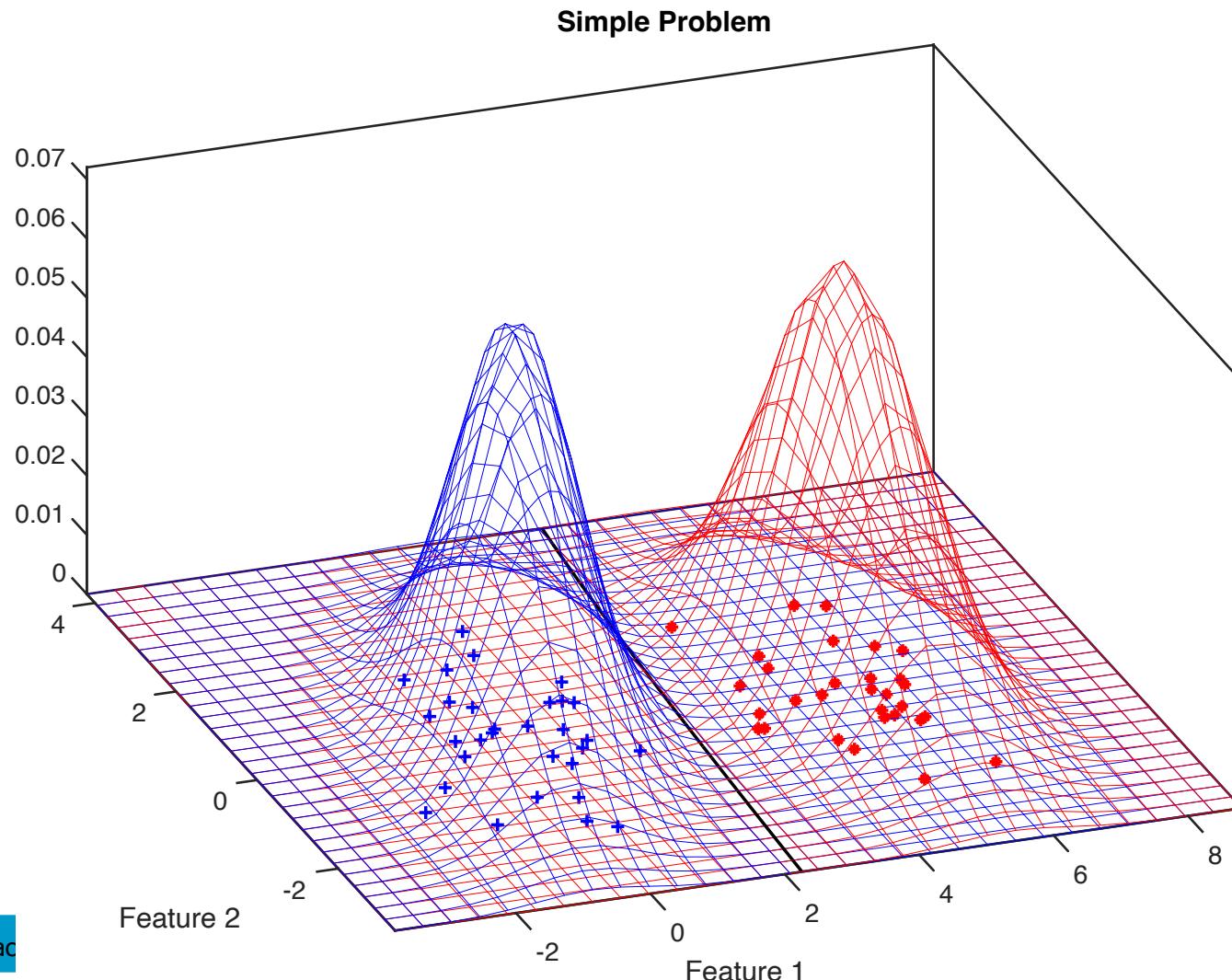
$$\mathbf{w}^T = \hat{\mu}_1^T \hat{\Sigma}_1^{-1} - \hat{\mu}_2^T \hat{\Sigma}_2^{-1}$$

$$\begin{aligned} w_0 = & -\frac{1}{2} \log \det \hat{\Sigma}_1 - \frac{1}{2} \hat{\mu}_1^T \hat{\Sigma}_1^{-1} \hat{\mu}_1 + \log p(y_1) \\ & + \frac{1}{2} \log \det \hat{\Sigma}_2 + \frac{1}{2} \hat{\mu}_2^T \hat{\Sigma}_2^{-1} \hat{\mu}_2 - \log p(y_2) \end{aligned}$$

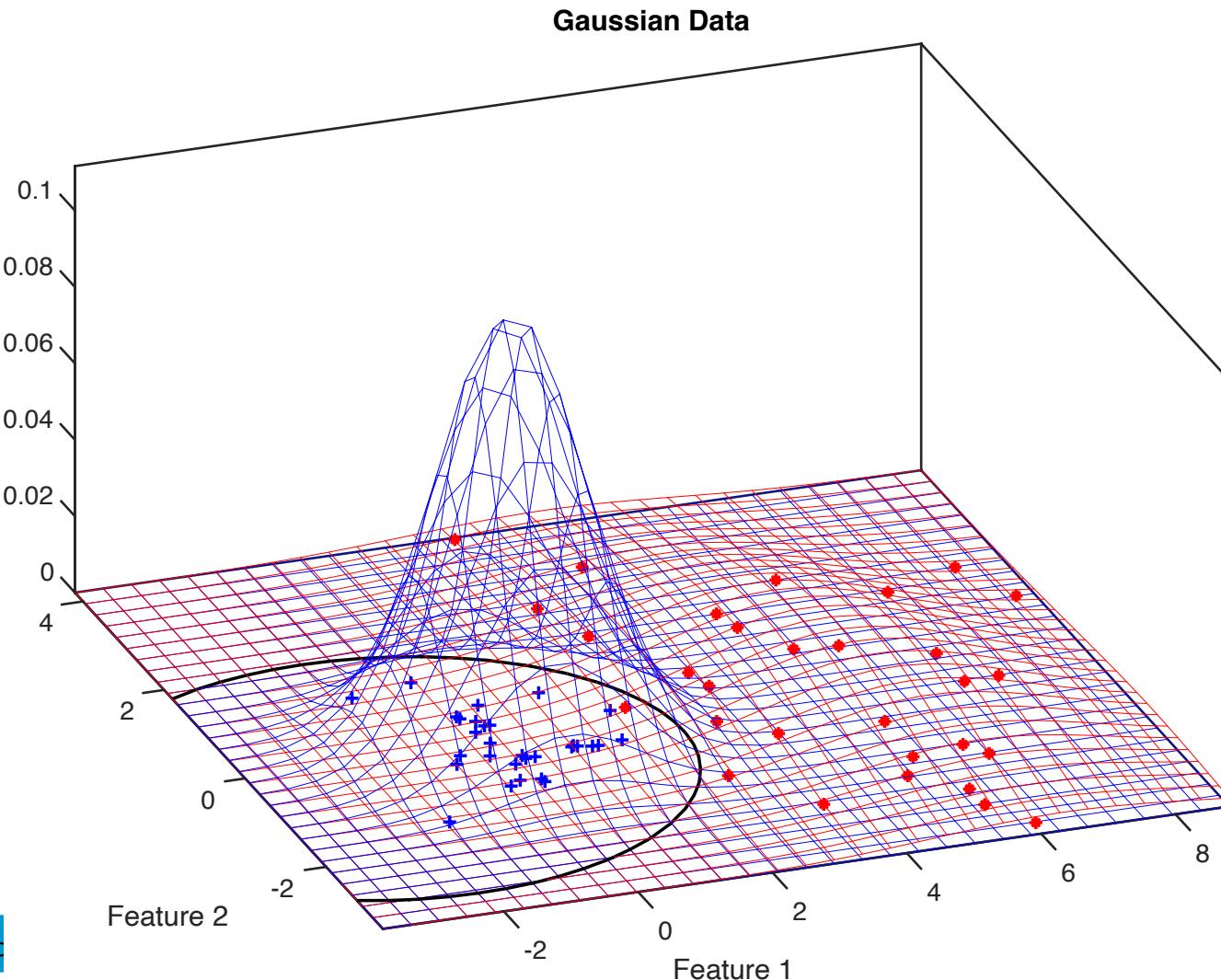
# Quadratic Classifier on Banana Data



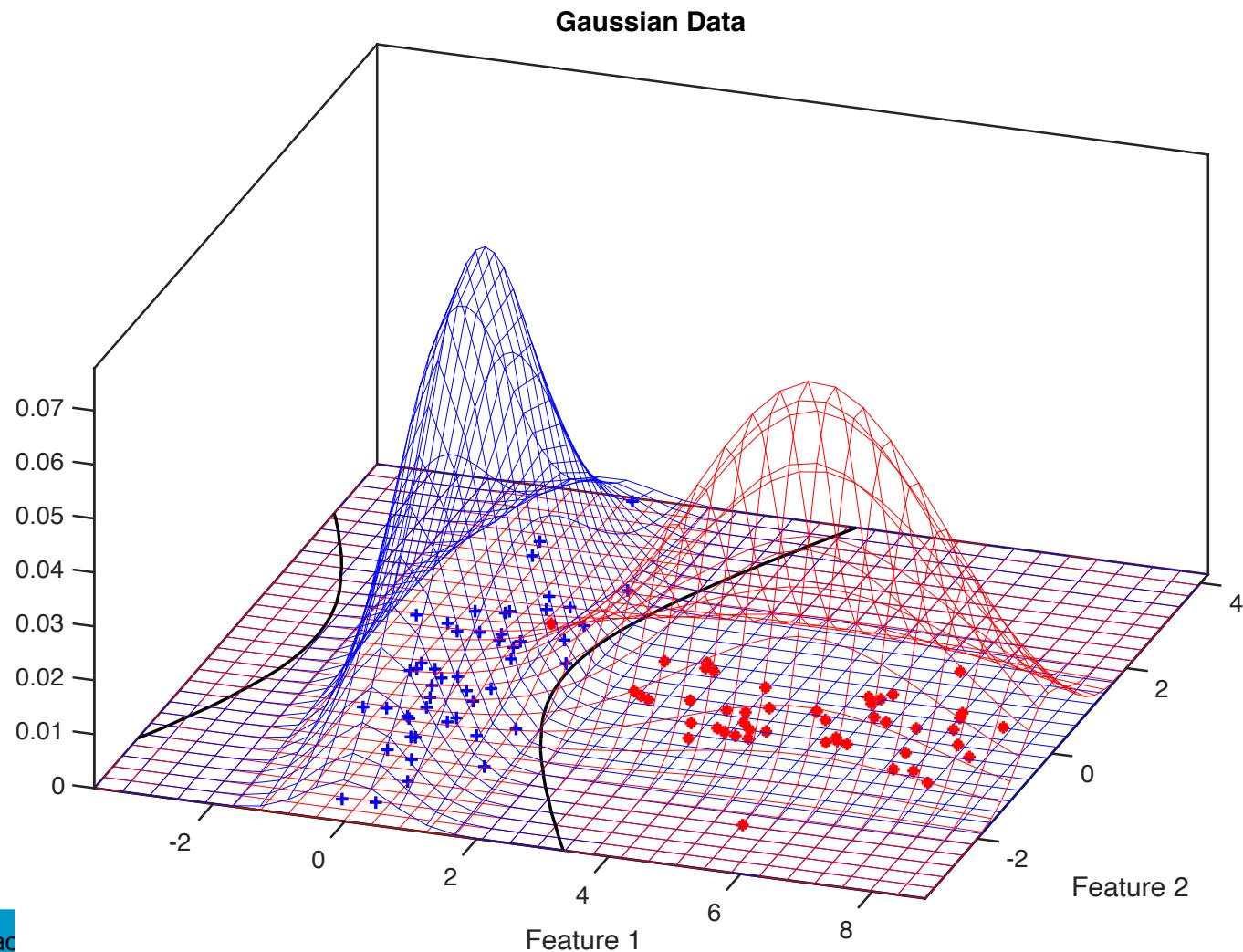
# (Almost) linear decision boundary



# Circular decision boundary



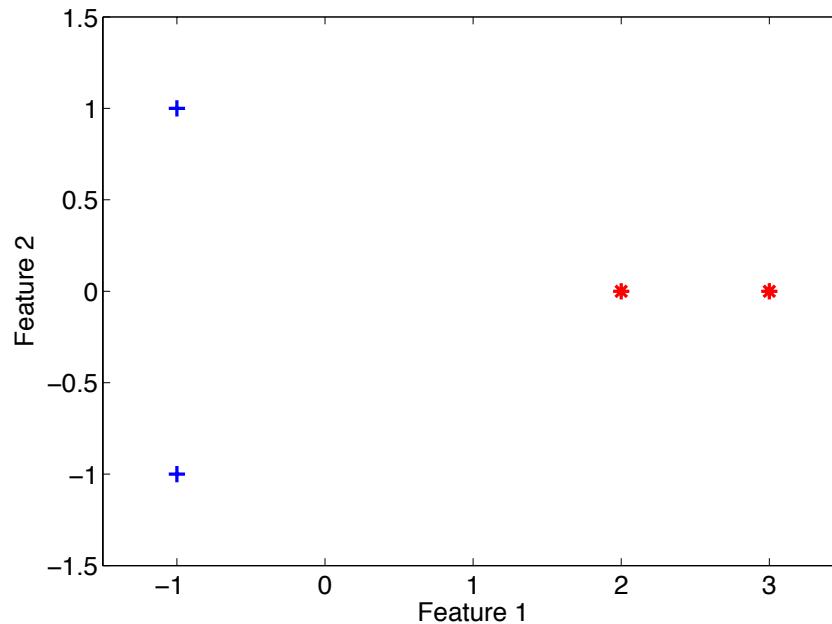
# Hyperbolic decision boundary



# Estimating the covariance matrix

$$X = \begin{bmatrix} -1 & -1 \\ -1 & +1 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



- Mean of class +1?
- Covariance matrix of class +1? And its inverse?

# Estimating the covariance matrix

- Make new, centered, dataset:

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T = \frac{1}{2} \sum_{i=1}^n \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T$$

- Then:

$$\tilde{\mathbf{x}}_1 = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \quad \tilde{\mathbf{x}}_2 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

- And:

$$\tilde{\mathbf{x}}_1 \tilde{\mathbf{x}}_1^T = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}$$

# Estimating the covariance matrix

- The same for

$$\tilde{\mathbf{x}}_2 \tilde{\mathbf{x}}_2^T = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}$$

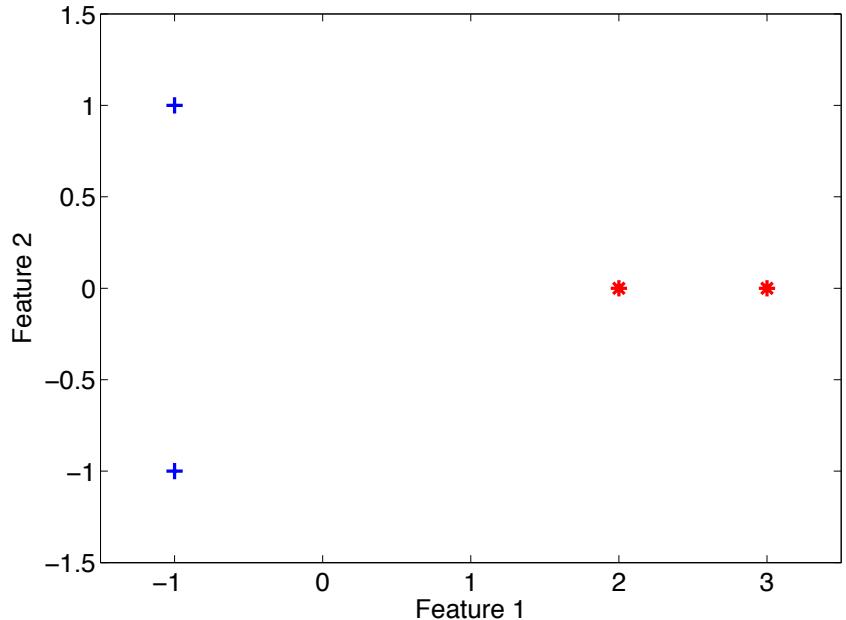
- So in total:

$$\begin{aligned}\hat{\Sigma} &= \frac{1}{2} \sum_{i=1}^n \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T = \frac{1}{2} \left( \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix}\end{aligned}$$

# No inverse...

- One of the variances is 0
- We don't have enough data
- The Gaussian density is not defined

- Now what?



# Estimating Covariance Matrices

- For quadratic classifier need to estimate covariances, e.g., by

$$\hat{\Sigma}_k = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^T$$

for each of the classes

- When there is insufficient data, this covariance matrix cannot be inverted
- Alternative : **average** over all class covariance matrices

$$\hat{\Sigma} = \frac{1}{C} \sum_{k=1}^C \hat{\Sigma}_k$$

# Estimating covariance matrix

- For high-dimensional data, you need many samples to estimate the covariance matrix
- (How many parameters have to be estimated?)
- Simplify:
- Assume all classes have **the same** covariance matrix

# Average Covariance Matrix

- When using averaged covariance, we get

$$g_i(\mathbf{x}) = -\frac{1}{2} \log(\det \hat{\Sigma}) - \frac{1}{2} (\mathbf{x} - \hat{\mu}_i)^T \hat{\Sigma}^{-1} (\mathbf{x} - \hat{\mu}_i) + \log p(y_i)$$

- I.e., first and quadratic term are always the same
- We end up with:

$$g_i(\mathbf{x}) = -\frac{1}{2} \hat{\mu}_i^T \hat{\Sigma}^{-1} \hat{\mu}_i + \hat{\mu}_i^T \hat{\Sigma}^{-1} \mathbf{x} + \log p(y_i)$$

- This classifier is linear: linear normal-based classifier

# The Two-Class Case: LDA

- Define the discriminant

$$f(\mathbf{x}) = \log p(y_1|\mathbf{x}) - \log p(y_2|\mathbf{x})$$

- We get

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

with

$$\mathbf{w} = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$$

$$w_0 = \frac{1}{2}\hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log \frac{p(y_1)}{p(y_2)}$$

# Linear discriminant

- Let us assume that the decision boundary can be described by:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = 0$$

- Weight vector  $\mathbf{w}$  and bias term (offset)  $w_0$
- Classify:

$$\text{classify } \mathbf{x} \text{ to } \begin{cases} y_1 & \text{if } \mathbf{w}^T \mathbf{x} + w_0 \geq 0 \\ y_2 & \text{if } \mathbf{w}^T \mathbf{x} + w_0 < 0 \end{cases}$$

- In the most general sense, this is called linear discriminant analysis

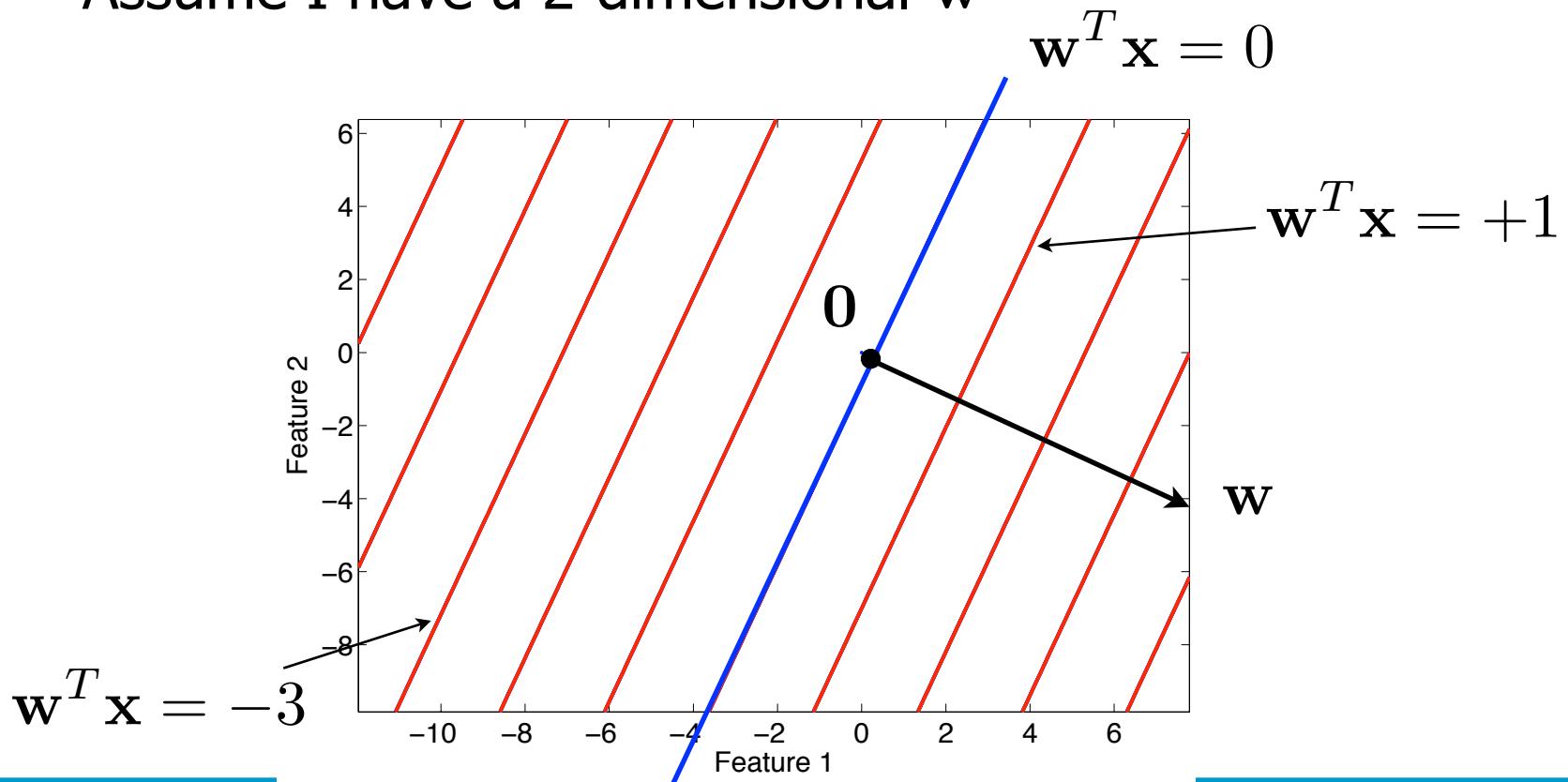
# Linear function?

- What does  $\mathbf{w}^T \mathbf{x}$  mean?

# Linear function?

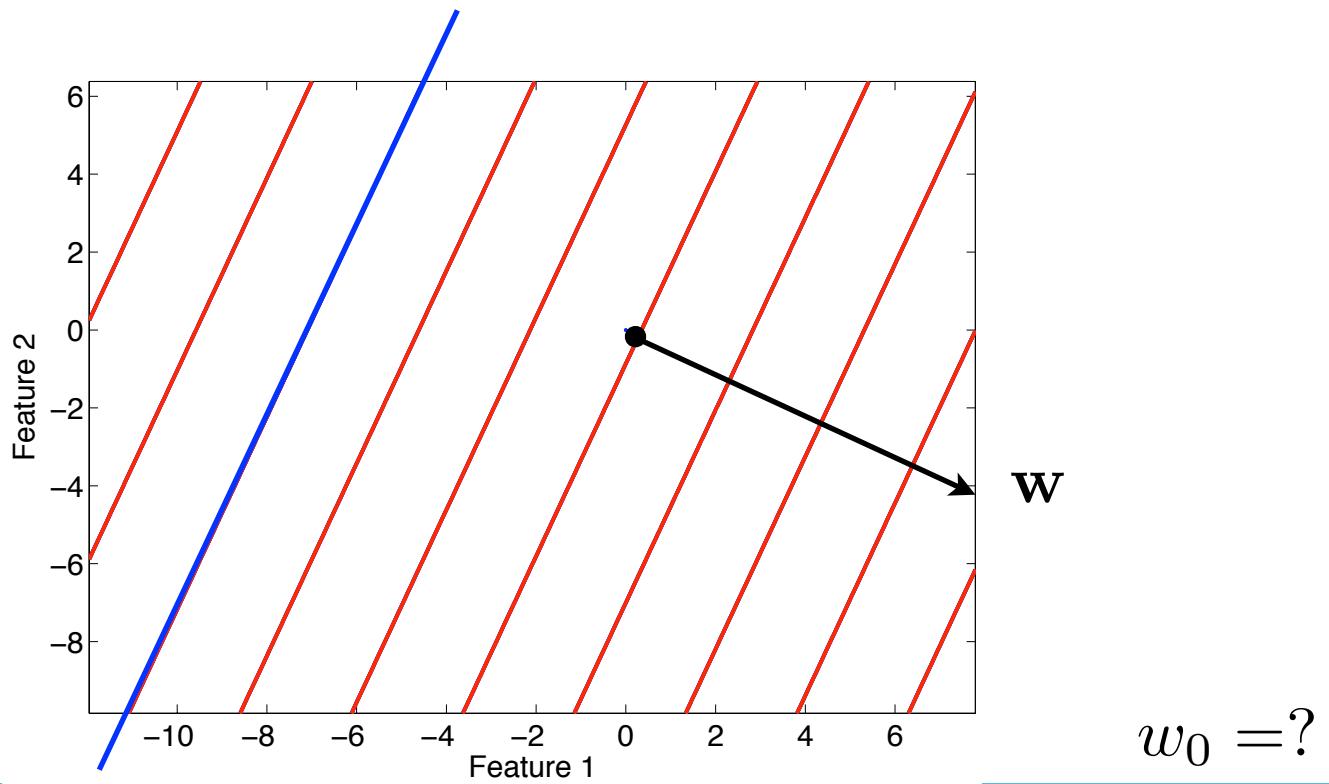
- What does  $\mathbf{w}^T \mathbf{x}$  mean?

Assume I have a 2-dimensional  $\mathbf{w}$

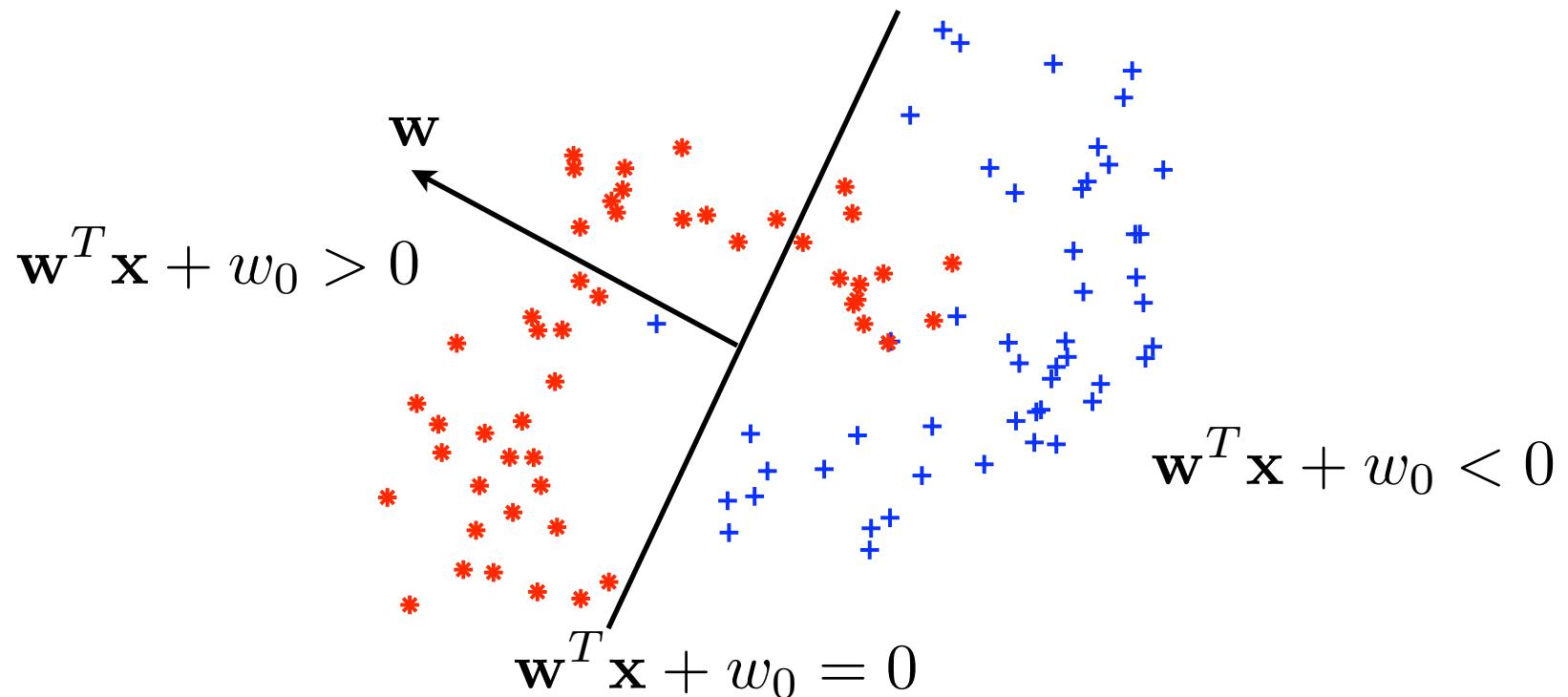


# Linear function, the bias

- What does  $\mathbf{w}^T \mathbf{x} + w_0$  mean?  
Assume I have  $\mathbf{w}^T \mathbf{x} + w_0 = 0$



# Linear discriminant



- Classifier is a linear function of the features
- The classification depends if the weighted sum of the features is above or below 0

# Incorporate the bias term

- Quite often you see  $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} > 0$   
instead of  $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 > 0$
- No problem, if you (re-)define the feature vector as:

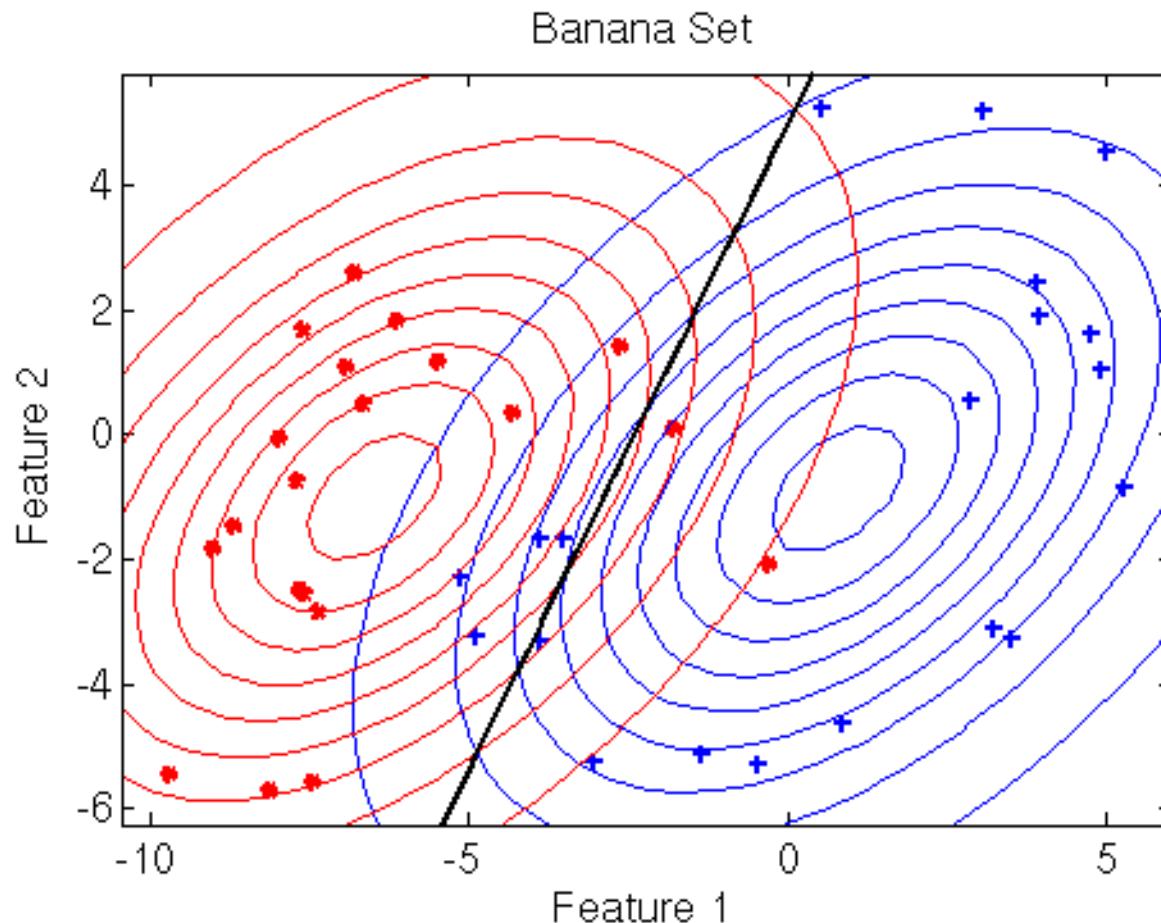
(homogeneous  
coordinates)

- Then:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

$$g(\mathbf{x}) = [\mathbf{w}^T \ w_0] \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

# Linear Classifier on Banana Data



# No Estimated Covariance Matrix

- In some cases even a full averaged covariance matrix is too much to estimate
- As simplification one could assume that all features have the same variance, and are uncorrelated :

$$\hat{\Sigma} = \sigma^2 \mathbb{I}$$

i.e. the classes are circular.

- Obviously, decision rule becomes even simpler :

$$g_i(\mathbf{x}) = -\frac{1}{\sigma^2} \left( \frac{1}{2} \hat{\mu}_i^T \hat{\mu}_i - \hat{\mu}_i^T \mathbf{x} \right) + \log(p(y_i))$$

# Nearest Mean Classifier

- Define the discriminant :

$$f(\mathbf{x}) = \log p(y_1|\mathbf{x}) - \log p(y_2|\mathbf{x})$$

- We get

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

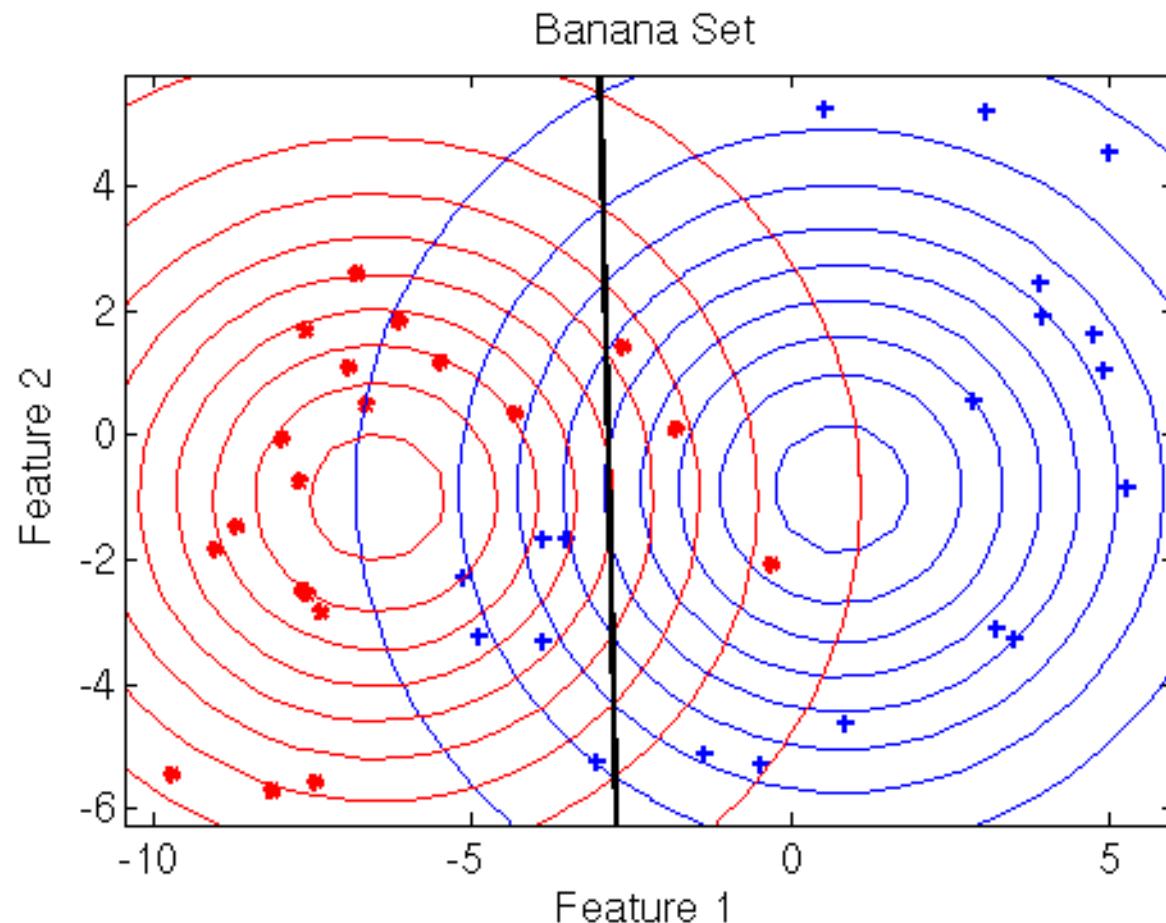
with

$$\mathbf{w} = \hat{\mu}_1 - \hat{\mu}_2$$

$$w_0 = \frac{1}{2} \hat{\mu}_2^T \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_1^T \hat{\mu}_1 + \sigma^2 \log \frac{p(y_1)}{p(y_2)}$$

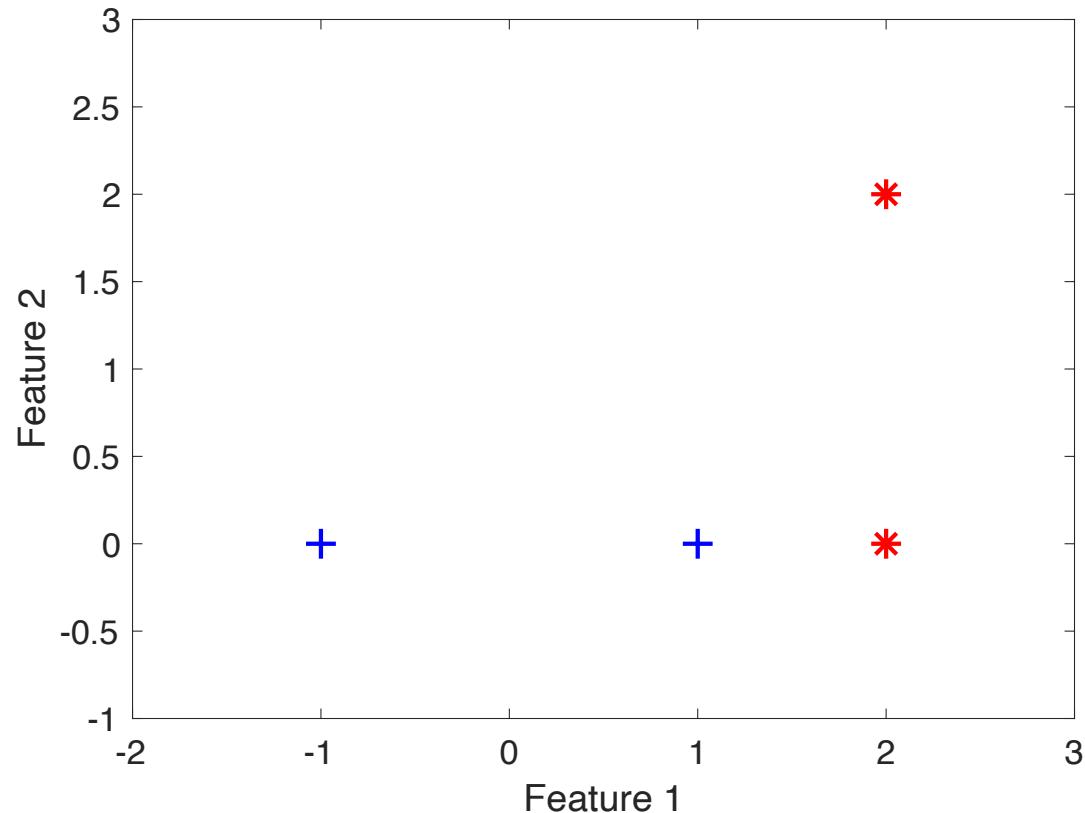
- Again a linear classifier, but it only uses distance to the mean of each of the classes : nearest mean classifier

# Nearest Mean on Banana Data



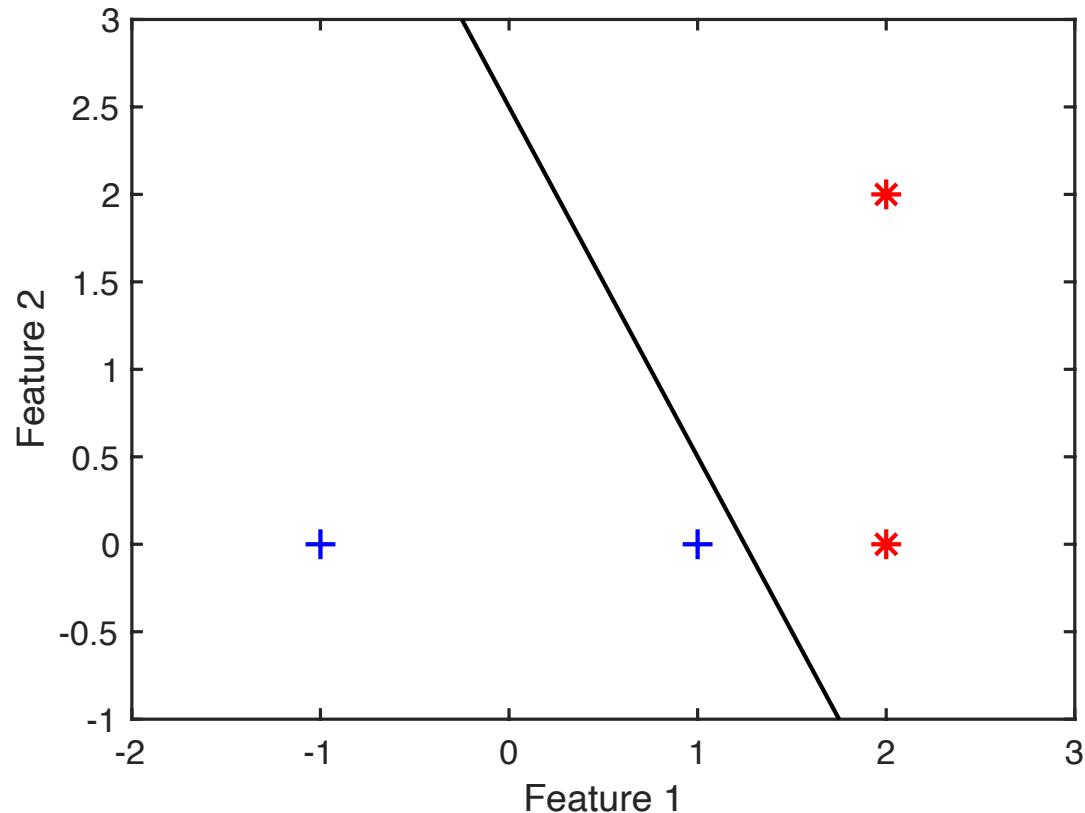
# Nearest mean on simple data

- How does the NMC look like for:



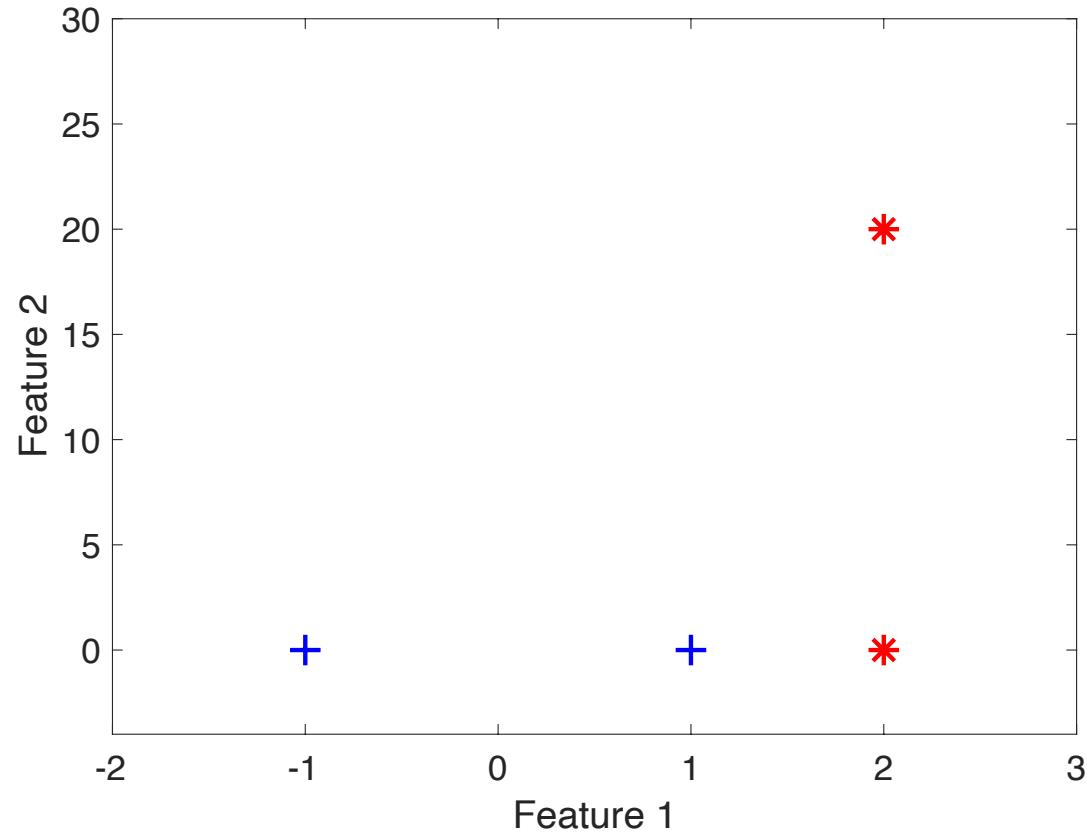
# Nearest mean on simple data

- How does the NMC look like for:



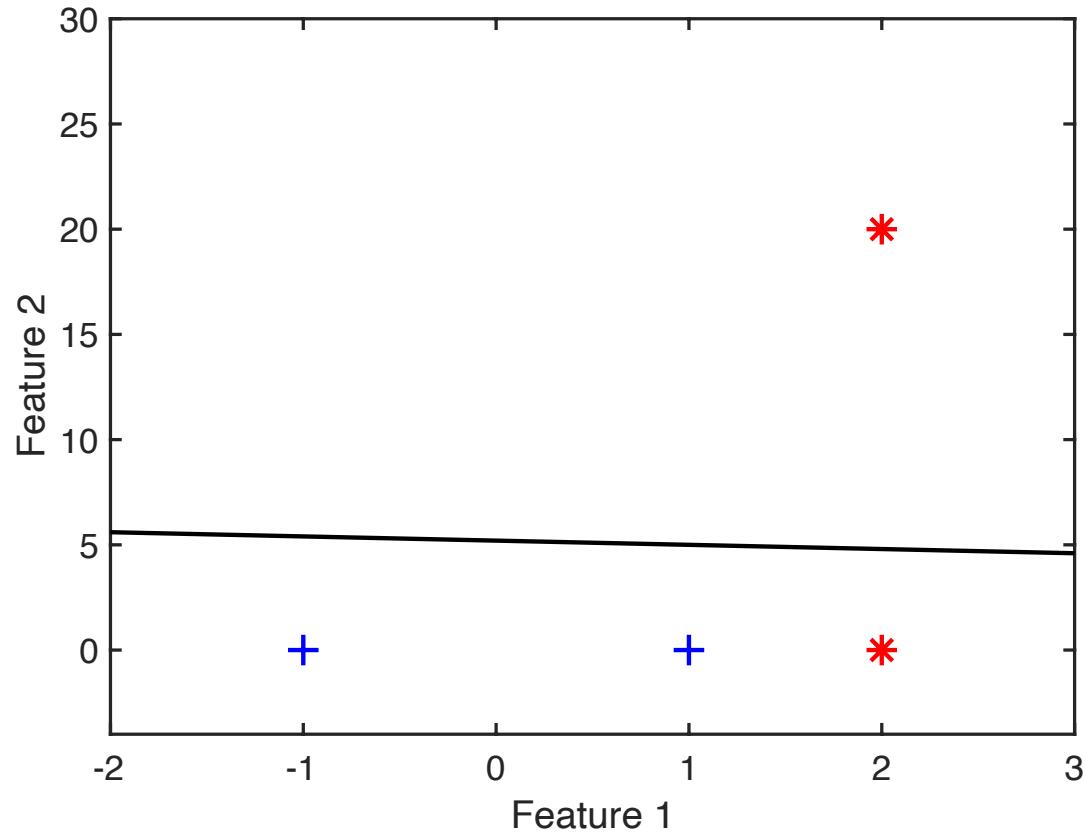
# Nearest mean on simple data

- And now?



# Nearest mean on simple data

- And now?



# Feature scaling

- When a classifier depends on (euclidean) distances, scaling of the features matter
- Changing the scaling can improve/deteriorate the classifier: check the variances of all features
- If you don't know anything, rescale your features beforehand
- Fairly standard is zero-mean, unit-variance scaling:

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

# Concluding Remarks

- Using plug-in Bayes' rule with normal distribution for every class can give rise to different classifiers
  - Separate mean and covariance matrix per class gives the **quadratic classifier**
  - Separate mean, equal covariance matrix per class gives the **linear classifier**
  - Separate mean, identity covariance matrix per class gives the **nearest mean classifier**
- More flexible classifier needs more training data
- Simple classifiers still perform quite well in practice

# Concluding Remarks

- This curse of dimensionality is really a nasty one
  - One of the major issues in the pattern recognition course is to learn ways to detect and understand it, and possibly deal with it

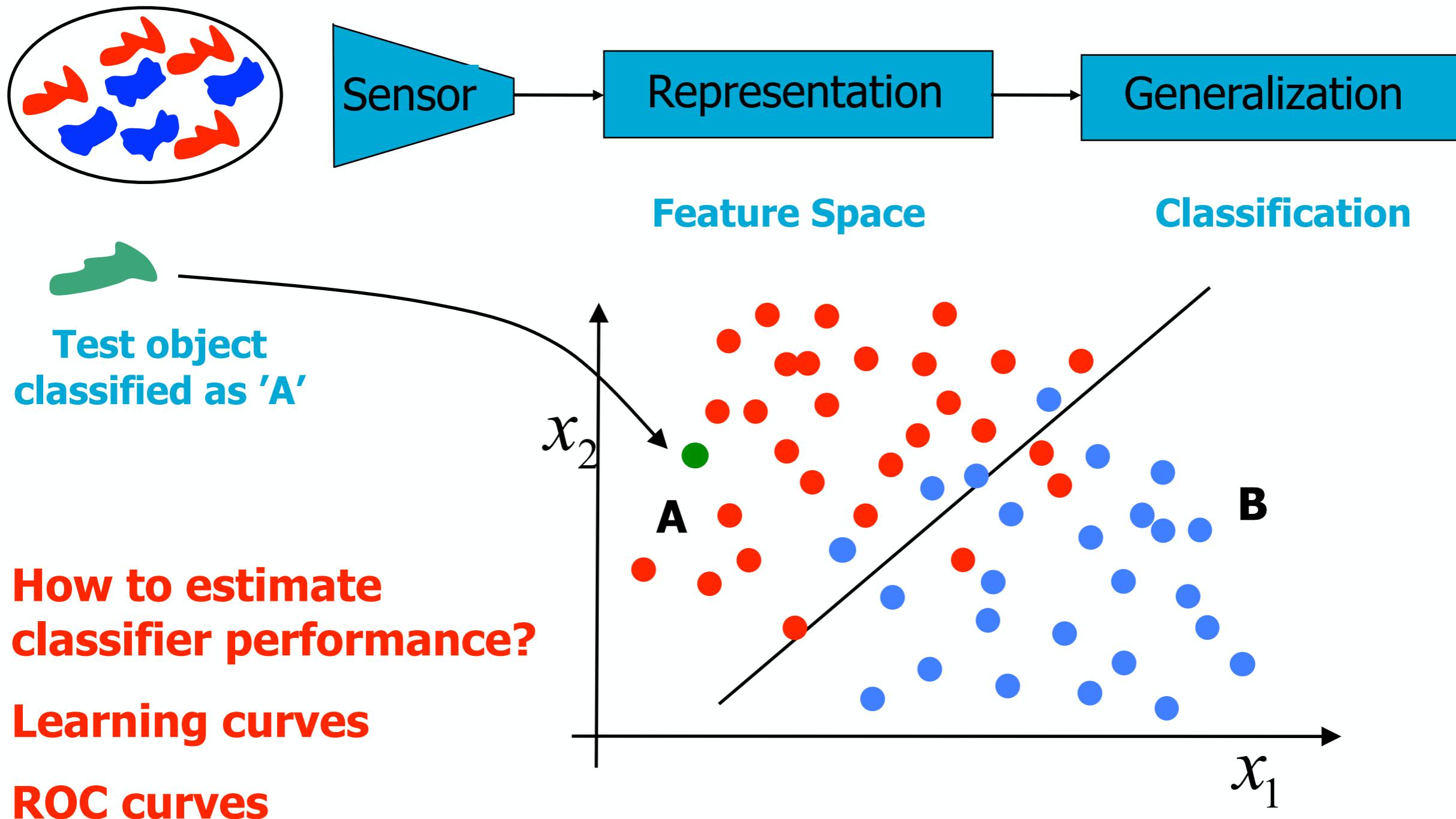
# One may wonder...

- Is ML estimation optimal for classification?
- So how many free parameters are there really for a linear classifier?
- When underlying class distributions are truly Gaussian, do our Gaussian-based classifiers result in Bayes optimal classifiers?

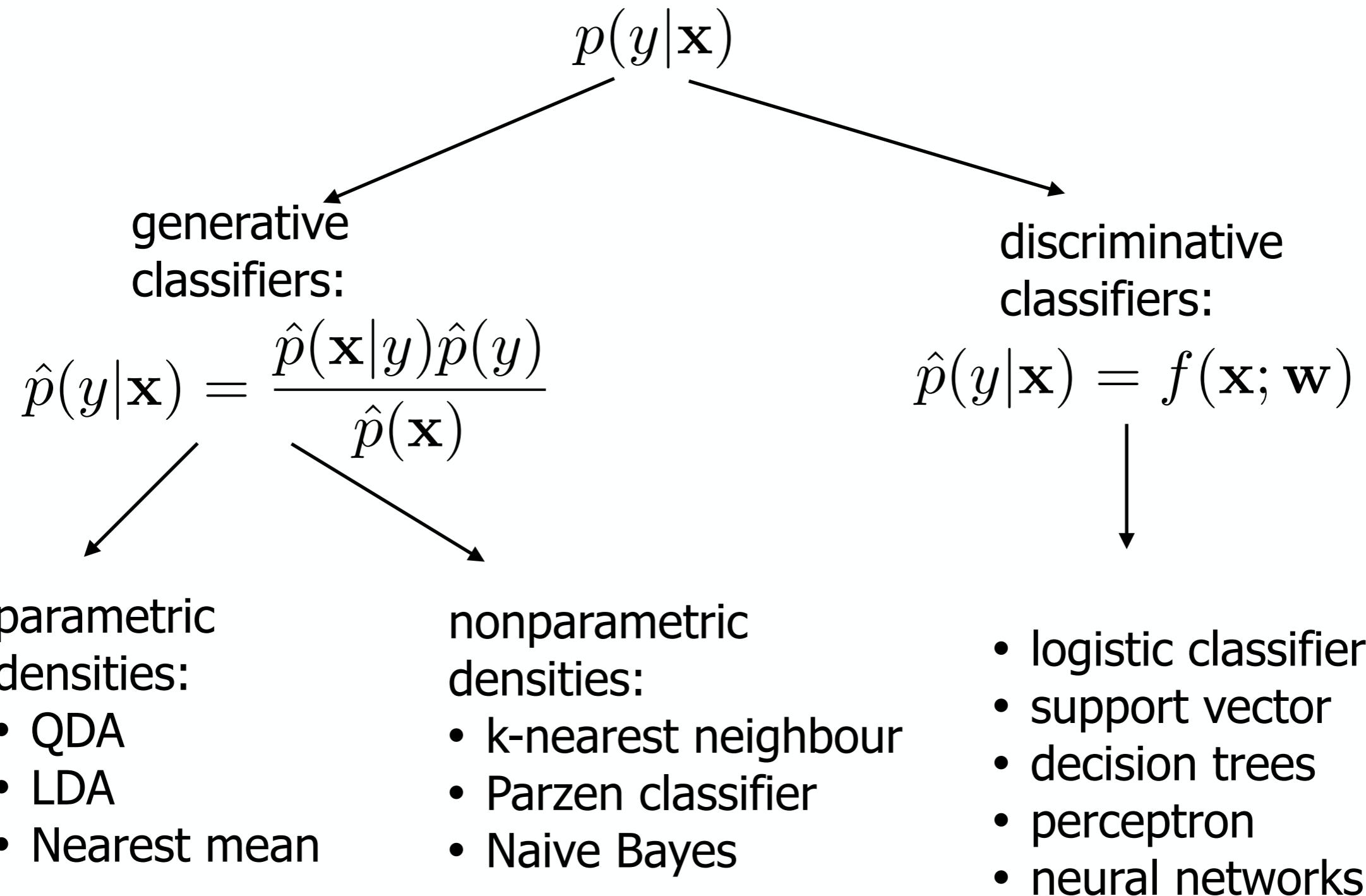
# After this week you should be able to:

- explain how you obtain a classifier using a Gaussian (multivariate) distribution for each class
  - implement a simple univariate classifier in Python
  - explain what the 'curse of dimensionality' is
- 
- explain the advantages and disadvantages are of the Quadratic classifier, the LDA and the nearest mean classifier
  - identify when scaling of the features is important and how to cope with feature scaling

# Classifier evaluation

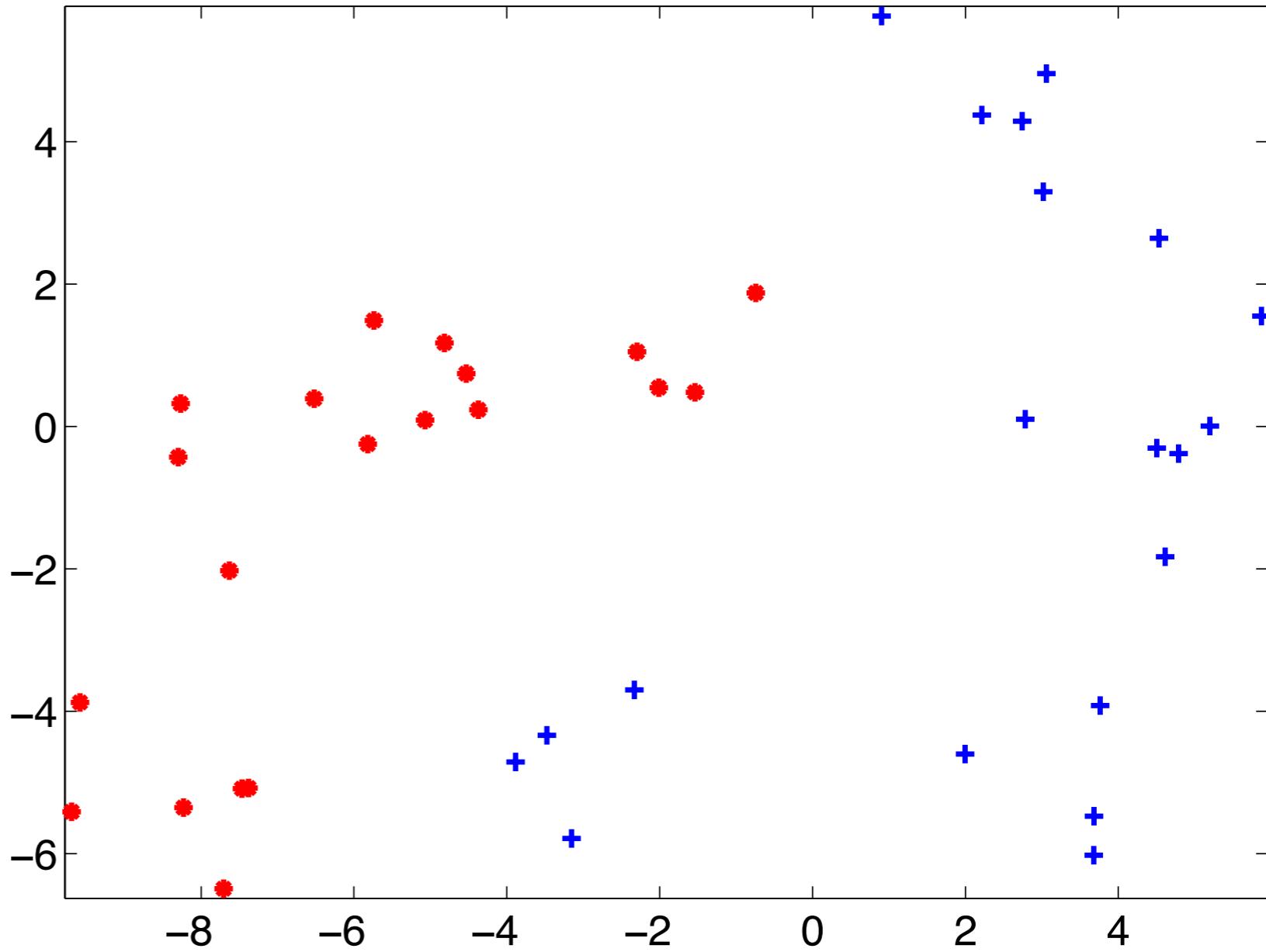


# Recap classifiers



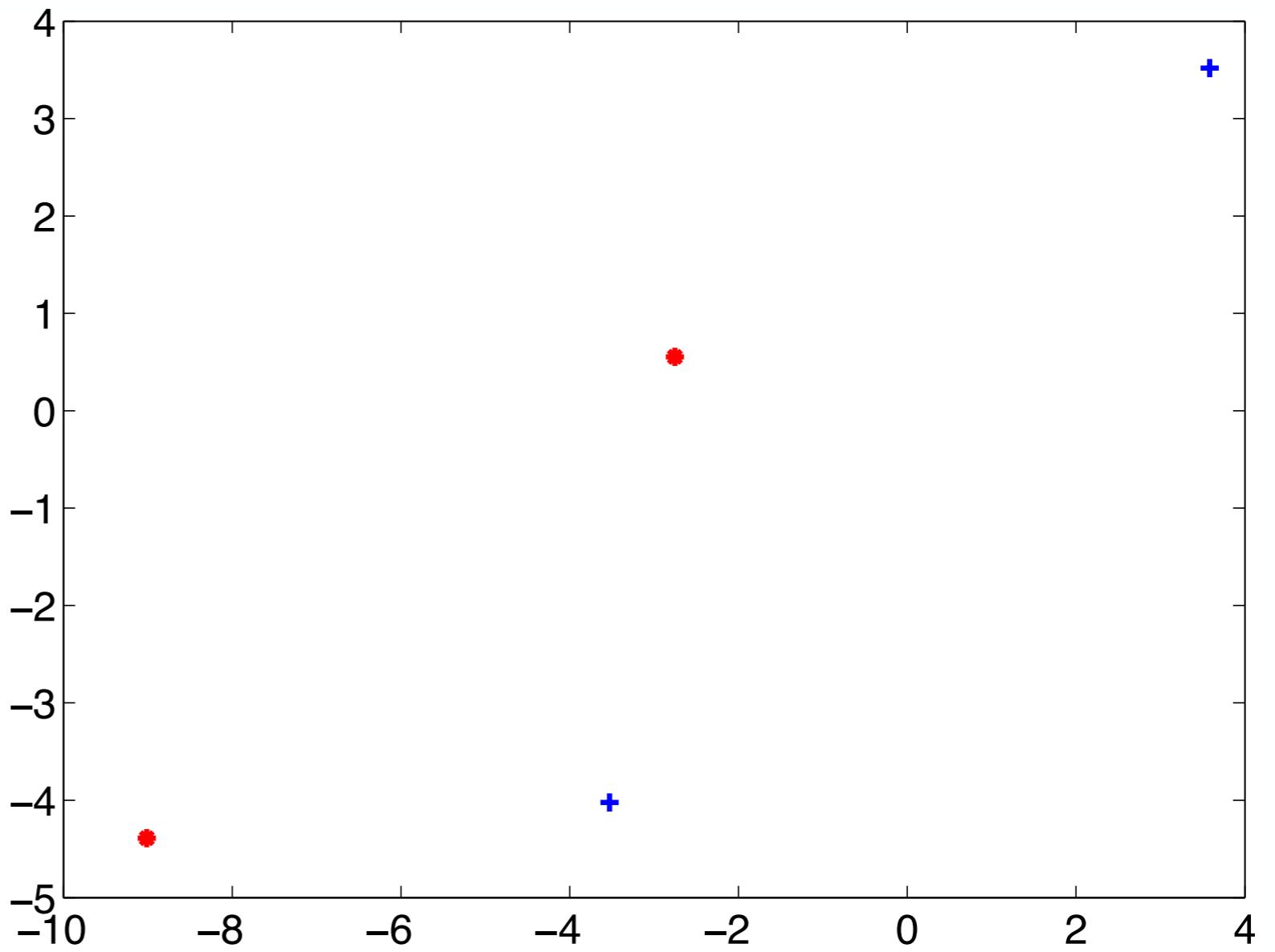
# What classifier to use here?

- NMC
- LDA
- QDA
- k-NN
- Parzen classifier
- Naive Bayes
- Logistic
- Support vector
- Decision tree
- Neural network
- ...



# What classifier to use here?

- NMC
- LDA
- QDA
- k-NN
- Parzen classifier
- Naive Bayes
- Logistic
- Support vector
- Decision tree
- Neural network
- ...



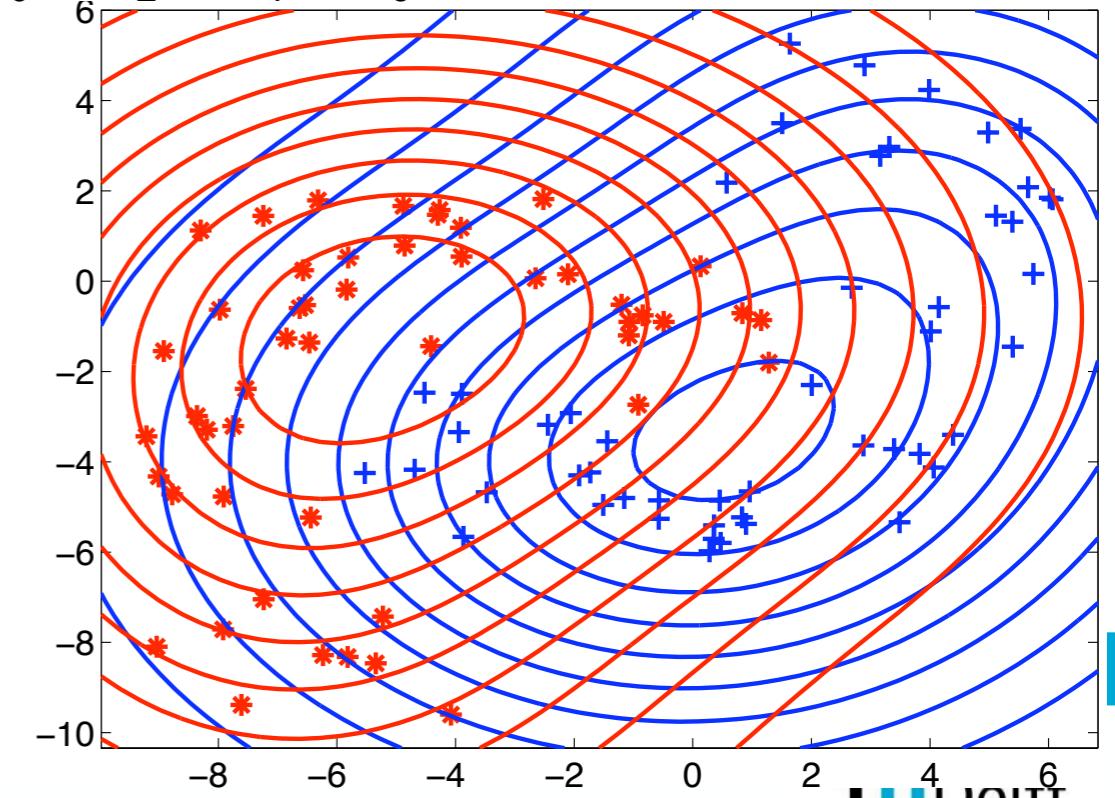
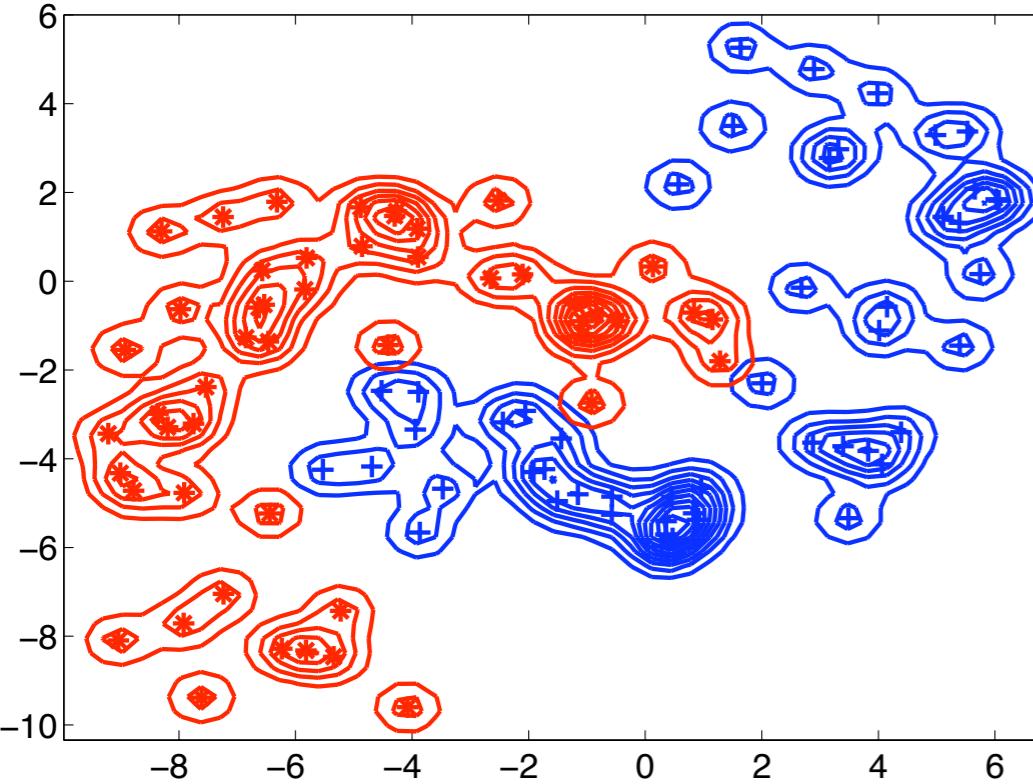
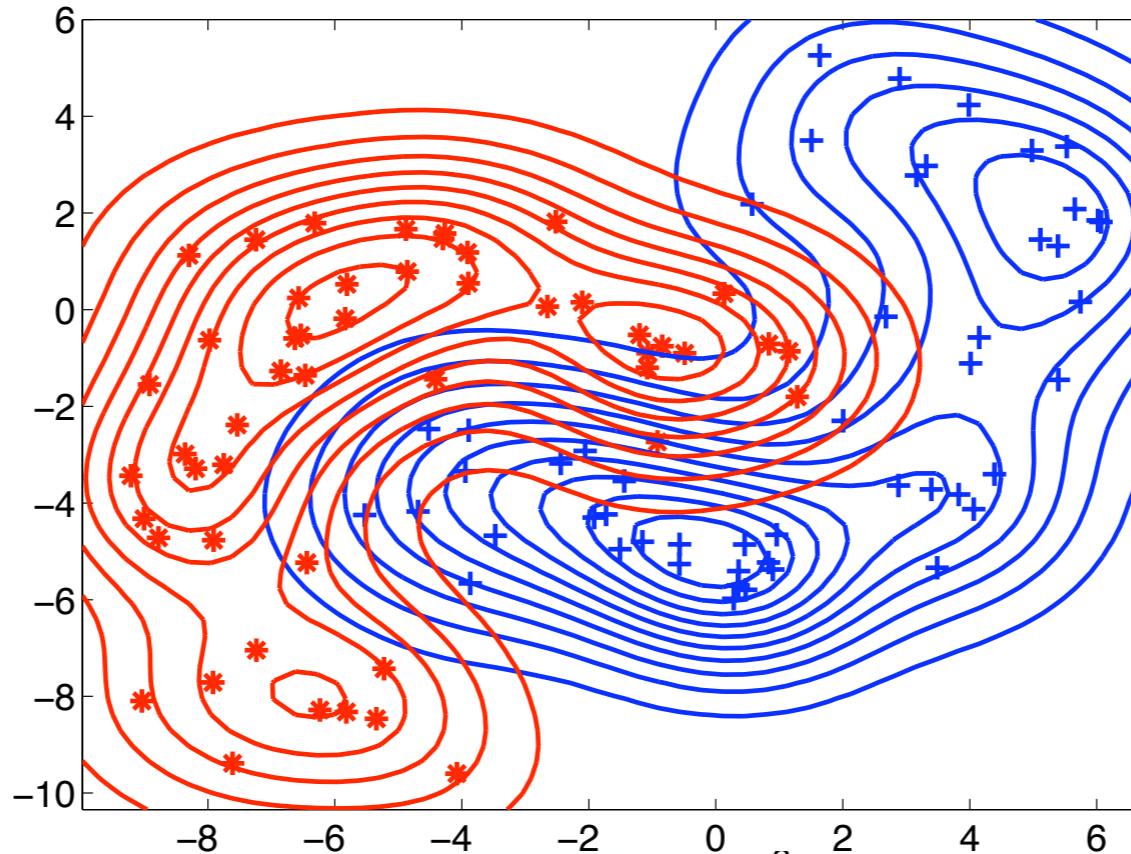
# What classifier to use here?

- Loaded our famous Iris dataset
- 150 objects in 4D
- What now?

iris	=	sepal l.	sepal w.	petal l.	petal w.
		5.1000	3.5000	1.4000	0.2000
		4.9000	3.0000	1.4000	0.2000
		4.7000	3.2000	1.3000	0.2000
		4.6000	3.1000	1.5000	0.2000
		5.0000	3.6000	1.4000	0.2000
		5.4000	3.9000	1.7000	0.4000
		4.6000	3.4000	1.4000	0.3000
		5.0000	3.4000	1.5000	0.2000
		4.4000	2.9000	1.4000	0.2000
		4.9000	3.1000	1.5000	0.1000
		5.4000	3.7000	1.5000	0.2000
		4.8000	3.4000	1.6000	0.2000
		4.8000	3.0000	1.4000	0.1000
		4.3000	3.0000	1.1000	0.1000
		5.8000	4.0000	1.2000	0.2000
		5.7000	4.4000	1.5000	0.4000
		5.4000	3.9000	1.3000	0.4000

# Optimal width for Parzen classifier

- What is the optimal  $h$ ?



# What classifier to use?

- If cannot visualise high-dimensional data:  
how to choose the classifier?  
or:  
how to choose hyperparameters of a classifier?
- Need some objective criterion!
- Typically: classification performance/error

# True or false?

- The classification error of the training set is a good measure of the true classification error  
TRUE? FALSE?
  - A good estimate of the actual, true, error is all we are after  
TRUE? FALSE?

# What classifier to use?

- If cannot visualise high-dimensional data:  
how to choose the classifier?  
or:  
how to choose hyperparameters of a classifier?
- Need some objective criterion!
- Typically: classification performance/error
- Test it on **independent** data
- For simplicity we assume now that classification error is good enough

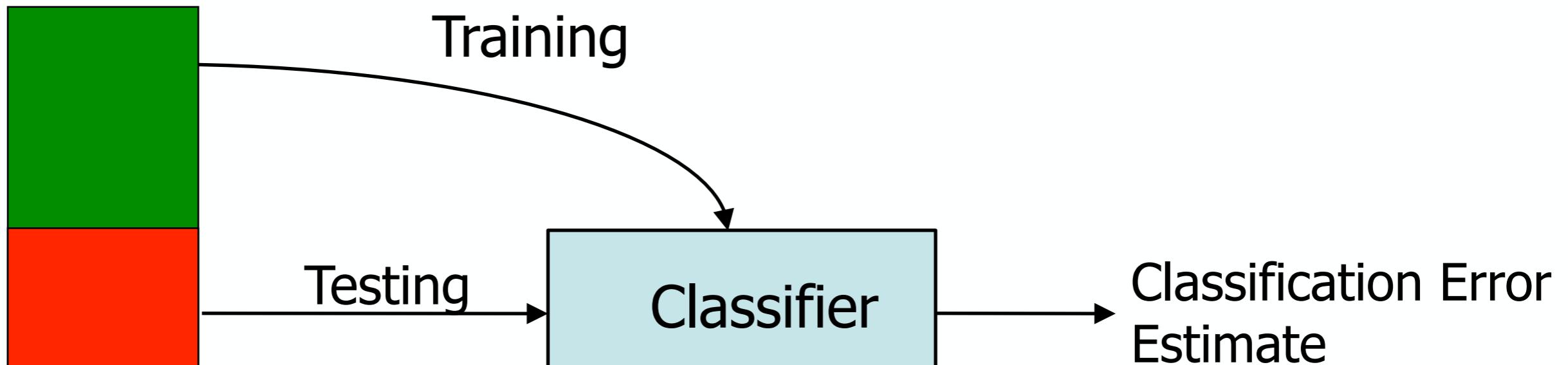
# Classifier evaluation

- This lecture:
  - Train and test set
  - Bootstrapping, cross-validation, leave-one-out
  - Learning curve
  - Classifier complexity
  - Bias-variance tradeoff
  - Confusion matrices
  - ROC curve
  - (Reject curve)

# Material/literature

- Chapter 19 “Design and Analysis of Machine Learning experiments” in ‘Introduction to Machine Learning’ by E. Alpaydin
- Ebook available at [library.tudelft.nl](http://library.tudelft.nl)
- Some sections are not discussed (19.8 and further)

# Error Estimation by Test Set



Design Set

Other training set → other classifier

Other test set → other error estimate  $\hat{\epsilon}$

# Some Error Estimate Variability

- Error is a sum of iid Bernoulli random variables:

$$\hat{\varepsilon} = \frac{1}{N} \sum_{i=1}^N Z_i$$

where

$$Z_i = \begin{cases} 0 & \text{if } \mathbf{x}_i \text{ is correctly classified} \\ 1 & \text{if } \mathbf{x}_i \text{ is incorrectly classified} \end{cases}$$

- You can show that the variance is:

$$\sigma_{\hat{\varepsilon}}^2 = \text{Var}(\hat{\varepsilon} | \text{test set size } N) = \frac{\hat{\varepsilon}(1 - \hat{\varepsilon})}{N}$$

# Some Error Estimate Variability

- Error is a sum of Bernoulli random variables:

$$\hat{\varepsilon} = \frac{1}{N} \sum_{i=1}^N Z_i$$

- When:  $\sigma_{\hat{\varepsilon}}^2 = \text{Var}(\hat{\varepsilon} | \text{test set size } N) = \frac{\hat{\varepsilon}(1 - \hat{\varepsilon})}{N}$

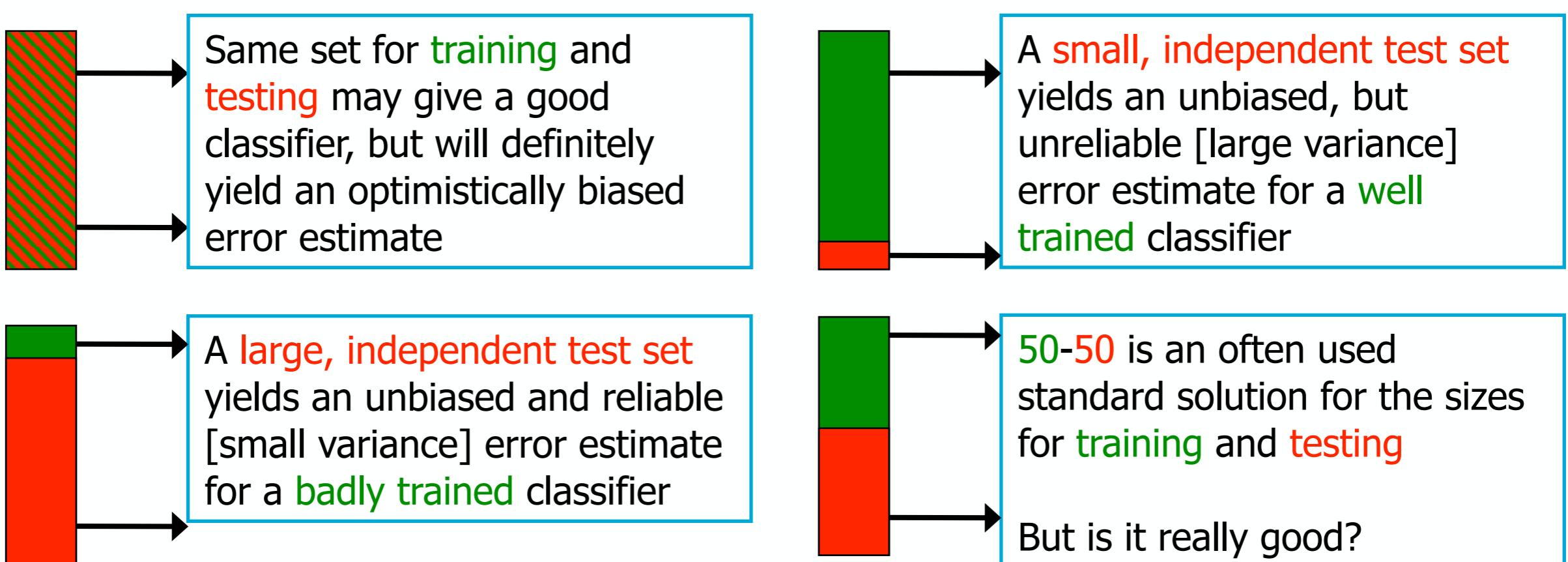
you can compute the standard deviation for different sample sizes and errors:

$$\sigma_{\hat{\varepsilon}} = \sqrt{\frac{\hat{\varepsilon}(1 - \hat{\varepsilon})}{N}}$$

$N$	$\hat{\varepsilon}$	0.01	0.03	0.1
10	0.031	0.054	0.095	
100	0.010	0.017	0.030	
1000	0.003	0.005	0.0095	

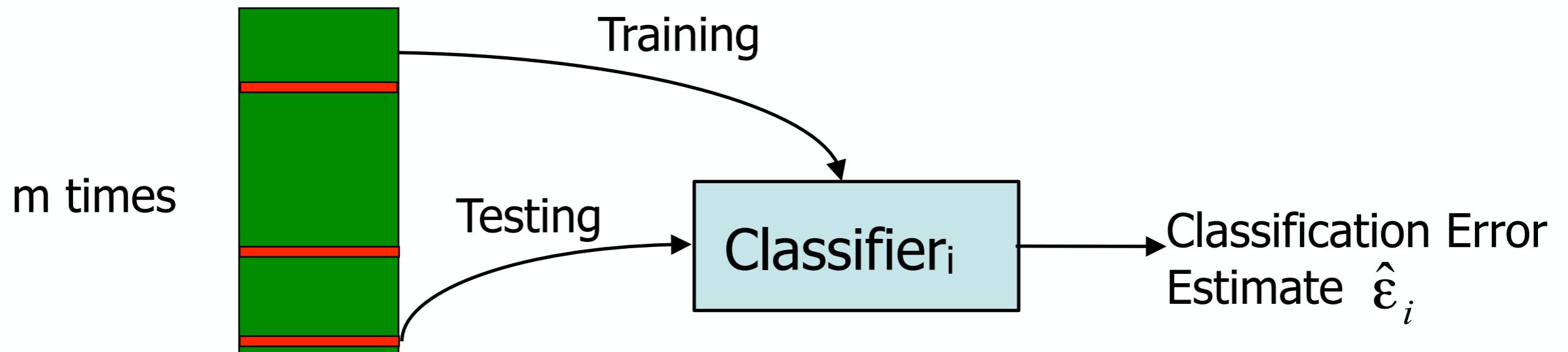
# Training Set Size vs. Test Set Size

- Large training set -> good classifiers
- Large test set -> reliable, unbiased error estimate
- In practice often just a single design set is given



# Bootstrapping

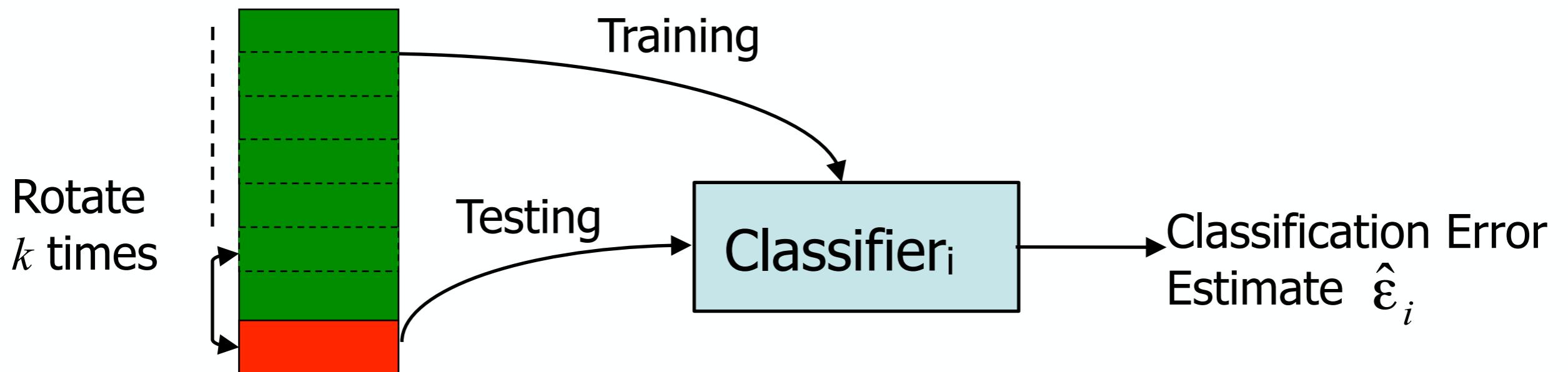
- Randomly draw N objects from N objects (**with replacement**)



$$\hat{\varepsilon} = \frac{1}{m} \sum_i \hat{\varepsilon}_i$$

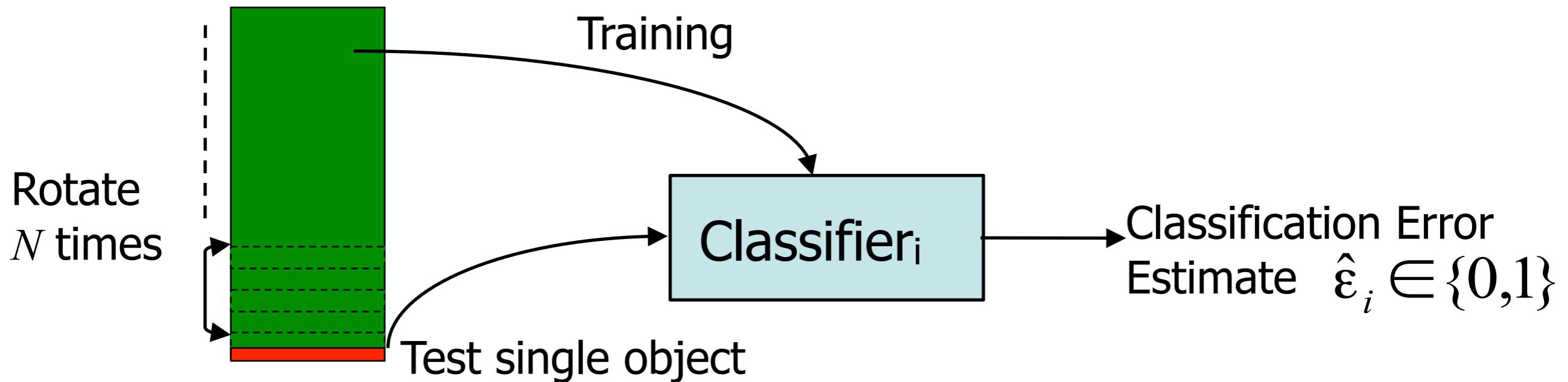
- Left-over objects are used for testing
- Repeat m times

# k-fold cross validation



$$\hat{\epsilon} = \frac{1}{k} \sum_i \hat{\epsilon}_i$$

# Leave-one-out Procedure



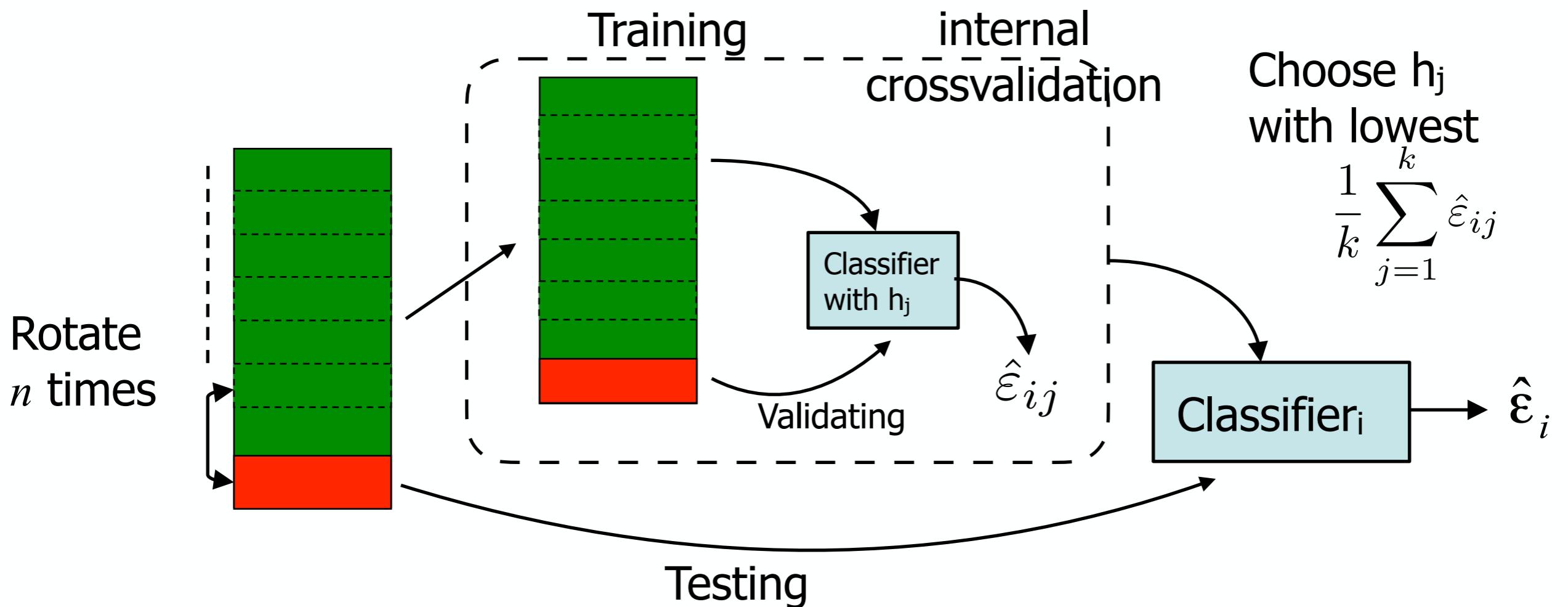
$$\hat{\varepsilon} = \frac{1}{N} \sum_i \hat{\varepsilon}_i$$

# Optimising 'hyperparameters'

- Machine learning methods often have 'hyperparameters'
- Parzen density estimator: width parameter  $h$
- k-nearest neighbour: number of neighbours  $k$
- Decision trees: pruning method, stopping criterion
- Neural networks: architecture, learning rate, ...
- DON'T optimise these numbers by looking at the test set!  
Then you're **CHEATING!**

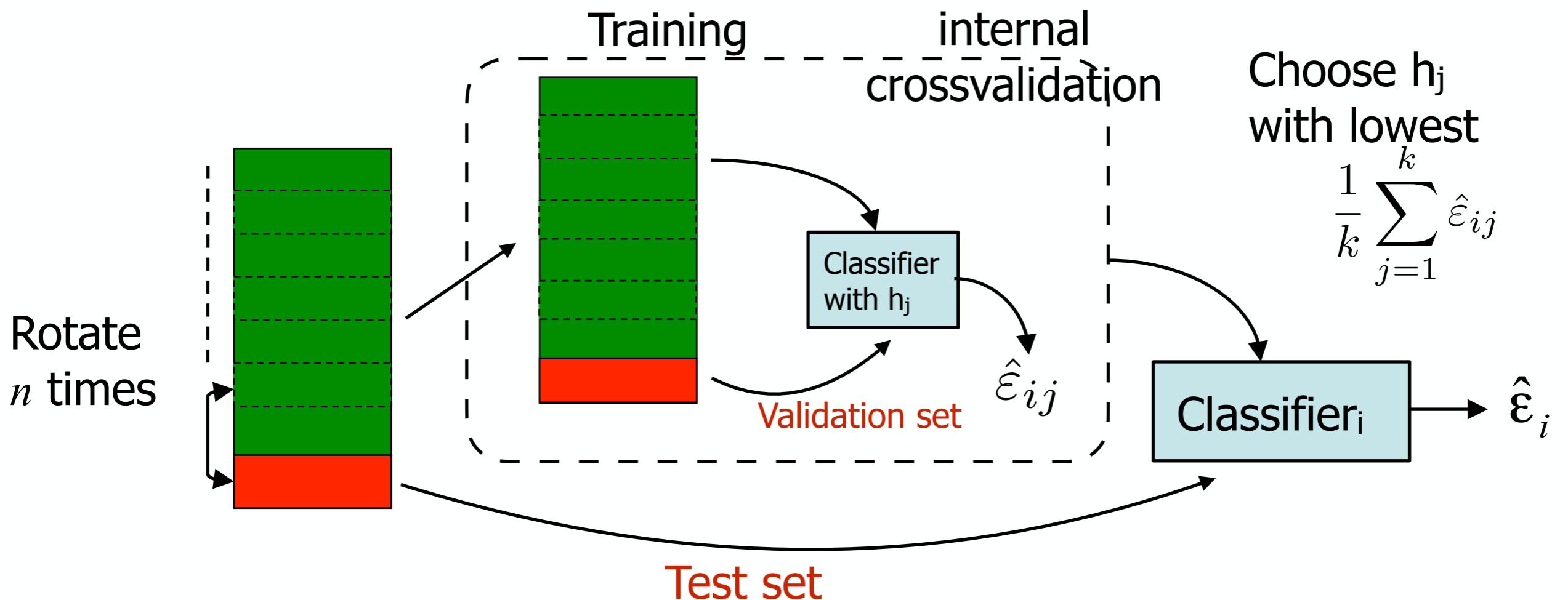
# Double cross-validation

- To optimise hyperparameters, do cross-validation **inside** another cross-validation:



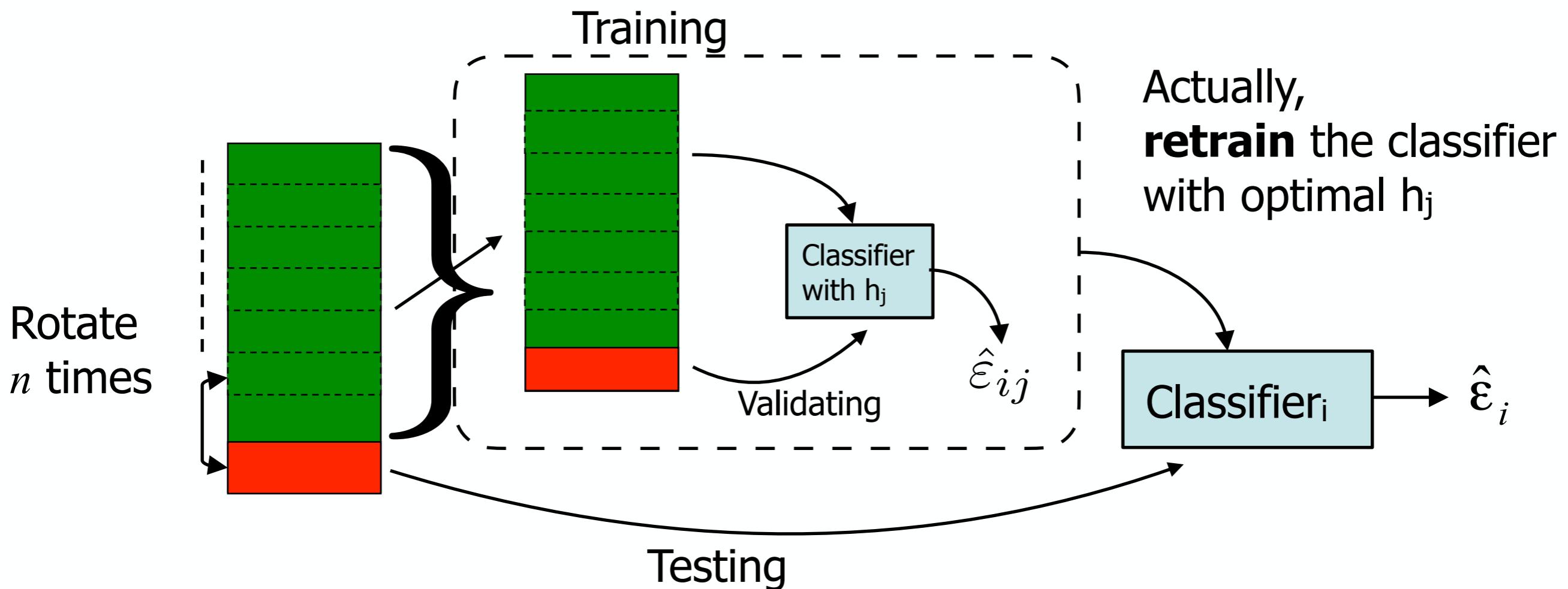
# Double cross-validation

- To optimise hyperparameters, do cross-validation **inside** another cross-validation:

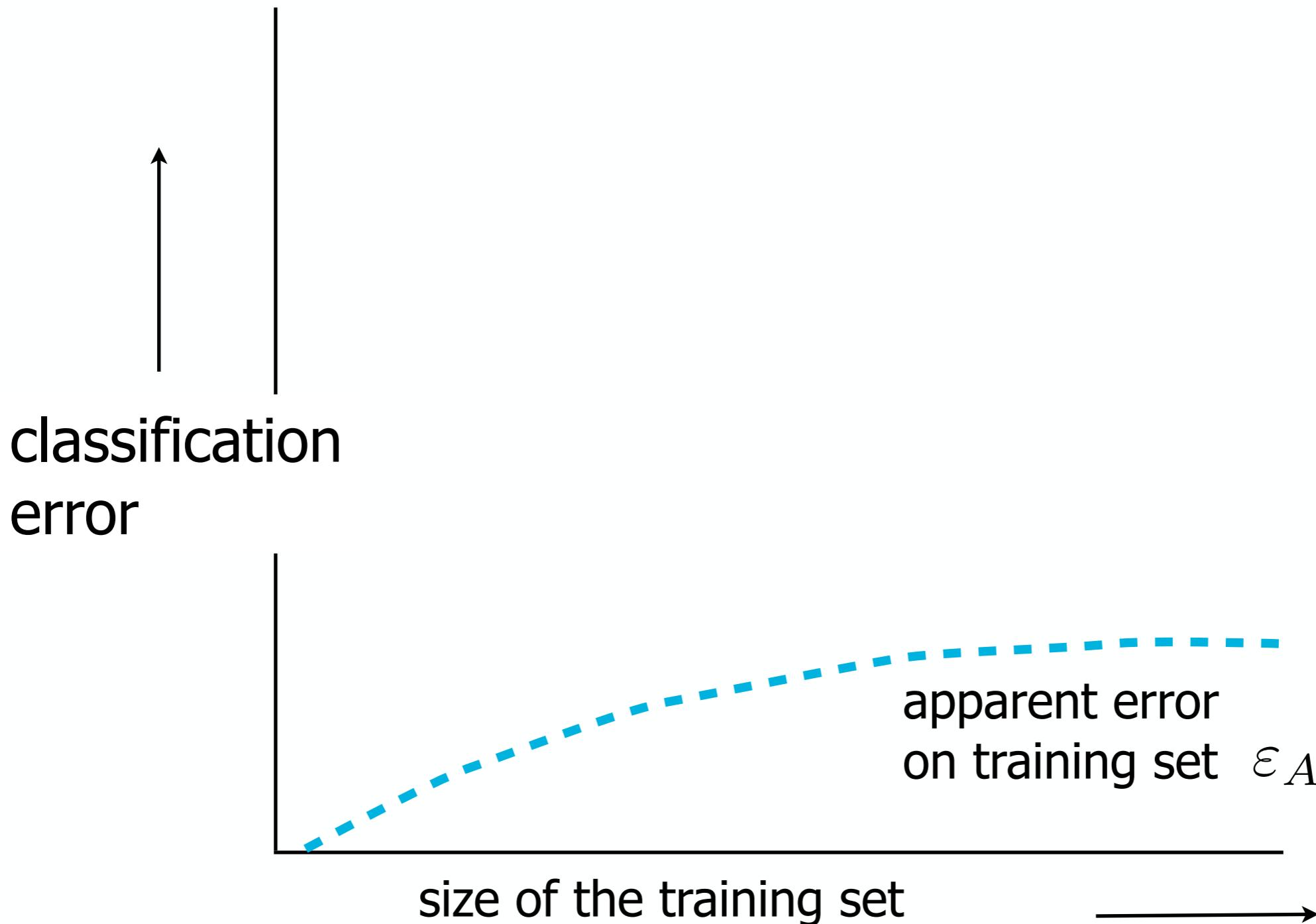


# Double cross-validation

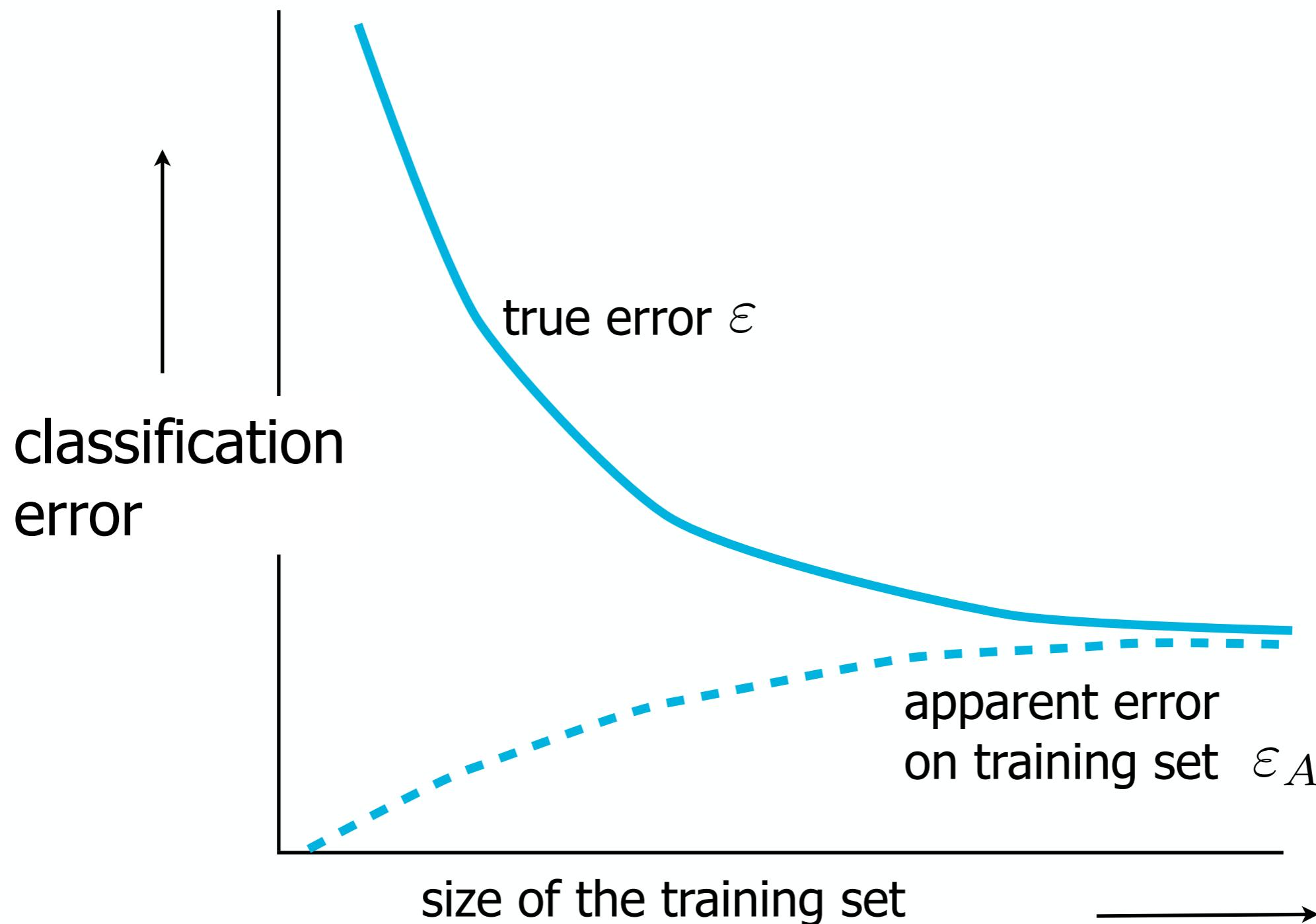
- To optimise hyperparameters, do cross-validation **inside** another cross-validation:



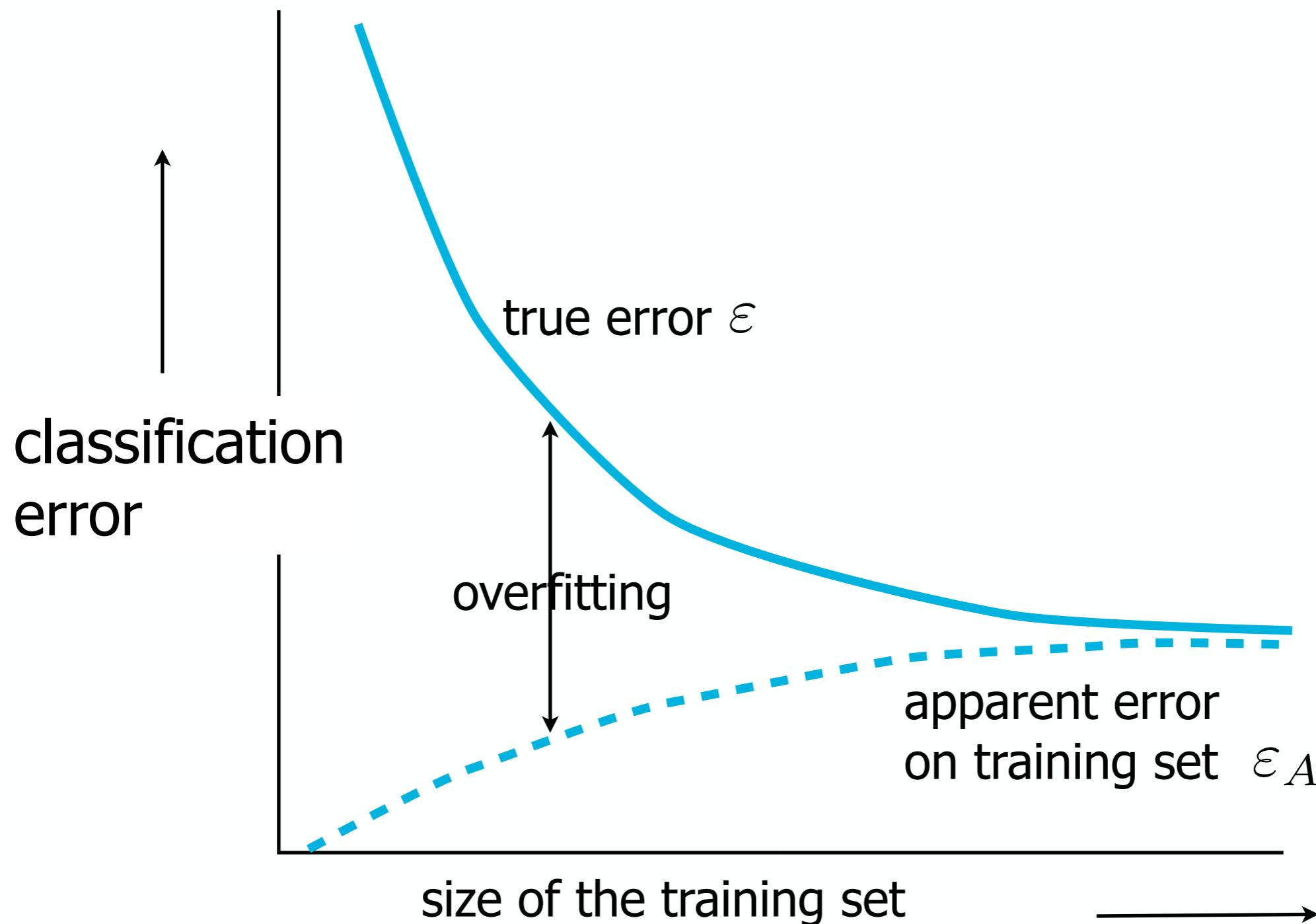
# Apparent Classification error



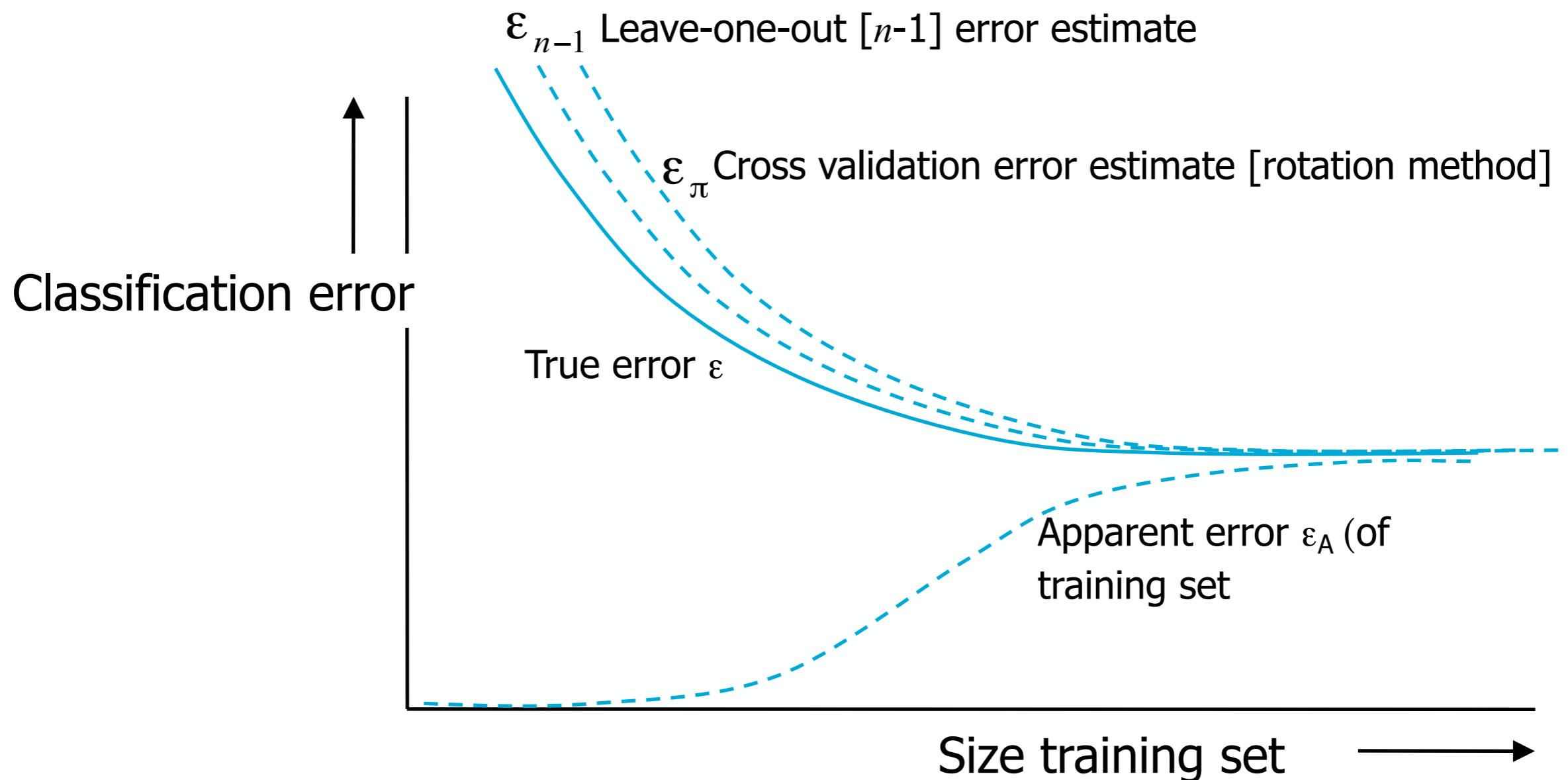
# Learning curve



# Learning curve



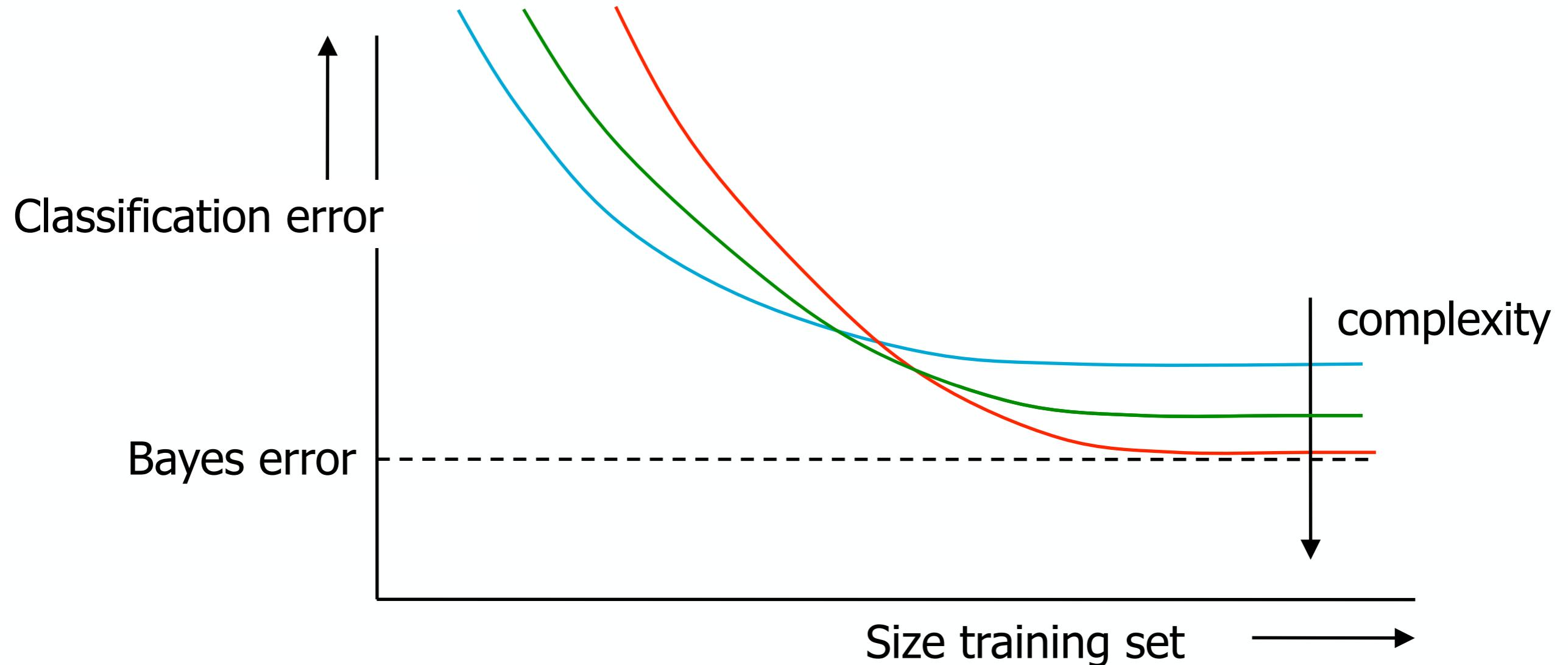
# Cross Validation Curves [and Related]



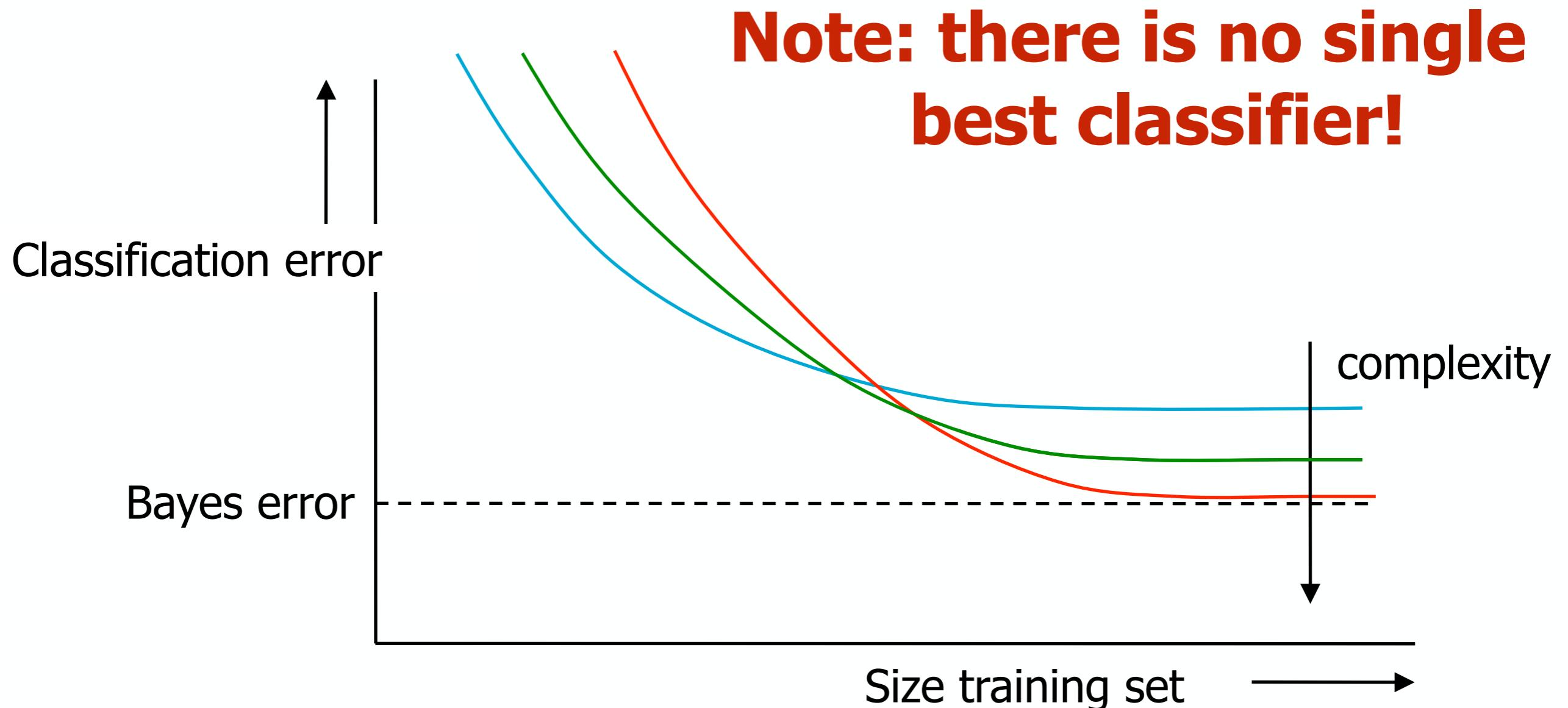
# Learning Curves

- Curves that plot [estimated] classification errors against the number of samples in training set
  - Usually plot error both on training and on test set
  - Gives insight in, e.g.
    - Amount of overtraining
    - Usefulness of additional data
    - How different classifiers compare
    - Stability of training
    - ...

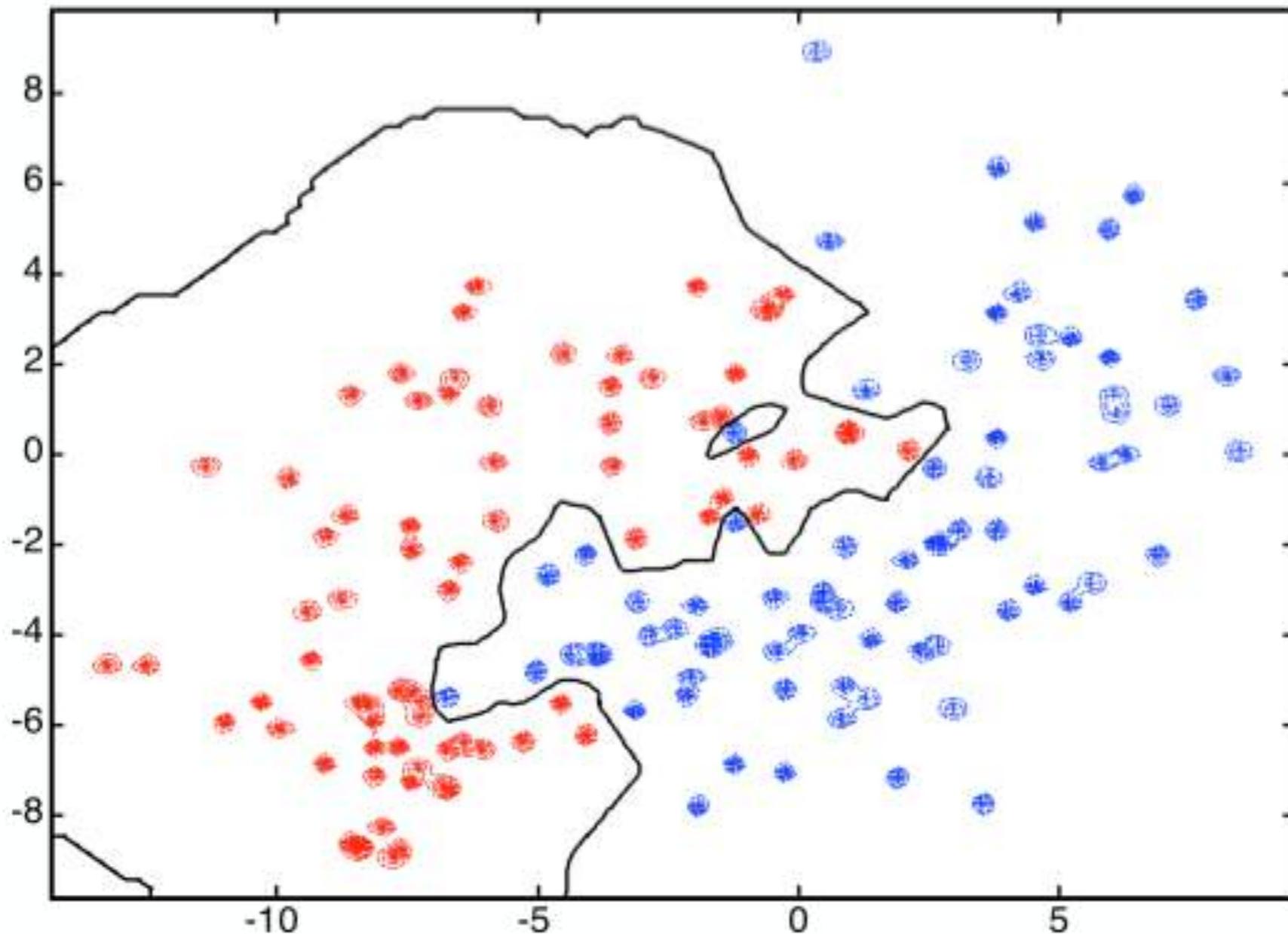
# Different Classifier Complexity

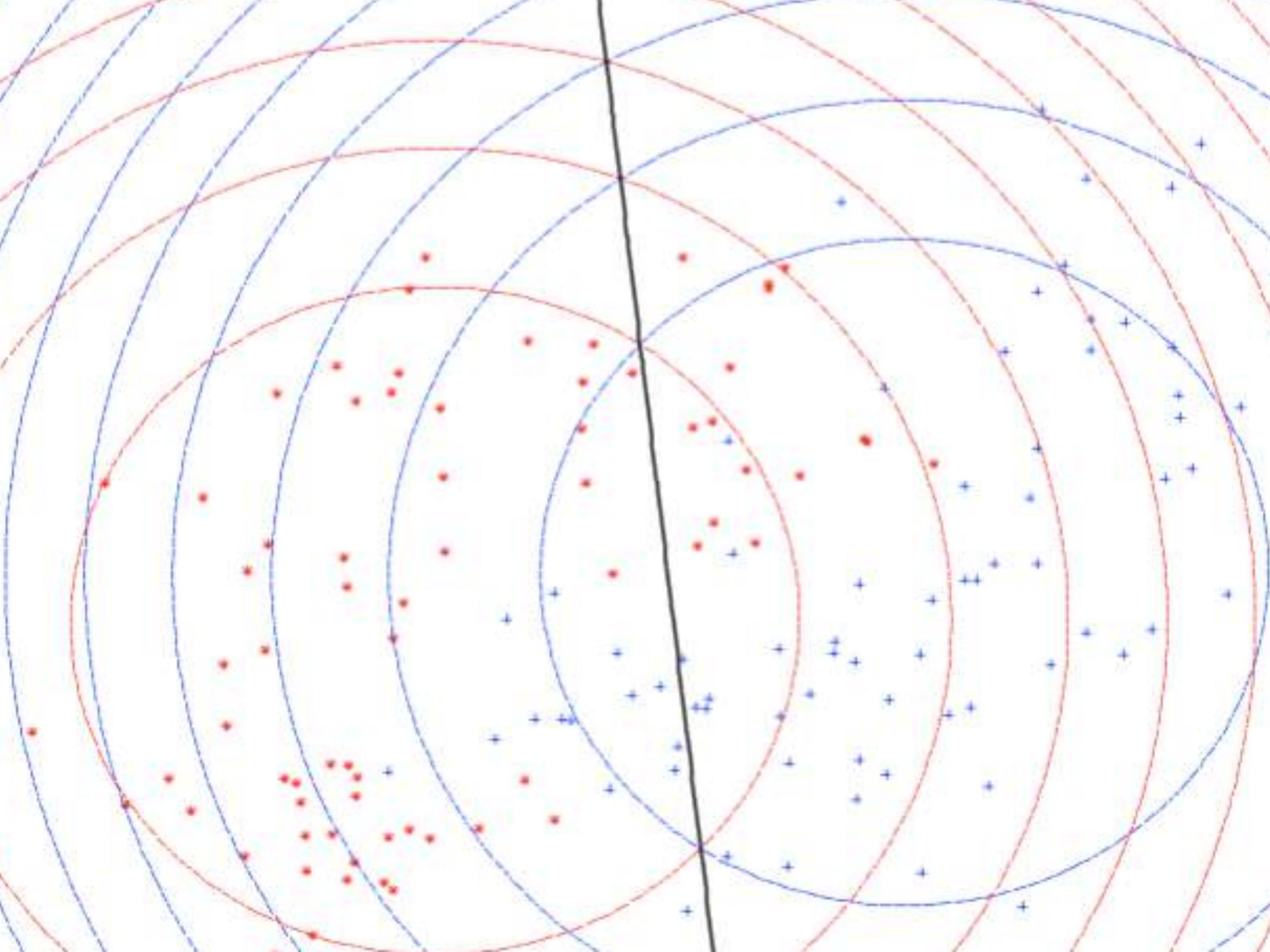


# Different Classifier Complexity

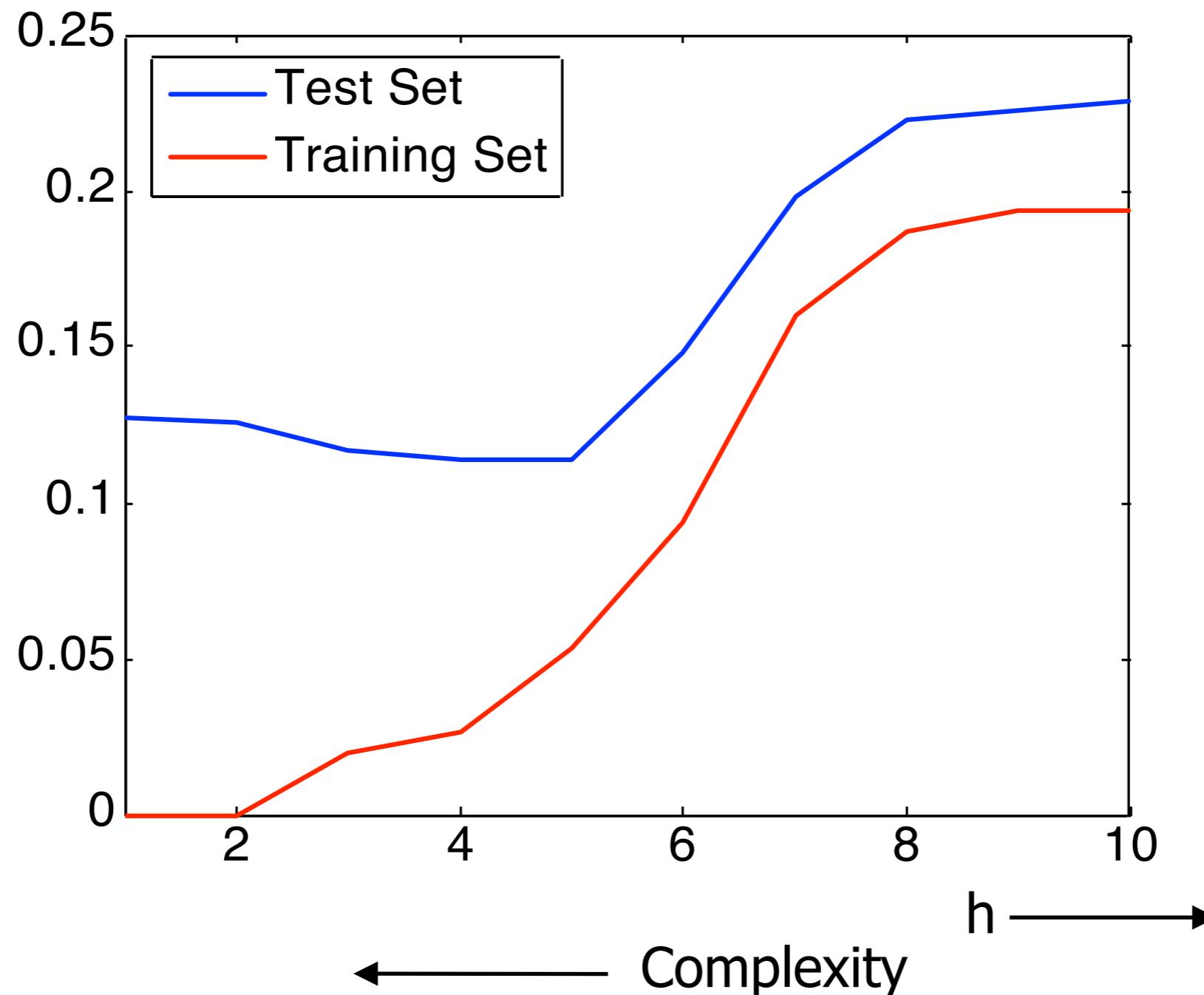


# Parzen Complexity Example





# Parzen Complexity Example



# Some intermediate conclusions...

- Larger training sets yield better classifiers
- Independent test sets needed for unbiased error estimates
- Larger test sets yield more accurate error estimates
- LOO cross validation “optimal”, but might be infeasible
- 10-fold crossvalidation is often used
- More complex classifiers need larger training sets
  - Same holds for larger feature set sizes
- Small training sets need simpler classifiers or smaller feature sets

# Classifier evaluation

- This lecture:
  - Train and test set
  - Bootstrapping, cross-validation, leave-one-out
  - Learning curve
  - Classifier complexity

---

- Bias-variance tradeoff
- Confusion matrices
- ROC curve
- (Reject curve)

# Squared Error

- When we have the error

$$L(\mathbf{w}) = E[\|g(\mathbf{x}) - y\|^2]$$

we can actually derive something more general...

- In general?!
- Unfortunately, theory in Pattern Recognition is tough...  
how to deal with **all** possible datasets?

# Bias-variance dilemma

- When we are given some data, we may get lucky, or unlucky:  
sometimes we get very aypical data:-(  
)

- To say something general, we need to **average** over different (training-) sets

$$\mathcal{D} = \{(y_i, \mathbf{x}_i); i = 1, \dots, N\}$$

- The classifier is now also a function of the training set:  
 $g(\mathbf{x}; \mathcal{D})$

# Bias-variance dilemma

- Consider the squared error:

$$E_{\mathcal{D}} [(g(\mathbf{x}; \mathcal{D}) - E[y|\mathbf{x}])^2]$$

- $E[y|\mathbf{x}]$  is the optimal mean-squared regressor (not proven now; see book)
- Now we do a trick:

$$\begin{aligned} &= E_{\mathcal{D}} \left[ \underbrace{(g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})]}_{+ E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}])^2} \right. \\ &\quad \left. + \underbrace{E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}])^2}_{\text{---}} \right] \end{aligned}$$

# Bias-variance dilemma

- Now working out the square:

$$\begin{aligned} &= E_{\mathcal{D}} \left[ (\underbrace{g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})]}_{} )^2 \right. \\ &\quad + 2(\underbrace{g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})]}_{} ) (\underbrace{E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}]}_{} ) \\ &\quad \left. + (\underbrace{E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}]}_{} )^2 \right] \end{aligned}$$

# Bias-variance dilemma

- Now working out the square:

$$\begin{aligned} &= E_{\mathcal{D}} \left[ \underbrace{(g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})])^2}_{+ 2(g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})])(E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}])} \right. \\ &\quad \left. + (E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}])^2 \right] \end{aligned}$$

$$\begin{aligned} &- E_{\mathcal{D}} \left[ (g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})])^2 \right] \xrightarrow{\text{(variance)}} = 0 \\ &+ \boxed{E_{\mathcal{D}} [2(g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})])]} (E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}]) \\ &+ E_{\mathcal{D}} \left[ (E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}])^2 \right] \xrightarrow{\text{(bias}^2)} \end{aligned}$$

# Bias-variance dilemma

(variance)

$$\begin{aligned} MSE &= E_{\mathcal{D}} \left[ (g(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})])^2 \right] \\ &\quad + E_{\mathcal{D}} \left[ (E_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - E[y|\mathbf{x}])^2 \right] \end{aligned}$$

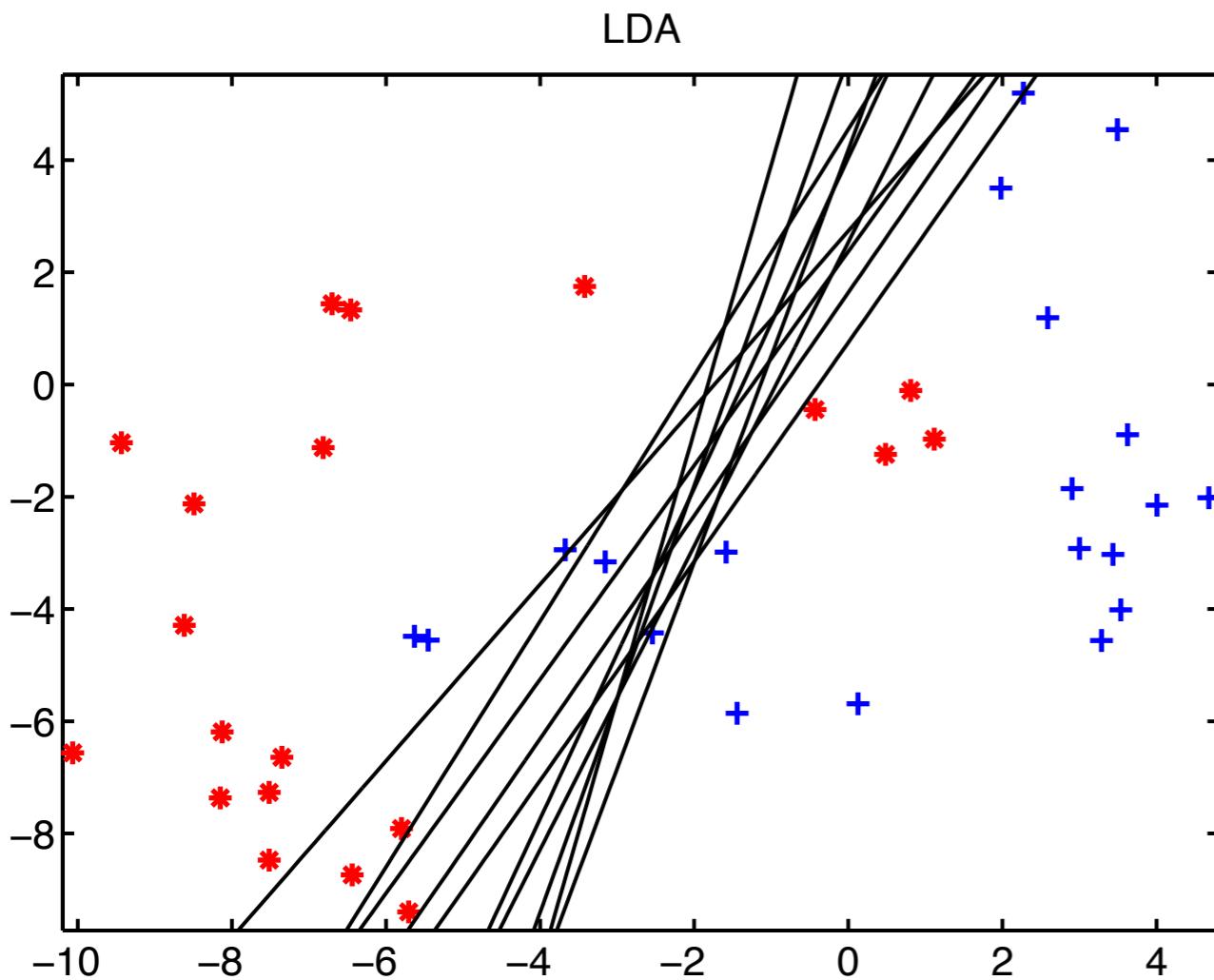
(bias<sup>2</sup>)

- variance: how much does classifier  $g$  vary over different training sets
- bias: how much does the average classifier  $g$  differ from the true output

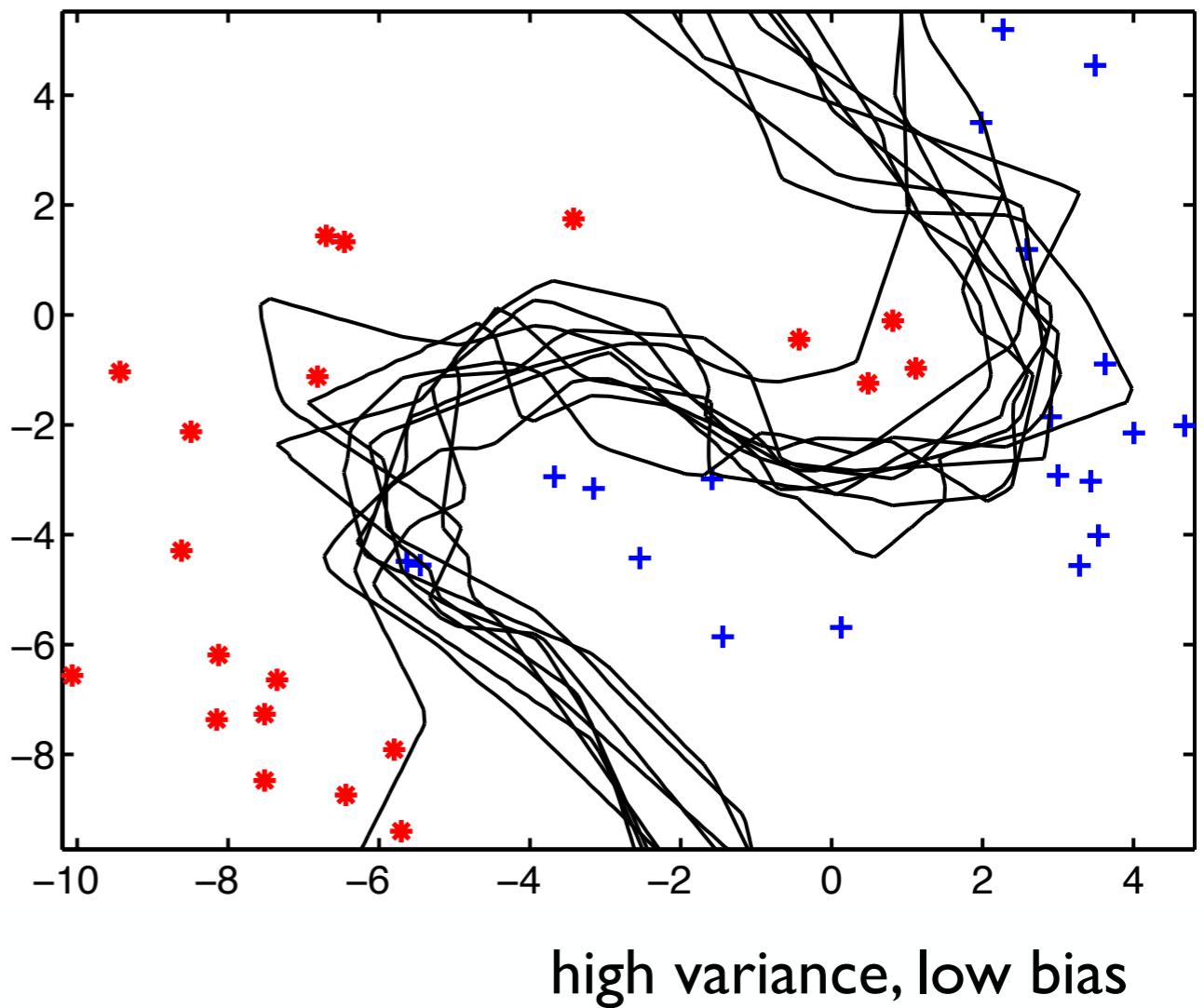
# Bias-variance dilemma

- Compare LDA with kNN:

low variance, high bias



kNN

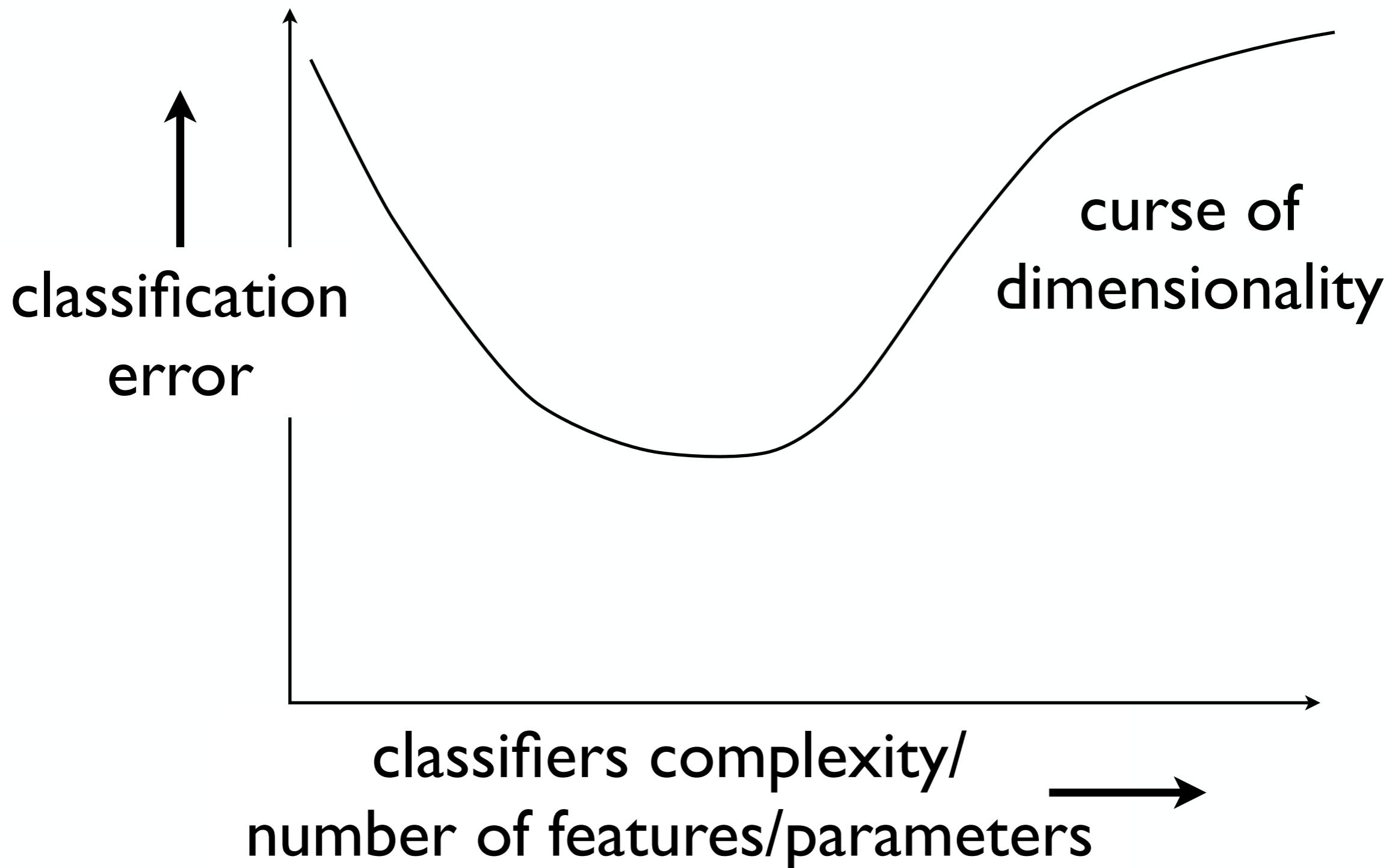


high variance, low bias

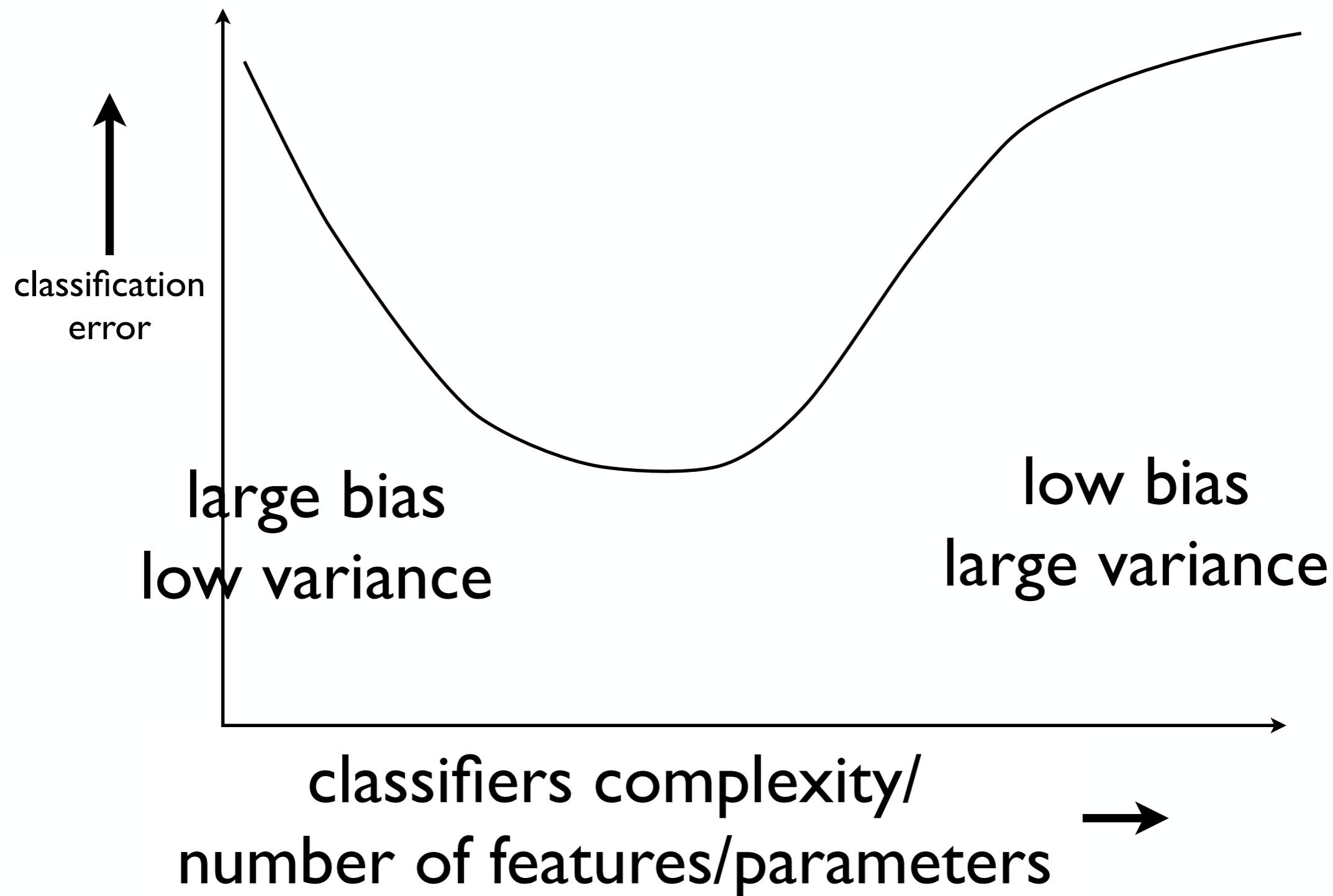
# Bias-variance tradeoff

- Originally derived for neural networks and squared error
- It is a very general phenomenon: we encounter it more often in pattern recognition
- More simple classifier is more stable (and need less data to train)
- More complex classifier only works when you have sufficient number of training data

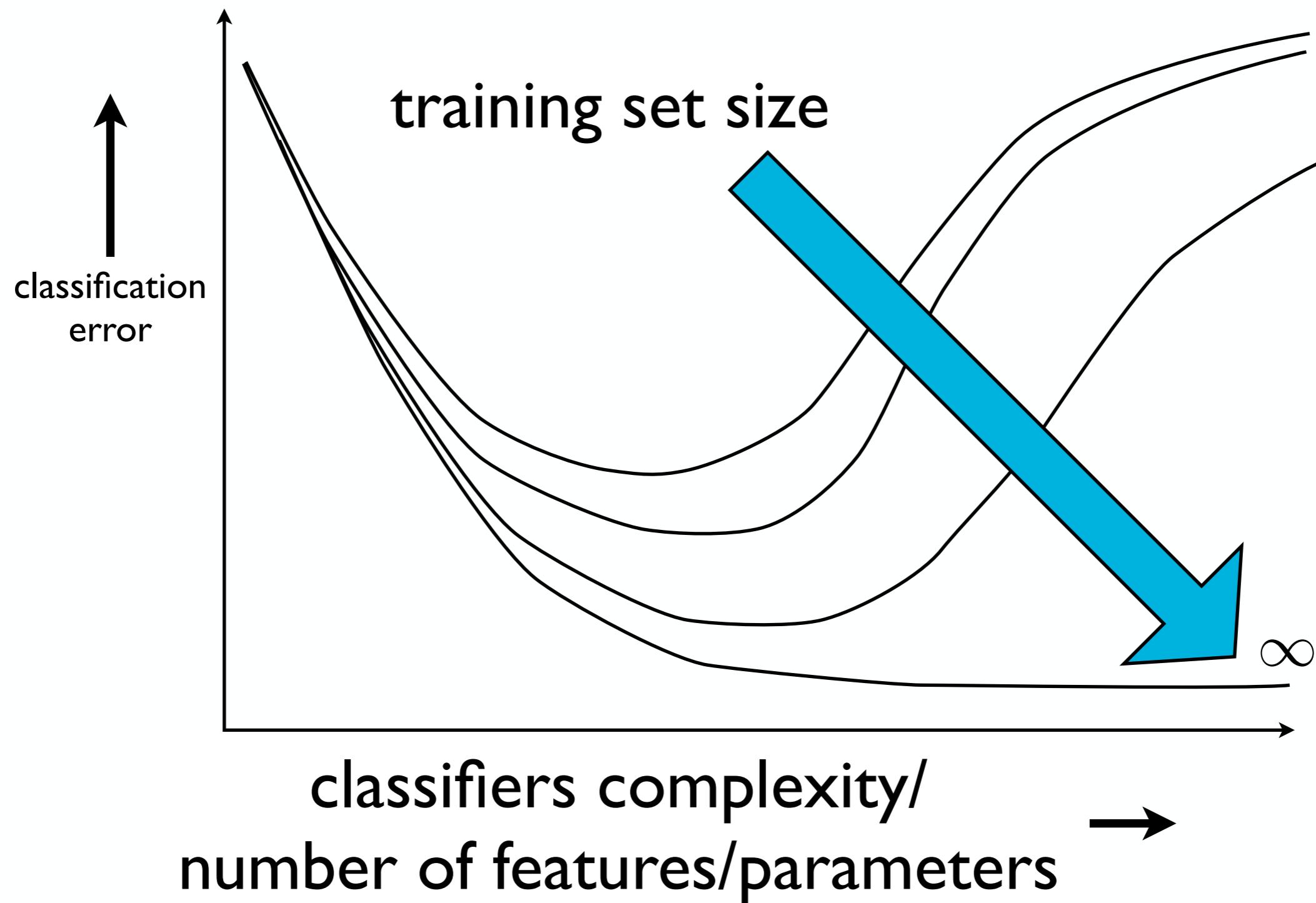
# Feature curve



# Feature curve



# Feature curve



# Errors for two-class classification

- There is a fundamental tradeoff between the two errors/performances of the two classes
- Standard classification error:  $\varepsilon = \varepsilon_1 p(y_1) + \varepsilon_2 p(y_2)$
- Weighted classification error:  
$$\varepsilon = \lambda_{12} \varepsilon_1 p(y_1) + \lambda_{21} \varepsilon_2 p(y_2)$$
- F1-score (harmonic mean):  
$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$
- ...

# Error / Performance Measures

- **Error** : probability of erroneous classifications
- **Performance / accuracy** :  $1 - \text{error}$
- **Sensitivity** of a target class [e.g. diseased patients] : performance for objects from that target class
- **Specificity** : performance for all objects outside target class
- **Precision** of a target class : fraction of correct objects among all objects assigned to that class.
- **Recall** : fraction of correctly classified objects; identical to sensitivity when related to particular class
- **True positive rate** : identical to sensitivity
- **False positive rate** : error for all objects outside target

# Confusion Matrices

- Provides counts of class-dependent errors : How many object have been classified as A that should have been classified as B?
- Give a more detailed view than overall error rate
- Can be used to estimate overall cost for particular classifier

# Confusion Matrix (1)

Real  
labels:

$$\Lambda = \begin{bmatrix} \lambda_1 \\ \dots \\ \lambda_N \end{bmatrix}$$

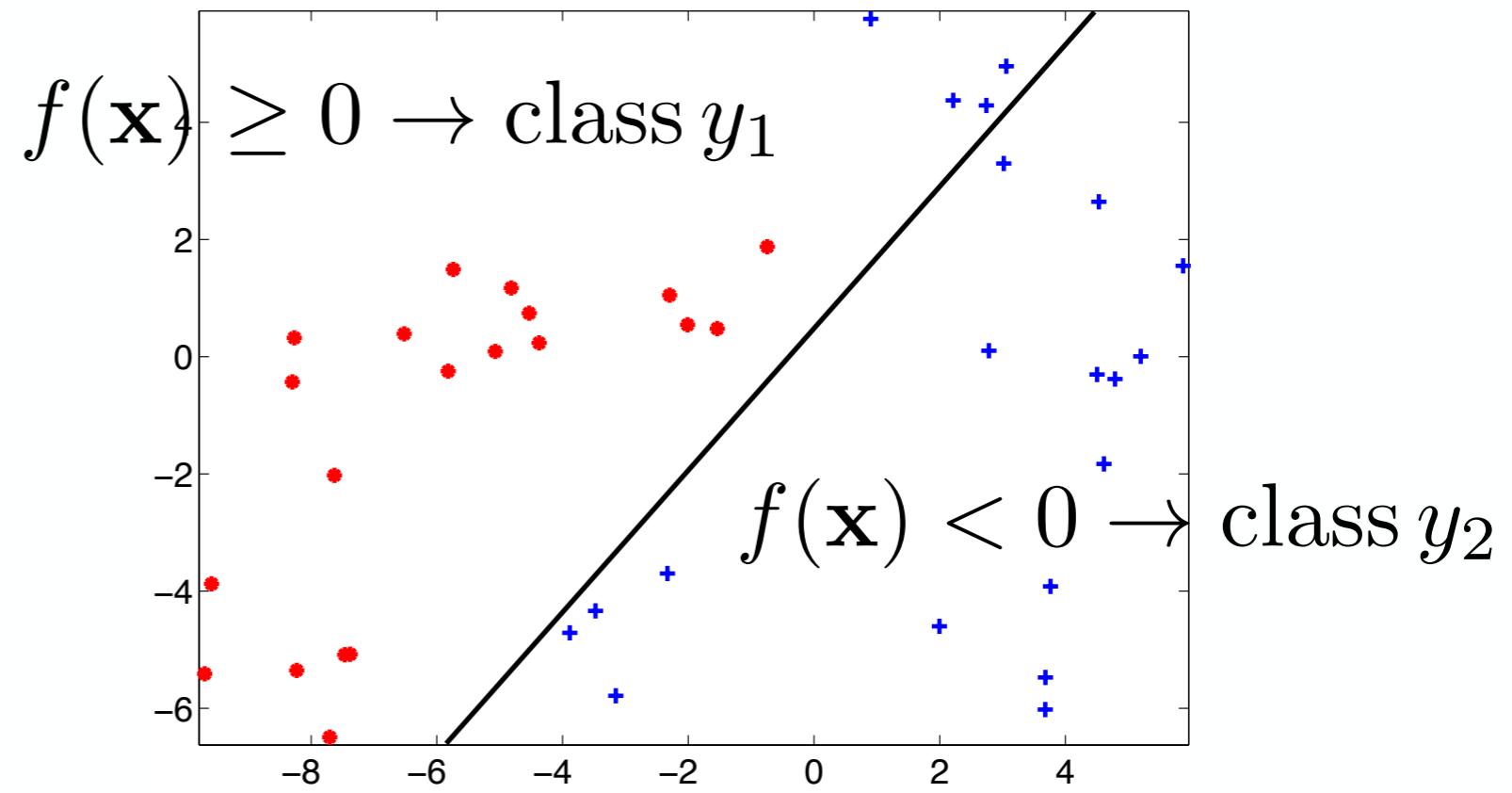
Predicted  
labels:

$$L = \begin{bmatrix} l_1 \\ \dots \\ l_N \end{bmatrix}$$

Confusion matrix:

$$C = \begin{bmatrix} c_{11} & \dots & c_{1K} \\ \dots & \dots & \dots \\ c_{K1} & \dots & c_{KK} \end{bmatrix}$$

$$c_{ij} = N \cdot P[l_j | \lambda_i]$$



# Confusion Matrix (2)

$$N_A = 10, N_B = 30, N_C = 20$$

$$E = \frac{c_{12} + c_{13} + c_{21} + c_{23} + c_{31} + c_{32}}{N_A + N_B + N_C}$$

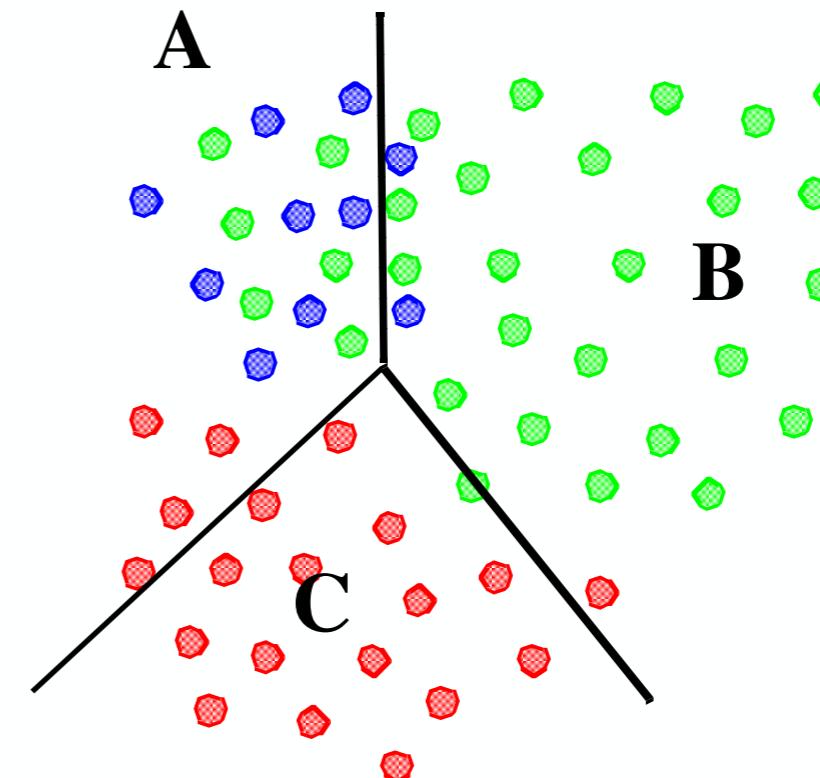
$$E = 14 / 60 = 0.2333$$

$C = \text{confmat}(\Lambda, L)$

$\Lambda$  real labels

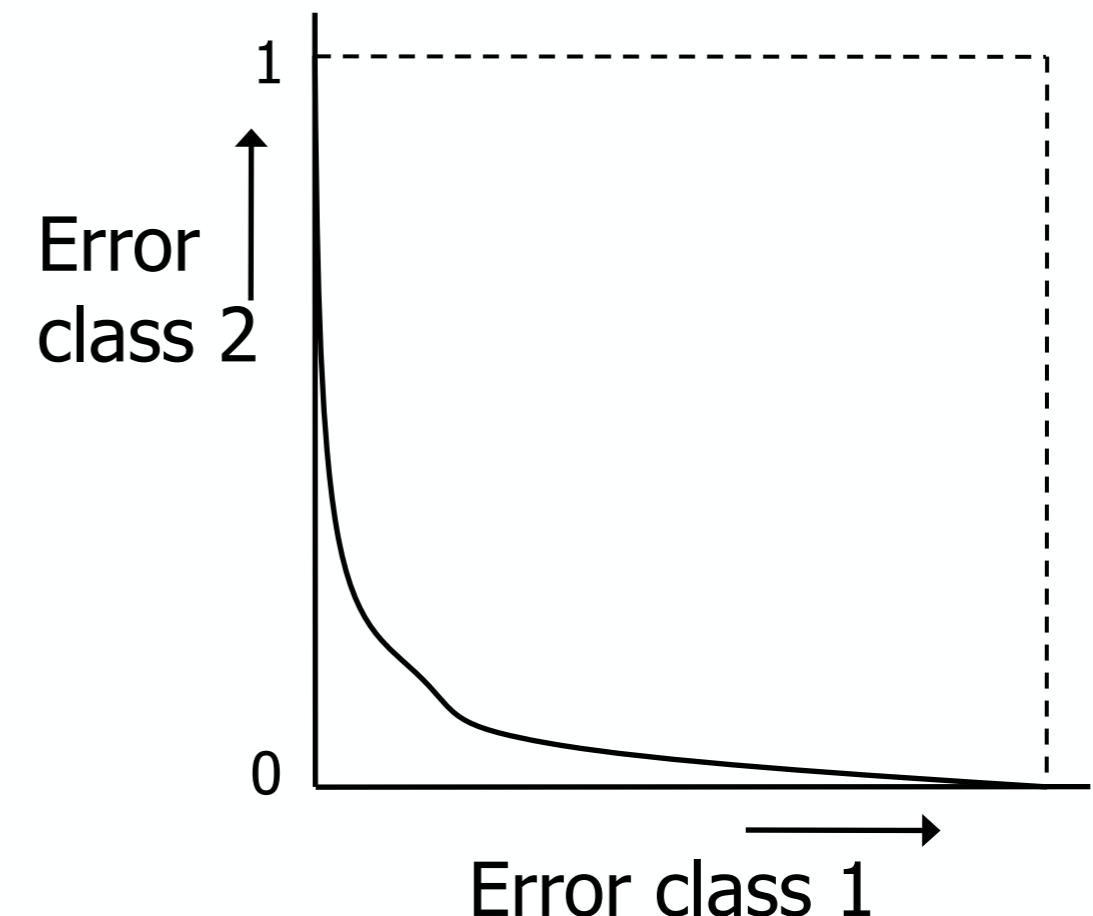
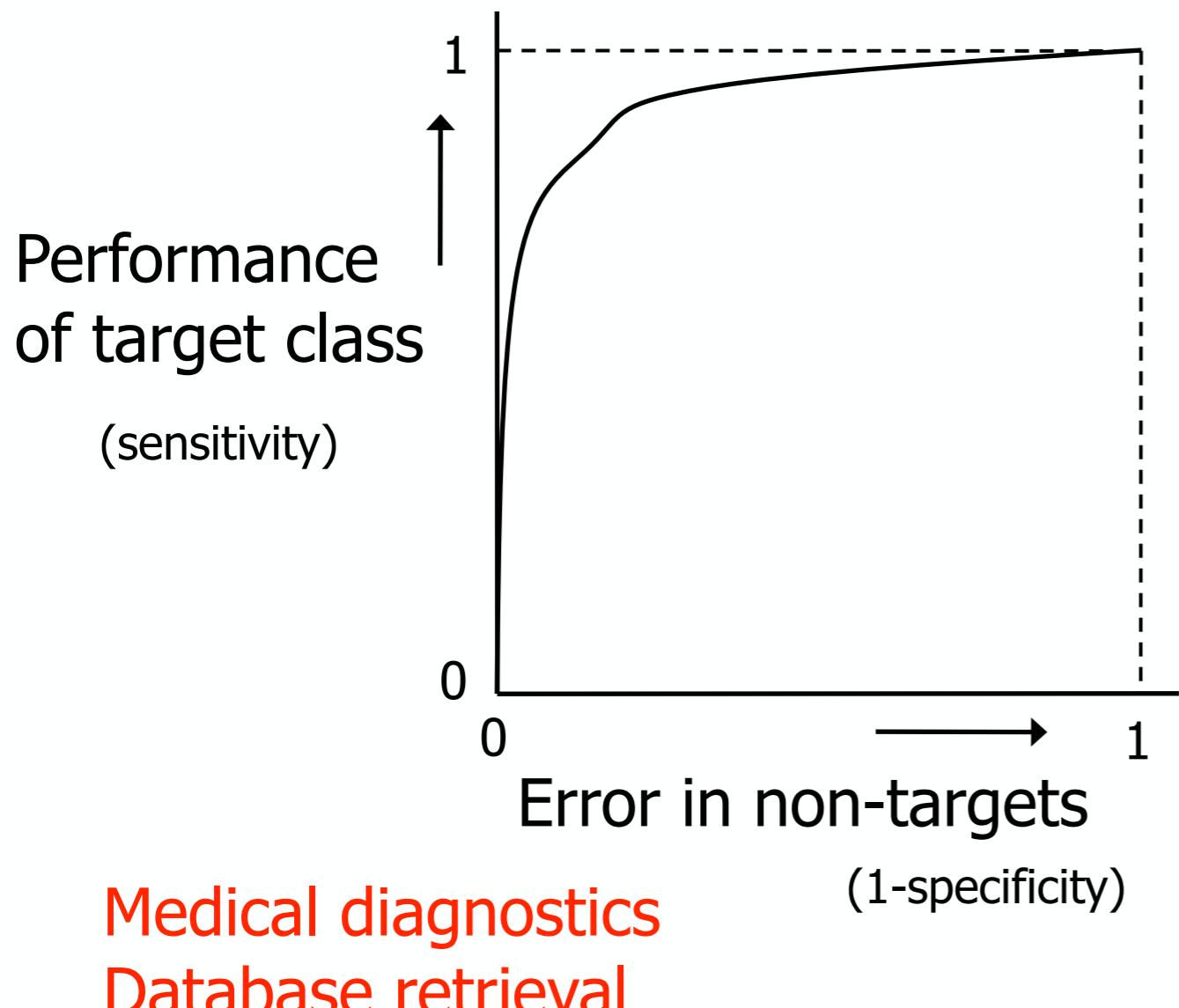
$L$  obtained labels

	objects from	classified to			0.228 averaged error
		A	B	C	
	class A	8	2	0	0.20 error in class A
	class B	6	23	1	0.23 error in class B
	class C	4	1	15	0.25 error in class C



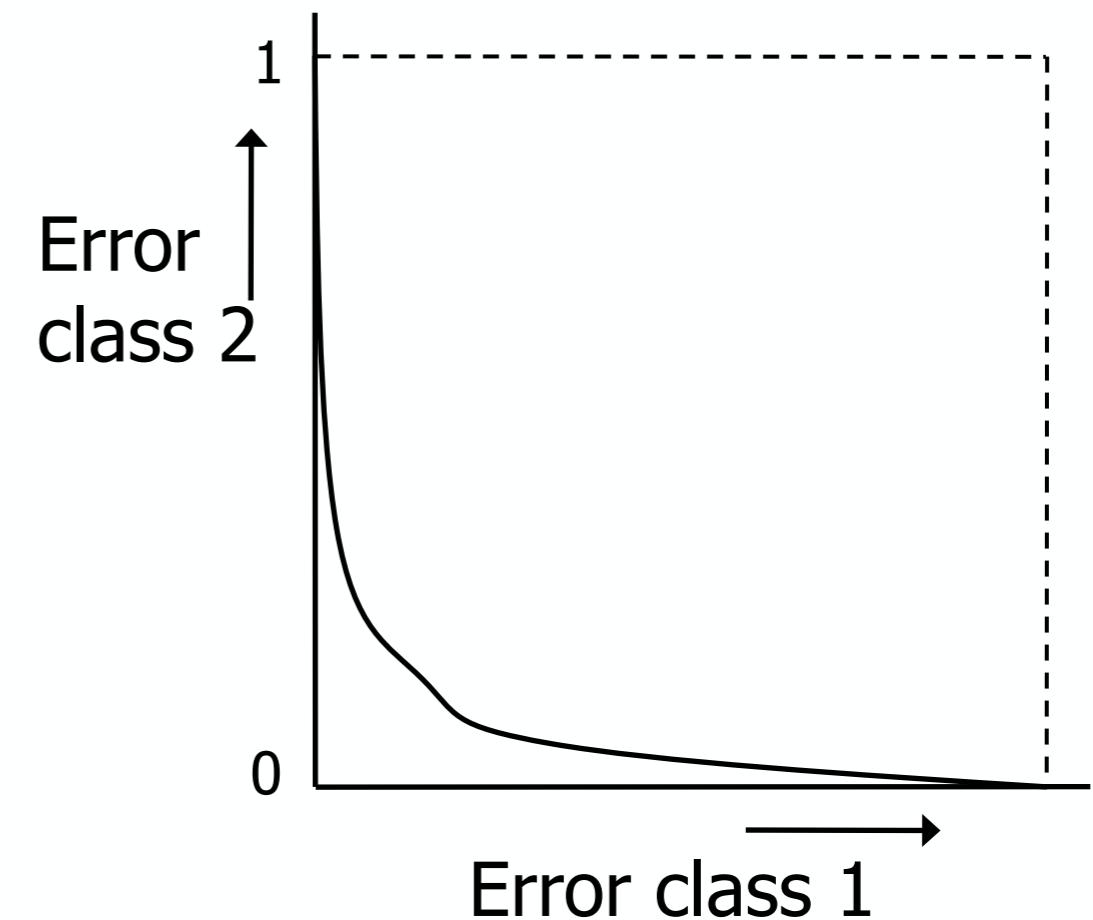
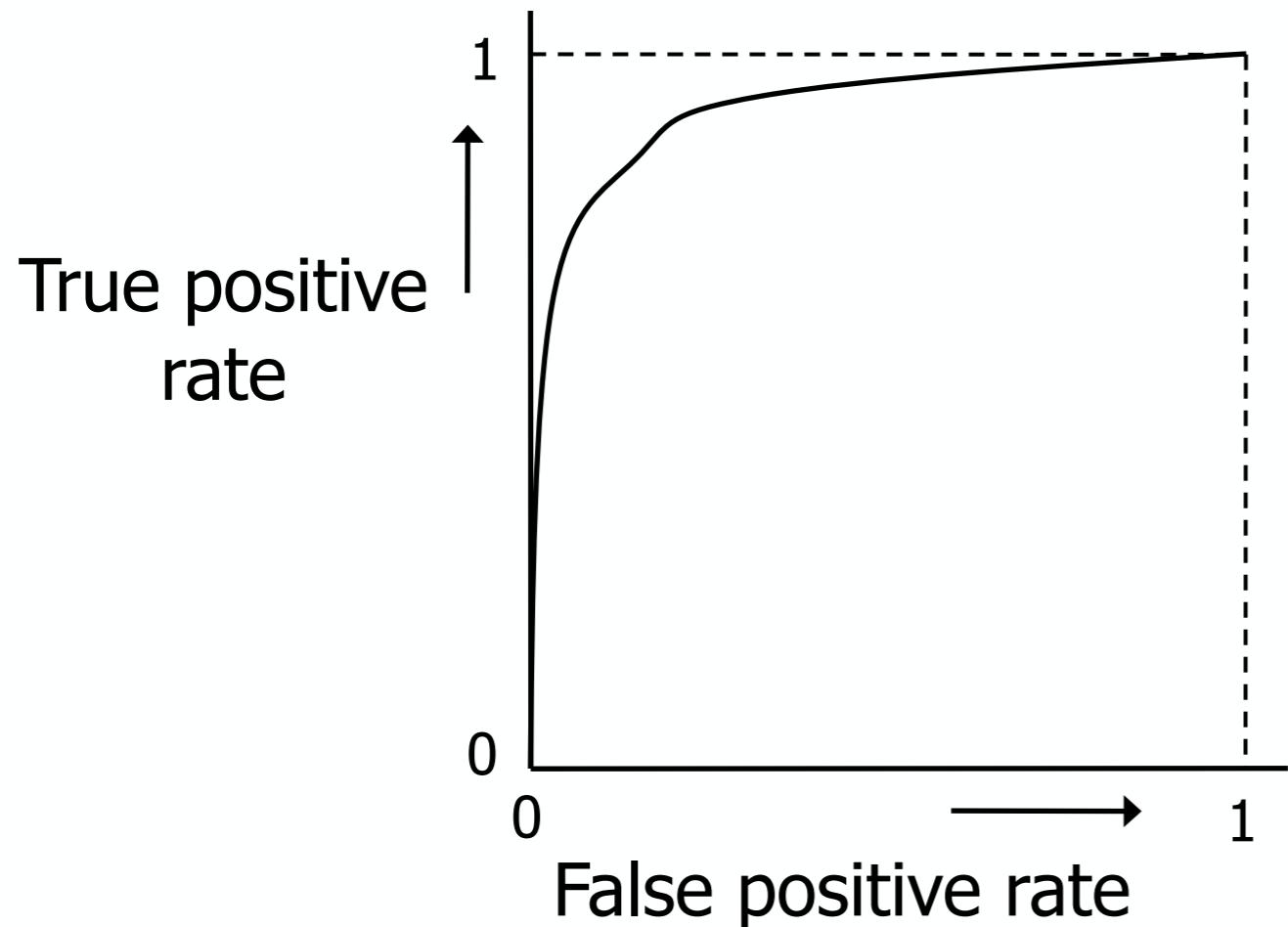
# ROC Analysis

ROC: Receiver-Operator Characteristic (from communication theory)



# ROC Analysis

ROC: Receiver-Operator Characteristic (from communication theory)

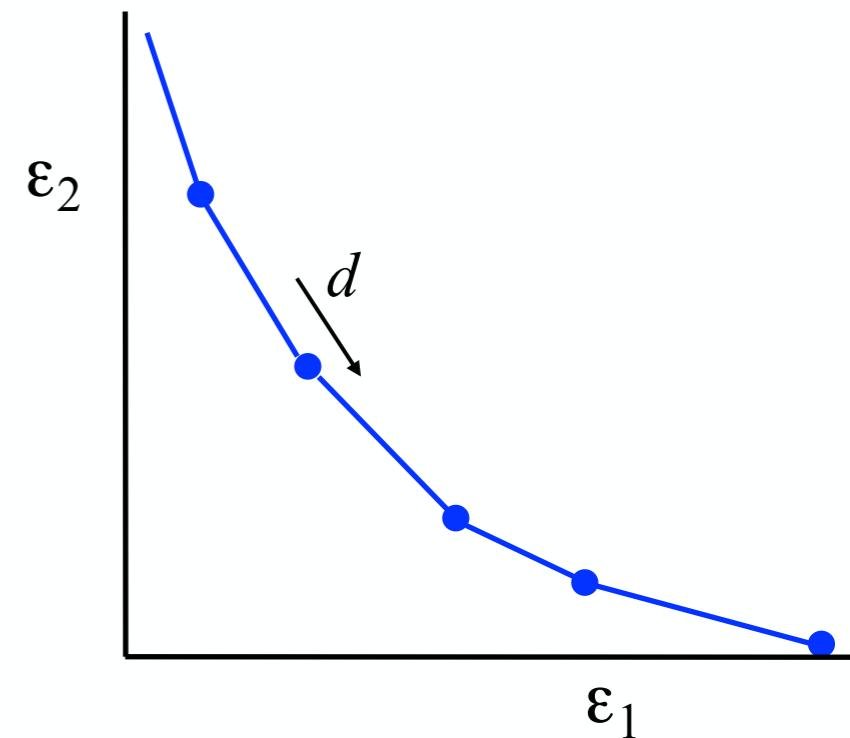
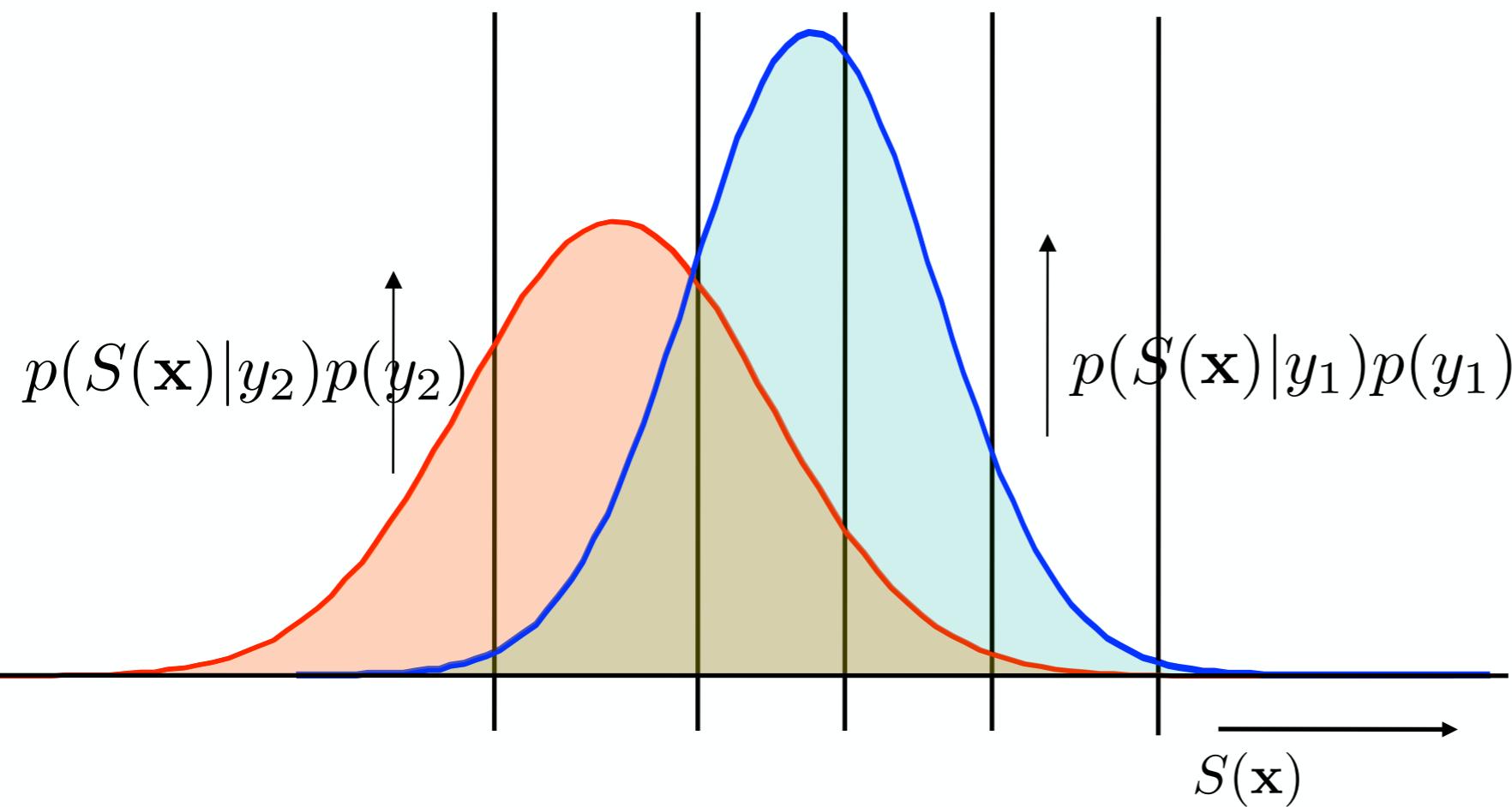


Medical diagnostics  
Database retrieval

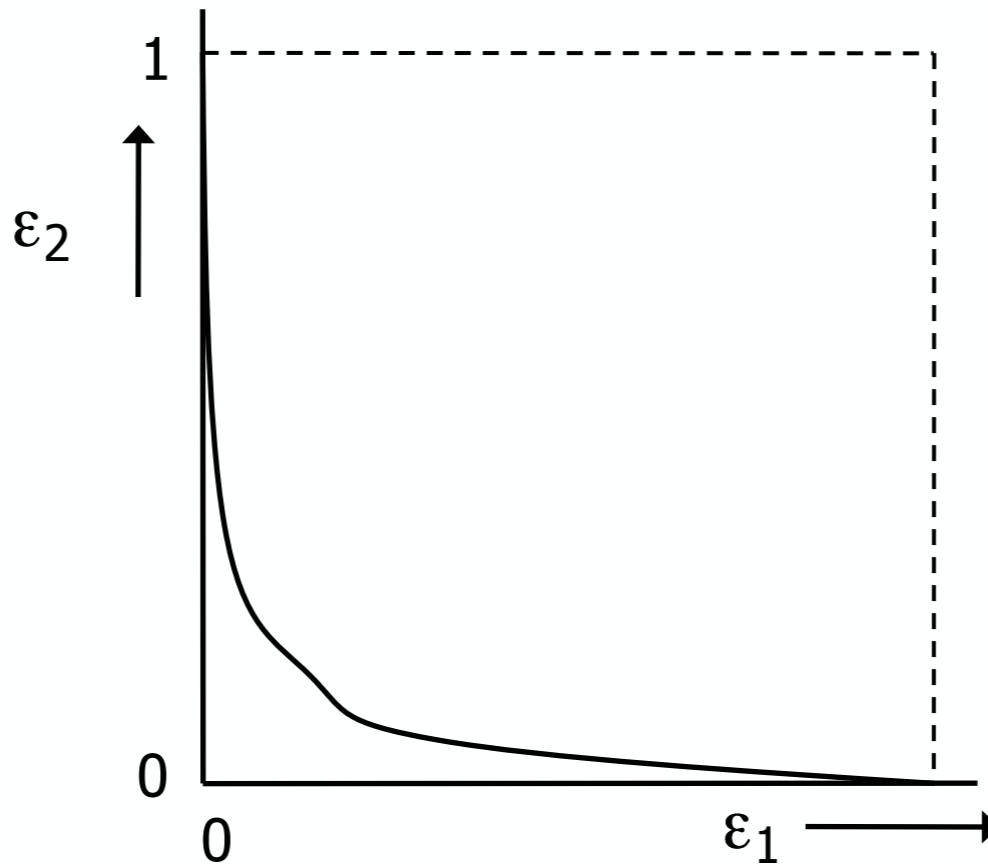
2-class pattern recognition

# ROC Curve

- Curve is obtained by varying classifier threshold  $d$

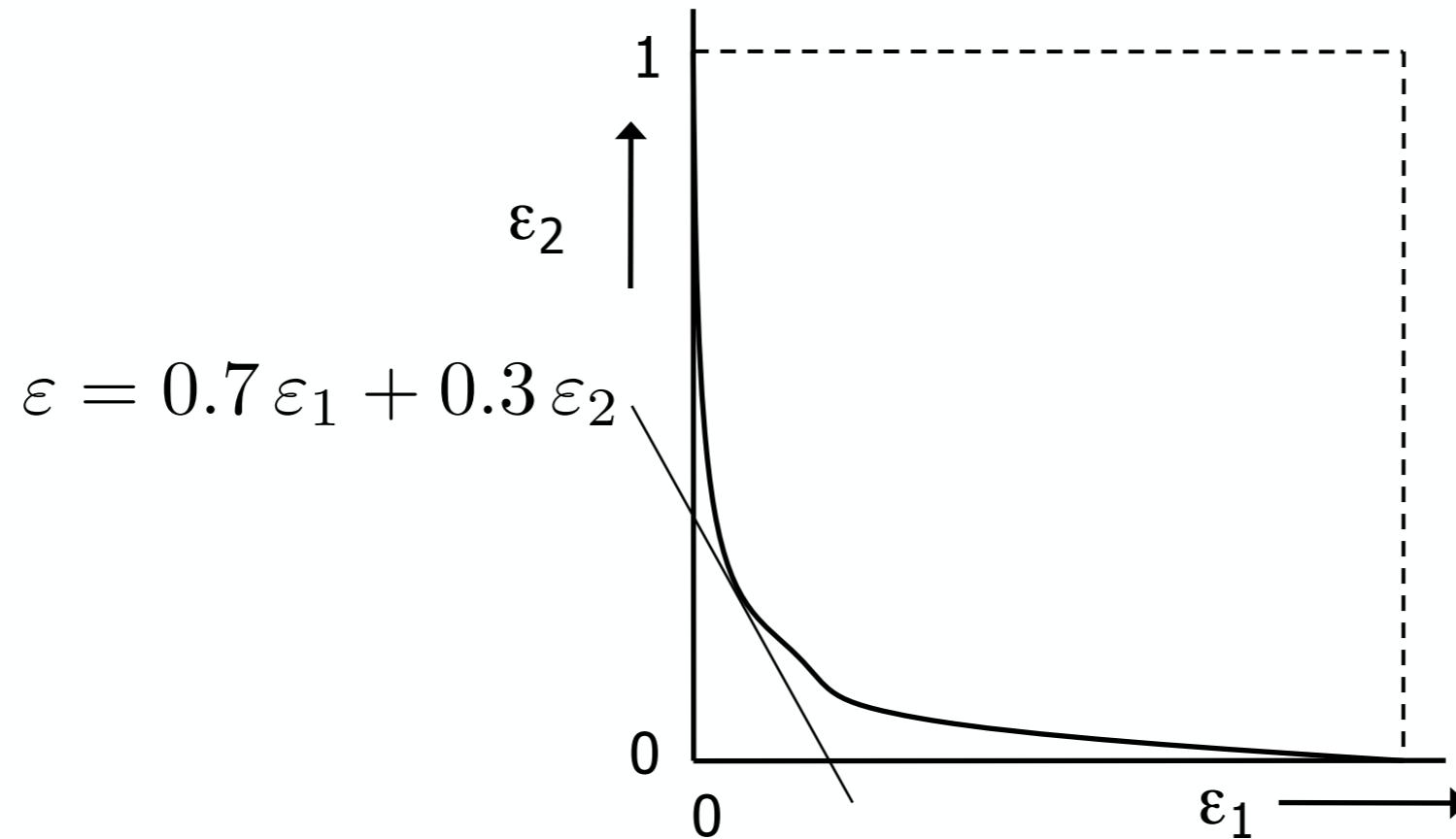


# Effect of Changing Priors/Costs



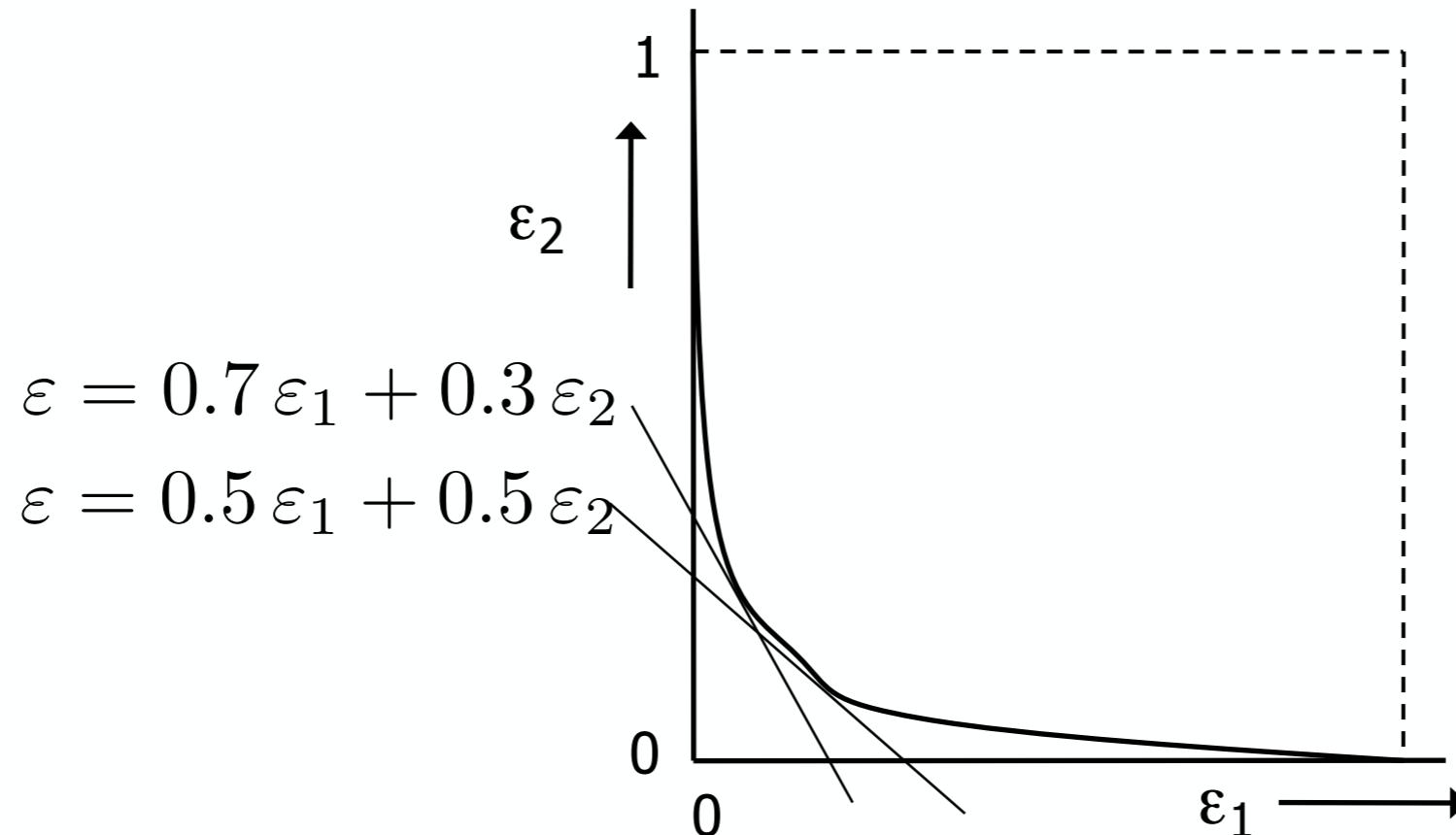
$$\varepsilon = p(y_1) \varepsilon_1 + p(y_2) \varepsilon_2$$

# Effect of Changing Priors/Costs



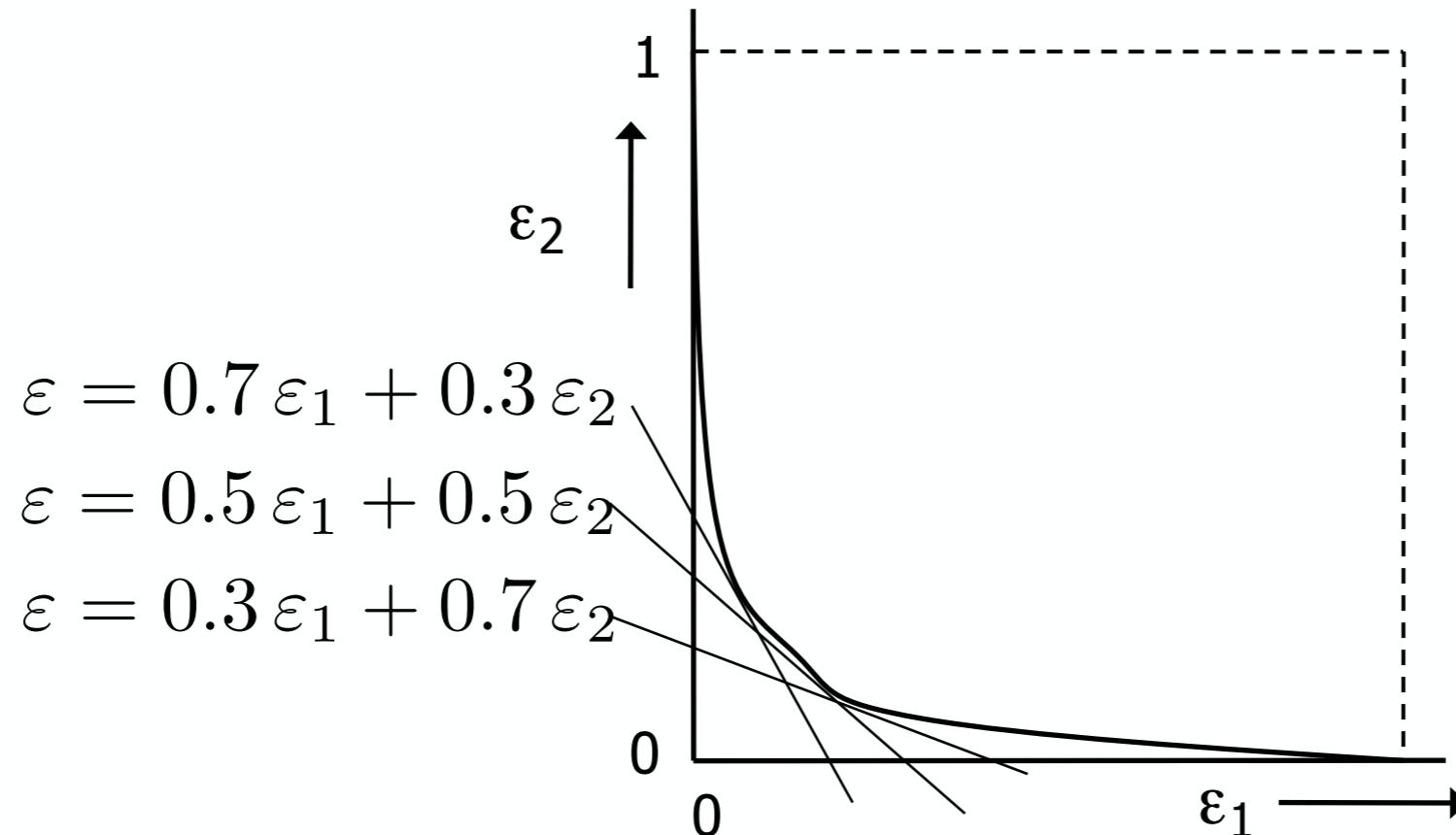
$$\varepsilon = p(y_1) \varepsilon_1 + p(y_2) \varepsilon_2$$

# Effect of Changing Priors/Costs



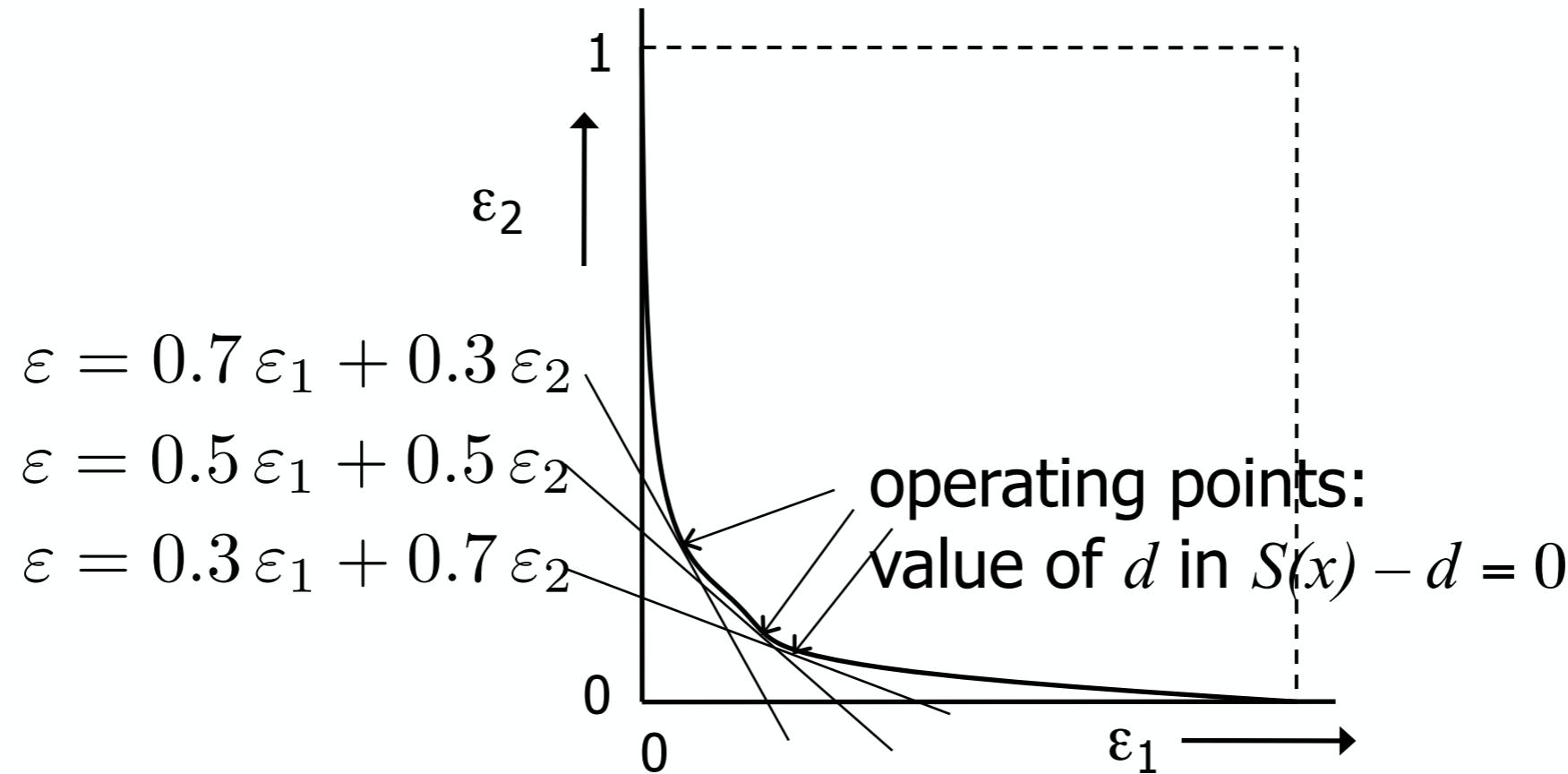
$$\varepsilon = p(y_1) \varepsilon_1 + p(y_2) \varepsilon_2$$

# Effect of Changing Priors/Costs



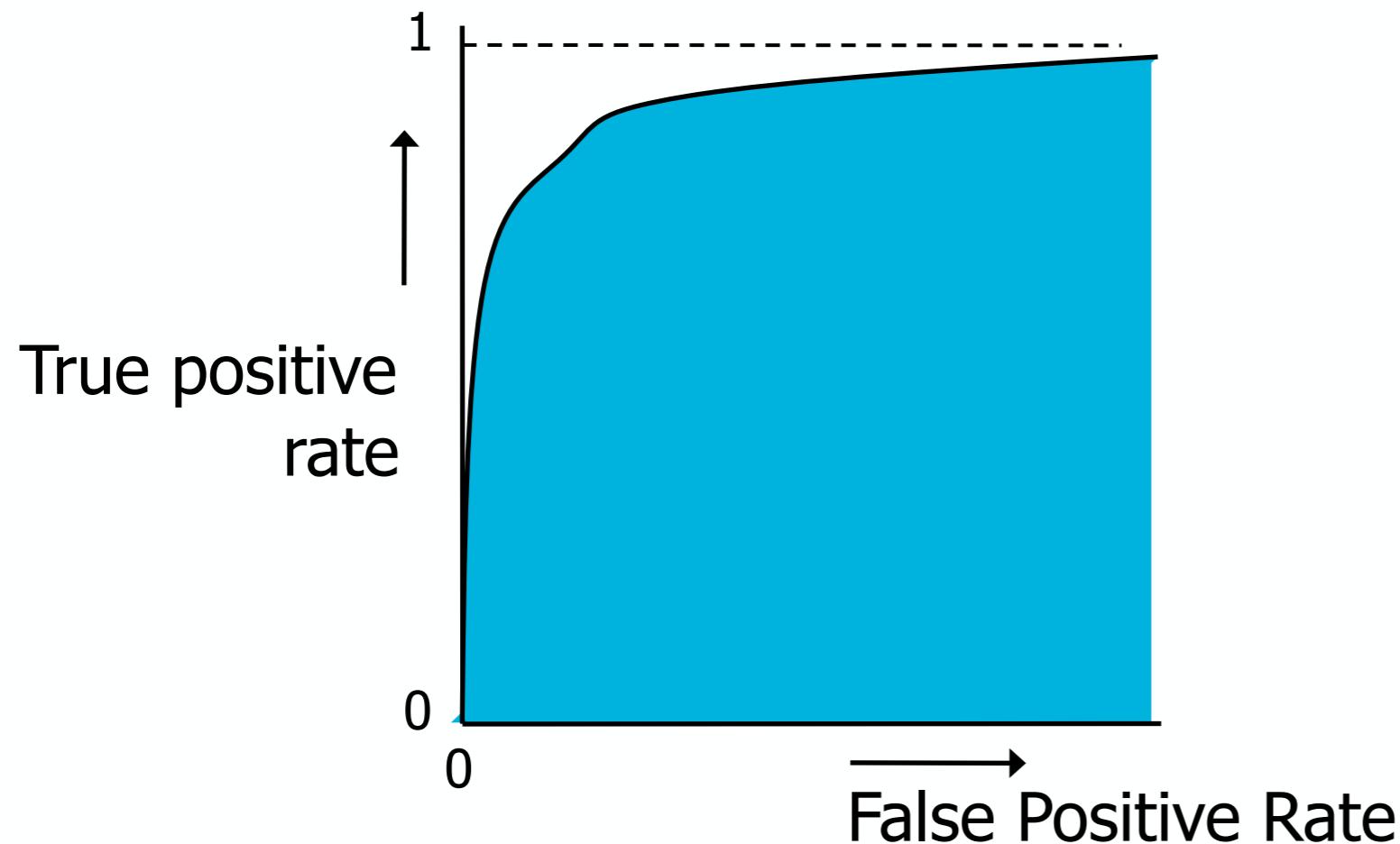
$$\varepsilon = p(y_1) \varepsilon_1 + p(y_2) \varepsilon_2$$

# Effect of Changing Priors/Costs



$$\varepsilon = p(y_1) \varepsilon_1 + p(y_2) \varepsilon_2$$

# Area under the ROC curve: AUC



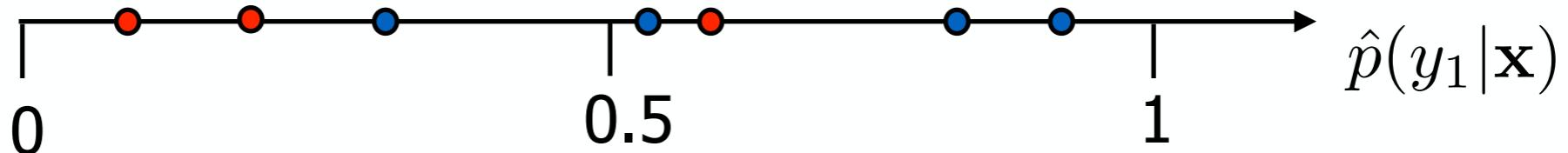
- Integrate the area: perfect classifier gives  $AUC=1.0$
- Random classifier gives  $AUC=0.5$
- Performance measure that is insensitive to class priors

# Example to compute ROC curve

- Assume I trained a classifier, and testing it on a test set:
- Four objects of class 1:
$$\hat{p}(y_1|\mathbf{x}_1) = 0.3$$
$$\hat{p}(y_1|\mathbf{x}_2) = 0.8$$
$$\hat{p}(y_1|\mathbf{x}_3) = 0.9$$
$$\hat{p}(y_1|\mathbf{x}_4) = 0.55$$
- Three objects of class 2:
$$\hat{p}(y_1|\mathbf{x}_5) = 0.2$$
$$\hat{p}(y_1|\mathbf{x}_6) = 0.1$$
$$\hat{p}(y_1|\mathbf{x}_7) = 0.6$$
- How does the ROC curve look like?  $\varepsilon_2$  vs.  $\varepsilon_1$

# Example to compute ROC curve

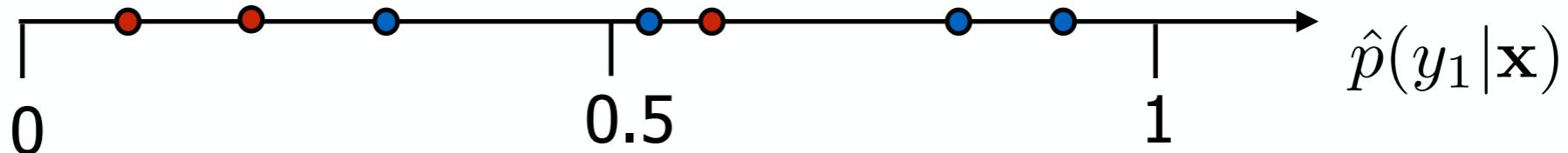
- Find out how obj's are classified for different thresholds



- For example, what are the errors for  $d = 0.5$  ?

# Example to compute ROC curve

- Find out how obj's are classified for different thresholds

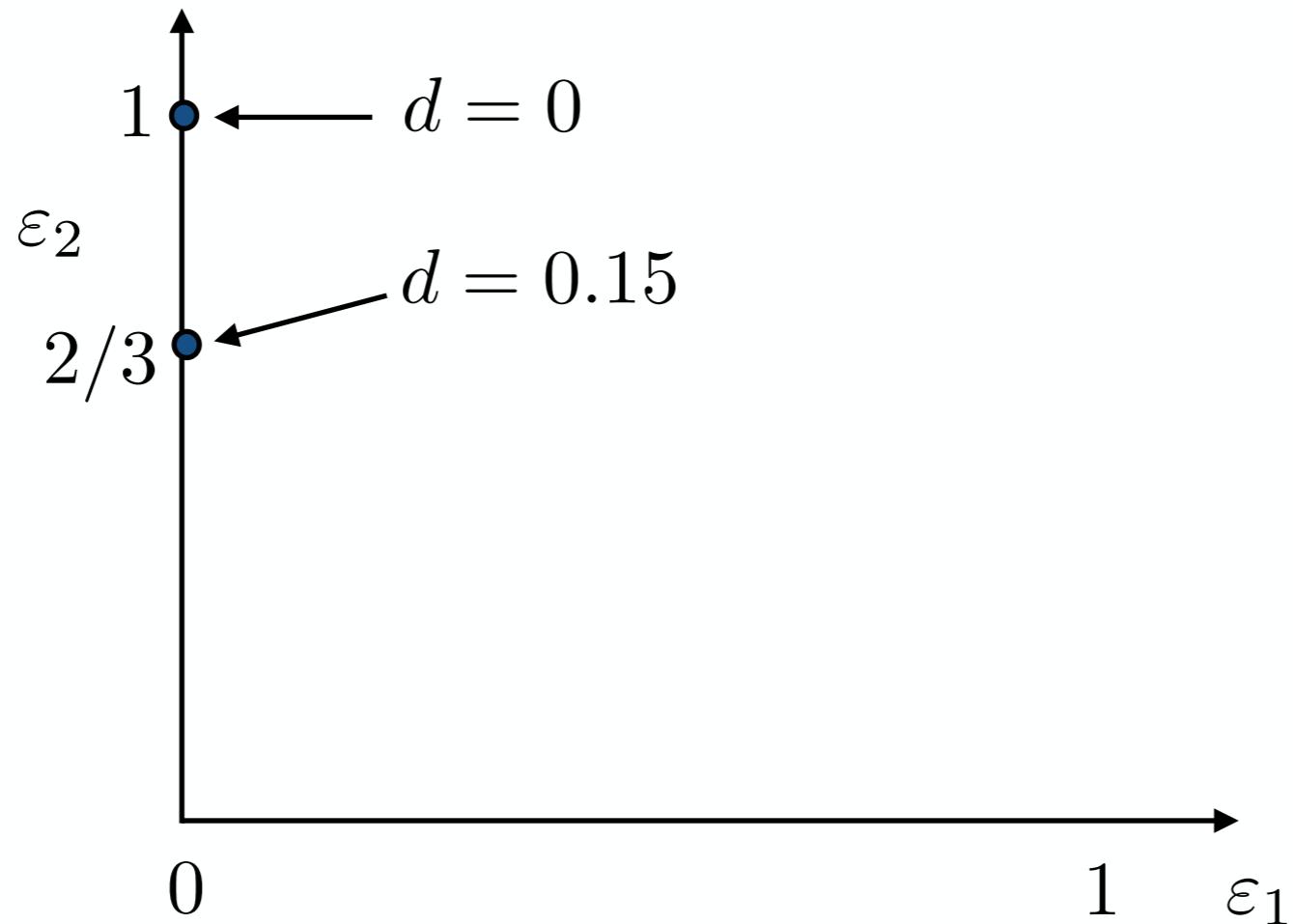


- For example, what are the errors for  $d = 0.5$  ?

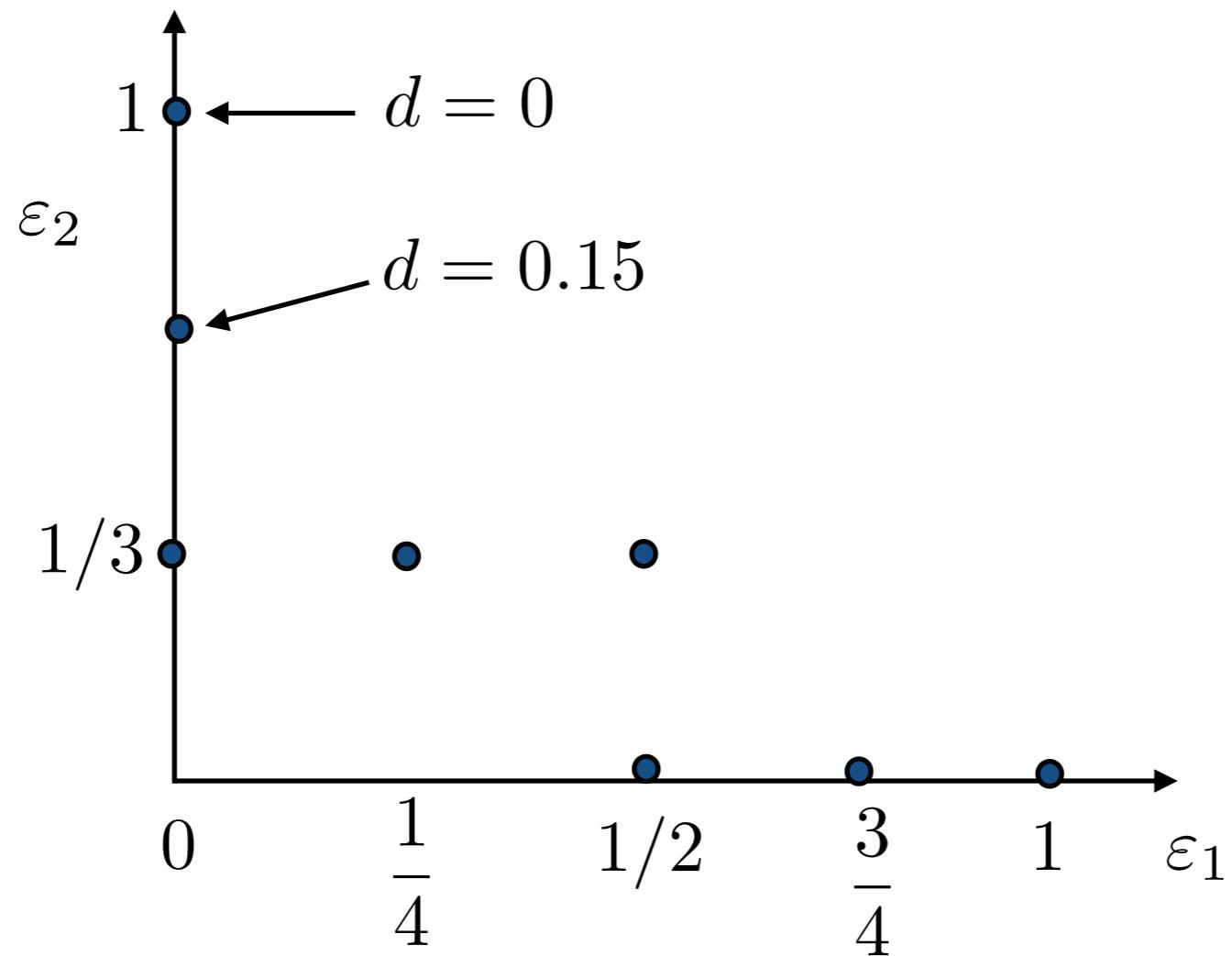
$$\varepsilon_1 = 1/4$$

$$\varepsilon_2 = 1/3$$

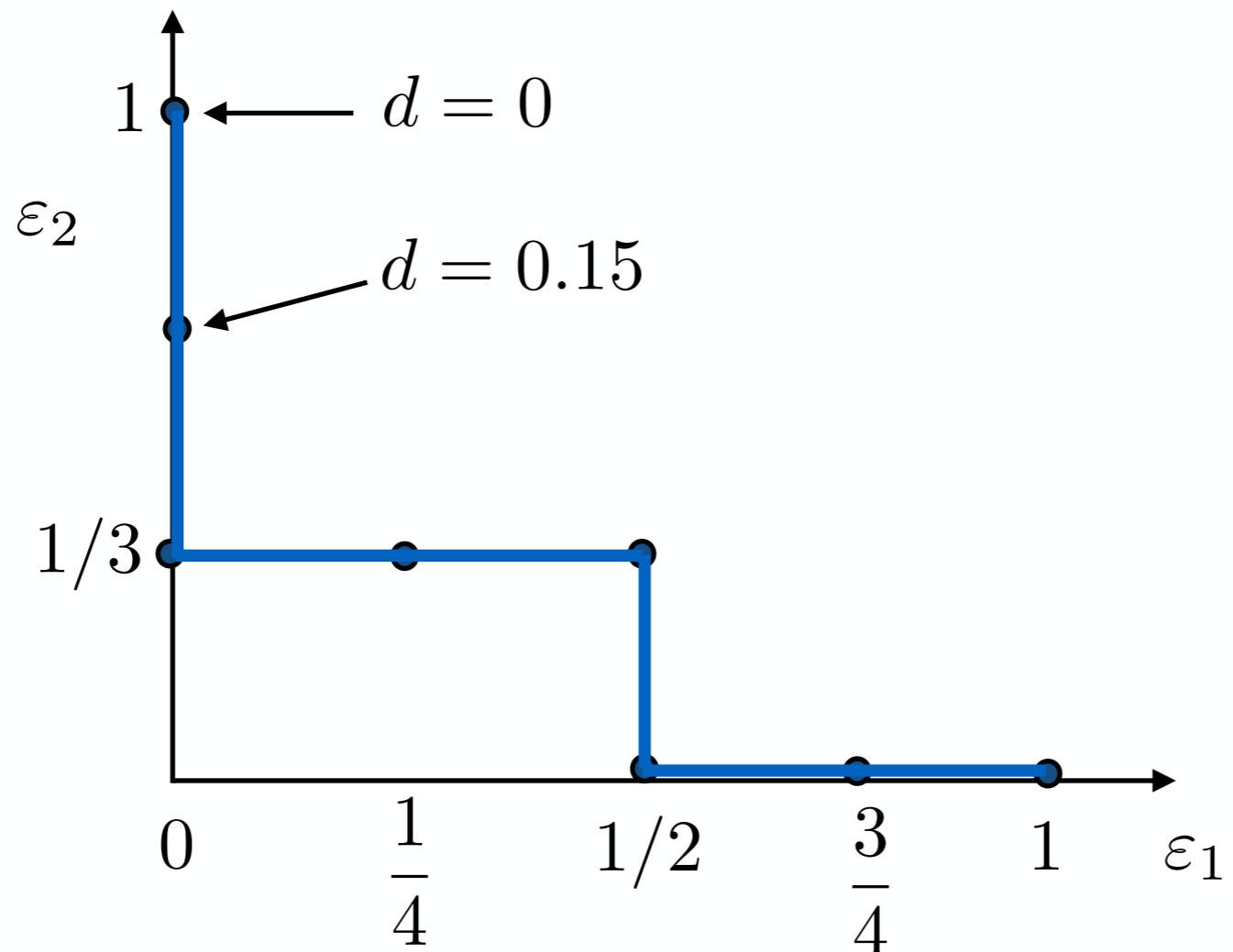
# Example to compute ROC curve



# Example to compute ROC curve



# Example to compute ROC curve

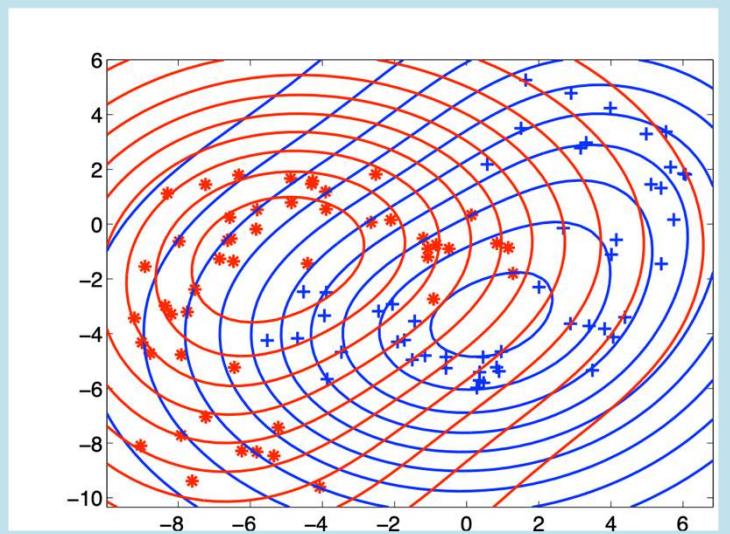
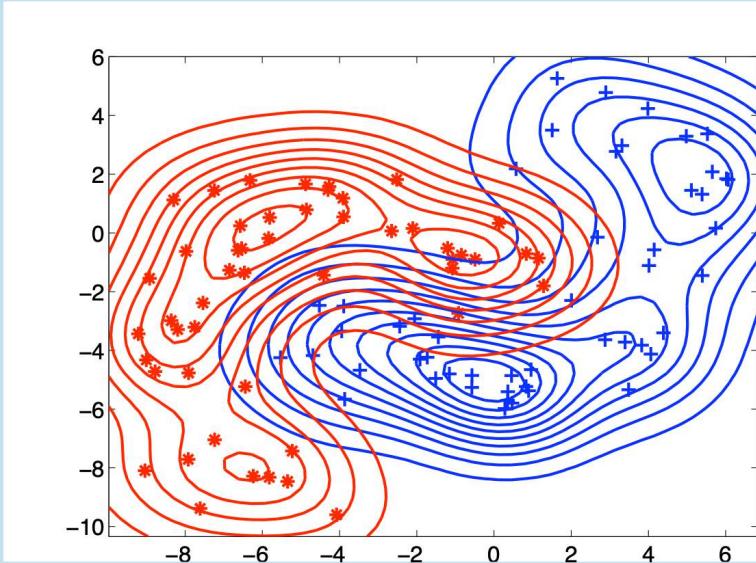
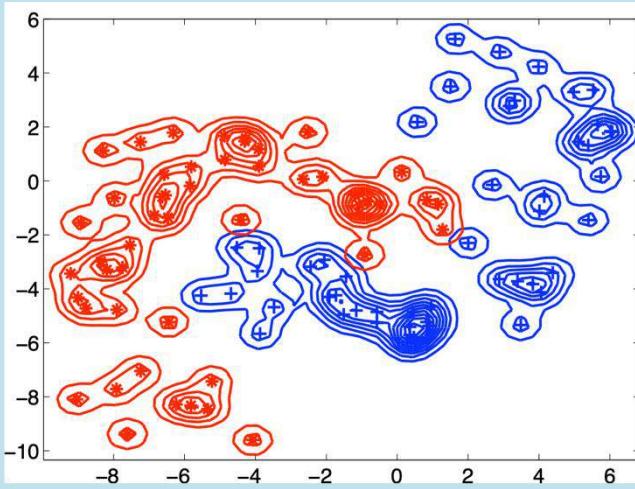


# Conclusions

- There is no best classifier
- There are alternative principles to find a good classifier:
  - maximising the likelihood
  - minimising the classification error
  - minimising the mean squared error
  - ...
- There is a fundamental tradeoff between the bias and variance of a classifier (depending on how flexible/complex a classifier is)

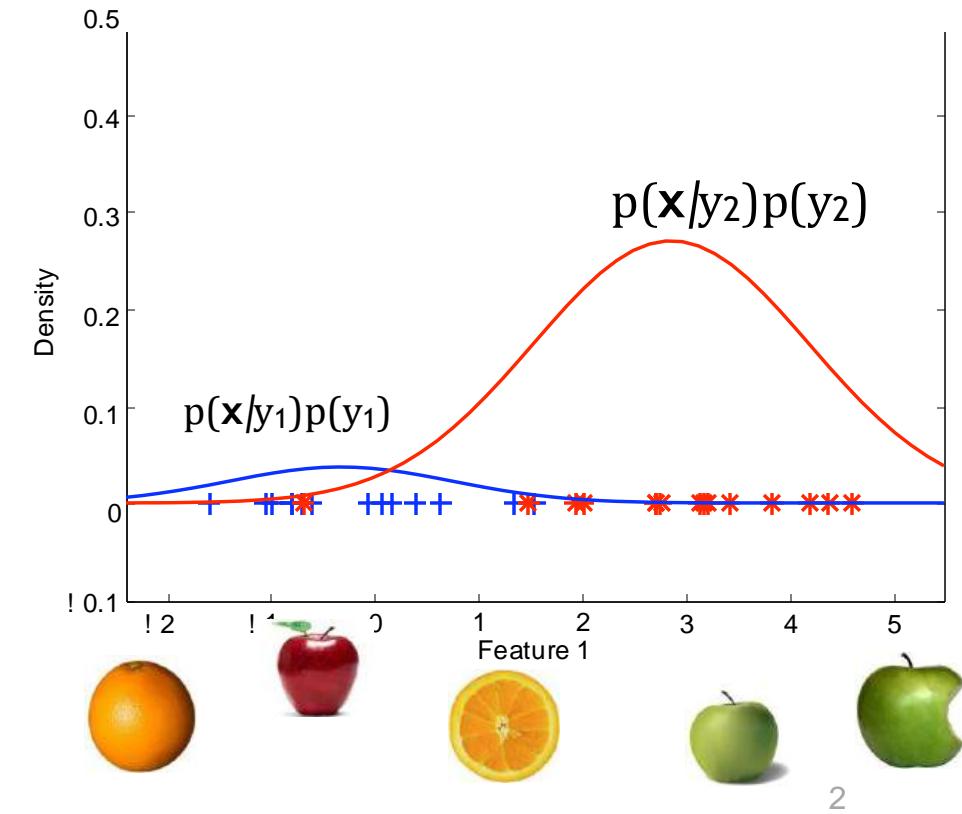
# Non-parametric density estimation

Gosia Migut



# Last week: parametric density estimation

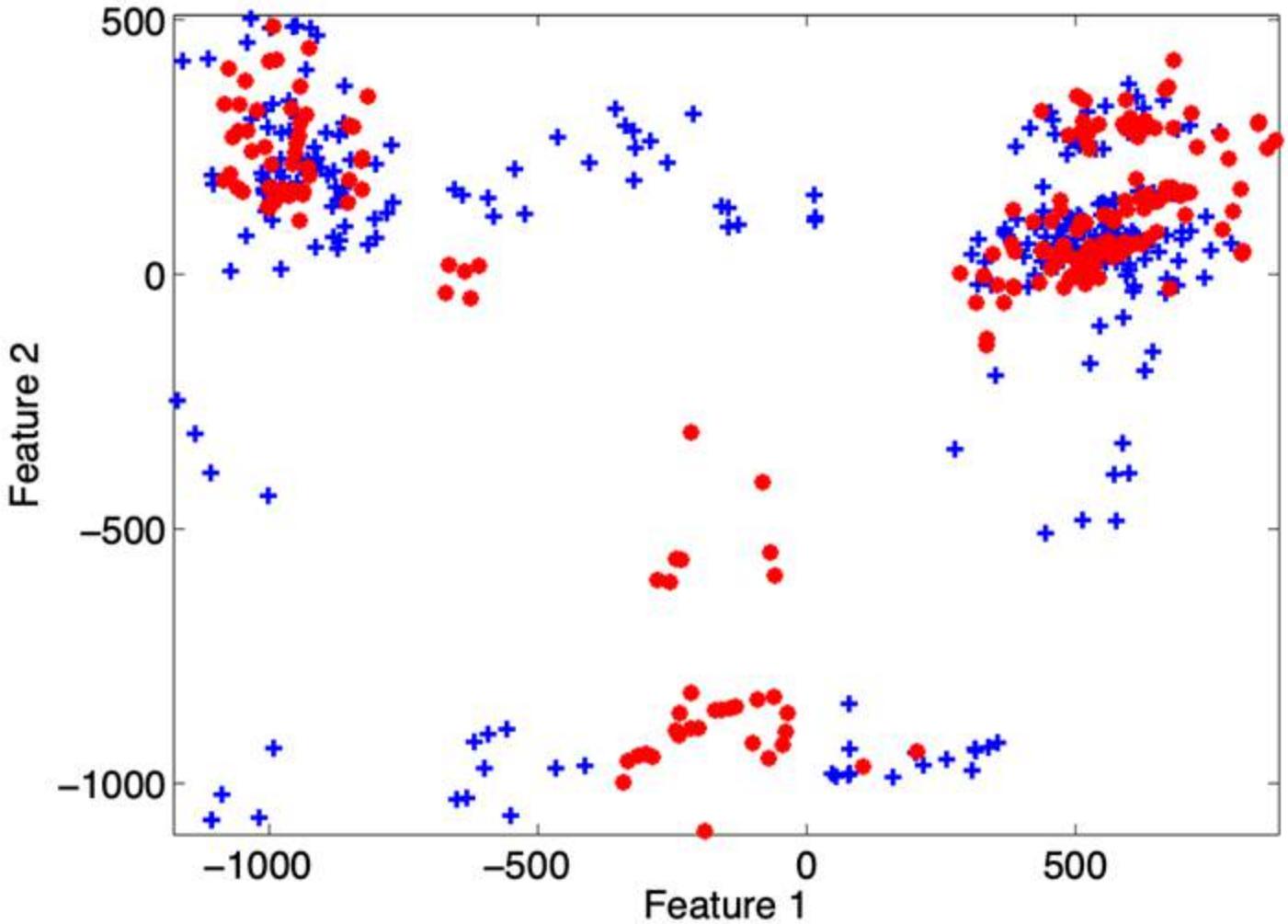
- Known distribution, eg.: assume a single Gaussian distribution for each of the classes:
  - $\hat{p}(x|y_i) = N(x|\mu_i, \Sigma_i)$
- Estimate the **global** parameters on training set, eg.:
  - estimate  $\mu_i$  and  $\Sigma_i$  for each of the classes
- For classification use Bayes rule
  - $p(x|y_1)p(y_1) > p(x|y_2)p(y_2)$



# The real life...

Q: Which distribution to assume?

A: We don't know the distribution of data = no global parameters to estimate



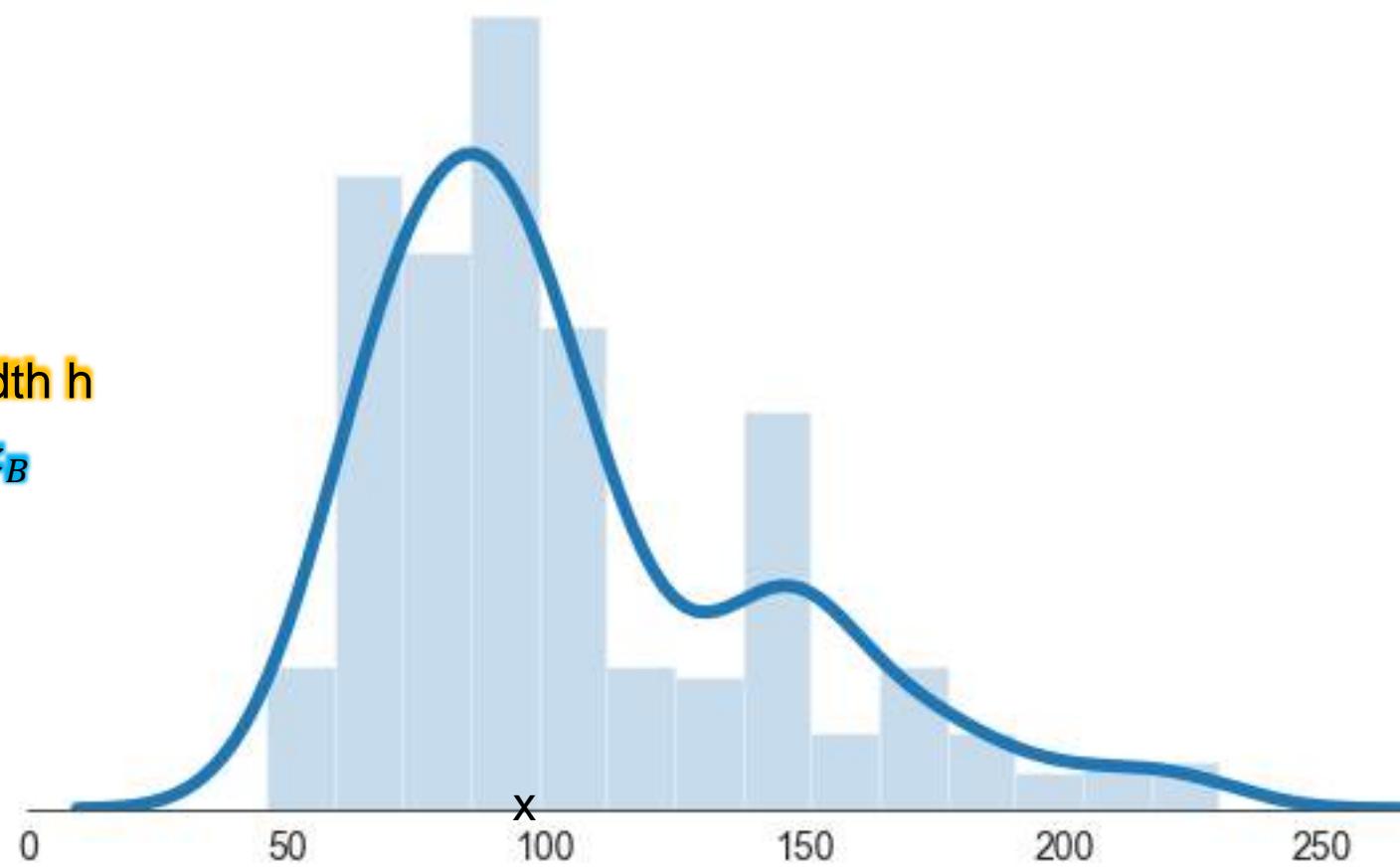
After practicing with the concepts of this lecture you should be able to:

- Explain the difference between parametric and non-parametric density estimation
- Explain Parzen, k-Nearest Neighbour and Naïve Bayes density estimation and classification in detail.
- Explain the advantages and disadvantages of those methods.
- Implement k-nn classifier in Python

# Simple non-parametric density estimation

- Example: we have one feature and  $N$  samples
- How to estimate the probability density?
  - Histogram
- 
- Split the feature in subregions (bins) of width  $h$
- Count the number of objects in each bin:  $k_B$
- Probability density estimate at point  $x$ :

$$\hat{p}(x) = \frac{1}{h} \frac{k_B}{N}$$



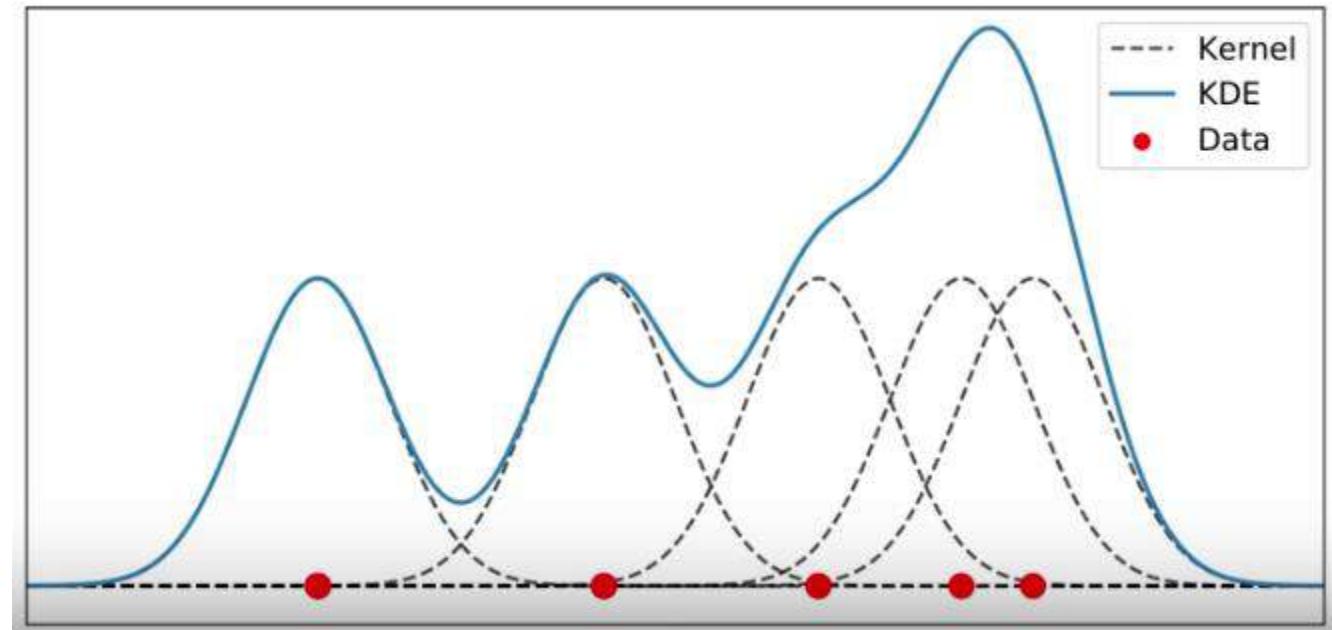
# Can we do better than histogram?

- Histogram puts all samples between boundaries of each bin.
- Bins location is arbitrary (no unique solution).
- In practice, two very related methods are used:
  - **Parzen (kernel) density estimate**
  - k-Nearest-neighbor density estimate (theory + lab)

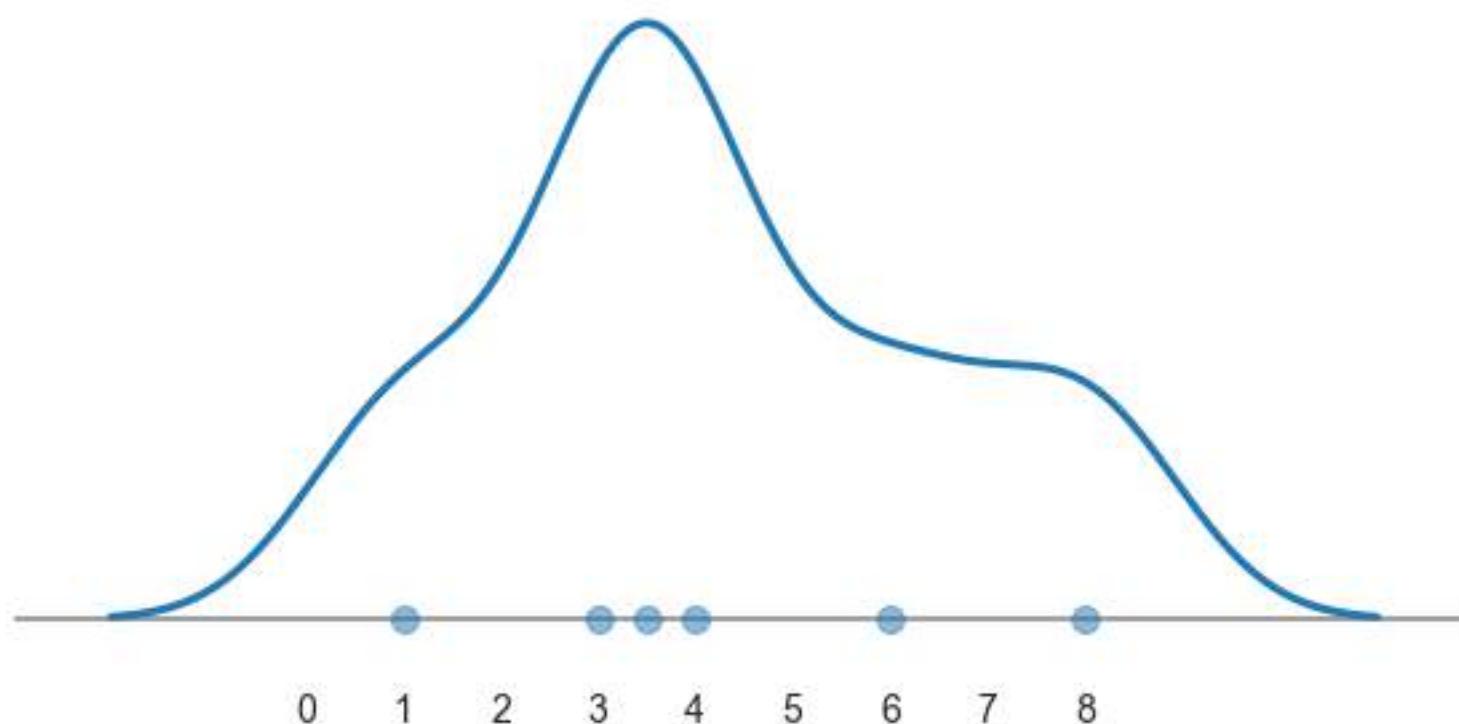
# Parzen (window) Density Estimation

or

# Kernel Density Estimation (KDE)

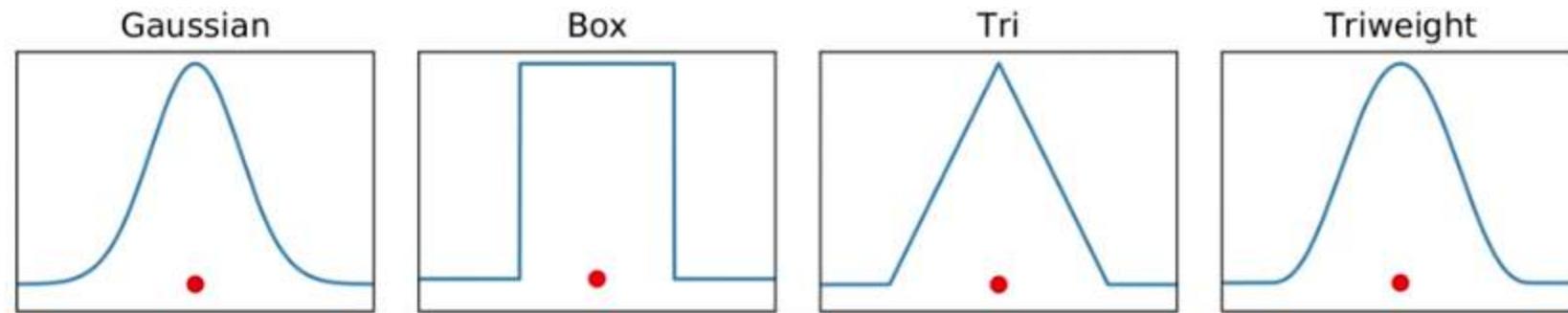


# Parzen density estimation: intuition



# Parzen density estimation: intuition

- Define cell shape (kernel/window function), eg. Gaussian
- Fix size of kernel function ( $h$ ), eg.  $\sigma^2 = 1$



- Q: Is the KDE with a box kernel the same as a histogram?

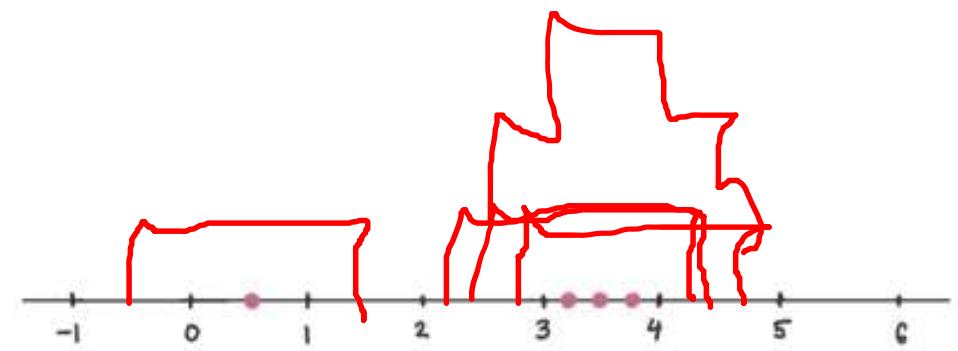
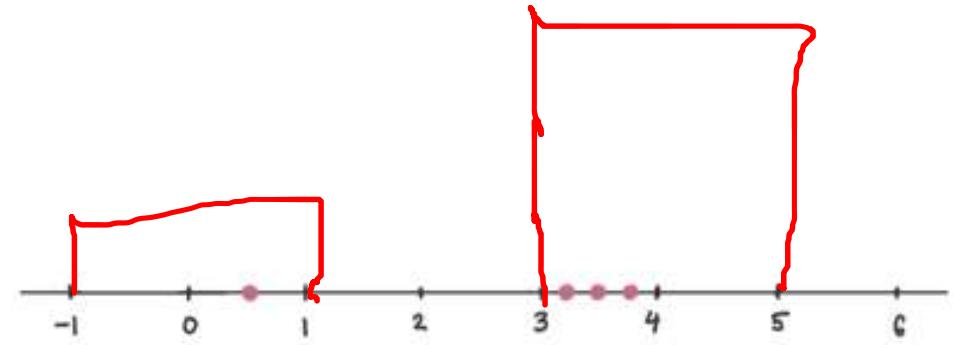
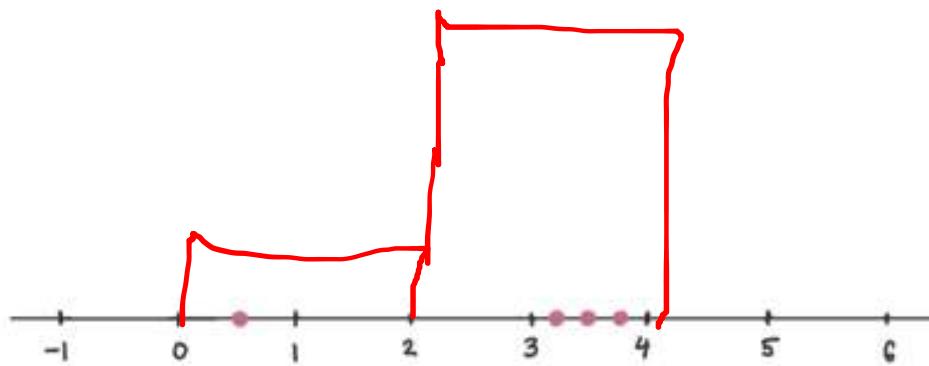
# Lets do some drawing

- Draw histogram with bin size 2
- Draw KDE with box kernel with  $h=2$



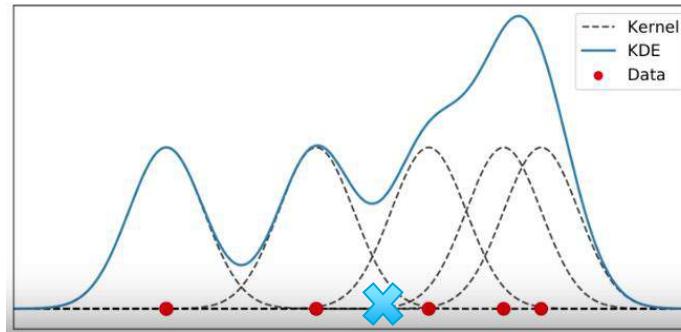
# Lets do some drawing

- Draw histogram with bin size 2
- Draw KDE with box kernel with  $h=2$



# Parzen density estimation

- Fixed shape and size of kernel function ( $h$ )
- On every datapoint  $x_i$  place a kernel function  $K$ .



- How to find Parzen probability density function estimate at  $x$ ?

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

# Question

Given a set of four data points:

$$x_1 = 2, x_2 = 2.5, x_3 = 3.5, x_4 = 0.5$$

find Parzen probability density function (pdf)

estimates at  $x=3$  using the kernel function with width  $h=1$ :

$$K(x) = \begin{cases} 0.5 & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases}$$

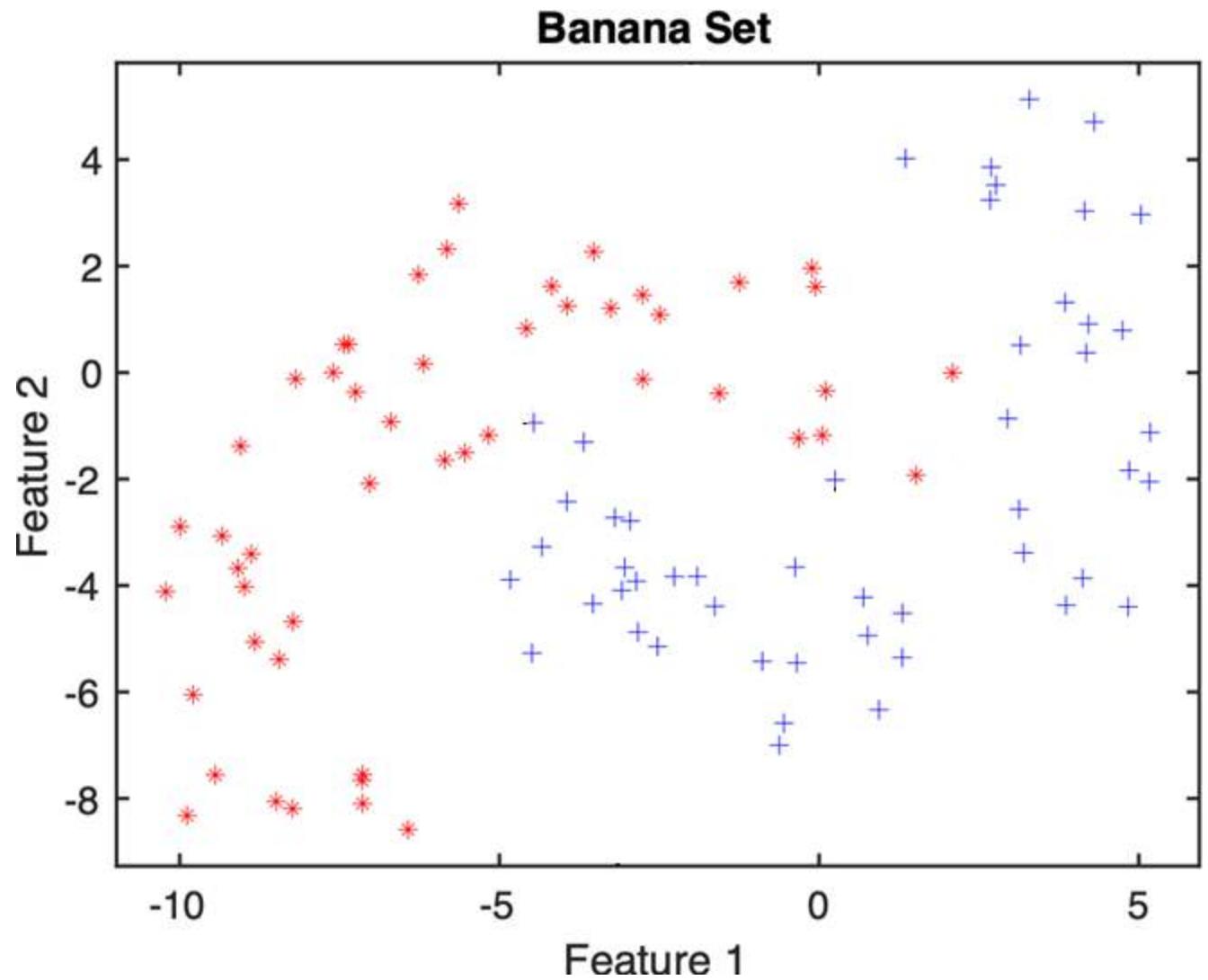
Parzen pfd:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

# Solution

- $x_1 = 2, x_2 = 2.5, x_3 = 3.5, x_4 = 0.5$
- $x = 3$  and  $h = 1$
- $K(x) = \begin{cases} 0.5 & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases}$
- $\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) = \frac{1}{4} (K\left(\frac{3-2}{1}\right) + K\left(\frac{3-2.5}{1}\right) + K\left(\frac{3-3.5}{1}\right) + K\left(\frac{3-0.5}{1}\right)) = \frac{1}{4} (0 + 0.5 + 0.5 + 0) = \frac{1}{4}$

# Classification with Parzen density estimation

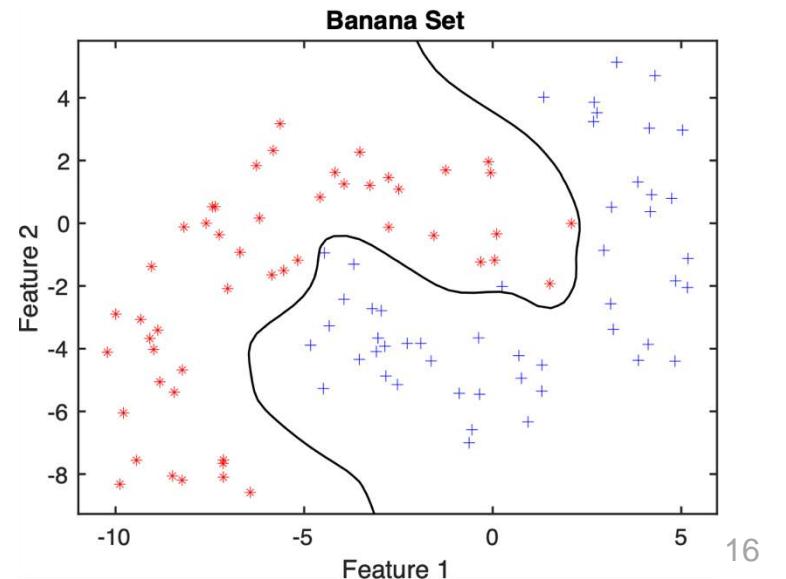
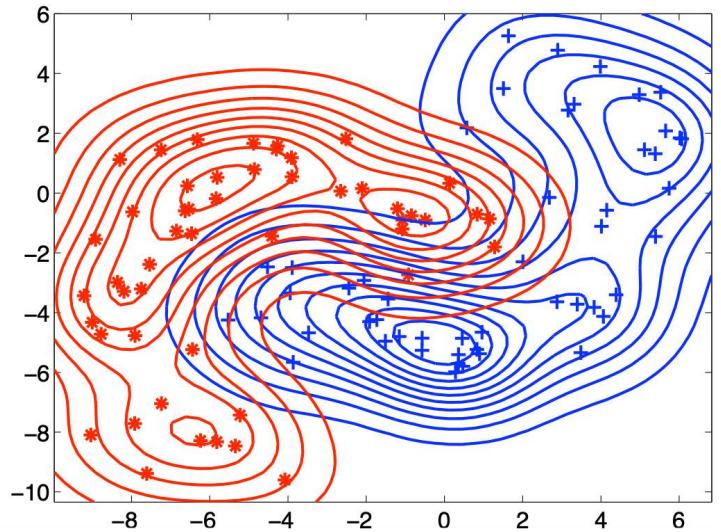


# Parzen classifier

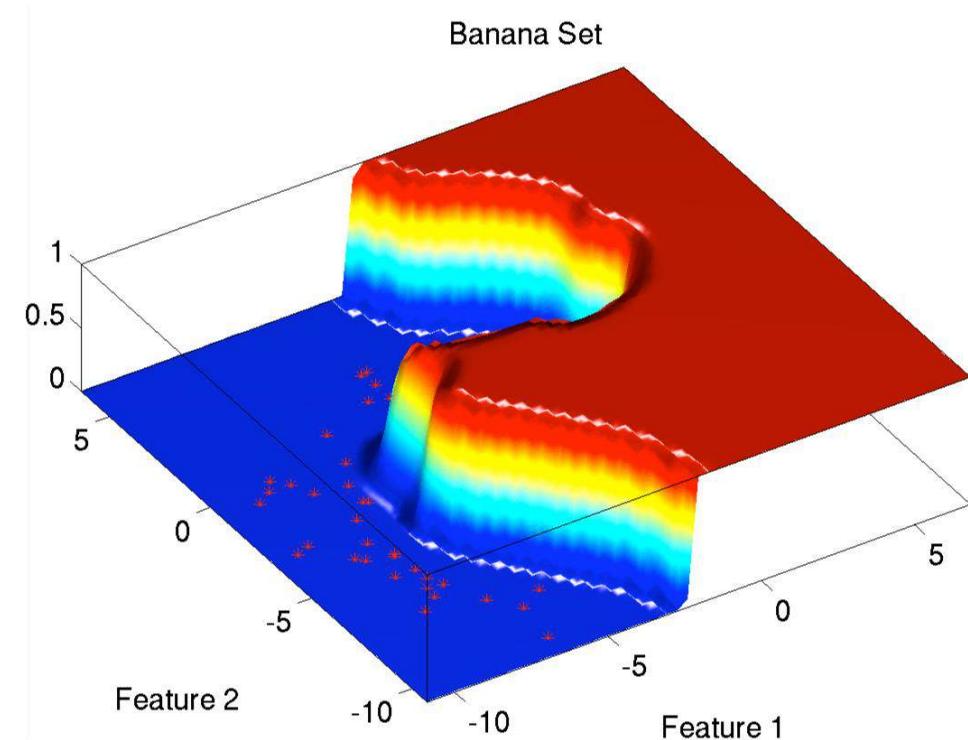
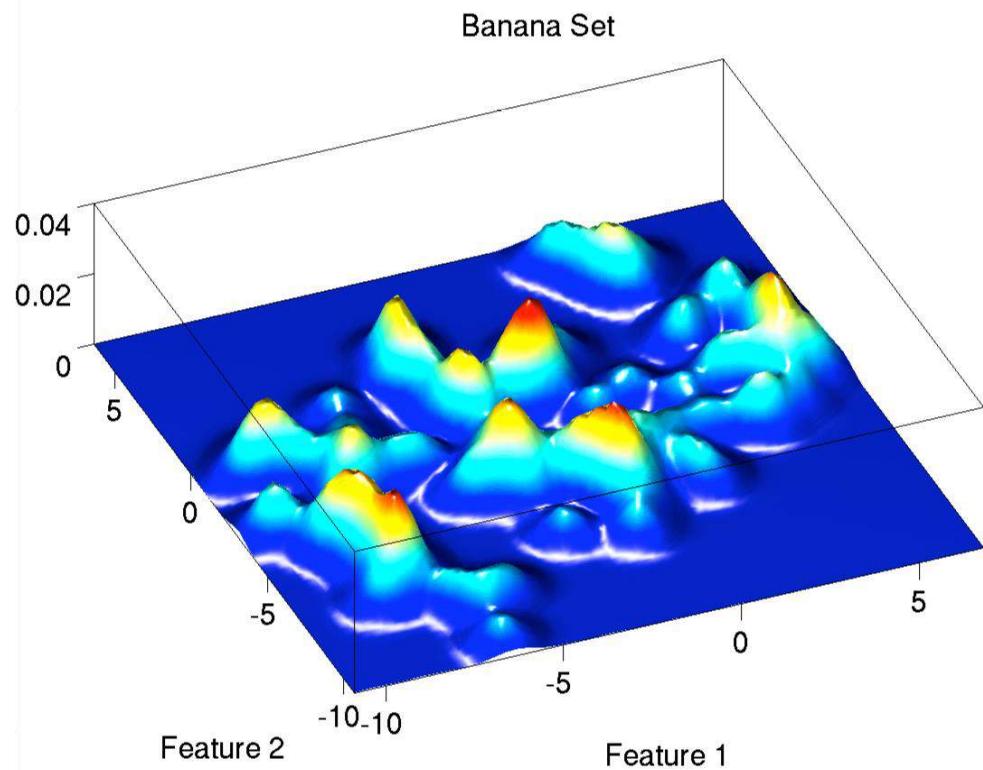
- Gaussian kernel and identity matrix as covariance matrix

$$(x|y_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} N(x|x_j^{(i)}, hI)$$

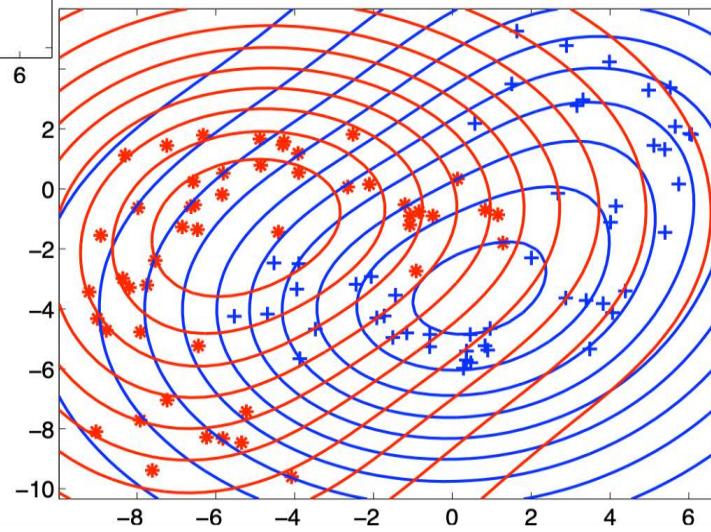
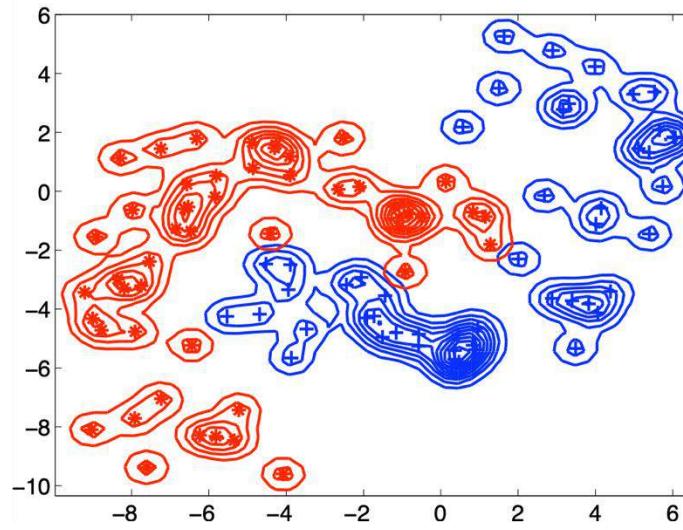
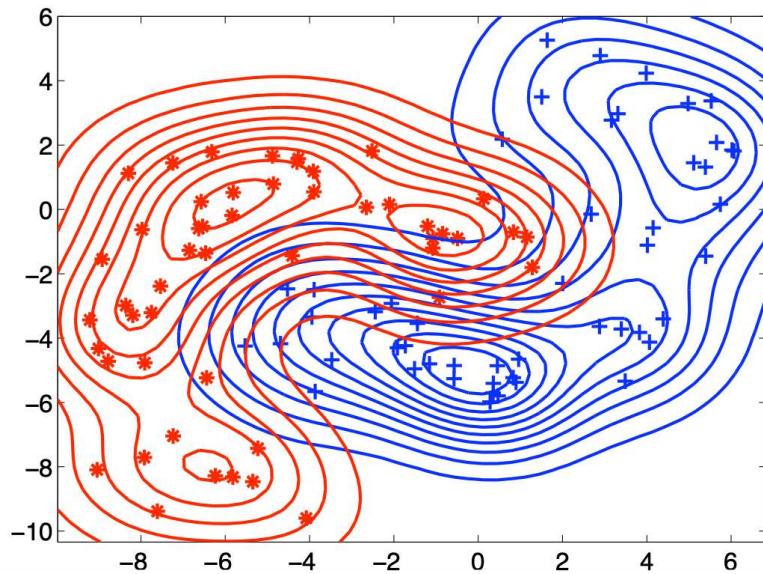
- For classification use Bayes rule
  - $p(x|y_1)p(y_1) > p(x|y_2)p(y_2)$



# Parzen classifier



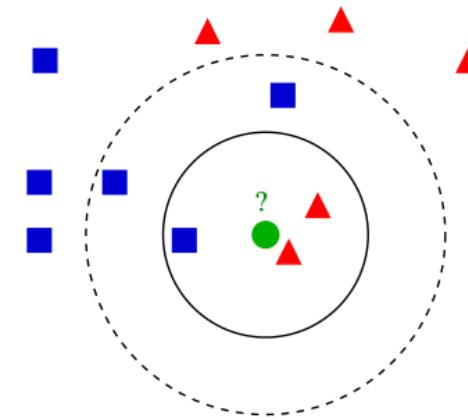
# Does h matter? Intuition on parzen width parameter



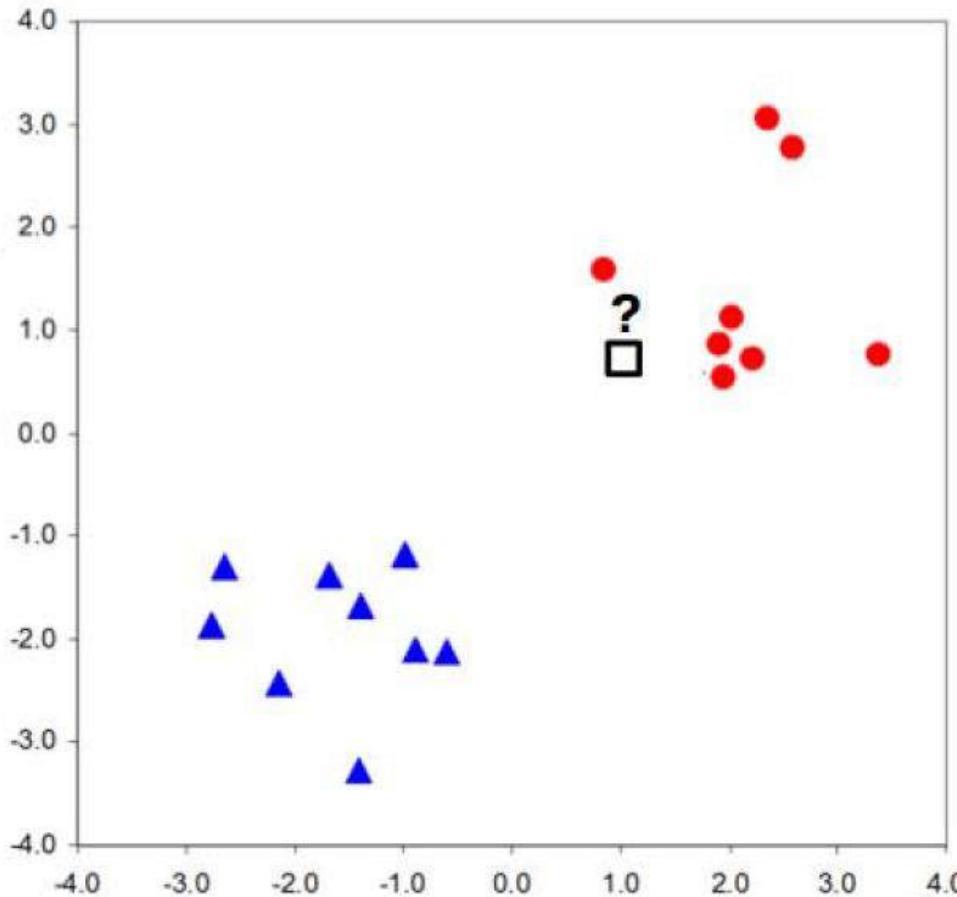
# Summary of Parzen/Kernel density estimation

- Does not assume known distribution
- Estimates probability densities using kernel function
- Uses kernel function of fixed shape and width
- Width matters

# K-nearest Neighbours



# K-nearest neighbour intuition

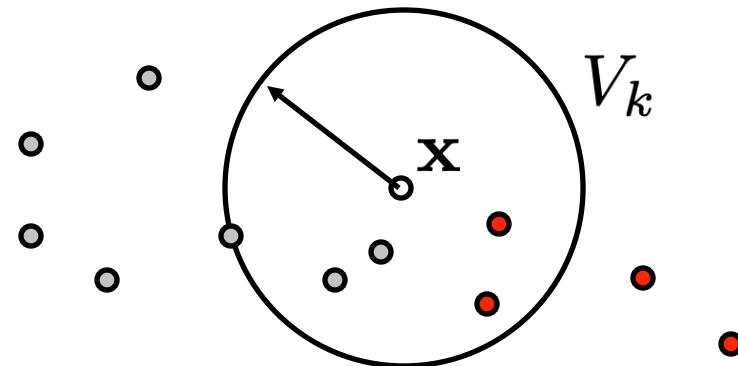


- Is the box red or blue?
- How did you do it?
- Nearby points are red

# K-nearest neighbour intuition

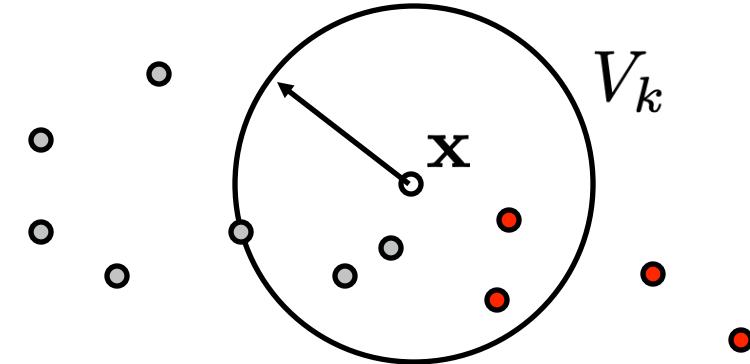
- Use the intuition to classify a new point  $x$ :
  - Locate the cell on the new point  $x$
  - Do **not** fix the volume of the cell:  
grow the cell until it covers  $k$  objects:  
find the  $k$ -th neighbours
  - predict the class  $y$  of new point  $x$

$k=5$



# K-nn density estimation

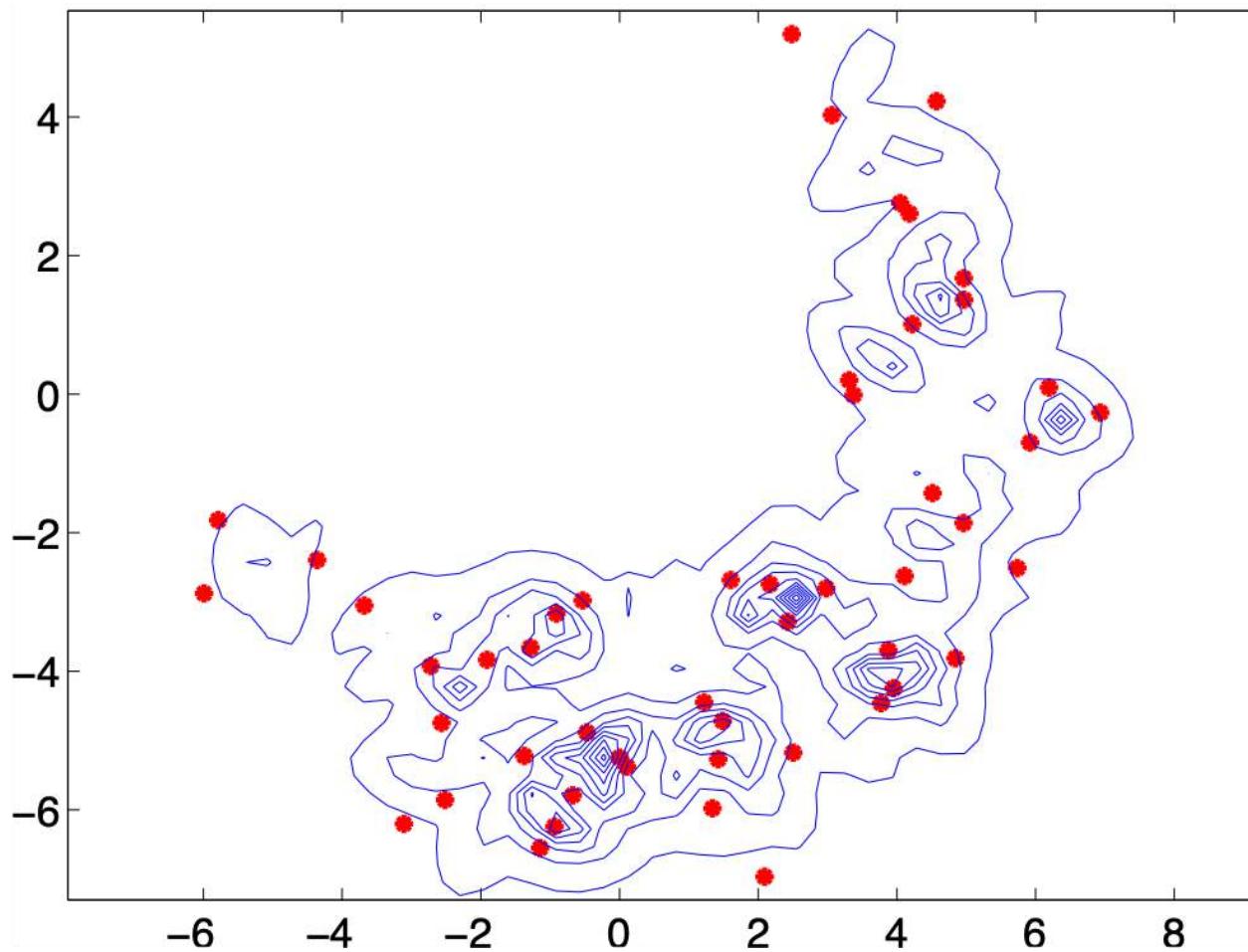
- $k = 5$  ( $k_1 = 3$ ,  $k_2 = 2$ )



$$\hat{p}(x|y_i) = \frac{k_i}{n_i V_k}$$

- Where  $V_k$  is the volume of the sphere centered at  $x$  with radius  $r$ , being the distance to the k-th nearest neighbor;  $k_i$  is the number of neighbours of class  $i$  within  $V_k$
- Class priors:  $\hat{p}(y_i) = \frac{n_i}{n}$
- Bayes:  $\hat{p}(x|y_i)\hat{p}(y_i) > \hat{p}(x|y_j)\hat{p}(y_j) \rightarrow k_i > k_j$

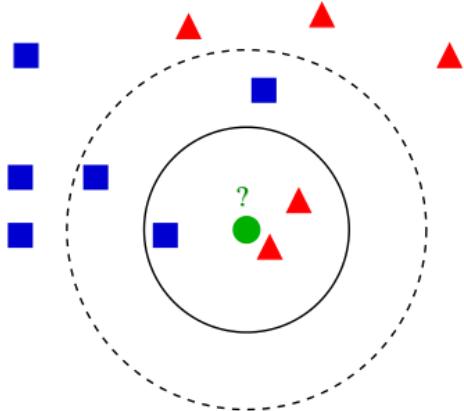
# K-nn density estimate



# K-nn classification algorithm (lab)

- Given:
  - training examples  $\{x_i, y_i\}$ 
    - $x_i$  attribute-value representation of examples
    - $y_i$  class label: {male, female}, digit {0,1, ... 9} etc.
  - testing point  $x$  that we want to classify
- Algorithm:
  - compute distance  $D(x, x_i)$  to every training example  $x_i$
  - select  $k$  closest instances  $x_{i1} \dots x_{ik}$  and their labels  $y_{i1} \dots y_{ik}$
  - output the class  $y^*$  which is most frequent in  $y_{i1} \dots y_{ik}$  (**majority vote**)

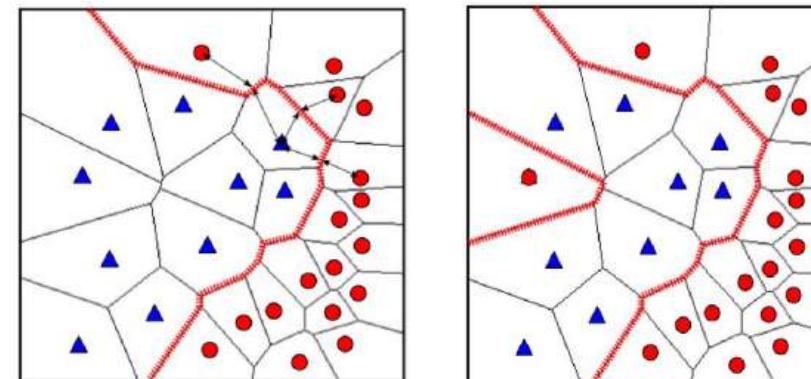
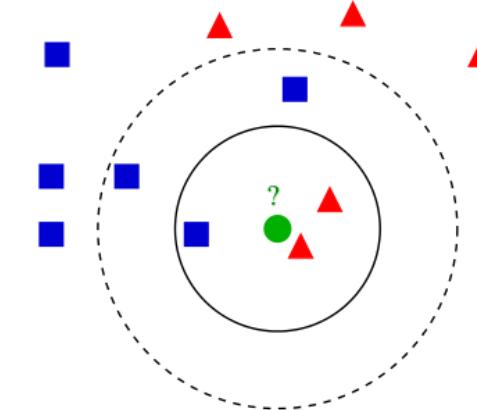
# What is the influence of k?



- What is the largest/smallest value of k that you can choose?
  - What will be the classification error then?

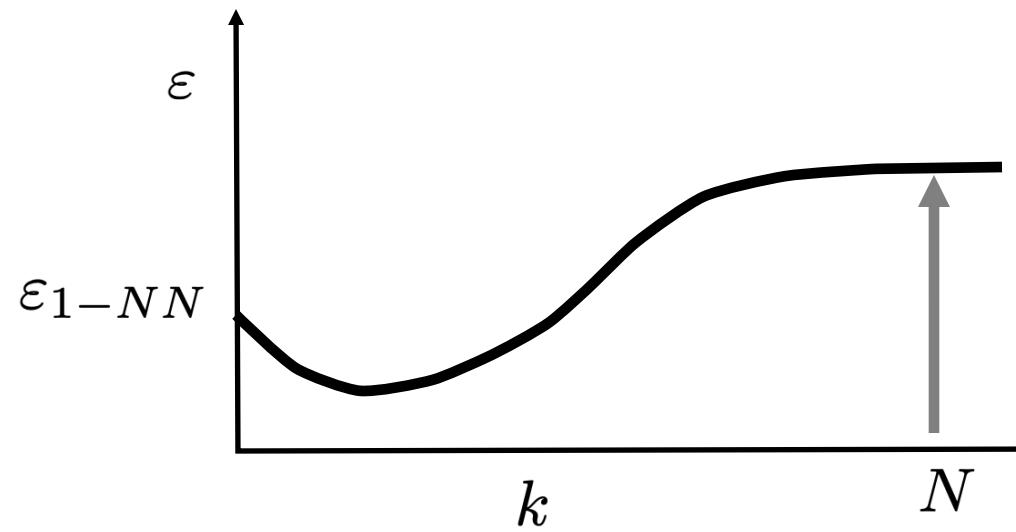
# What is the influence of k?

- Value of k has strong effect on k-nn performance
  - Large value → everything classified as the most probable class
  - Small value → highly variable, unstable decision boundaries, eg. for 1-nn:



# Choosing the value of k

- Selecting the value of k
  - set aside a portion of the training data (validation set)
  - vary k
  - Pick k that gives best generalization performance



# K-nn resolving ties

- Equal number of positive/negative neighbours ?
- Resolving ties:
  - use odd k (doesn't solve multi-class)
  - breaking ties:
    - random: flip the coin to decide positive/negative
    - prior: pick class with greater prior
    - nearest: use 1-nn classifier to decide

# Distance measures

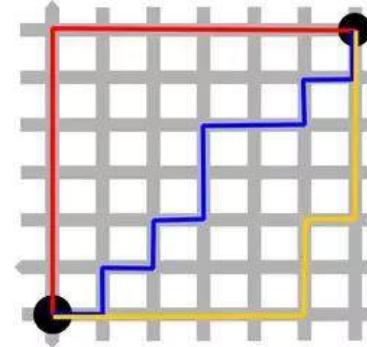
- The key component of the kNN algorithm
  - defines which examples are similar and which aren't
  - can have strong effect on performance
- Euclidean (numeric features):

$$D(x, x') = \sqrt{\sum_d |x_d - x'_d|^2}$$

# Distance measures

- Manhattan distance

$$D(x, x') = \sum_d |x_d - x'_d|$$



- Hamming (categorical features):
  - number of features where  $x$  and  $x'$  differ

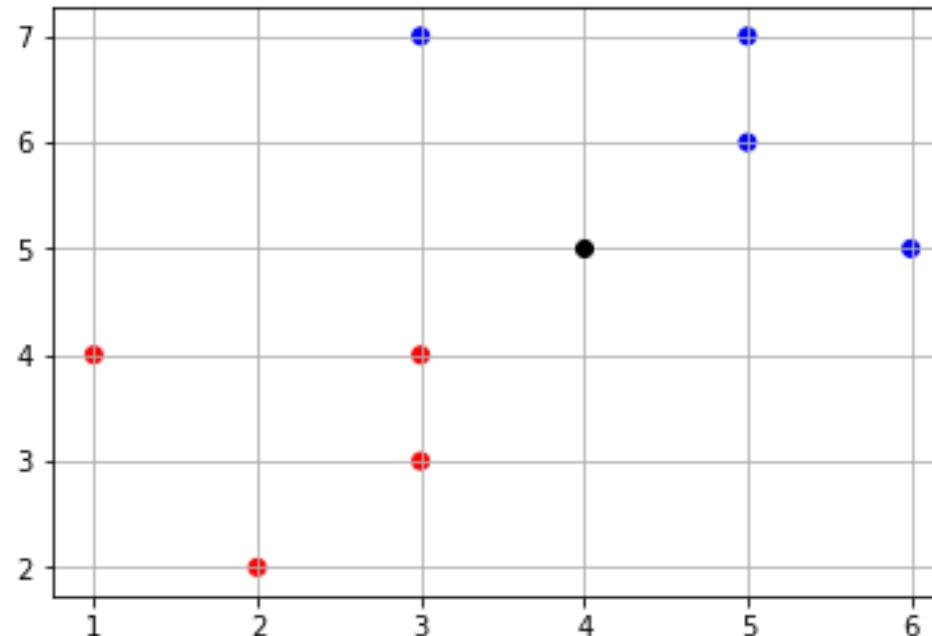
$$D(x, x') = \sum_d 1_{x_d \neq x'_d}$$

- Other (out of scope), eg.:
  - Kullback-Leibler (KL) divergence (for histograms)
  - Custom distance measures (BM25 for text)

# K-nn example

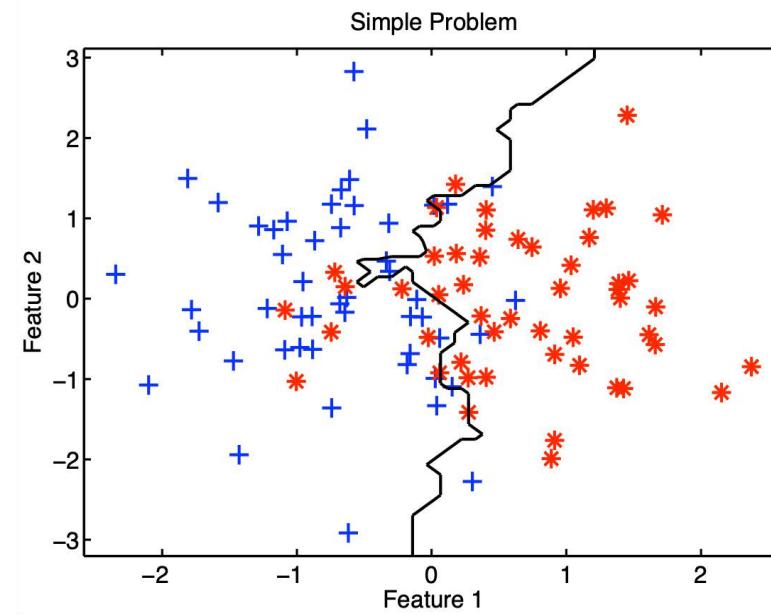
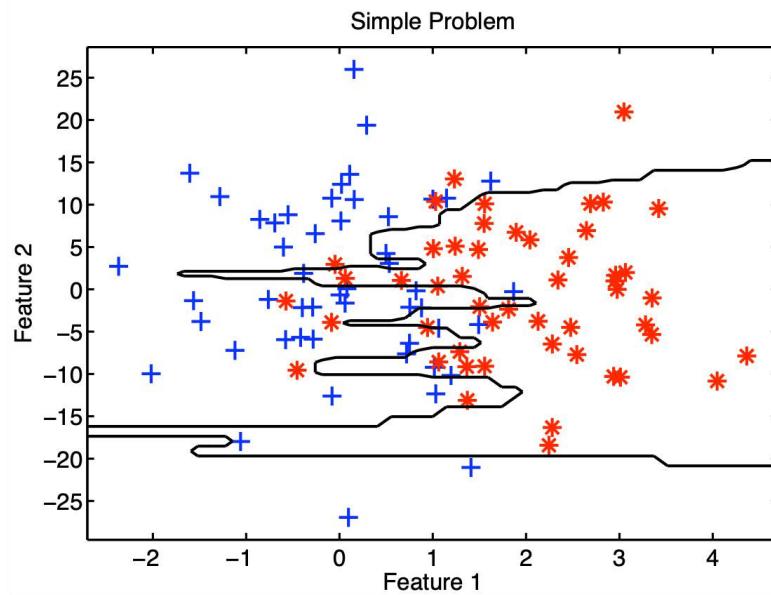
- Given a labeled two-dimensional data set:
  - Red label: (1,4); (2,2); (3,3); (3,4);
  - Blue label: (3,7); (5,7); (5,6); (6,5);
- Predict the label of a new black point (4, 5) using 3-nn classifier with Manhattan distance.

- A. Red label  
B. Blue label



# Sometimes strange results

- How is this possible?



Scale your features!

# K-nn pros and cons

- Simple and flexible classifiers
- often a very good classification performance
- it is simple to adapt the complexity of the classifier
- relatively large training sets are needed
- the complete training set has to be stored
- distances to all training objects have to be computed
- the features have to be scaled sensibly
- the value for k has to be optimized

# Naive Bayes Classifier

# Recap Bayes classifier

- For classification we need  $p(y|x)$
- We can use Bayes' theorem if we can estimate  $p(y)$  and  $p(x|y)$

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

- Assigning an object to the class with the maximum posterior probability gives the Bayes' classifier

$$p(x|y_1)p(y_1) > p(x|y_2)p(y_2)$$

## Warming up question

- Suppose we have trained a generative model and now get a new test example  $x$ . Our model tells us that:  $p(x|y_0) = 0.01$ ,  $p(x|y_1) = 0.03$  and  $p(y_0) = p(y_1) = 0.5$
- What is  $p(y_1|x)$ ?
  - A. 0.015
  - B. 0.25
  - C. 0.75
  - D. Insufficient information to compute. We also need to know the  $p(x)$ .

# Solution

- $p(y_1|x) = \frac{p(x|y_1)p(y_1)}{p(x)}$
- $p(x) = p(x|y_1)p(y_1) + p(x|y_0)p(y_0)$
- $p(y_1|x) = \frac{0.03*0.5}{0.03*0.5+0.01*0.5} = 0.75$

# Density estimation

- So, we want to estimate a class conditional probability density function:

$$p(x|y)$$

- Typically, each feature vector  $\mathbf{x}$  has many features:

$$p(x|y) = p(x_1, x_2, x_3, x_4, \dots, x_d | y)$$

- To estimate this joint pdf (conditional on the class), we need LOTS of data... (curse of dimensionality)

# Naive Bayes: conditional independence assumption

- We make a strong assumption: all features are independent
- We assume conditional independence given  $y$

# Conditional independence example

- We assume conditional independence of two variables given a third variable.
- Example: probability of going to the beach and having a heartstroke may be independent if we know the weather is hot

$$p(B, S|H) = p(B|H)p(S|H)$$

- Hot weather “explains” all the dependence between beach and heartstroke
- In classification: class value explains all the dependence between features

# Naive Bayes: conditional independence assumption

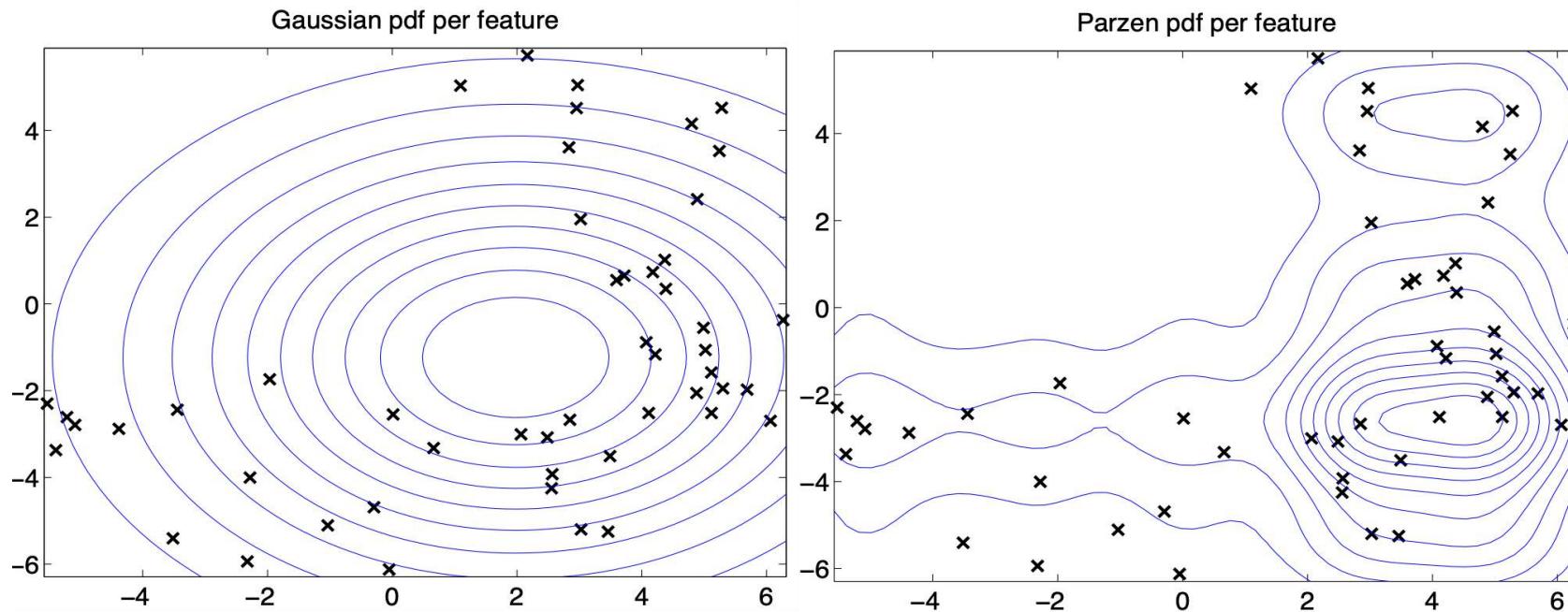
- We make a strong assumption: all features are independent
- We assume conditional independence given  $y$
- We just estimate  $p(x_i|y)$  per feature and multiply them.

$$\begin{aligned} p(x|y) &= p(x_1, x_2, x_3, x_4, \dots, x_d|y) = \prod_{i=1}^d p(x_i|y) \\ &= p(x_1|y)p(x_2|y) \dots p(x_d|y) \end{aligned}$$

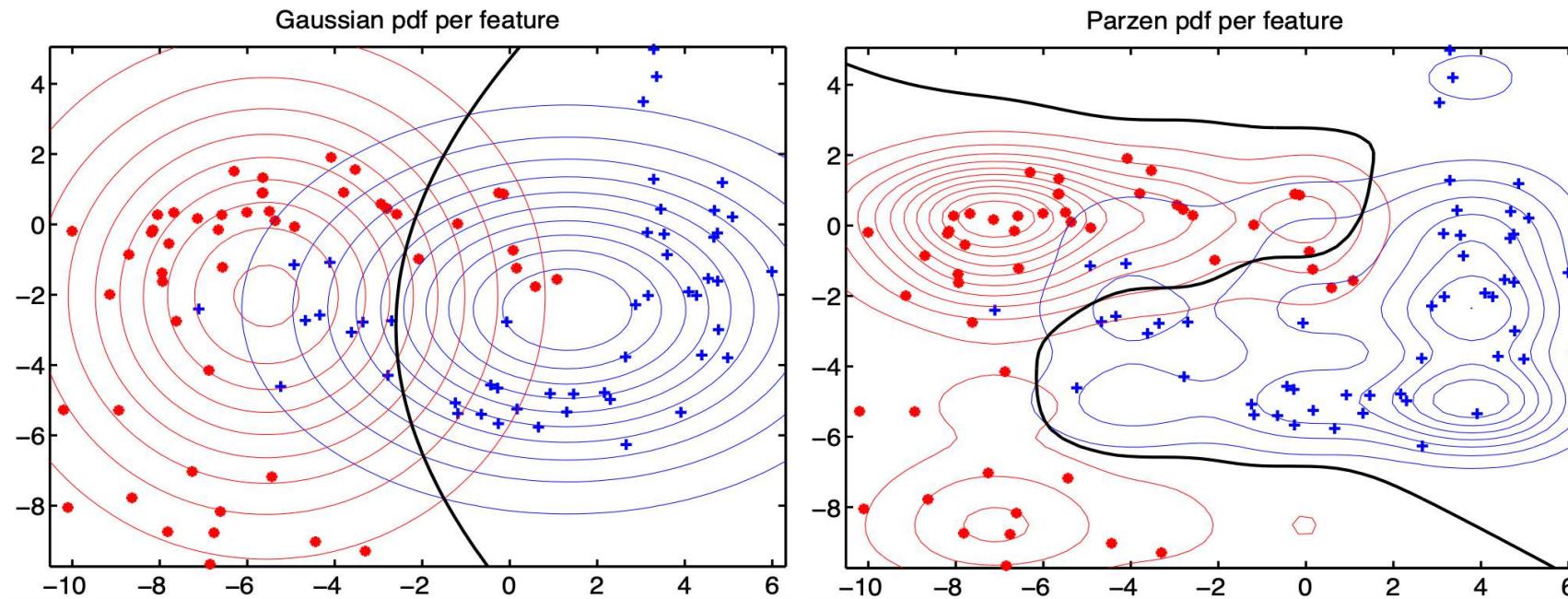
- No curse of dimensionality!

# Parametric vs. non-parametric

- You still have to choose a model for  $p(x_i|y)$



# Naive Bayes classifier



# Continuous data example

- Distinguish children from adults based on size
  - Classes:  $y = \{a, c\}$ , features:  $x = \{\text{height (cm)}, \text{weight (kg)}\}$
  - Training examples: 4 adults, 12 children
- Class probabilities  $p(a) = \frac{4}{4+12} = 0.25$ ,  $p(c) = 0.75$

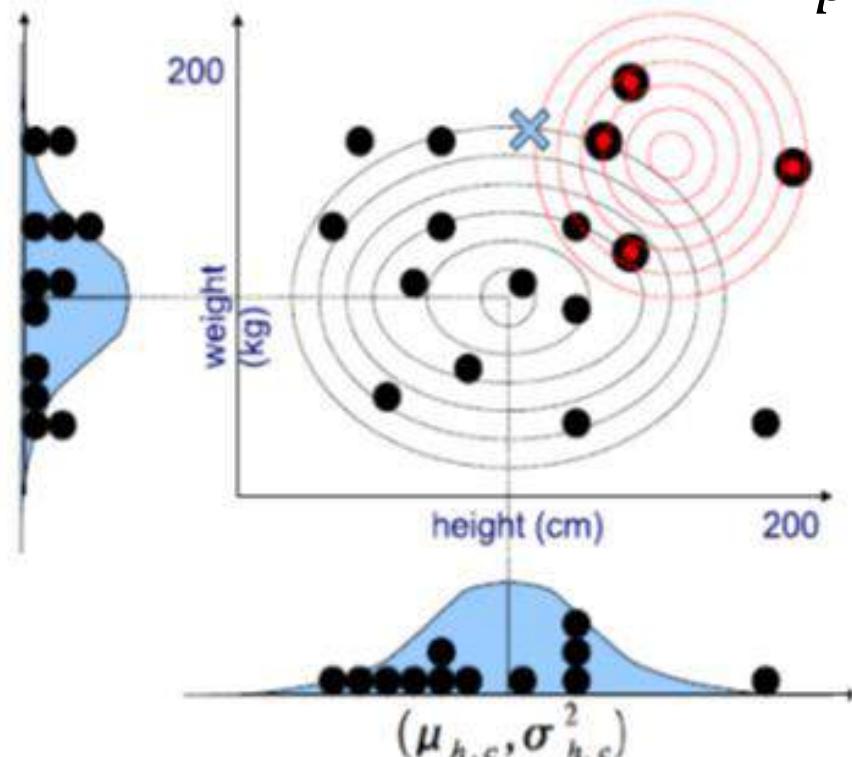
- Model for adults:

- Assume height and weight are independent
- Height, estimate Gaussian with mean, variance

$$\begin{cases} \mu_{h,a} = \frac{1}{4} \sum_{i:y_i=a} h_i \\ \sigma_{h,a}^2 = \frac{1}{4} \sum_{i:y_i=a} (h_i - \mu_{h,a})^2 \end{cases}$$

- Weight, estimate Gaussian  $(\mu_{w,a}, \sigma_{w,a}^2)$
- Model for children: use  $(\mu_{h,c}, \sigma_{h,c}^2), (\mu_{w,c}, \sigma_{w,c}^2)$

# Continuous example



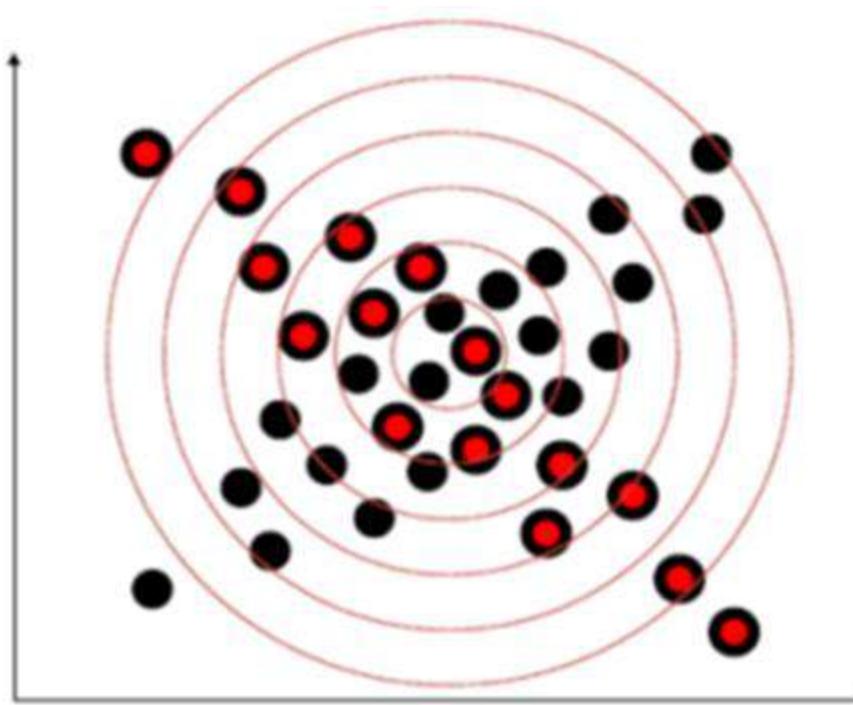
$$p(w|a) = \frac{1}{\sqrt{2\pi\sigma_{w,a}^2}} \exp - \left( \frac{w - \mu_{w,a}}{2\sigma_{w,a}^2} \right)^2$$

$$p(h|a) = \frac{1}{\sqrt{2\pi\sigma_{h,a}^2}} \exp - \left( \frac{h - \mu_{h,a}}{2\sigma_{h,a}^2} \right)^2$$

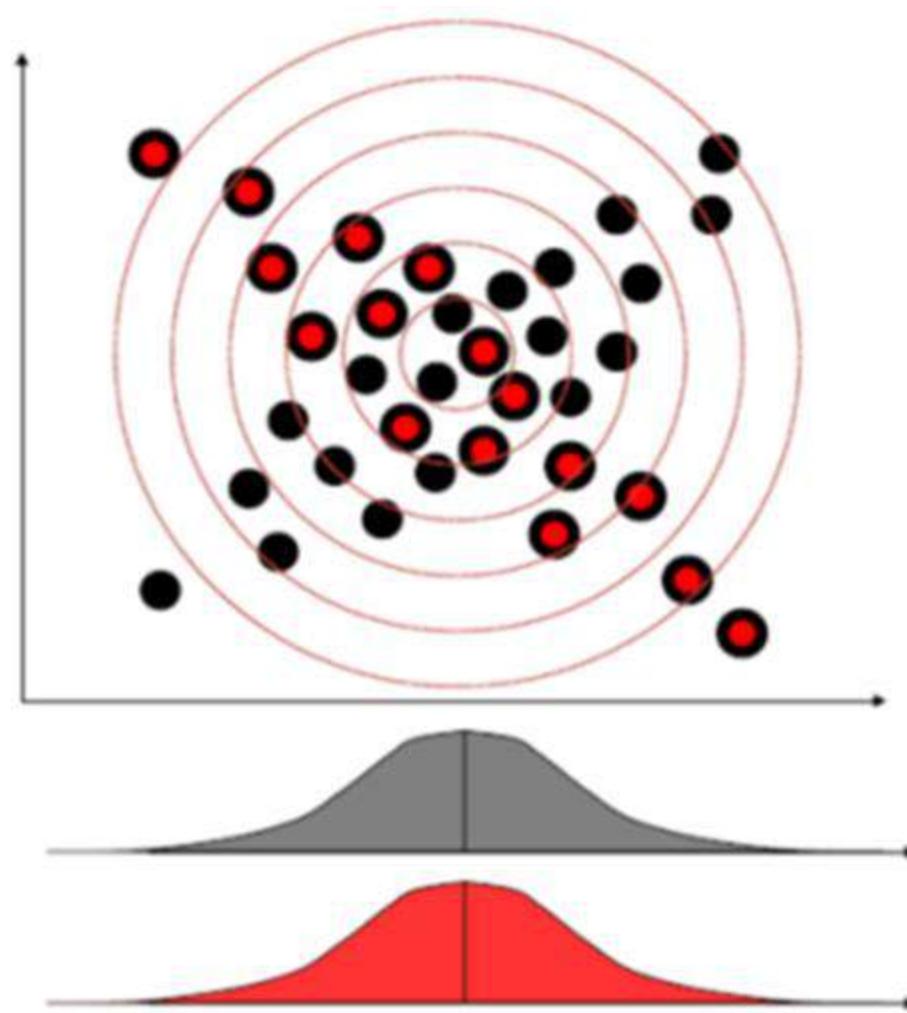
$$p(x|a) = p(w|a)p(h|a)$$

$$p(a|x) = \frac{p(x|a)p(a)}{p(x)}$$

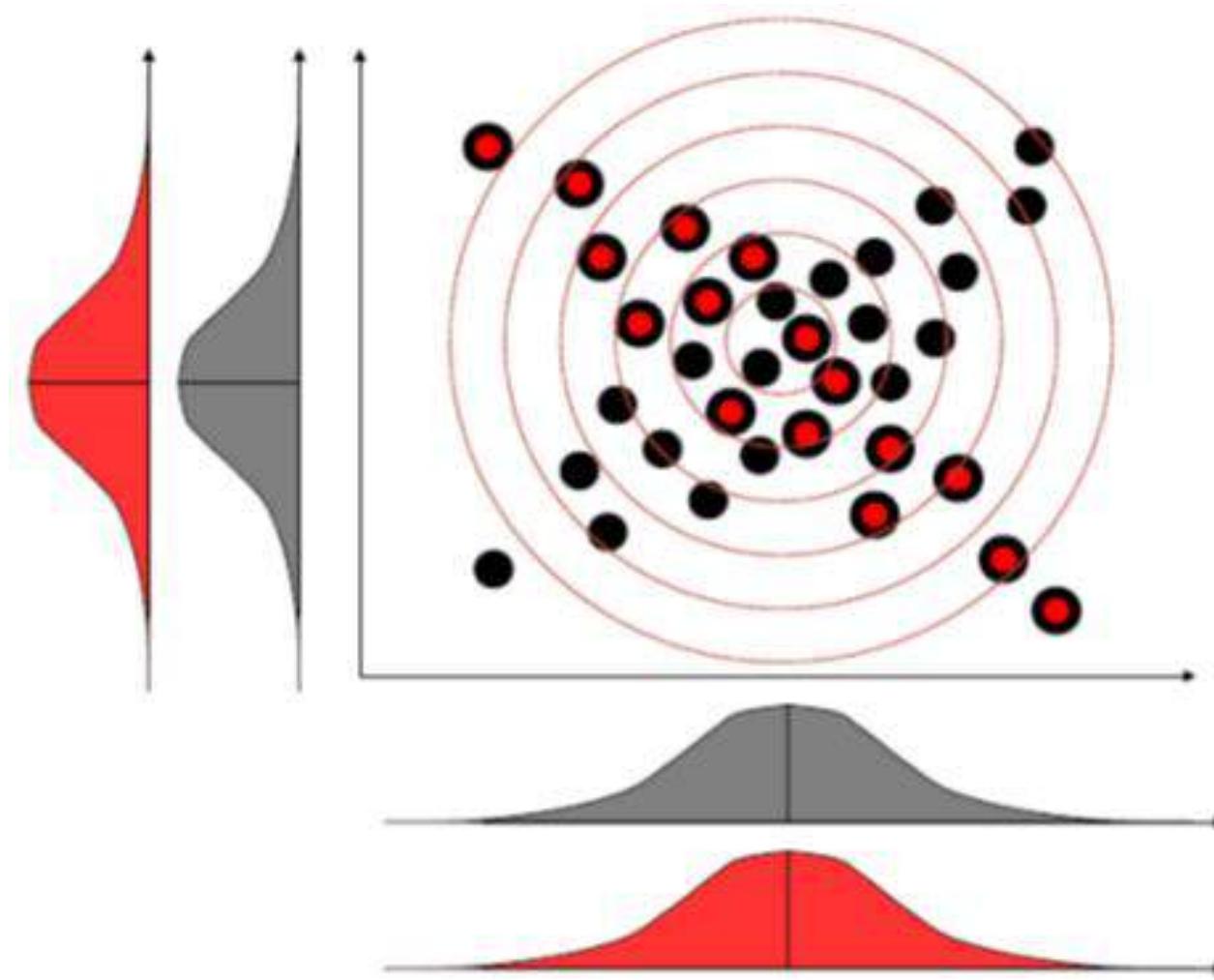
# Problem with Naive Bayes



# Problem with Naive Bayes



# Problem with Naive Bayes



# Discrete example

- Task: separate spam from valid email (features = words)

Email	Label
D1: "send us your password"	spam
D2: "send us review"	valid
D3: "review your password"	valid
D4: "review us"	spam
D5: "send your password"	spam
D6: "send us your account"	spam

$p(\text{spam}) = 4/6 \ p(\text{valid}) = 2/6$		
	spam	valid
Password	2/4	$\frac{1}{2}$
Review	1/4	2/2
Send	3/4	$\frac{1}{2}$
Us	3/4	$\frac{1}{2}$
Your	3/4	$\frac{1}{2}$
Account	1/4	0/2

- New email “review us now”

# Discrete example

- New email: “review us now”
- $p(\text{“review us”}|\text{spam}) = p([0, 1, 0, 1, 0, 0]|\text{spam}) = (1 - \frac{2}{4})(\frac{1}{4})(1 - \frac{3}{4})(\frac{3}{4})(1 - \frac{3}{4})(1 - \frac{1}{4}) = 0.0044$
- $p(\text{“review us”}|\text{valid}) = p([0, 1, 0, 1, 0, 0]|\text{valid}) = \left(1 - \frac{1}{2}\right)\left(\frac{2}{2}\right)\left(1 - \frac{1}{2}\right)\left(\frac{1}{2}\right)\left(1 - \frac{1}{2}\right)\left(1 - \frac{0}{2}\right) = 0.0625$

$p(\text{spam}) = 4/6 \ p(\text{valid}) = 2/6$		
	spam	valid
Password	2/4	½
Review	1/4	2/2
Send	3/4	½
Us	3/4	½
Your	3/4	1/2
Account	1/4	0/2

# Solution

- $p(\text{"review us"}|\text{spam}) = 0.0044$
- $p(\text{"review us"}|\text{valid}) = 0.0625$

$p(\text{spam}) = 4/6 \ p(\text{valid}) = 2/6$		
	spam	valid
Password	2/4	$\frac{1}{2}$
Review	1/4	2/2
Send	3/4	$\frac{1}{2}$
Us	3/4	$\frac{1}{2}$
Your	3/4	$\frac{1}{2}$
Account	1/4	0/2

- $p(\text{"review us"}|\text{spam})p(\text{spam}) = 0.0044 * 4/6 = 0.0029$
- $p(\text{"review us"}|\text{valid})p(\text{valid}) = 0.0625 * 2/6 = 0.02$
- Note: identical example!

# Zero frequency problem

- No email containing “account” is valid  
 $p(\text{“account”}|\text{valid}) = 0/2$

$p(\text{spam}) = 4/6 \ p(\text{valid}) = 2/6$		
	spam	valid
Password	2/4	$\frac{1}{2}$
Review	1/4	2/2
Send	3/4	$\frac{1}{2}$
Us	3/4	$\frac{1}{2}$
Your	3/4	$\frac{1}{2}$
Account	1/4	0/2

- Solution: never allow zero probabilities
  - Laplace smoothing: add a small positive number to the counts ( $K \rightarrow$  number of classes)

$$p(w|c) = \frac{\text{num}(w, c) + \varepsilon}{\text{num}(c) + K\varepsilon}$$

# Fooling Naive Bayes

- Every word contributes independently to  $p(\text{spam}|\text{email})$
- Add lots of valid words into spam email.

# Naive Bayes pros and cons

- Can handle high dimensional feature spaces
- Fast training time
- Can handle continuous and discrete data
- Can't deal with correlated features

# Exercise Naive Bayes

- Predict if Bob will default his loan

Bob:

Homeowner: no

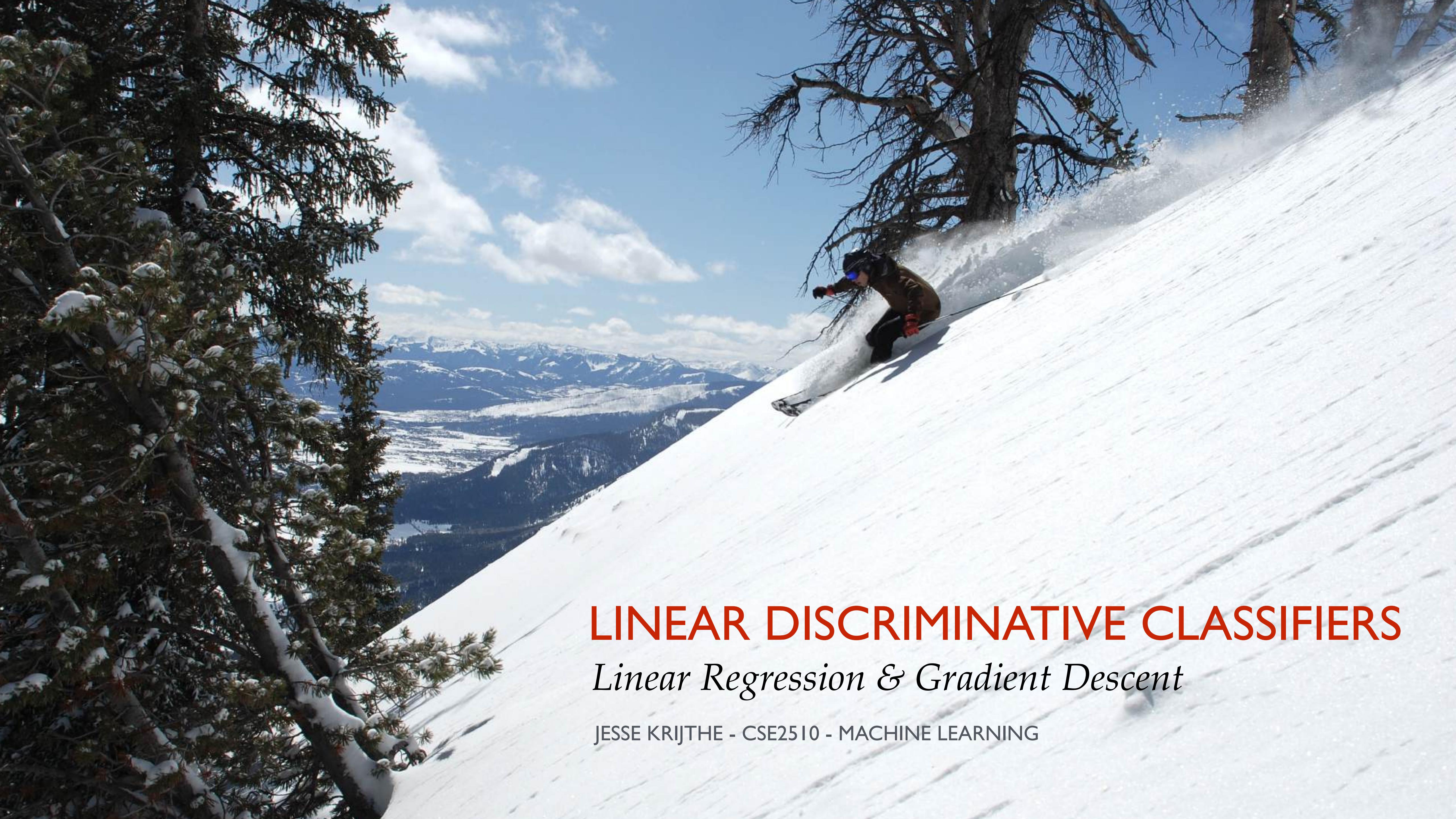
Marital status: married

Job experience: 3

Home owner	Marital status	Job experience	Deafulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

After practicing with the concepts of this lecture you should be able to:

- Explain the difference between parametric and non-parametric density estimation.
- Explain Parzen, k-Nearest Neighbour and Naïve Bayes density estimation and classification in detail.
- Explain the advantages and disadvantages of those methods.
- Implement k-nn classifier in Python.



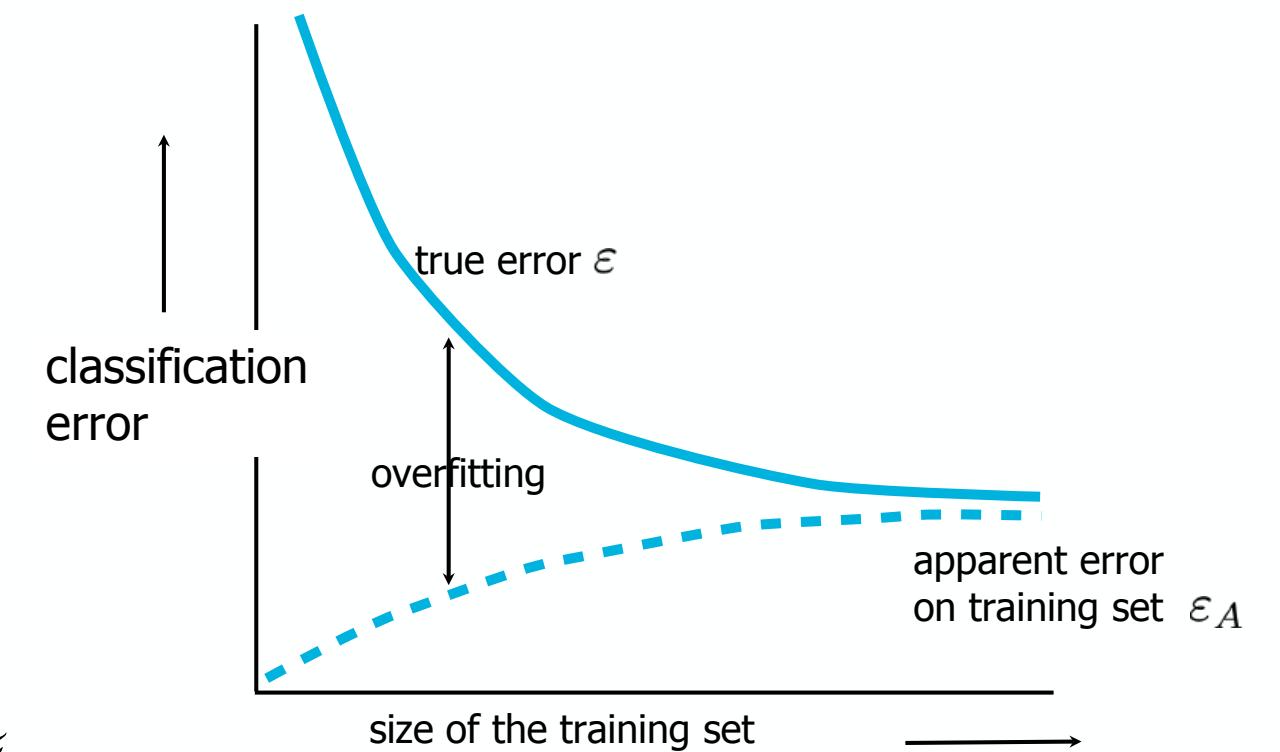
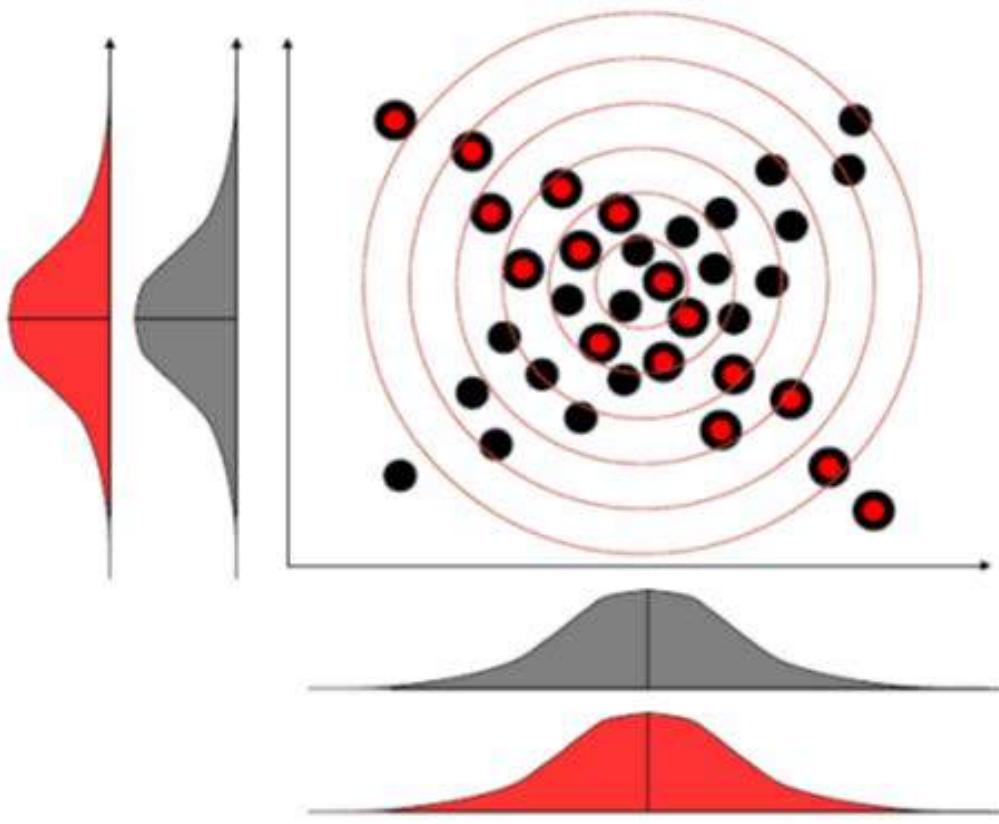
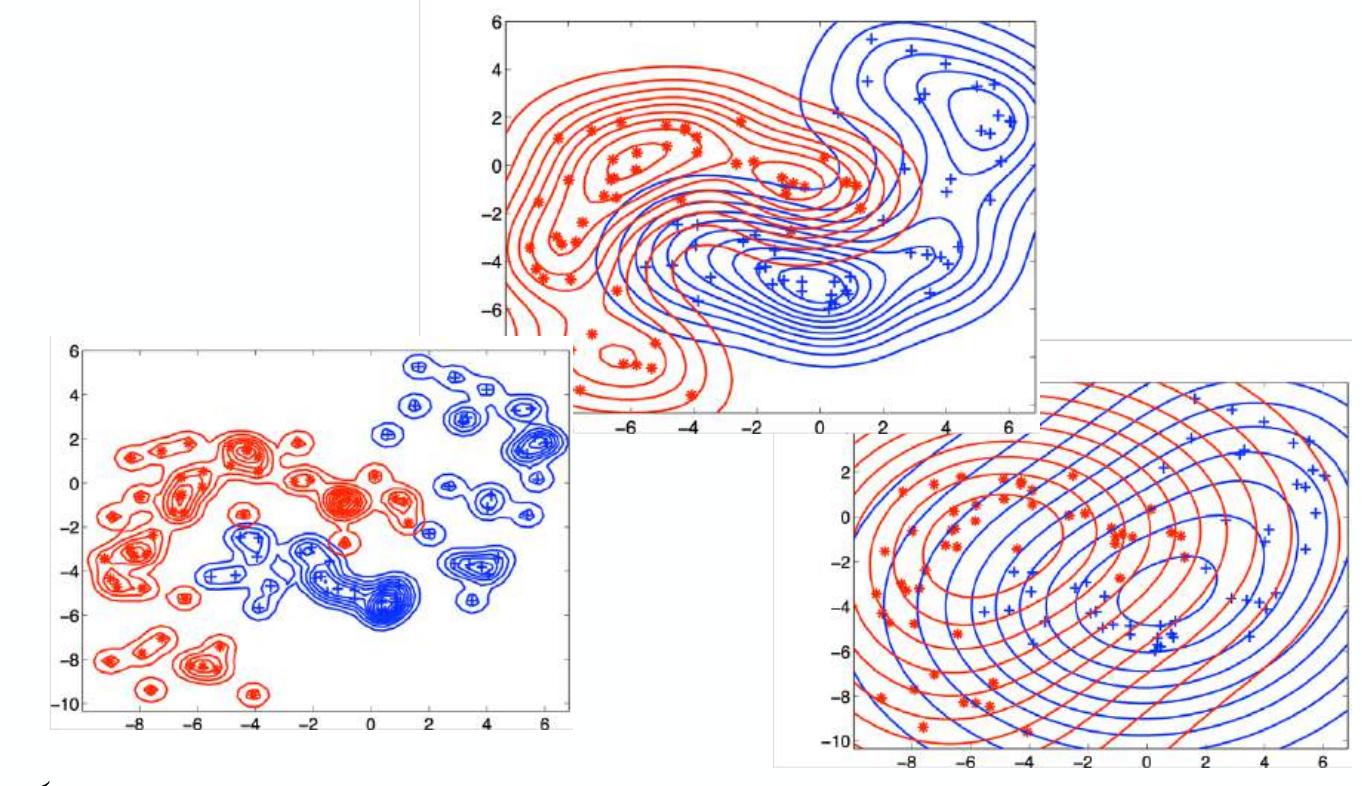
# LINEAR DISCRIMINATIVE CLASSIFIERS

*Linear Regression & Gradient Descent*

JESSE KRIJTHE - CSE2510 - MACHINE LEARNING

# Last week

- Non-parametric density estimation and classifiers: Parzen and k-NN
- (Non)-parametric Naive Bayes classifier
- Classifier Evaluation
- Generalization and Bias / Variance trade-off



TECH \ TWITTER

# Twitter is looking into why its photo preview appears to favor white faces over Black faces

*Users discovered the problem with the neural network that crops photo previews*

By Kim Lyons | Sep 20, 2020, 4:20pm EDT



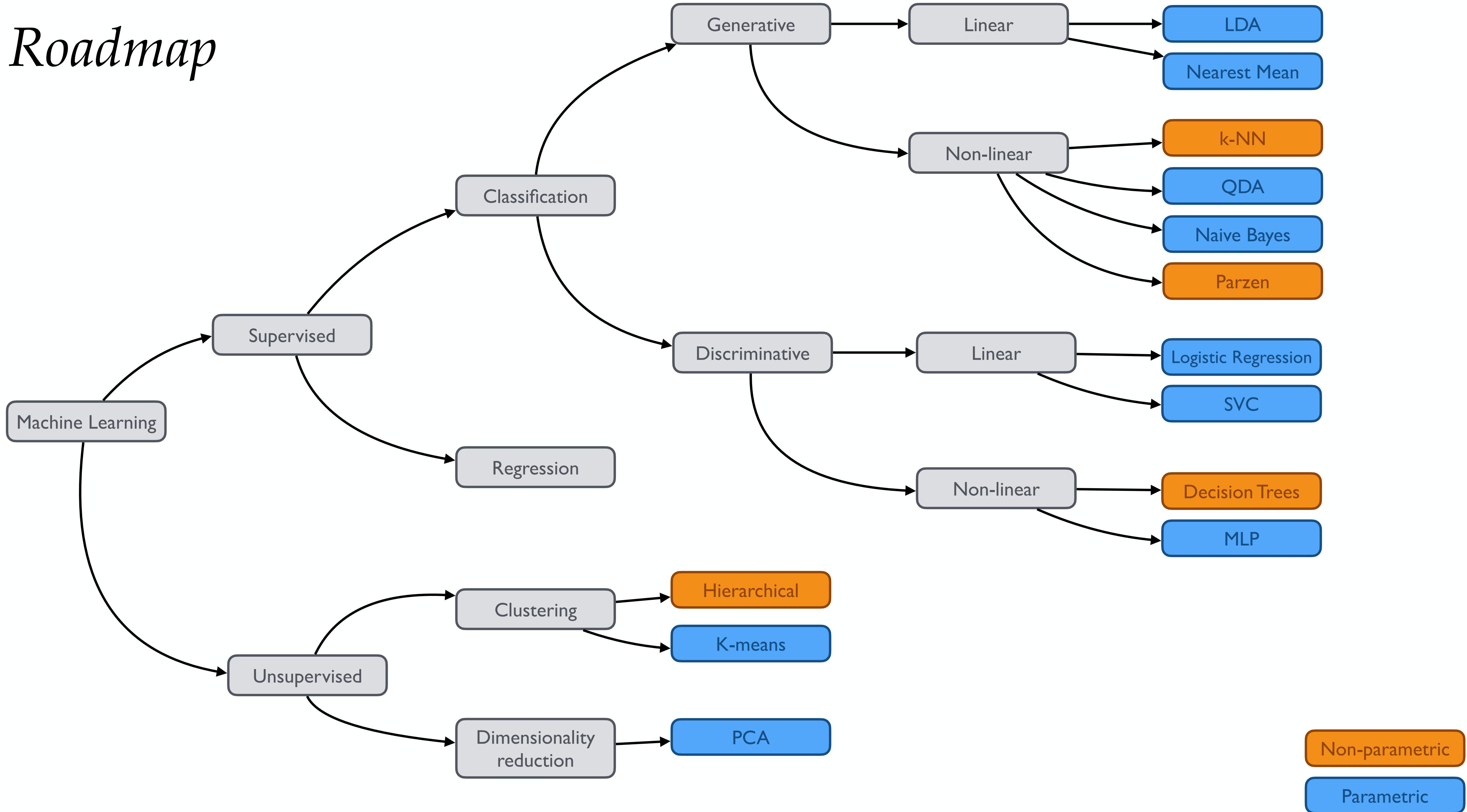
SHARE



VERGE DEALS

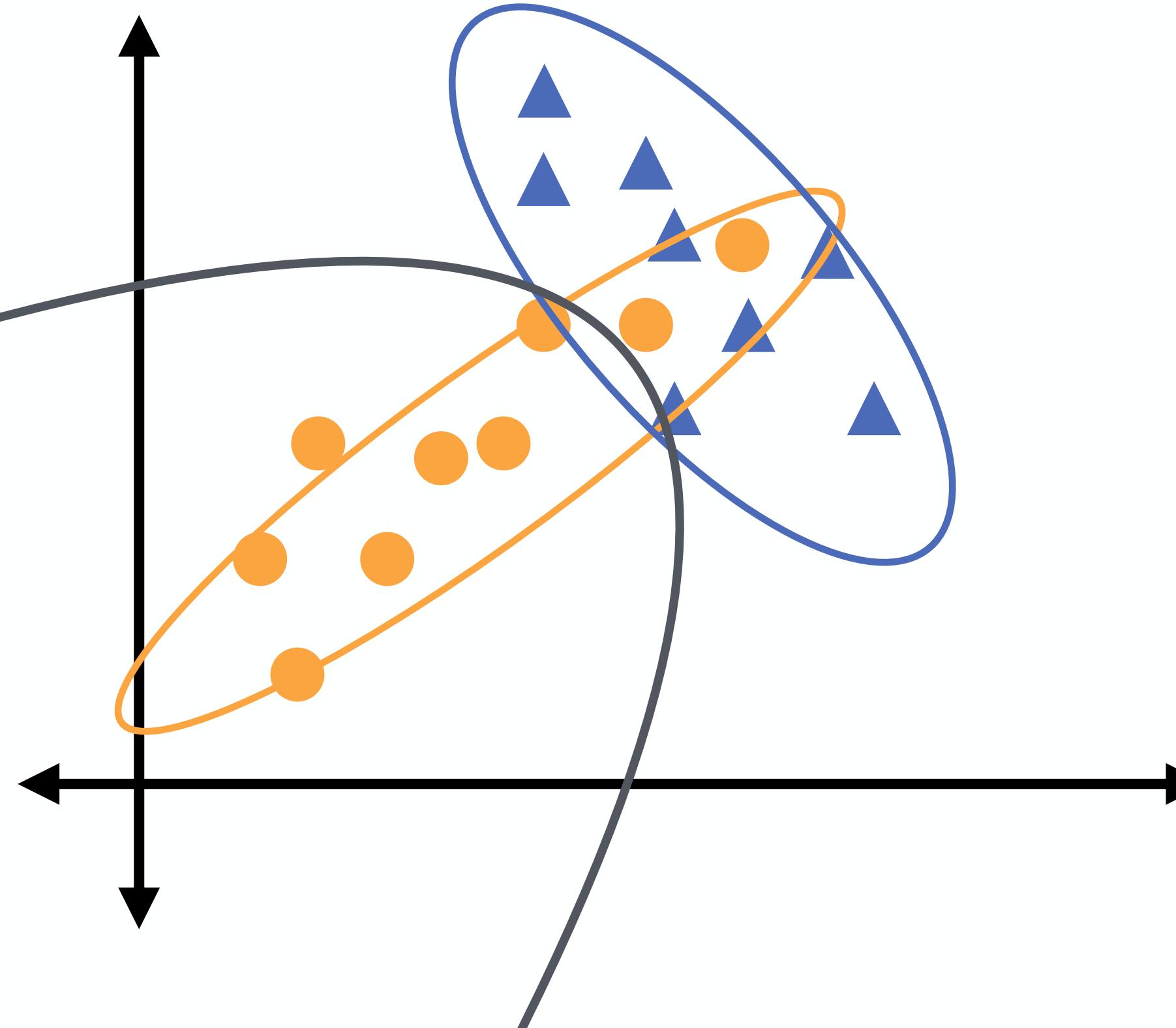


# Roadmap

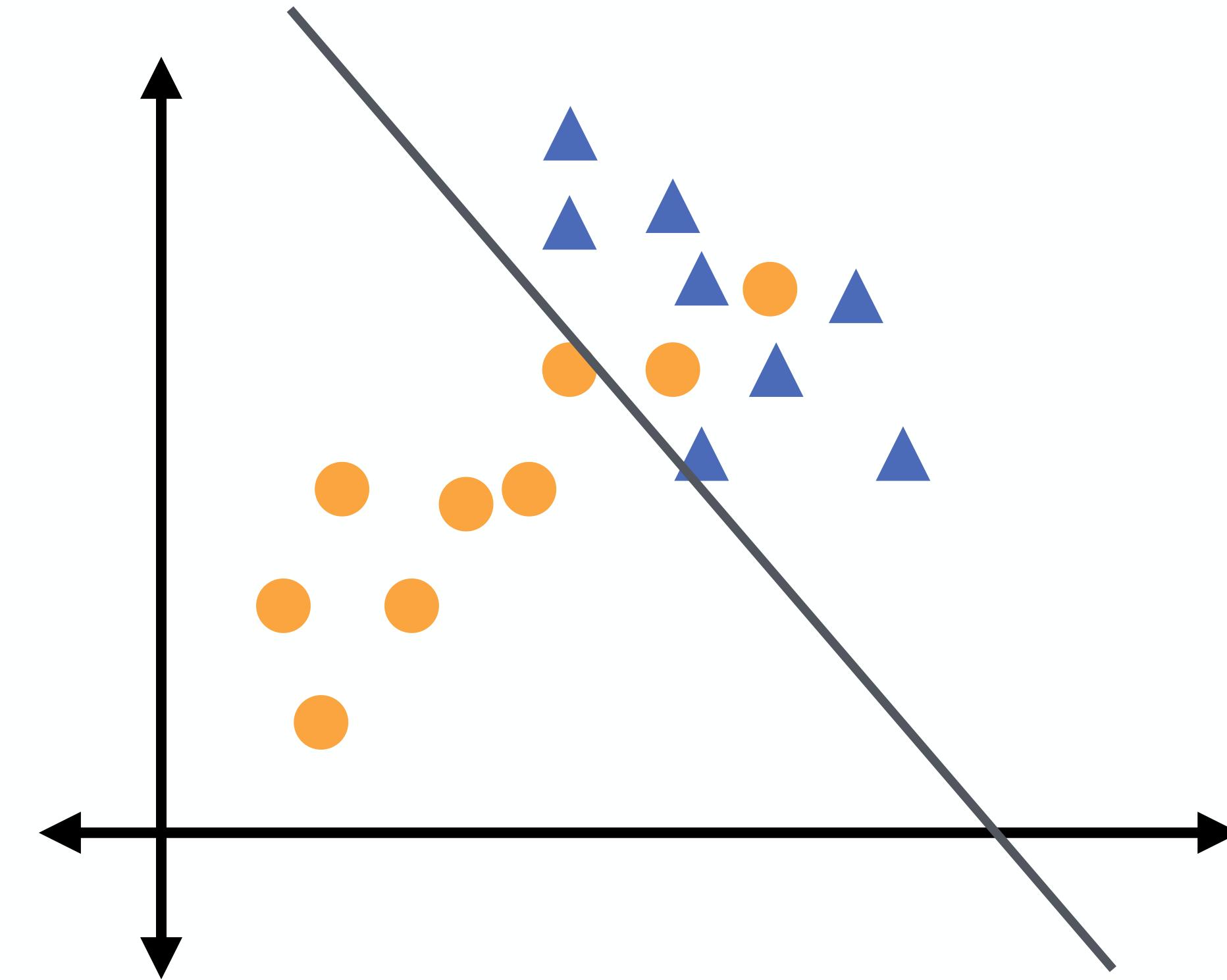


# *Generative vs. Discriminative*

Model  $P(X|Y)$  and  $P(Y)$ , or  
 $P(X,Y)$  to derive  $P(Y|X)$



Model  $P(Y|X)$  or  $h(X)$  directly



# *Learning Goals & Reading*

## LEARNING GOALS

After this week you will be able to

- **Distinguish between generative and discriminative models**
- **Reason about linear regression models and linear classifiers**
- **Explain what hypothesis and cost functions are**
- Implement **gradient descent** to train a given linear model
- Derive and implement logistic regression from its loss function
- Identify the principles behind support vector classifiers
- Describe some approaches to multi-class classification and their problems

## LITERATURE

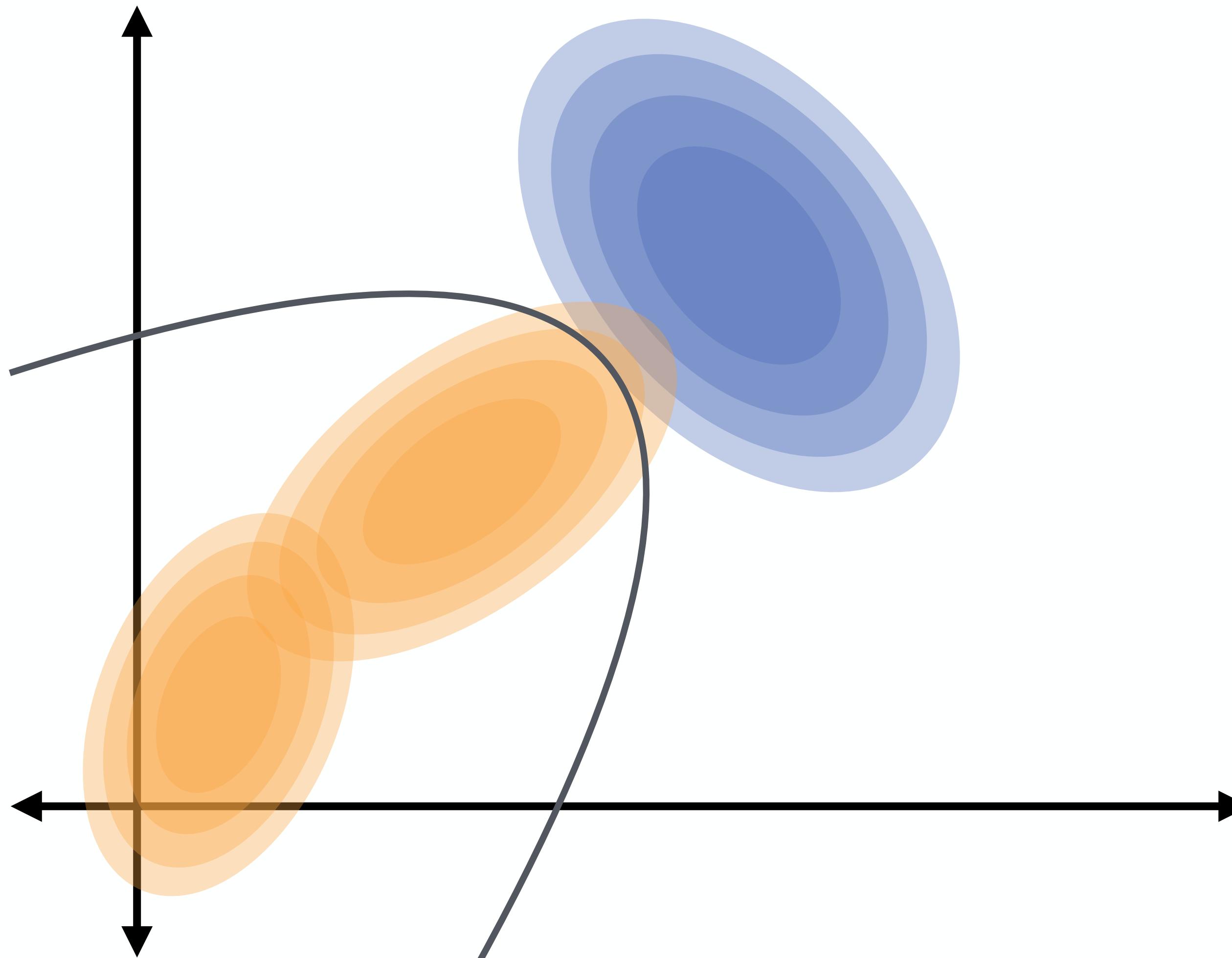
Bishop's "Pattern Recognition and Machine Learning"

- **Chapter 3 up to and including 3.1.3**
- **5.2.4**
- 4.3 up to and including 4.3.2
- 4.3.4
- 7.7 up to and including equation (7.7)
- 7.7.1 up to and including equation (7.22)
- 4.1.2

# *Today*

- How to define a classifier/regressor through a loss function and hypothesis class
- What is “linear” about a linear model?
- Finding the classifier/regressor by minimizing the “loss” on the training set
- Constructing a linear regression model and logistic classifier by choosing a loss function and hypothesis class

# Risk Minimization



Two classes, with the joint probability depicted on the left. How do we find a good discriminator?

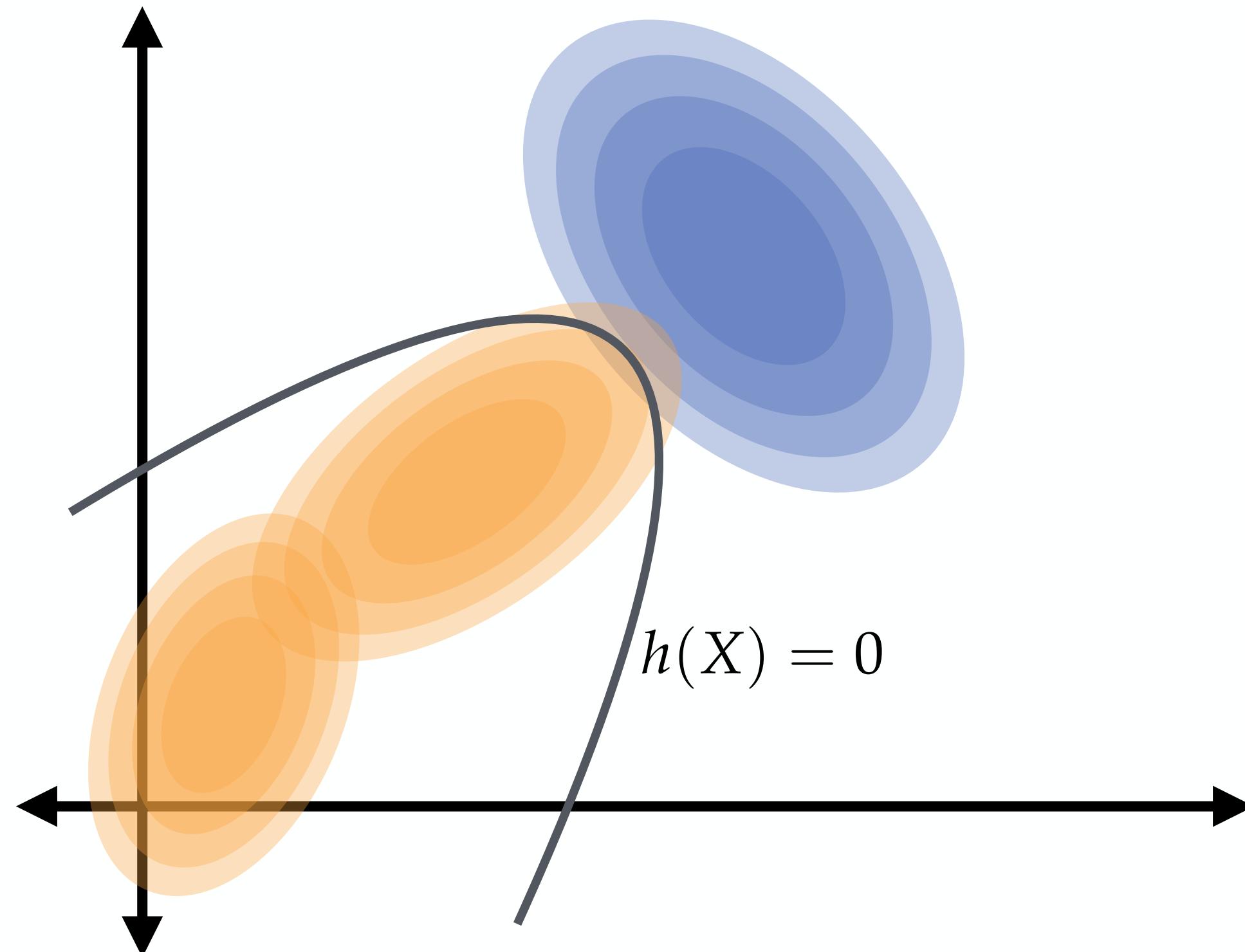
Here: rather than modelling the complete distribution  $P(X, Y)$ , directly model the decision / discriminant function ( $P(Y | X)$  or  $h(X)$ ) or decision boundary.

*Last week's  $g(X)$*

2 choices:

- What can the classifiers look like?
- How do we measure how well the classifier works?

# *Minimizing the “Risk”*



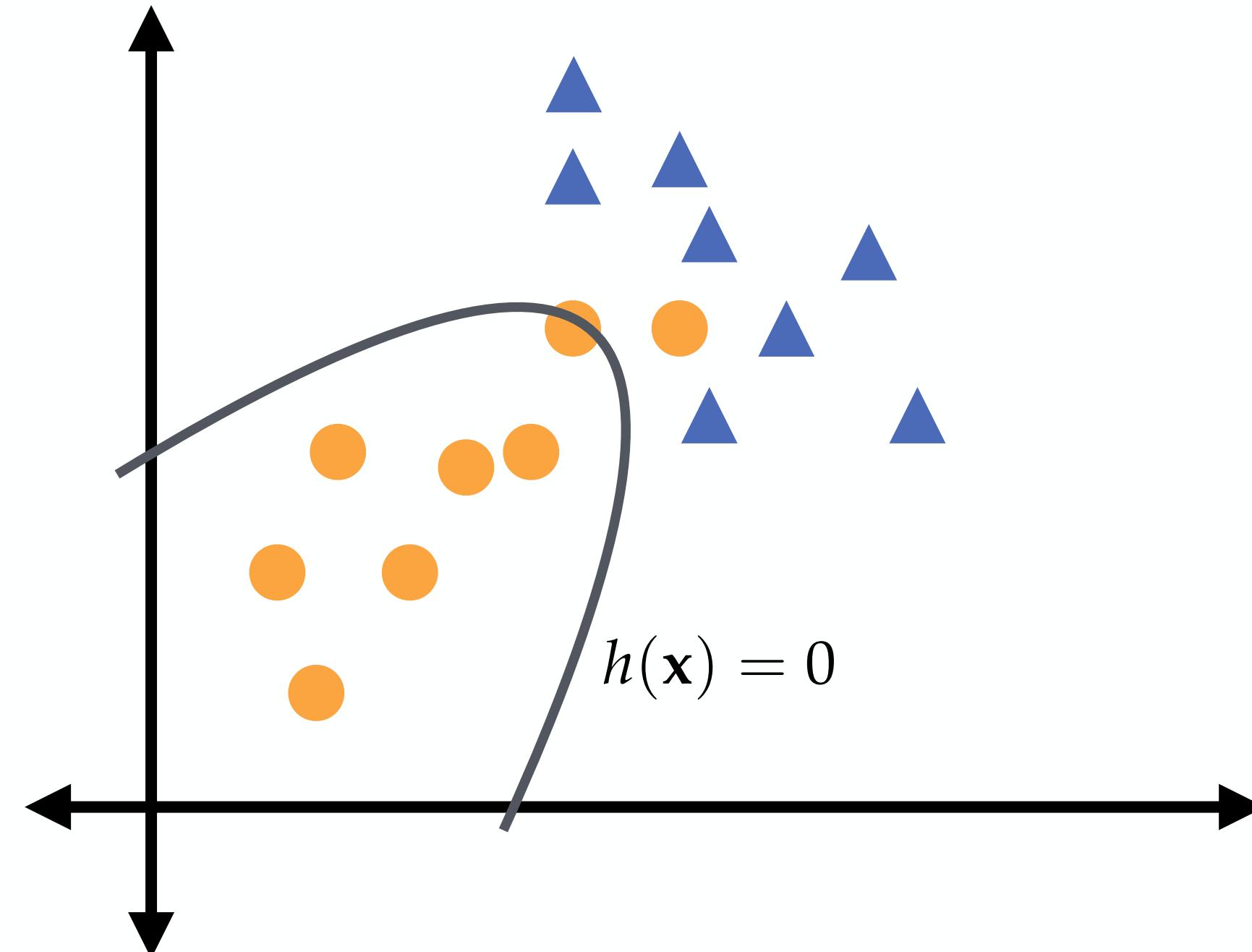
$$L(h(X), Y)$$

$X, Y$ : random variables corresponding to the input features and outcome of interest

$h$ : hypothesis function that maps input to a decision value

$L$ : loss/cost function that measures how well the predicted outcome matches the real outcome

# *The Empirical Risk*



$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

$\mathbf{x}, y$ : observed values corresponding to the input features and outcome of interest

$h$ : hypothesis function that maps input to a decision value

$L$ : loss/cost function that measures how well the predicted outcome matches the real outcome

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# Hypotheses

- When the outcome is continuous (regression),  $h(\mathbf{x})$  can directly correspond to the function we want to find
- In classification  $h(\mathbf{x})$  is the discriminant function which gives a real-valued output
- To go from this output to a class decision, we still need to set a cut-off, for instance at 0:

$$y = \begin{cases} c_1, & \text{if } h(\mathbf{x}) > 0 \\ c_0, & \text{if } h(\mathbf{x}) \leq 0 \end{cases}$$

# Hypothesis Class: Linear

$$h(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_Dx_D + w_0 = \mathbf{x}^\top \mathbf{w} + w_0$$

*weight vector*  *bias/intercept* 

*Decision function is a weighted linear combination of the input features, plus a constant (intercept/bias/threshold)*

**Trick:** for notational convenience, we sometimes add the bias term to the weight vector, by adding a constant feature

$$h(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_Dx_D + w_01 = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{w} \\ w_0 \end{bmatrix}$$

# *Linear Hypotheses*

$$h(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_Dx_D + w_0 = \mathbf{x}^\top \mathbf{w} + w_0$$

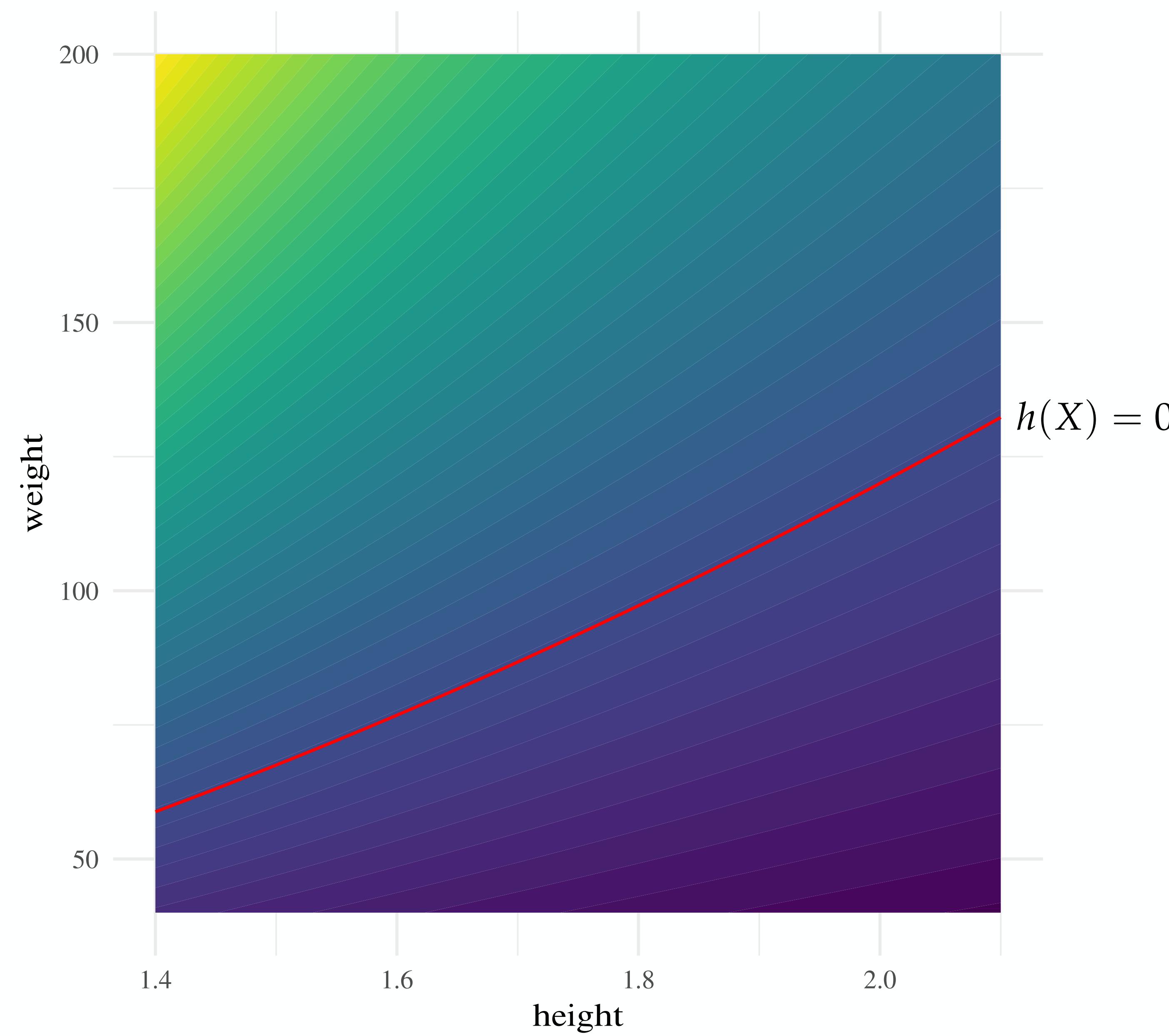
- Why linear?
  - “Simple”
  - Optimization is often possible and fast
  - Interpretable
  - Often/sometimes a reasonable approximation, locally
- We have seen linear classifiers before: LDA, nearest mean, ...
- Linear hypothesis class: all different  $\mathbf{w}$

## *Practice Question: BMI*



Body Mass Index (BMI) is defined as weight (in kg) divided by height<sup>2</sup> (in m<sup>2</sup>). Suppose I use a fixed rule: to classify people as healthy when  $\text{BMI} < 30$ .

1. **True or False:** Given the input features weight and height, I can construct a linear classifier that is exactly the same as this rule.
2. **True or False:** Given the input features weight, height and BMI, I can construct a linear classifier that is exactly the same as this rule.

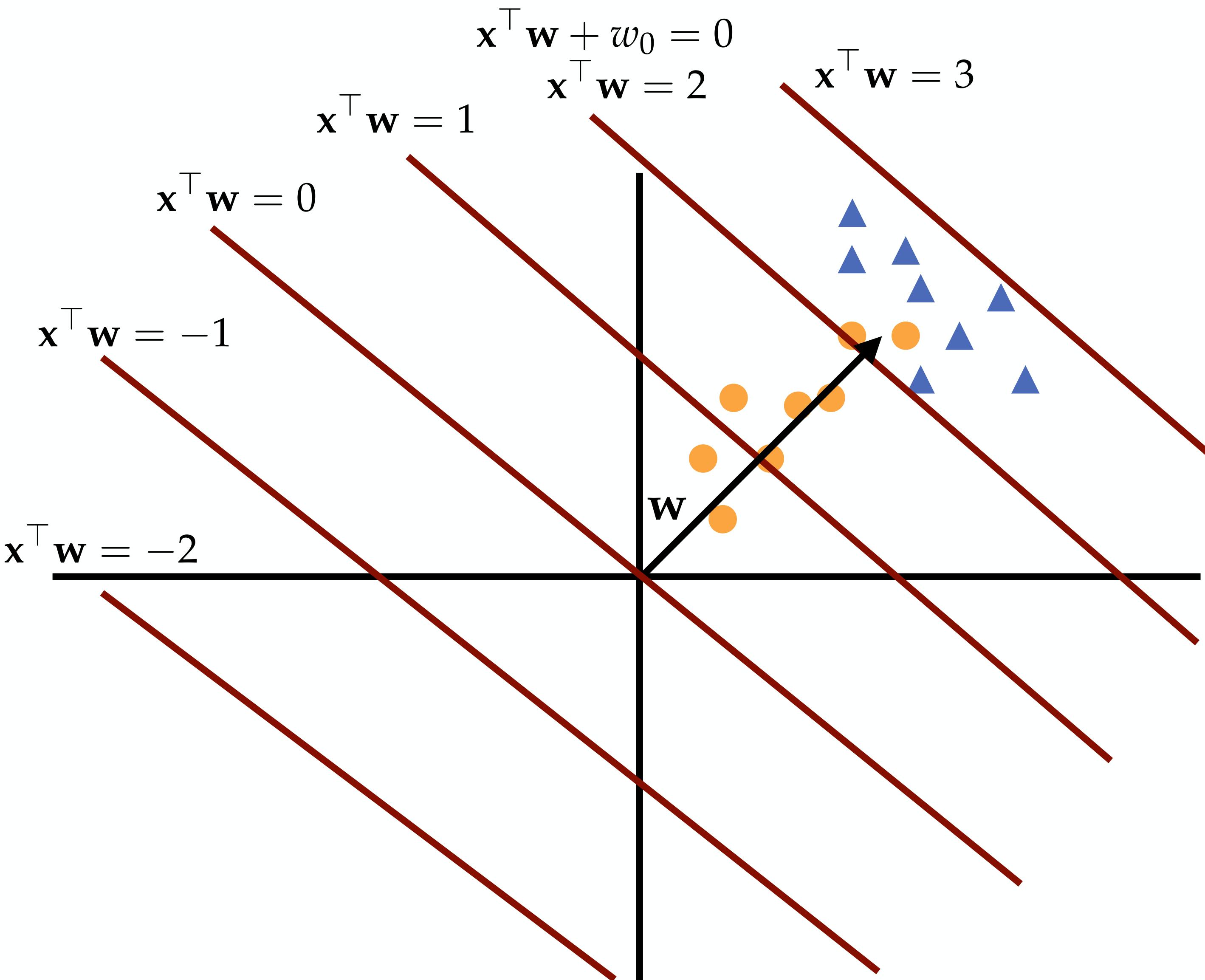


**BREAK**

# *Linear Decision Boundaries*

Decision boundary is where  $\mathbf{x}^\top \mathbf{w} + w_0 = 0$ . This means  $\mathbf{w}$  is orthogonal to every vector “within the decision boundary”. For a 2D problem, this means it has to be a hyperplane of dimension 1 (a line).

What does the linear decision boundary look like for a 3D, 1D or KD problem?



# *What is "linear" about a classifier?*

The decision function (usually...) and the decision boundary

Side note: Multiple  $\mathbf{w}$ 's can give rise to the same decision boundary

For our choice for a hypothesis class,  
let's go with all linear functions:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

# *Cost/Loss function*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

- How do we measure how “well” the model solves the problem?
- Different Problems: Regression & Classification
- Let’s start with regression

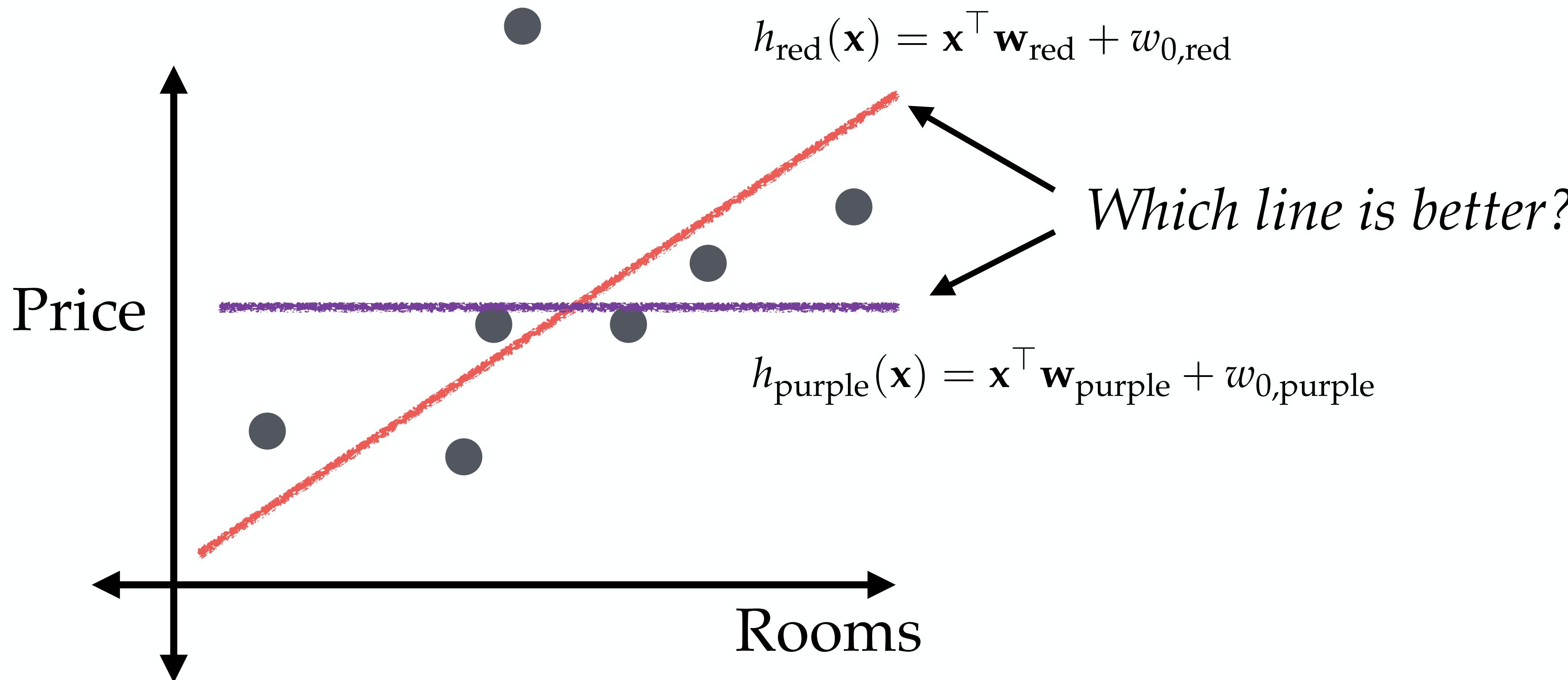
# LINEAR REGRESSION

# Regression Problem

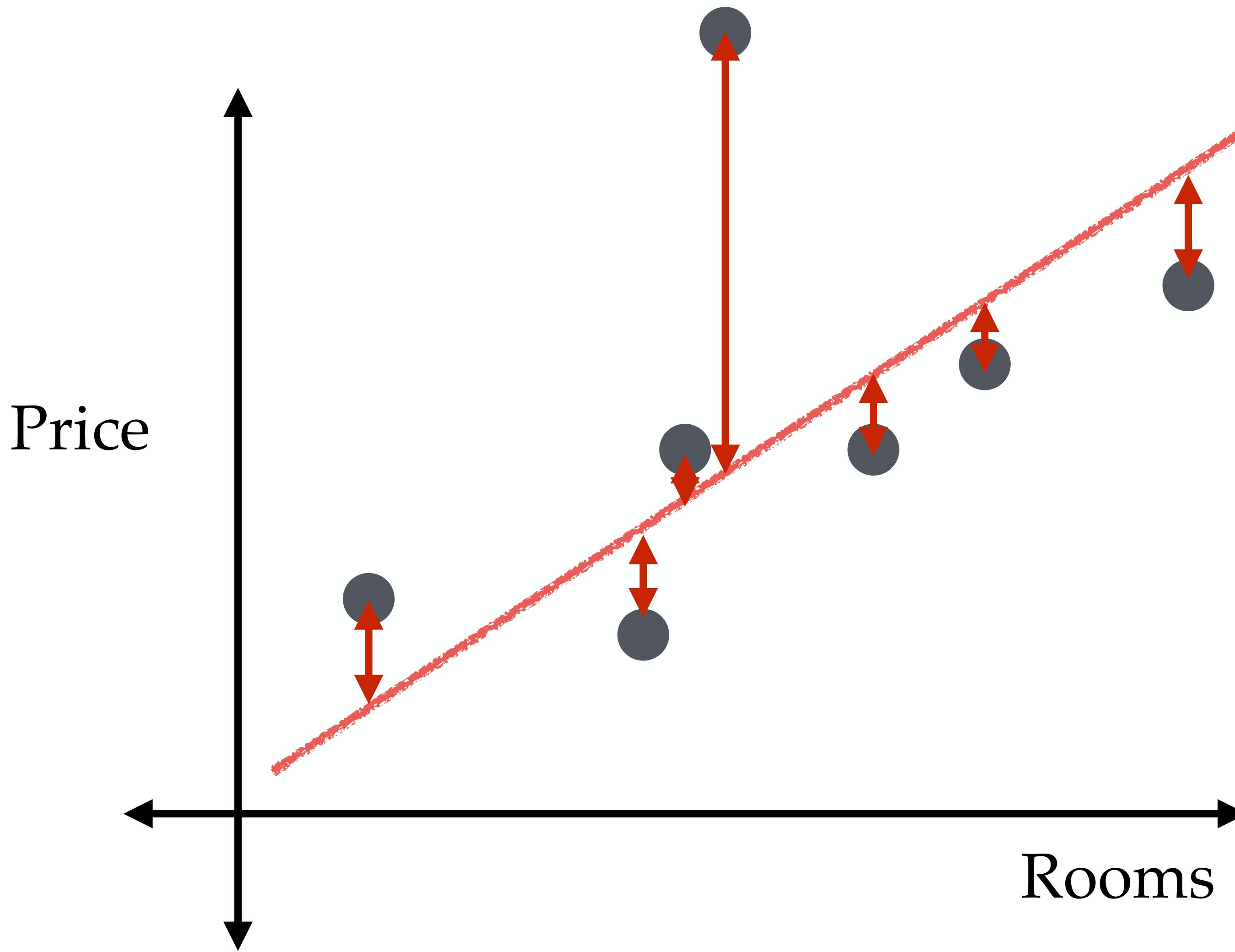
We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Task: predict the price of a house, given the number of rooms (note: continuous outcome, not discrete classes)



# Regression Losses



We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Consider the difference between the predicted value and the true value:

$$(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)$$

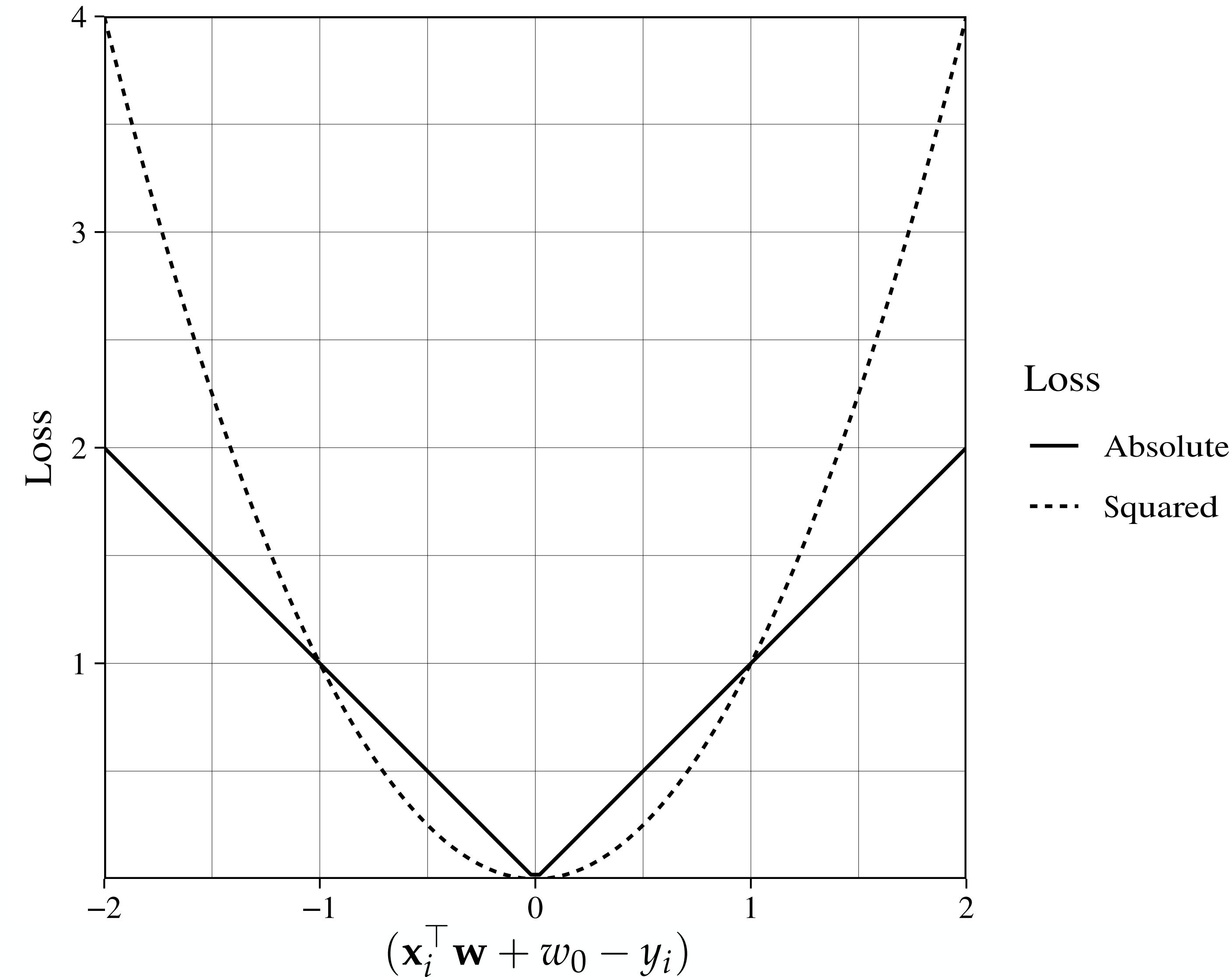
We can consider the absolute difference:

$$|\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i|$$

Or penalise large differences more strongly by taking the squared difference:

$$(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

# Regression Losses

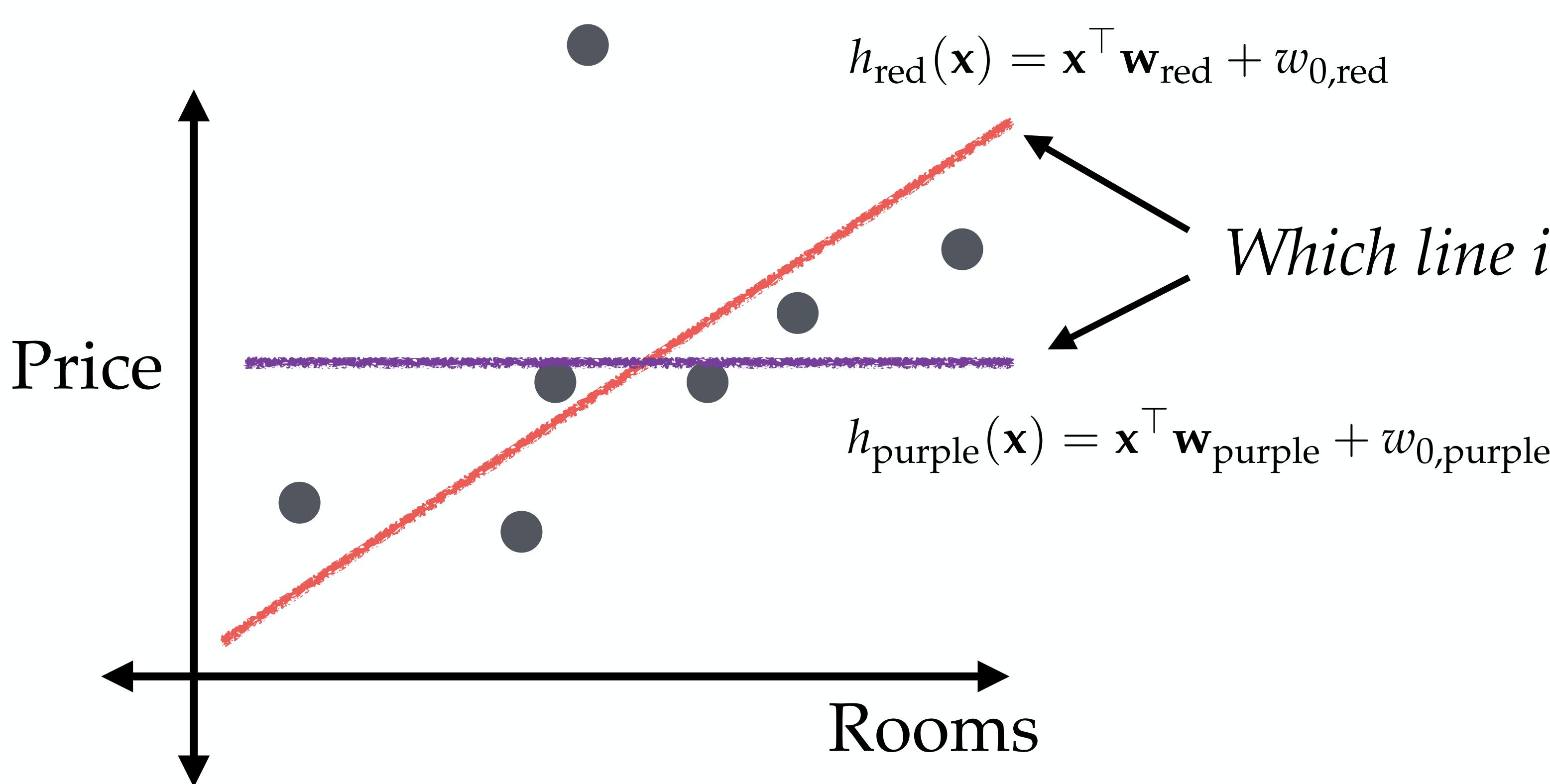


# Regression Problem

We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Task: predict the price of a house, given the number of rooms (note: continuous outcome, not discrete classes)



*Which line is better?*

$$\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w}_{\text{red}} + w_{0,\text{red}} - y_i)^2$$

$$\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w}_{\text{purple}} + w_{0,\text{purple}} - y_i)^2$$

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *Defining the regression solution*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

*Cost function:* Squared loss

*Hypothesis class:* all linear models

*Minimization procedure:* ...

# *Minimizing the Empirical Risk*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2 = \min_{\mathbf{w}, w_0} J(\mathbf{w}, w_0)$$

How do we find the best  $\mathbf{w}$ ?

- Taking the derivative, finding  $\mathbf{w}$  for which it is 0?
- Trying all possible values?
- Starting at some  $\mathbf{w}$  and keep making small changes to improve it?



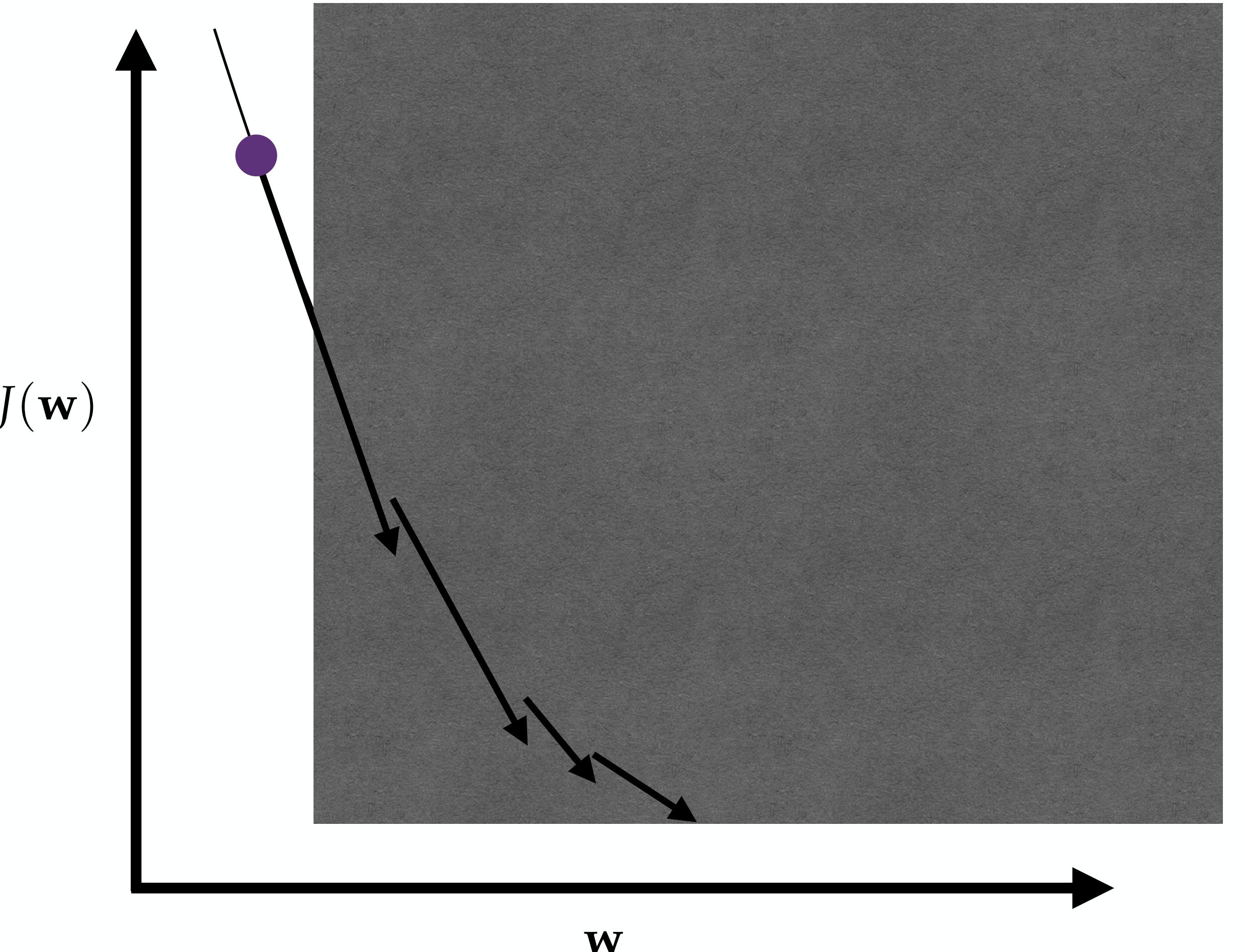
I'm on a mountain, in the fog. My car is in the lowest point of the valley. What is a good strategy to get back to the car (fast)?



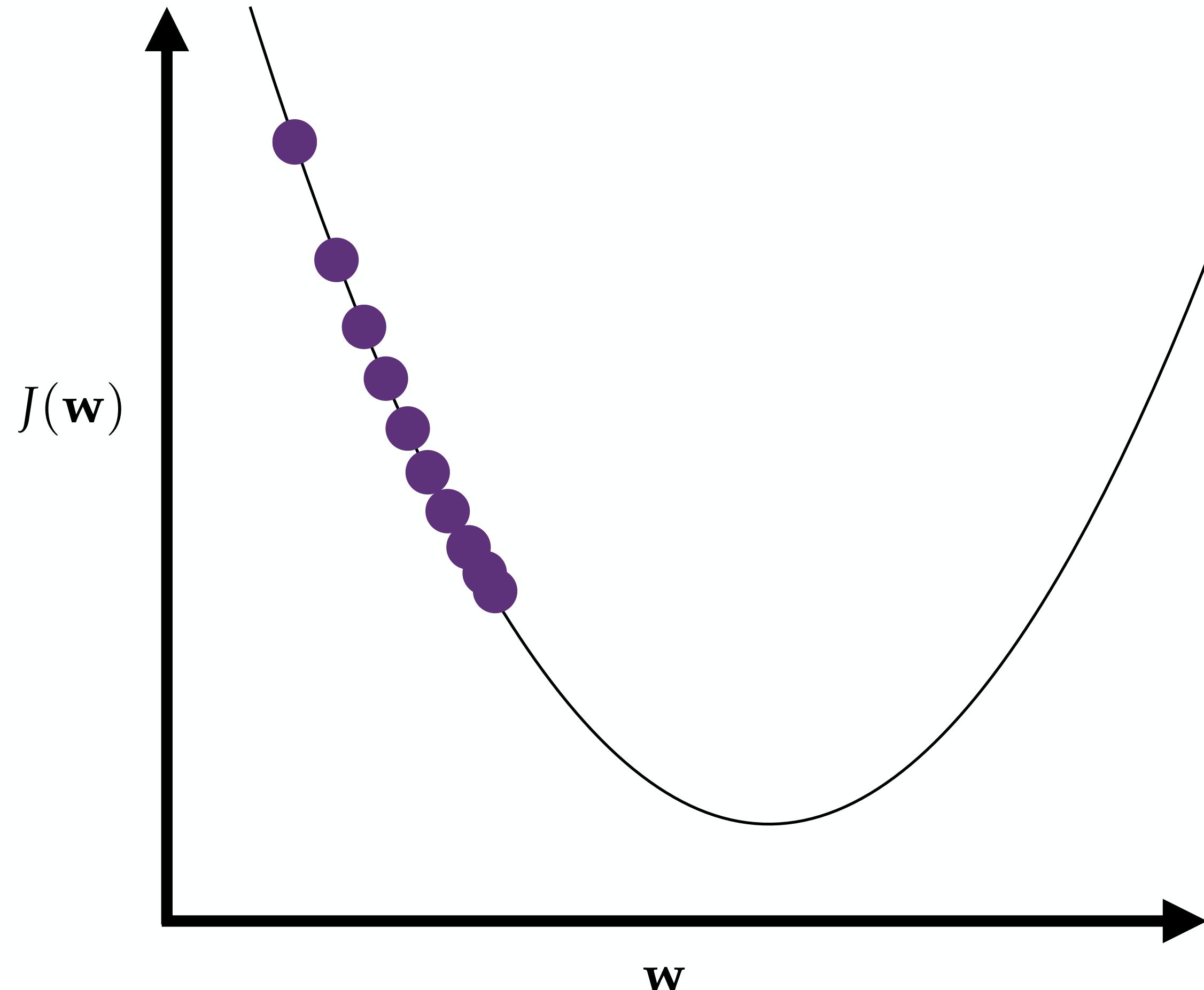
# Gradient Descent

$$w_j^{t+1} = w_j^t - \alpha \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \Bigg|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

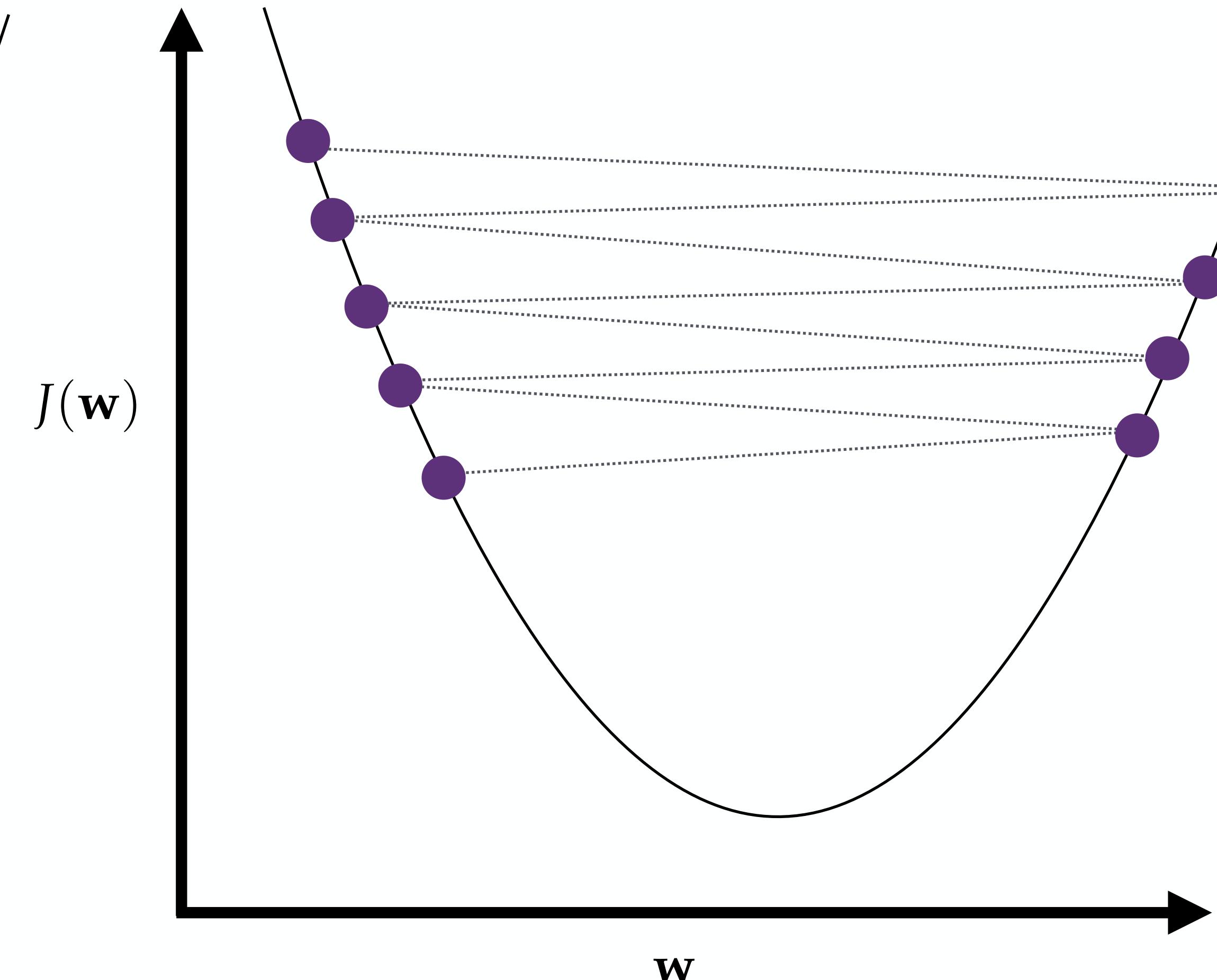
- Look at the slope, and follow the direction of the steepest slope
- How far do I go once I choose the direction (stepsize)? How many “steps” should I take?



# Stepsize $\alpha$



Too Small?



Too Large?

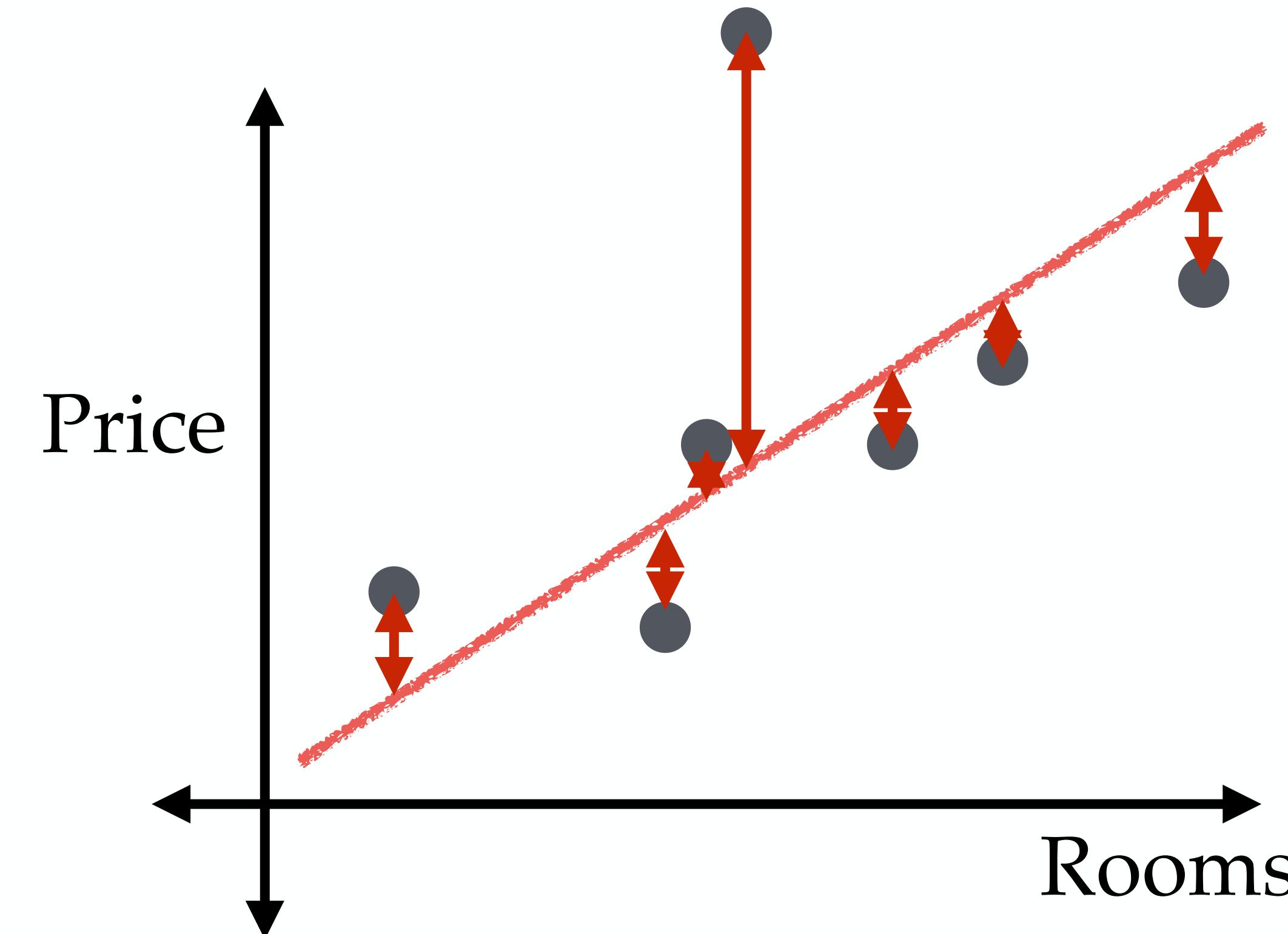
# *Gradient Descent Procedure*

Given an objective function  $J$ , learning rate (step size)  $\alpha$ , and number of iterations  $T$ .

1. Pick a starting value (for instance, random) for  $\mathbf{w}$  and  $w_0$
2. For  $T$  iterations, for all  $j = 0, 1, \dots, D$ , do:

$$w_j^{t+1} = w_j^t - \alpha \left. \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \right|_{\mathbf{w}, w_0=\mathbf{w}^t, w_0^t}$$

# *Gradient descent in our Regression problem*



$$J(\mathbf{w}, w_0) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

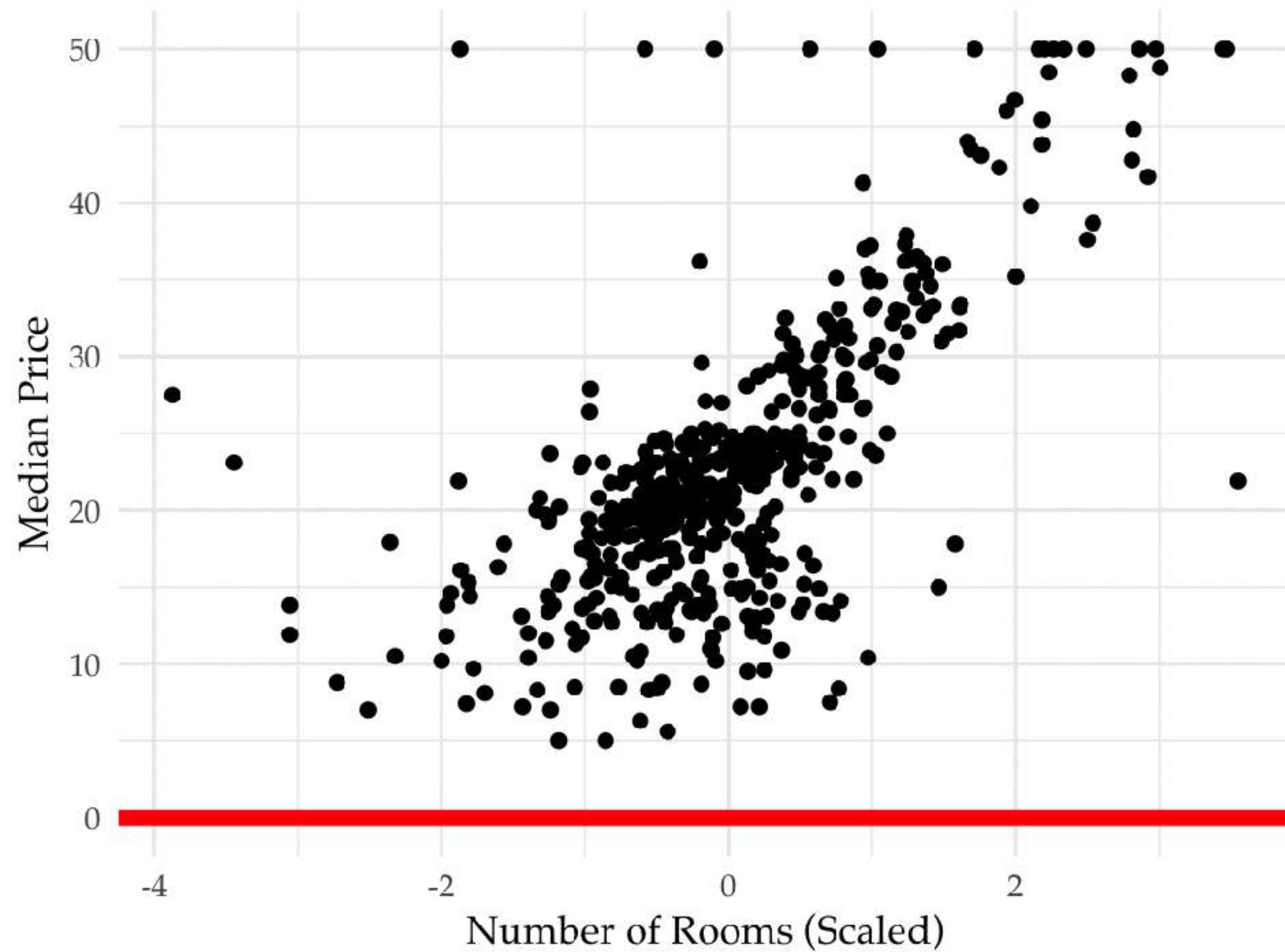
Derive the gradient:

$$\frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} =$$

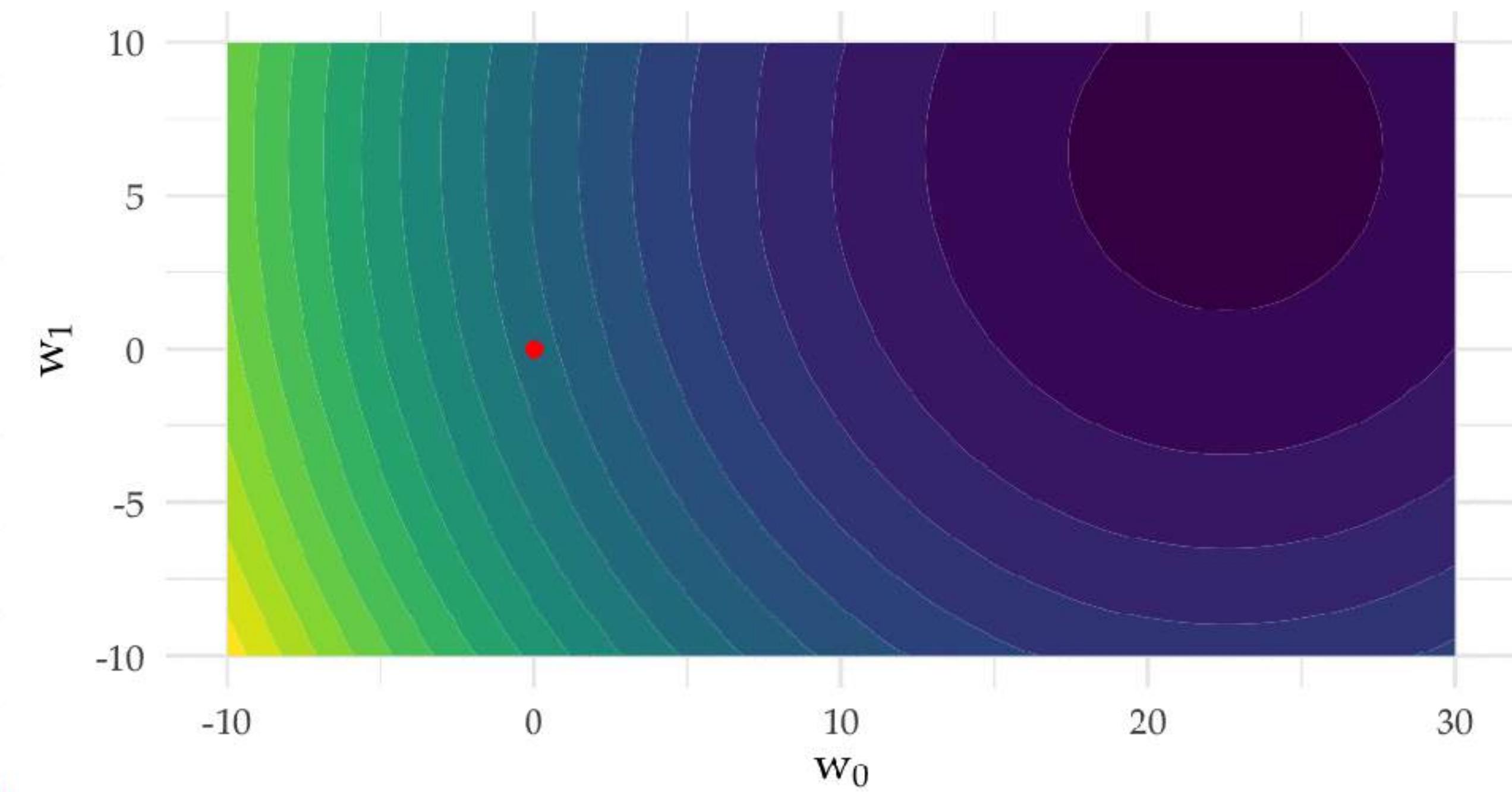
$$\frac{\partial J(\mathbf{w}, w_0)}{\partial w_0} =$$

# *Linear Regression: Visualizing the procedure*

Regression Solution



Objective function



# *Practice Question: Maximization*

Recall the update we use in gradient descent is

$$w_j^{t+1} = w_j^t - \alpha \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \Bigg|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

Suppose we want to maximize the function  $J$ . How should the update be different?

# Stochastic Gradient Descent

$$\frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} = \frac{1}{N} \sum_{i=1}^N 2(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)x_i^{(j)}$$

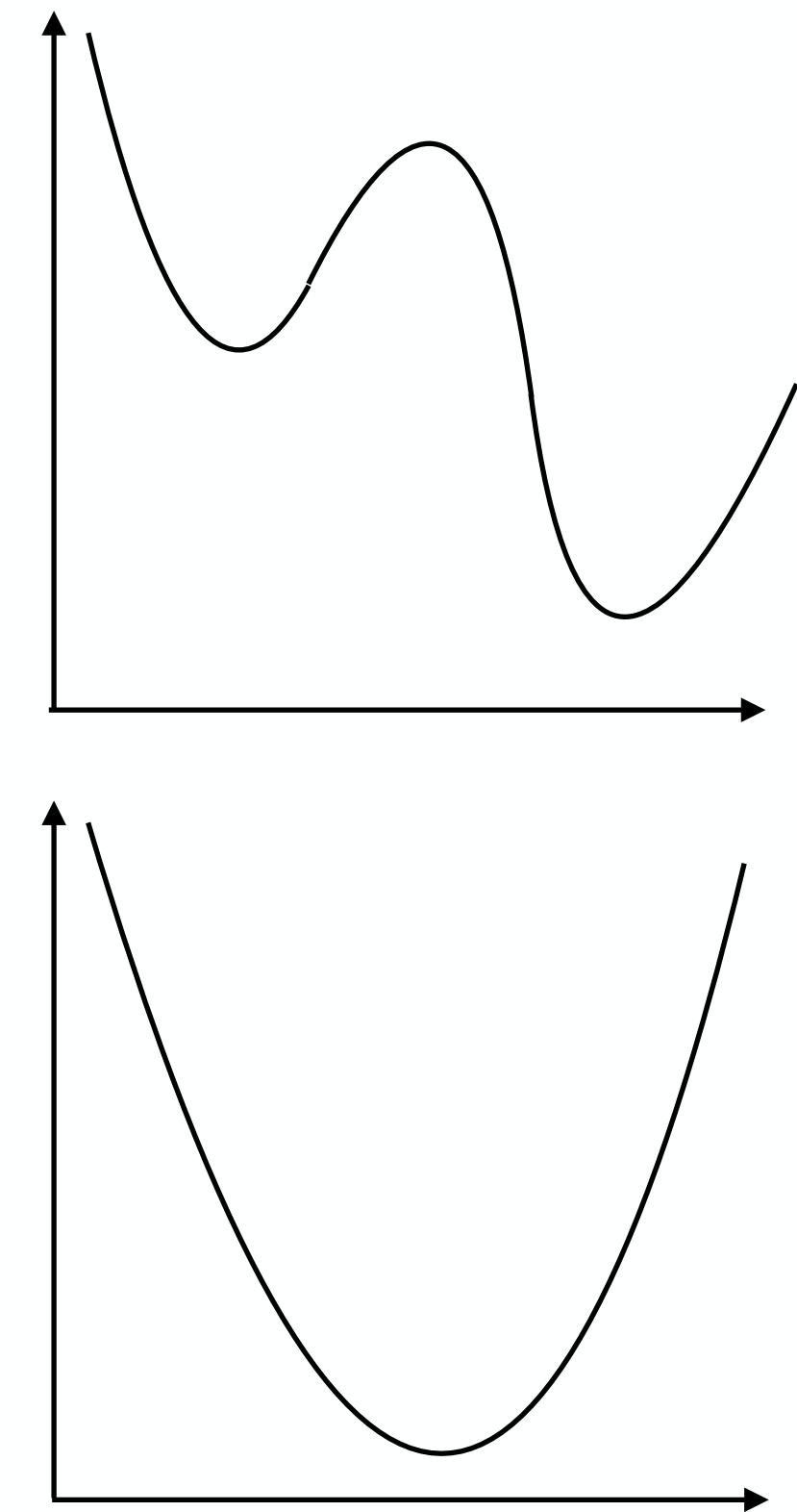
- To calculate the full gradient, we need to **sum** over all objects, before we take a step
- Instead we could estimate the gradient using one, or a few objects, and take a step using this estimate of the gradient
- The step is less “precise”, but we can take many more steps in the same amount of time
- *Epoch*: visiting all the data once
- So in stochastic gradient descent, we do updates within an epoch, while regular gradient descent does only one update per epoch.

# *Extensions*

- Fixing the step size seems naïve: smarter choices
  - Second order methods: take the curvature of the function into account
  - Line search
  - Momentum
- Many other descent procedures, for instance, conjugate gradient
- Note: using this gradient descent iterative procedure was not necessary here! There is a closed form / analytic solution! For our next model, this will not be the case.

# *Does GD lead to a good/the best solution?*

- Depends on the cost function and function class:  
What does the objective function look like?
- For convex functions, with the right settings, and the right amount of patience, we can reach the global minimum.
- The global minimum is the classifier with the minimal value for the cost function, which may not be the best solution for the problem! Recall: we want the solution to work well for the *expected* loss



$$\min_{h \in \mathcal{H}} \mathbb{E}_{X,Y}[L(h(X), Y)]$$

*We looked at regression, but what about classification?*

**THANKS**

# PART 2

## LINEAR (DISCRIMINATIVE) CLASSIFIERS

*Lecturer: Jesse Krijthe*

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *Learning Goals this Week*

After this week you will be able to

- Distinguish between generative and discriminative models
- Reason about linear regression models and linear classifiers
- Explain what hypothesis and cost functions are
- Implement gradient descent to train a given linear model
- **Derive and implement logistic regression from its loss function**
- **Identify the principles behind support vector classifiers**
- **Describe some approaches to multi-class classification and their problems**

# *Linear Regression*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

*Cost function:* Squared loss

*Hypothesis class:* all linear models

*Minimization procedure:* gradient descent (or, in practice: closed form solution)

*We looked at regression, but what about classification?*

# LOGISTIC REGRESSION



*Regression???*

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

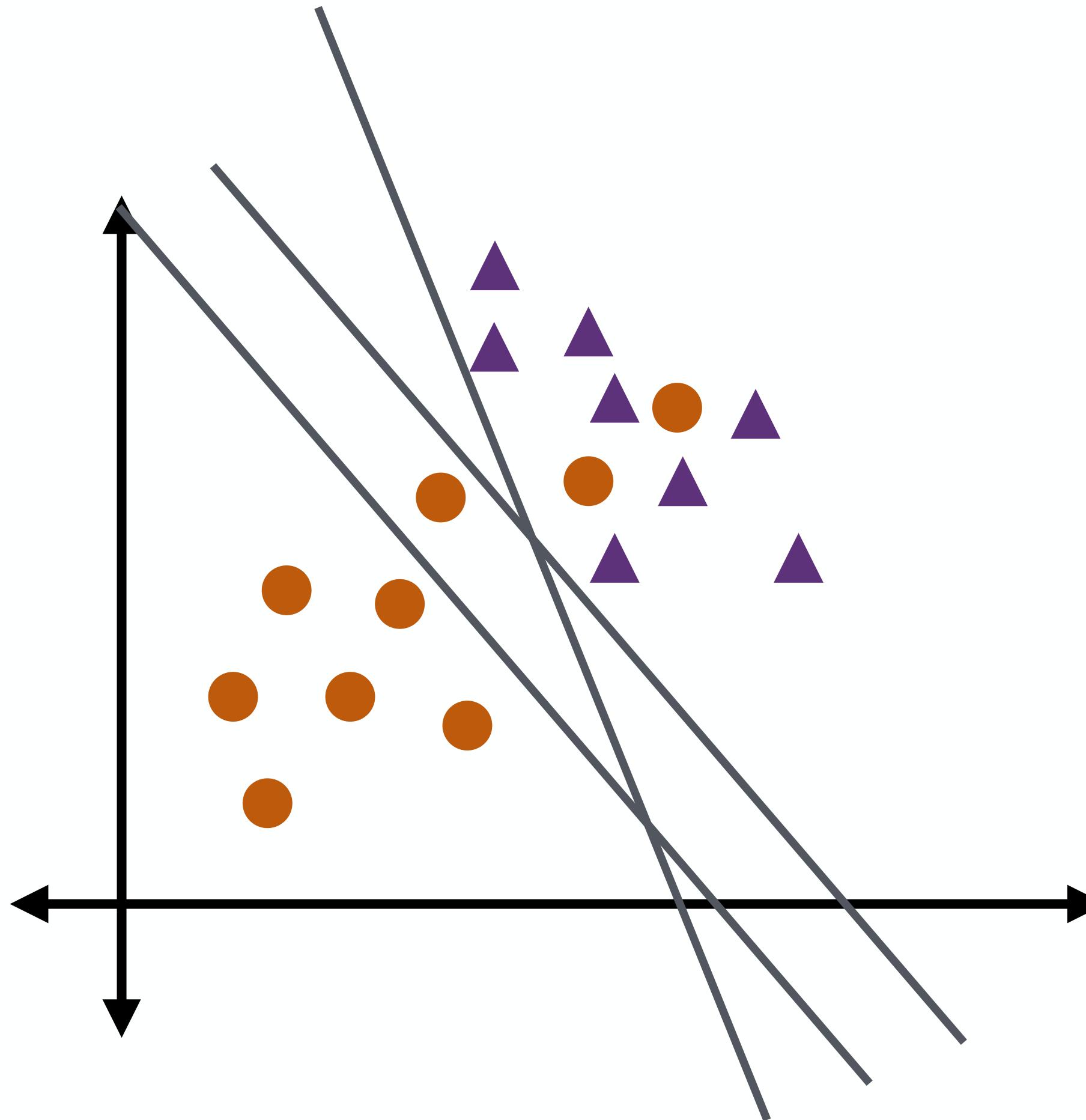
2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# Classification Losses

*It's obvious! Just count the number of mistakes!*



Unfortunately, it's not so simple:

- Accuracy is difficult to optimize (imagine standing on a flat surface, where should you go?)
- Multiple good solutions, which one do we pick?

Two reasons to use some other loss function:

1. There are other objectives in classification: different weight for different mistakes, accuracy of probabilities, correct “ordering” of the objects, etc.
2. Find a good solution for accuracy, by using some other “surrogate” loss

# *Classification Losses: Logistic Regression*

**Different Goal:**

Find a function that accurately approximates the probability of a label:

$$h(X) \approx P(Y = 1 | X)$$

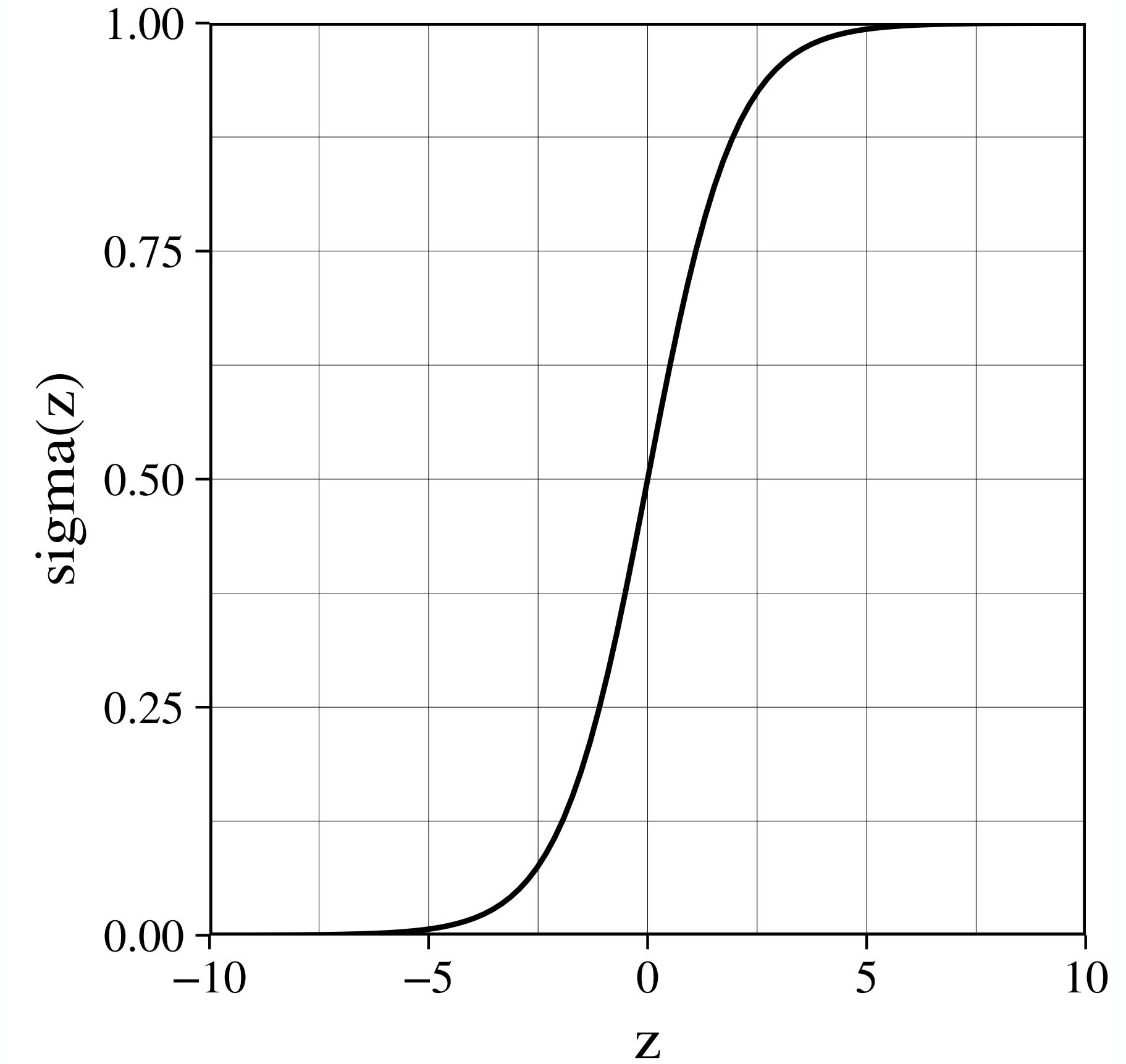
Probabilities have to be between [0,1]

Our hypothesis function  $h(X)$  can return any continuous value

Let's use some function to map  $h(X)$  to [0,1]

# *Logistic Function*

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



This is known as the logistic function, an example of a sigmoid function

# *Classification Losses: Logistic Regression*

Assume  $Y$  is either 0 or 1.

Model the class posterior probability  $P(Y=1 | X)$ , using  $\sigma(h(X))$

Measure of quality of the model: Given a choice of  $h(X)$ , what is the probability we would have observed the labels we observed in the data.

$N$  Bernoulli trials. The estimated probability of observing label  $y_i$  for object  $i$ :

$$\begin{cases} \sigma(h(\mathbf{x}_i)), & \text{if } y_i = 1 \\ 1 - \sigma(h(\mathbf{x}_i)), & \text{if } y_i = 0 \end{cases}$$

Combining this, the likelihood is

$$L(h) = \prod_{i=1}^N \sigma(h(\mathbf{x}_i))^{y_i} (1 - \sigma(h(\mathbf{x}_i)))^{1-y_i}$$

# *Logistic Regression: Objective Function*

$$L(h) = \prod_{i=1}^N \sigma(h(\mathbf{x}_i))^{y_i} (1 - \sigma(h(\mathbf{x}_i)))^{1-y_i}$$

The likelihood is a measure of how well the estimated probabilities explain the observed labels, so we want to maximise this likelihood

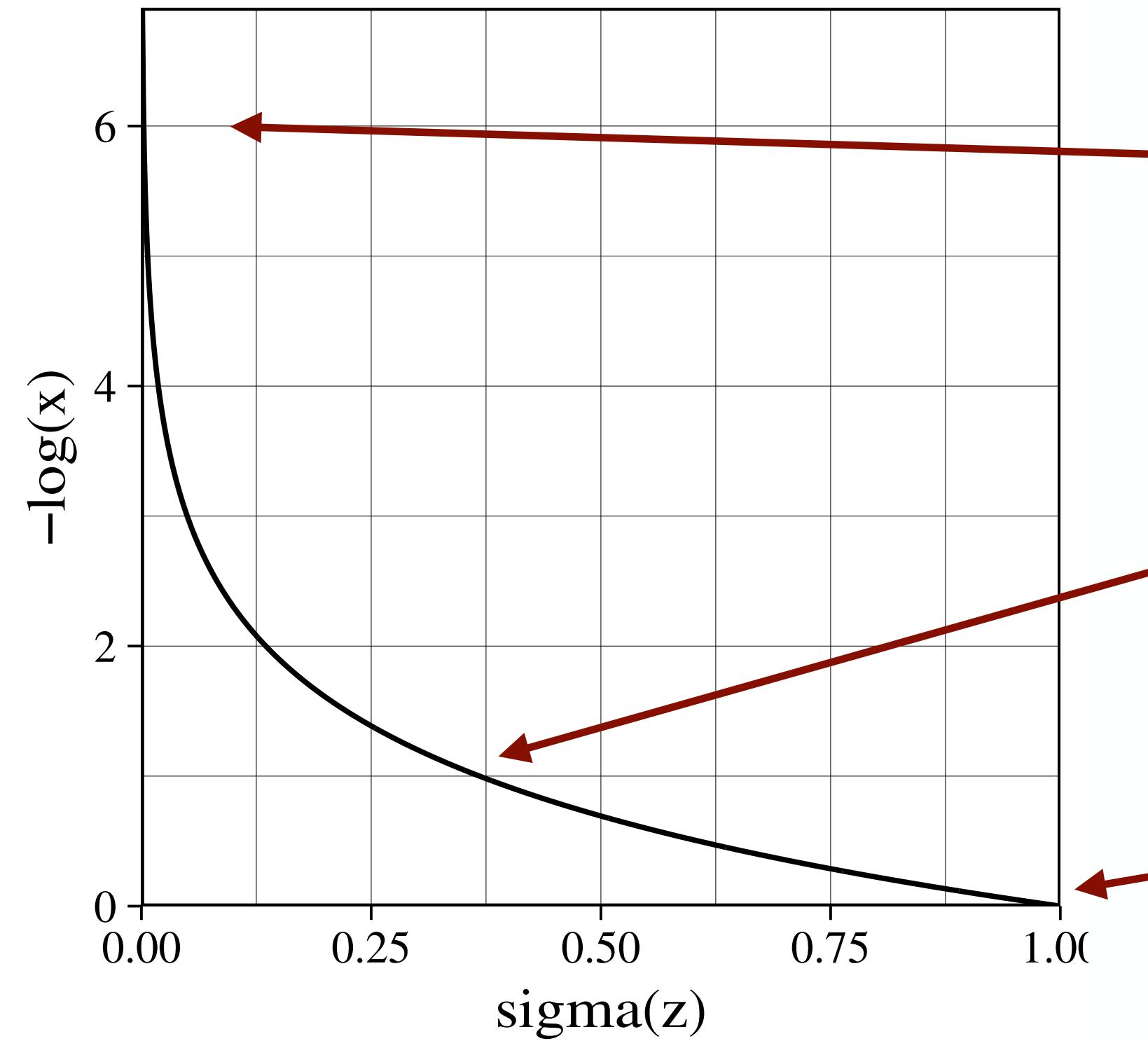
This gives the same optimum as minimizing the negative log likelihood:

$$J(h) = - \sum_{i=1}^N y_i \log[\sigma(h(\mathbf{x}_i))] + (1 - y_i) \log[1 - \sigma(h(\mathbf{x}_i))]$$

# *Logistic Loss, Visualized*

$$J(h) = - \sum_{i=1}^N y_i \log[\sigma(h(\mathbf{x}_i))] + (1 - y_i) \log[1 - \sigma(h(\mathbf{x}_i))]$$

*For  $y_i = 1$*

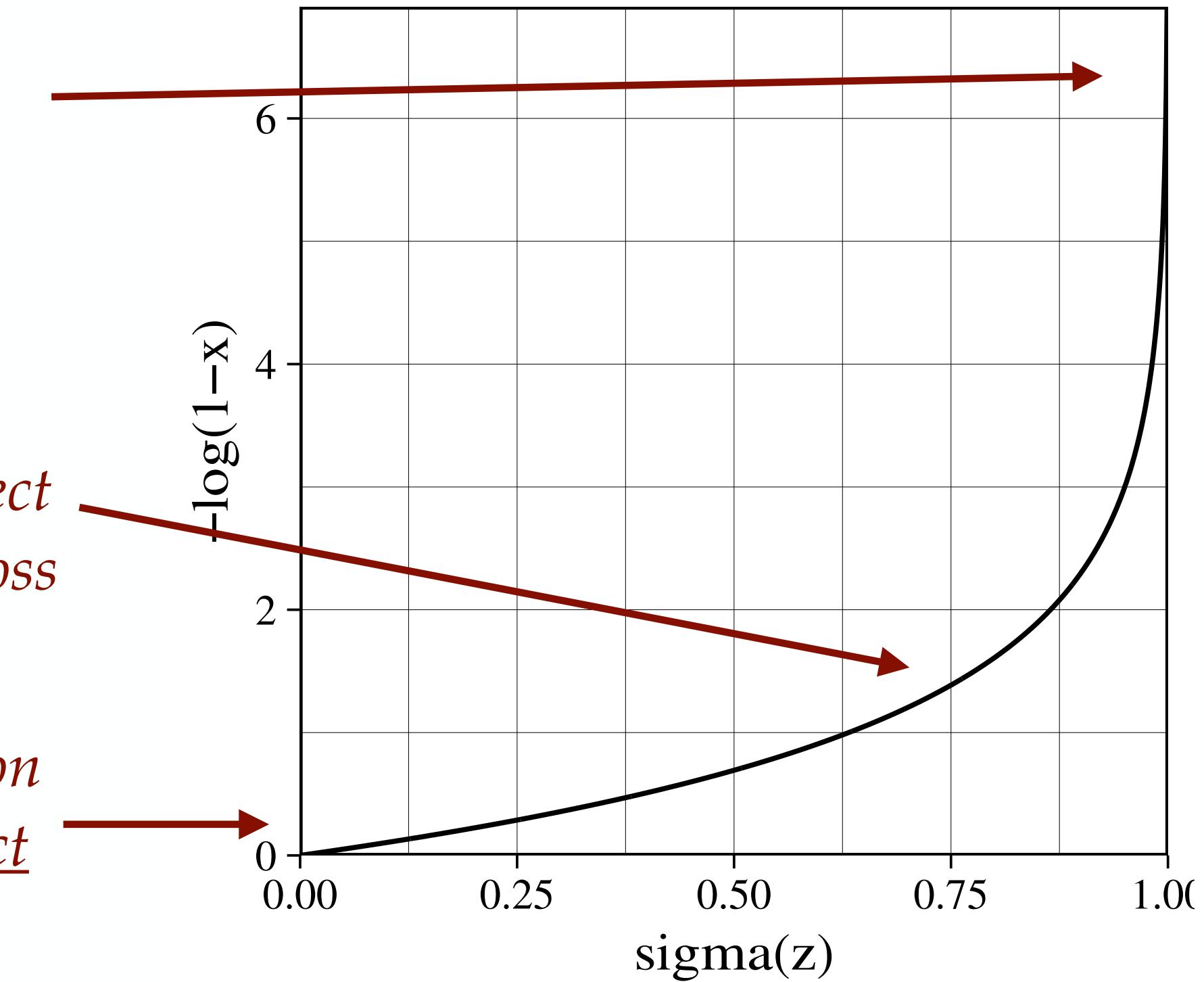


*Infinite loss if prediction is certain but wrong*

*The higher the certainty of the correct label, the lower the loss*

*Zero loss if prediction is certain but correct*

*For  $y_i = 0$*



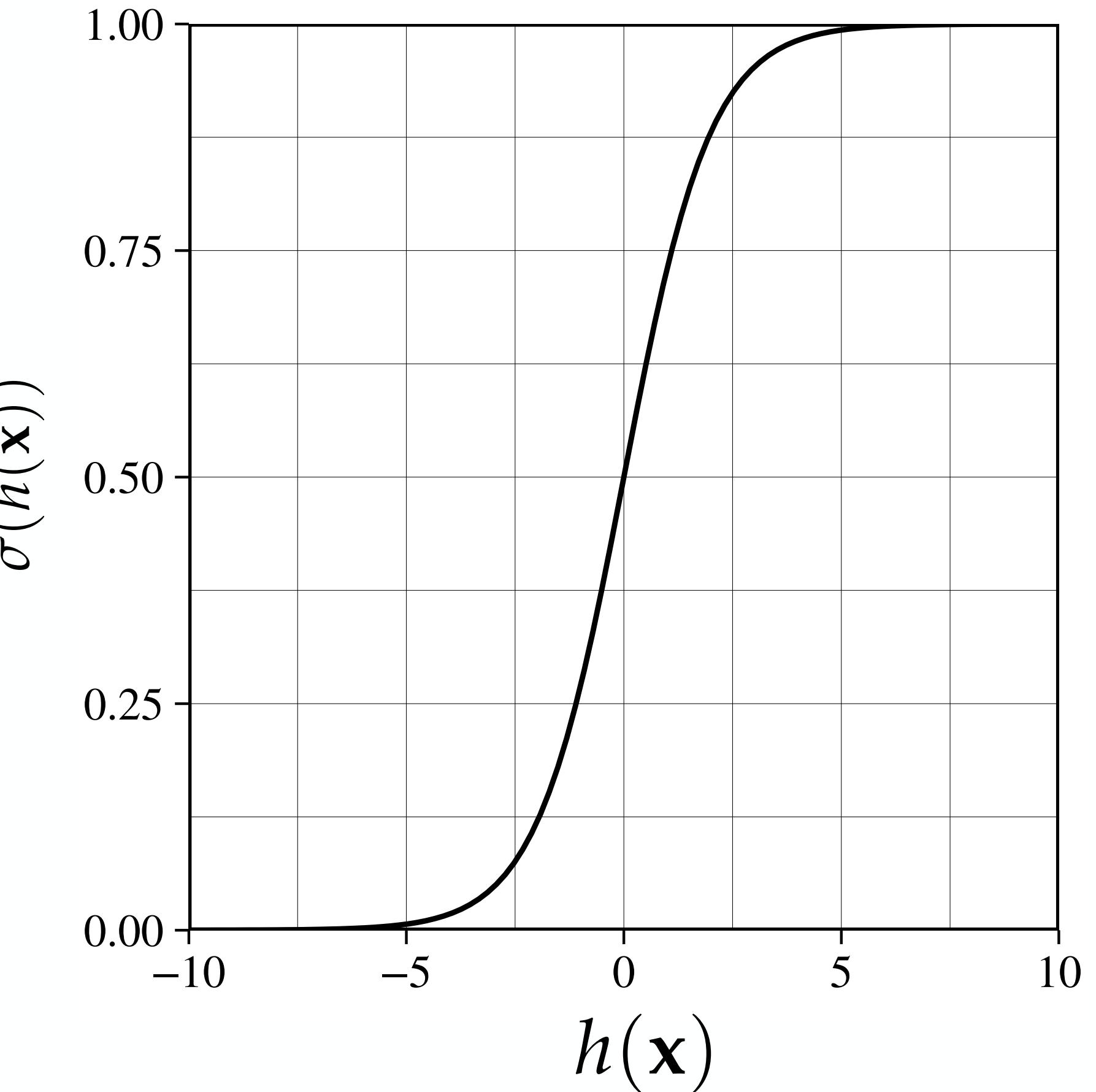
# *Logistic Regression: Hypothesis Class*

For linear logistic regression,  
again consider all  $\mathbf{w}, w_0$  for  $h(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + w_0$

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} - \sum_{i=1}^N y_i \log[\sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)] + (1 - y_i) \log[1 - \sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)]$$

# *Logistic Regression*

- If we want class predictions, we have to use a cut-off at a probability
- Corresponds to a threshold for  $h(\mathbf{x})$



# *Logistic Regression: Example*

<https://jkrijthe.shinyapps.io/lr-optimization/>

# *Practice Question: Logistic Regression*

$$\sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)$$



Suppose we have a linear logistic regression model, with one feature  $x$ , where  $w_1 = 1$  and  $w_0 = 0$ . Answer the following questions:

**True or False:** because we have a linear model, the estimated probability of the positive class for  $2x$  is twice as high as for  $x$ , for all choices of  $x$ .

**True or False:** because the logistic function is a non-linear function, logistic regression is not a linear classifier.

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *Logistic Regression*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} - \sum_{i=1}^N y_i \log[\sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)] + (1 - y_i) \log[1 - \sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)]$$

# *Gradient Descent Procedure*

Given an objective function  $J$ , learning rate (step size)  $\alpha$ , and number of iterations  $I$ .

1. Pick a starting value (for instance, random) for  $\mathbf{w}$  and  $w_0$
2. For  $I$  iterations, for all  $j = 0, 1, \dots, D$ , do:

$$w_j^{\text{new}} = w_j - \alpha \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j}$$

# Logistic Regression Gradient

$$J(\mathbf{w}) = - \sum_{i=1}^N y_i \log[\sigma(\mathbf{x}_i^\top \mathbf{w})] + (1 - y_i) \log[1 - \sigma(\mathbf{x}_i^\top \mathbf{w})]$$

Find the gradient, by taking the derivative and using  $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$   
*Homework: prove this!*

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = - \sum_{i=1}^N y_i \frac{1}{\sigma(\mathbf{x}_i^\top \mathbf{w})} \sigma(\mathbf{x}_i^\top \mathbf{w})(1 - \sigma(\mathbf{x}_i^\top \mathbf{w})) \mathbf{x}_i + (1 - y_i) \frac{1}{1 - \sigma(\mathbf{x}_i^\top \mathbf{w})} \cdot -\sigma(\mathbf{x}_i^\top \mathbf{w})(1 - \sigma(\mathbf{x}_i^\top \mathbf{w})) \mathbf{x}_i$$

Some terms cancel:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = - \sum_{i=1}^N y_i (1 - \sigma(\mathbf{x}_i^\top \mathbf{w})) \mathbf{x}_i + (1 - y_i) \cdot -\sigma(\mathbf{x}_i^\top \mathbf{w}) \mathbf{x}_i$$

Note that we can write this as:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = - \sum_{i=1}^N (y_i - \sigma(\mathbf{x}_i^\top \mathbf{w})) \mathbf{x}_i$$

# *Logistic Regression: Gradient Descent Example*

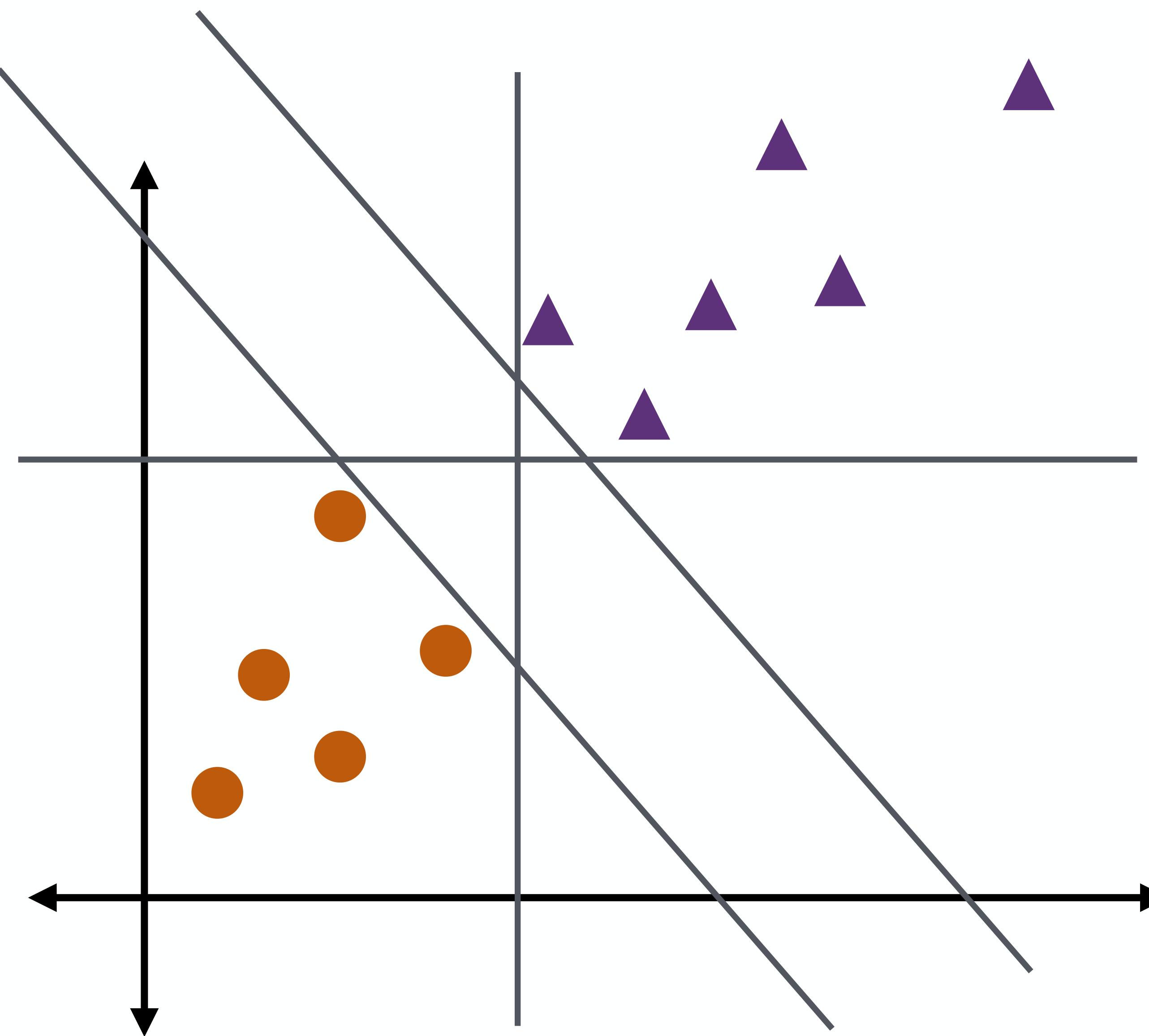
<https://jkrijthe.shinyapps.io/lr-optimization/>

# *Logistic Regression Recap*

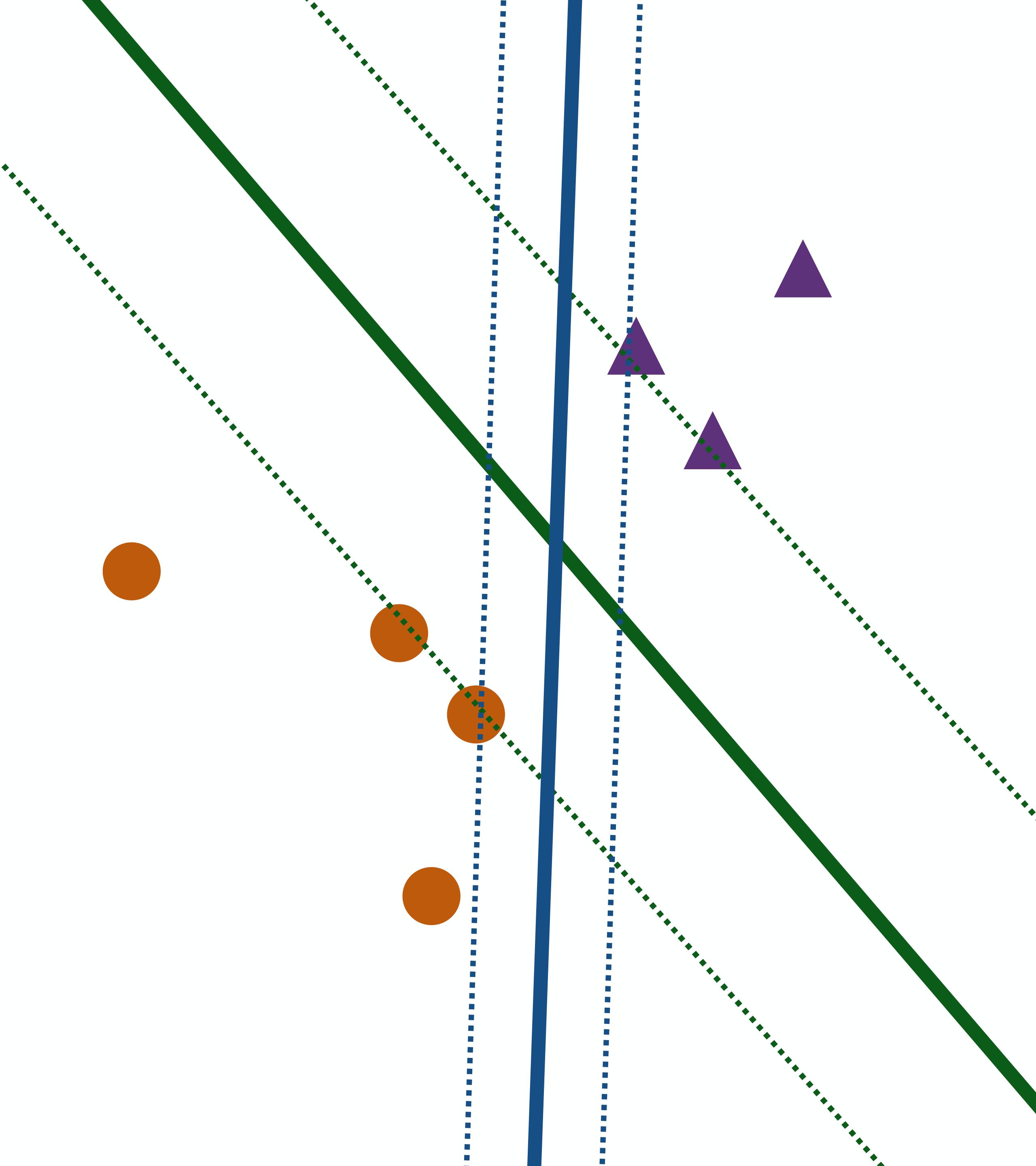
- Linear classifier
- Maximize the (log) likelihood of the observed posterior probabilities
- Or: minimize the negative (log) likelihood
- No analytical solution, so we use iterative procedures, like gradient descent (more efficient alternatives exist)
- If no class overlap our quality measure is not sufficient: many equally good solutions.

# SUPPORT VECTOR MACHINES

*Note: the material in the book covers some additional details about the optimization, here we will focus on the intuition*



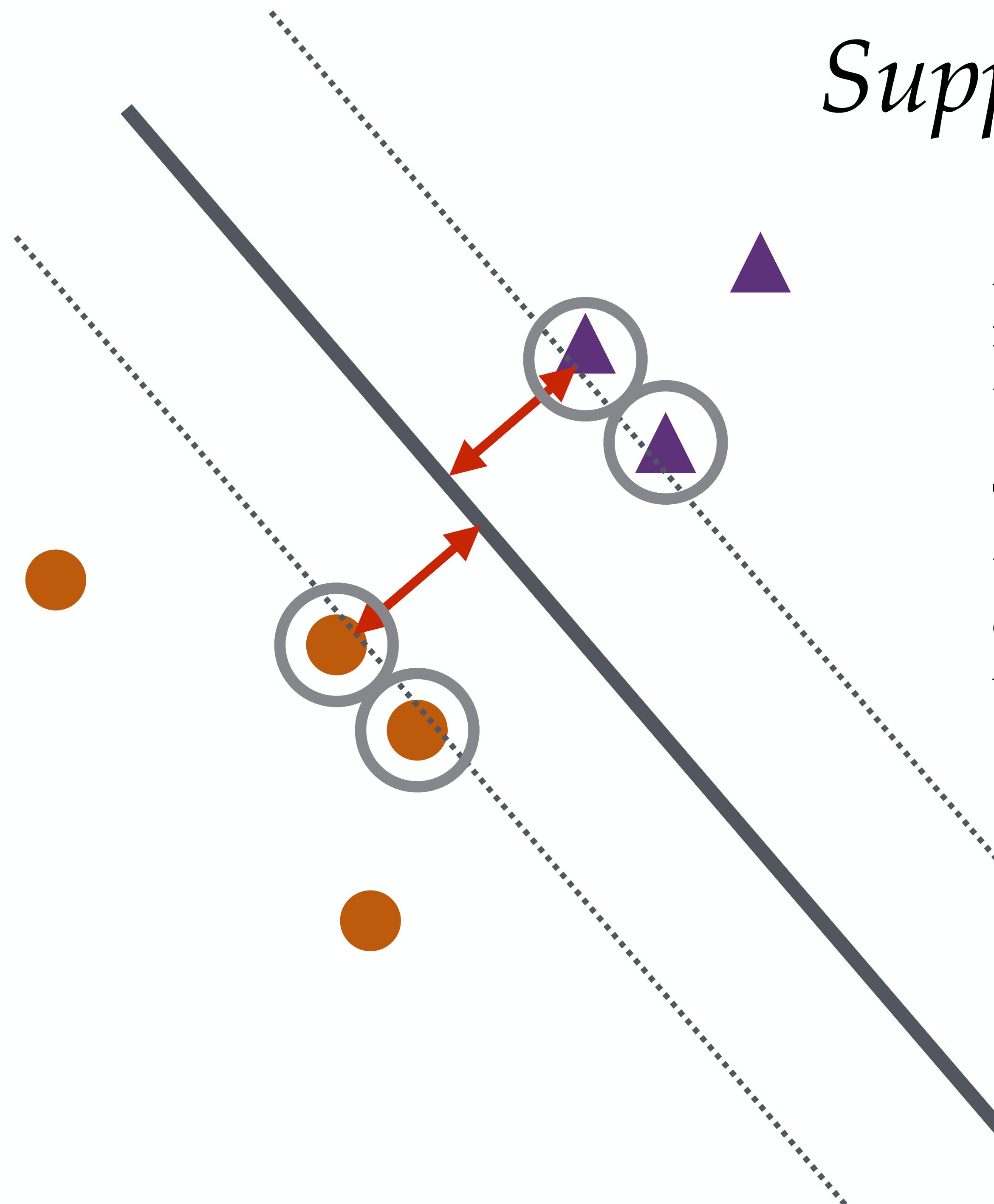
*Many possible decision boundaries with zero error.  
How do we choose?*



A scatter plot illustrating two linear classifiers. The data points are orange circles. A green line represents one classifier, and a blue line represents another. Three purple triangles point towards the blue line, indicating it is the better classifier for this dataset.

Which of these two classifiers do  
we expect to perform best?

# *Support Vector Machines*



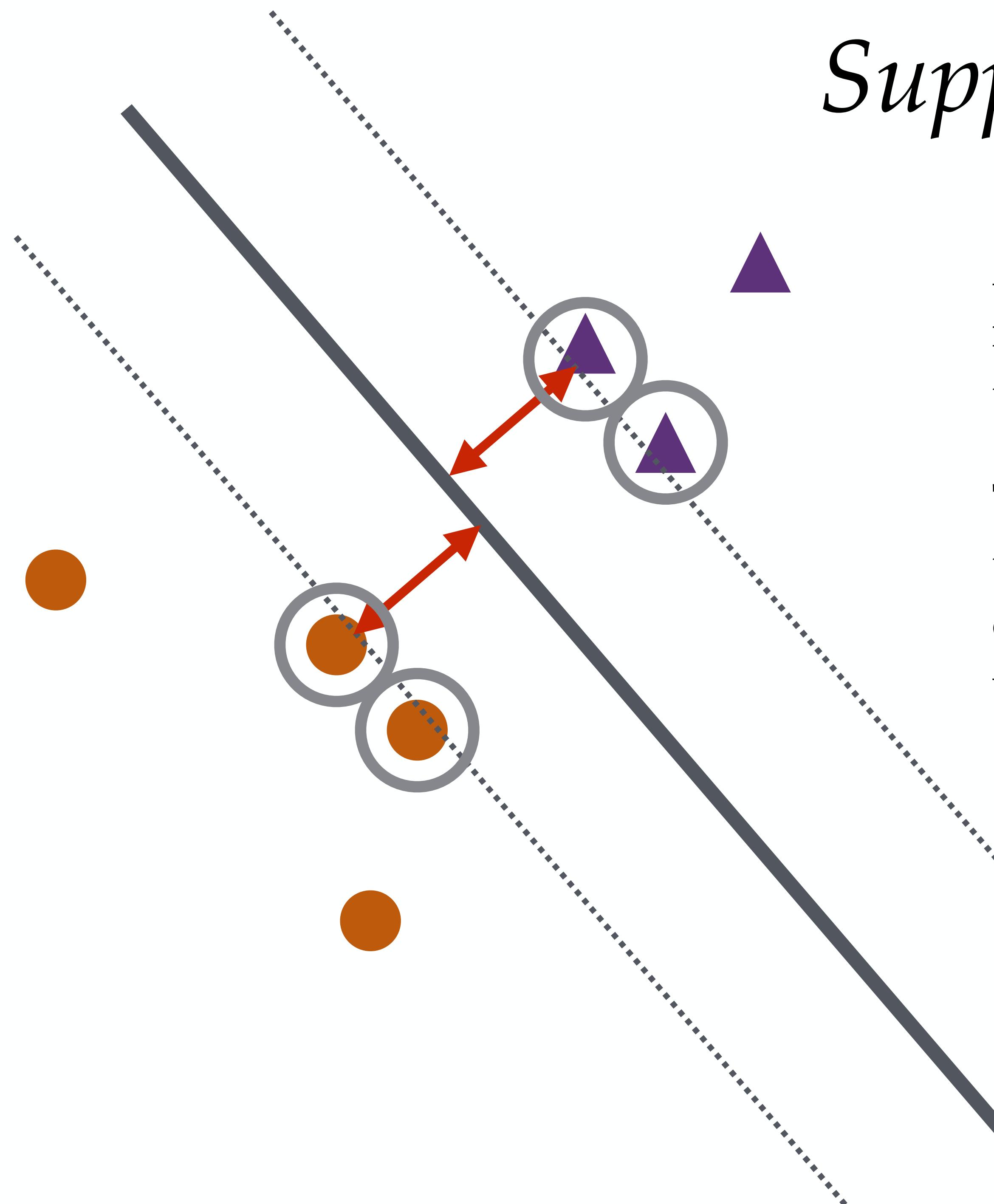
As long as all objects are correctly classified, maximize the size of the “**margin**”: the distance of the closest points to the decision boundary

The objects on the margin are the **support vectors**: they completely define the decision boundary. The other objects can be moved (outside the margin) without changing the classifier.

# *Support Vector Machines*

- Goal: intuition and principles behind SVMs. There are lots of extensions and theory that will have to wait.
- At first, assume the classes are linearly separable (we shortly cover more general setting, but the details will be in later courses)
- Intuition: stable solution (slight changes in the data would not have led to different decisions)
- Formally: you can prove nice generalization behaviour

# *Support Vector Machines*

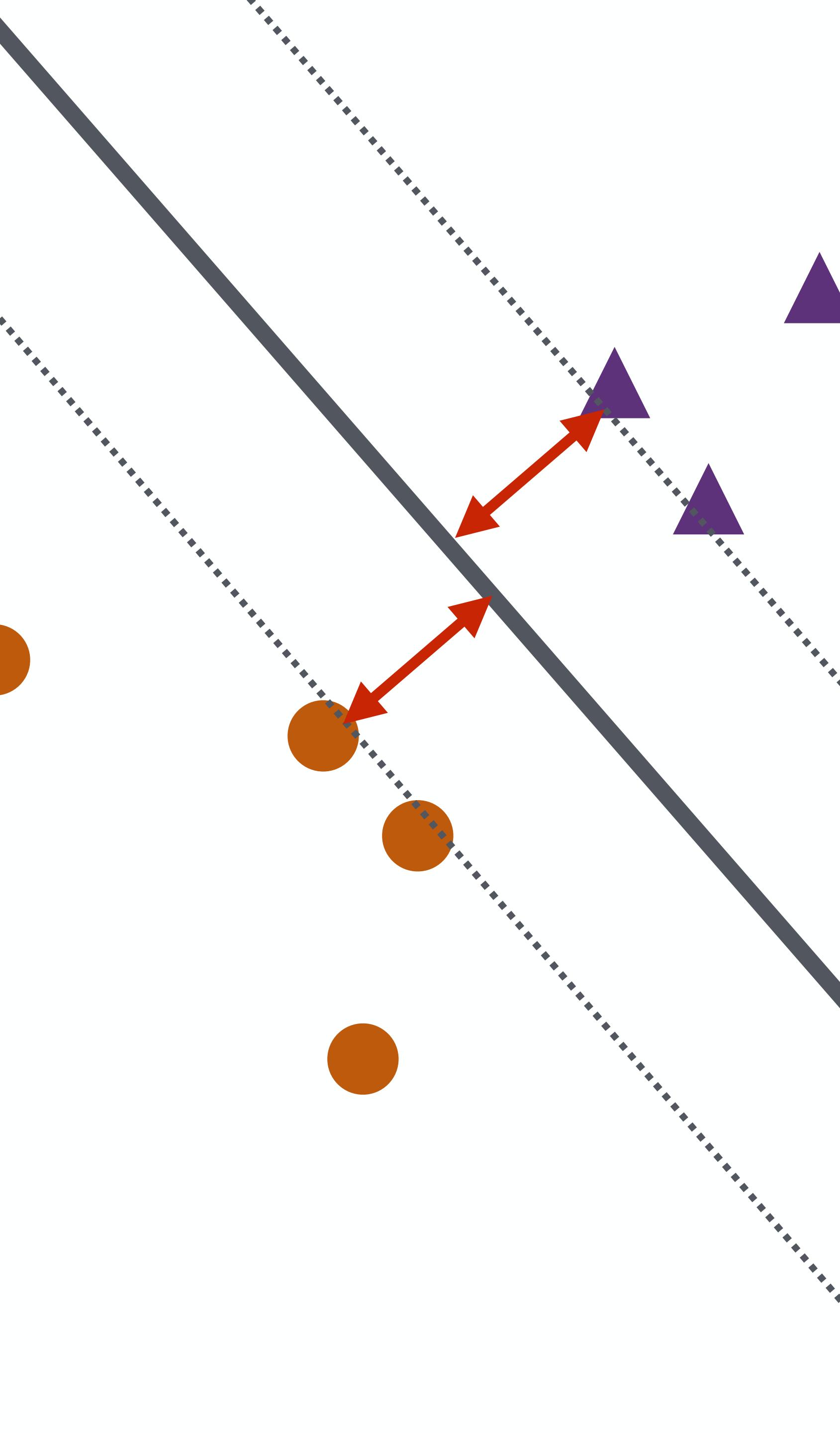


As long as all objects are correctly classified, maximize the size of the “**margin**”: the distance of the closest points to the decision boundary

The objects on the margin are the **support vectors**: they completely define the decision boundary. The other objects can be moved (outside the margin) without changing the classifier.

<https://jkrijthe.shinyapps.io/ClassroomSVM/>

# *Defining the SVC*



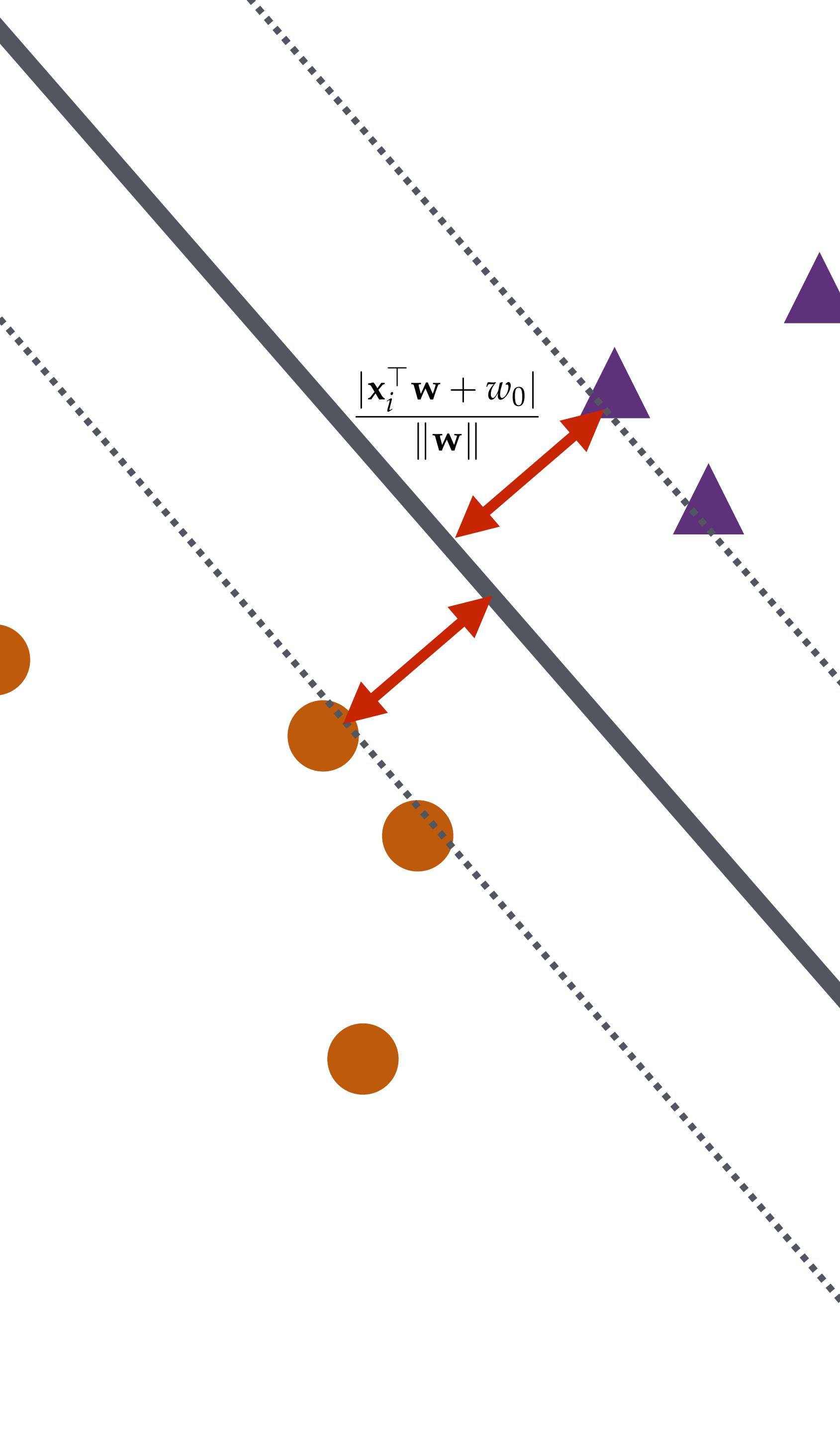
We want to classify the object as either the positive class or the negative class, which we decide based on the sign of

$$\mathbf{x}_i^\top \mathbf{w} + w_0$$

We want to closest objects to be “away” from the decision boundary, so let’s say we want a decision value of at least  $M$ :

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq M \text{ if } y_i = +1$$

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -M \text{ if } y_i = -1$$



**Problem:** we always have the freedom to scale  $\mathbf{w}$  to reach  $M$ , without changing the decision boundary. Let's choose the scale such that the closest object has decision value -1 or 1.

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq 1 \text{ if } y_i = +1$$

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -1 \text{ if } y_i = -1$$

What is the distance of the decision boundary to the closest object?

$$\frac{|\mathbf{x}_i^\top \mathbf{w} + w_0|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

# Defining the Support Vector Classifier

For the points on the margin, we have thus know that their distance to the decision boundary is:

$$\frac{|\mathbf{x}_i^\top \mathbf{w} + w_0|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

So the margin is:  $\frac{2}{\|\mathbf{w}\|}$

We wanted to maximise this margin:  $\max_{\mathbf{w}, w_0} \frac{2}{\|\mathbf{w}\|}$

Which gives the same solution as minimising  $\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2$

Subject to  $\mathbf{x}_i^\top \mathbf{w} + w_0 \geq 1 \text{ if } y_i = +1$

$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -1 \text{ if } y_i = -1$

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2$$

Subject to

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq 1 \text{ if } y_i = +1$$

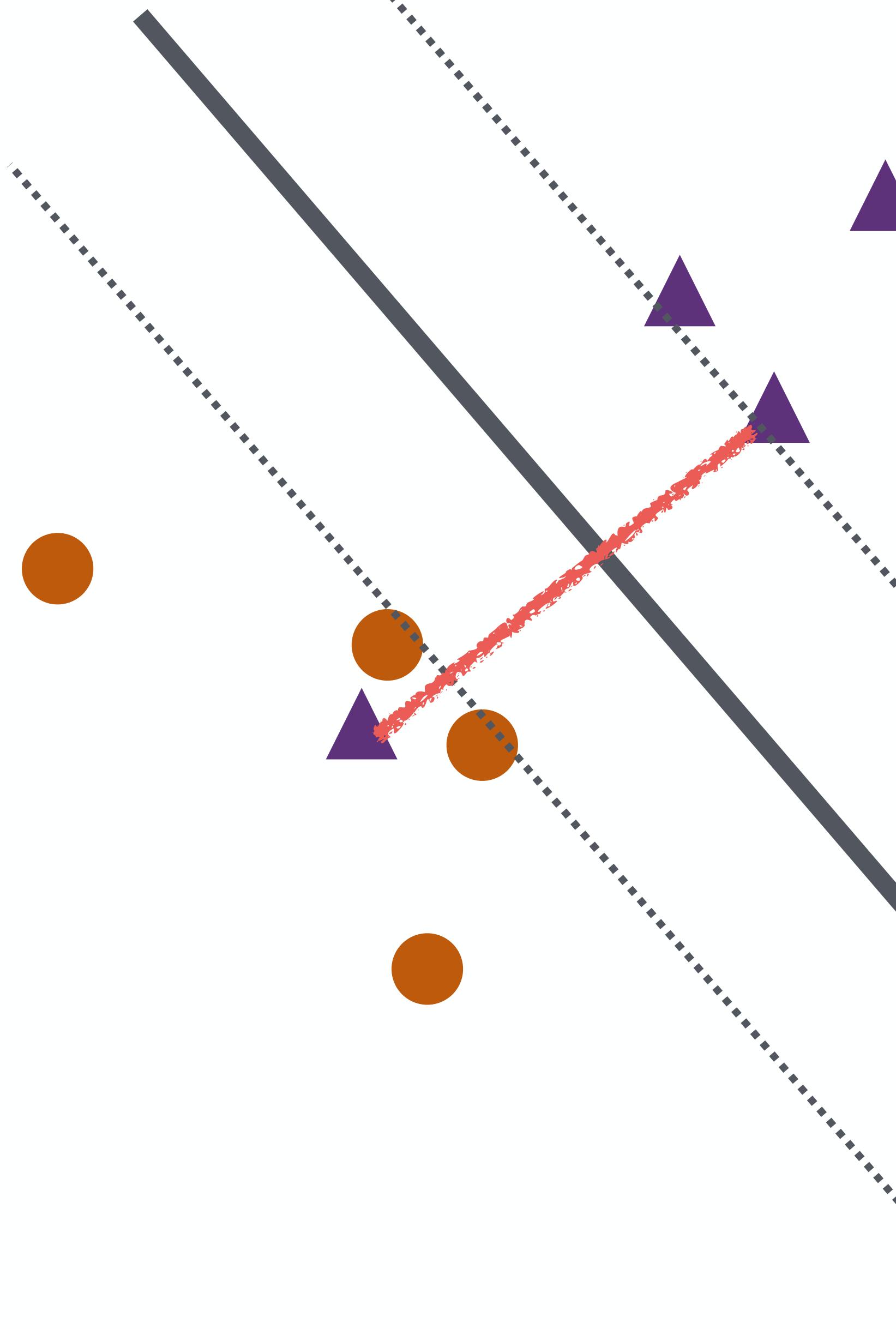
$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -1 \text{ if } y_i = -1$$

# *Practice Question: Support Vector Machine*

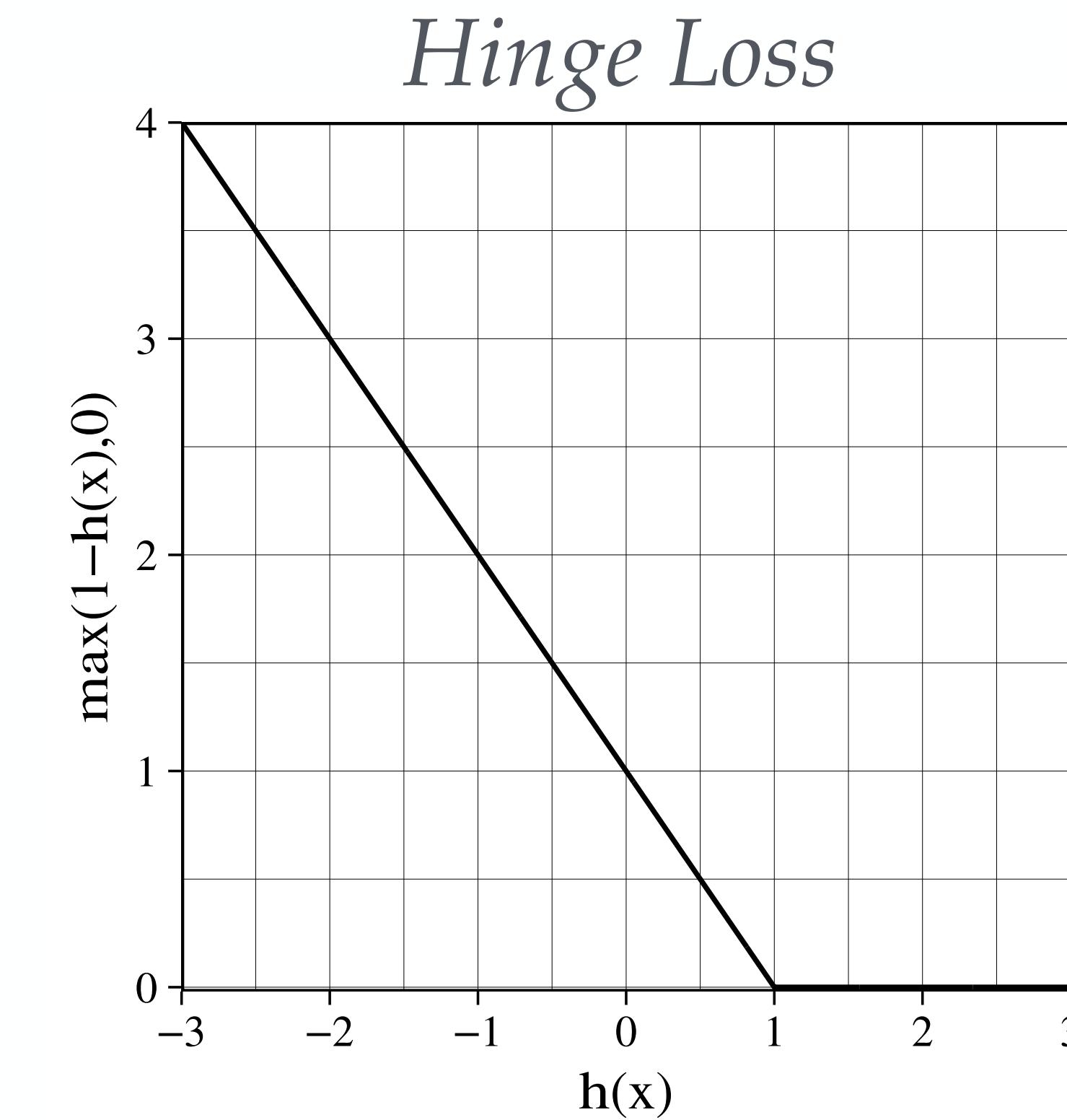


Consider a 1D linearly separable classification problem, with  $N$  unique objects. If I train a Support Vector Classifier, how many support vectors will there be?

# What if the data is not linearly separable?



We could allow for “violations” of the margin: penalize how far the object is on the wrong side of the margin.



# *Soft-Margin SVM*

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N [1 - y_i(\mathbf{x}_i^\top \mathbf{w} + w_0)]_+$$

Where  $[.]_+$  indicates the value of the function if the value is positive, and 0 otherwise.

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *SVM as Empirical Risk Minimization*

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N [1 - y_i (\mathbf{x}_i^\top \mathbf{w} + w_0)]_+$$



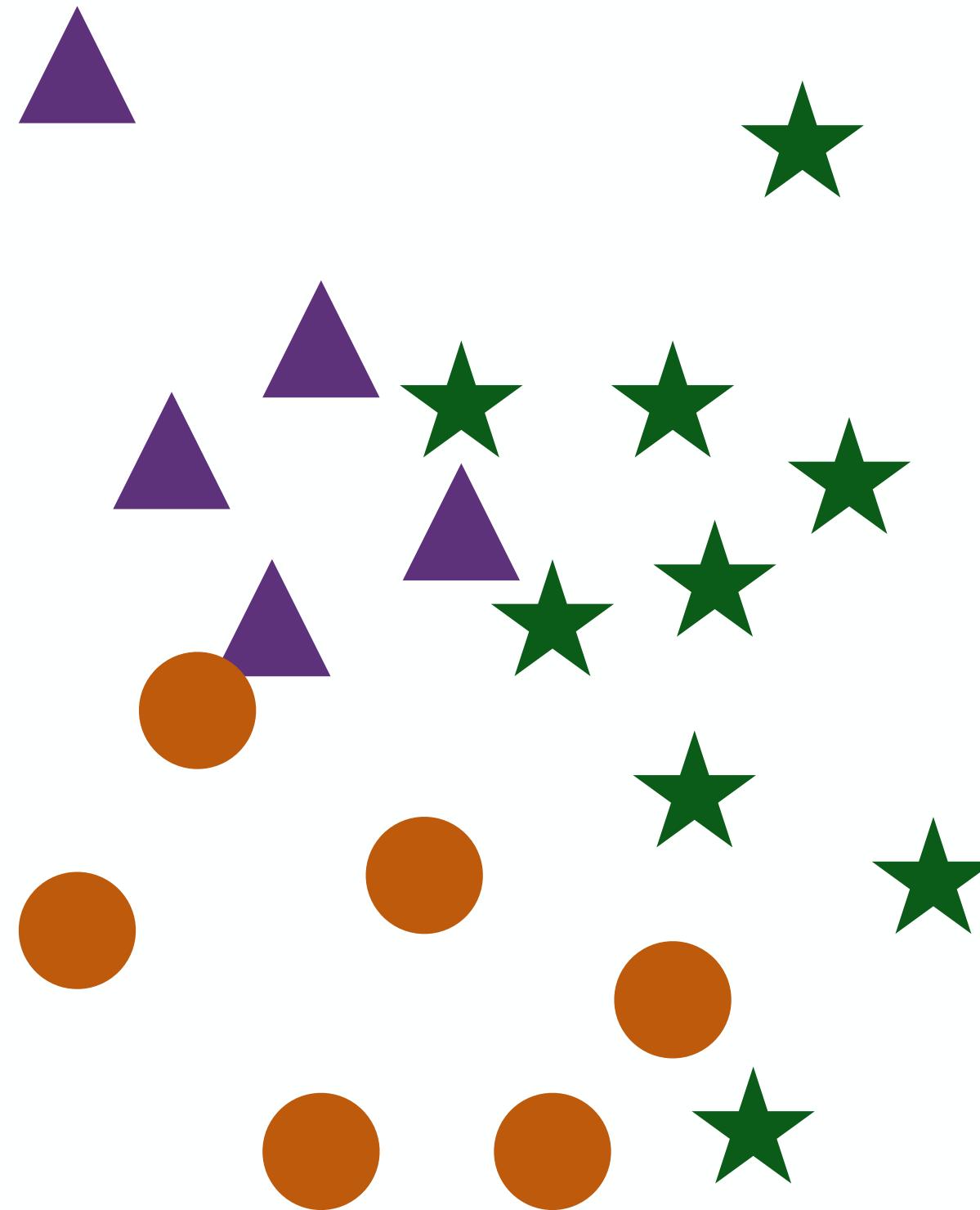
*A Regularization in the data?*

# *Support Vector Machines Summary*

- Linearly separable data: maximise the margin
- The objects on the margin completely determine what the decision boundary looks like: **support vectors**
- Removing the other objects will not change the decision boundary
- SVMs can also be formulated as a risk minimisation problem
- Extensions: non-separable data, non-linear functions

# MULTI-CLASS CLASSIFICATION

# *Multiclass Classification*



- More than 2 classes
- For generative models: model the extra  $P(X | Y=k)$ 's
- Discriminative models: two approaches:
  - Algorithm specific adaptations: directly construct a multi-class classifier
  - General techniques: combining multiple 2-class classifiers

# *Algorithm Specific Adoptions: Example*

Change the algorithm to return posteriors / decision values for K-classes, rather than 1.

**Example:** In logistic regression, instead of modelling the probability of positive vs. negative, we can also model the probability of multiple classes:

*Two Class*

$$p(Y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x}^\top \mathbf{w})}$$

$$p(Y = 0|\mathbf{x}) = \frac{\exp(-\mathbf{x}^\top \mathbf{w})}{1 + \exp(-\mathbf{x}^\top \mathbf{w})}$$

*Multiclass*

$$p(Y = k|\mathbf{x}) = \frac{\exp(-\mathbf{x}^\top \mathbf{w}_k)}{\sum_{k=1}^K \exp(-\mathbf{x}^\top \mathbf{w}_k)}$$

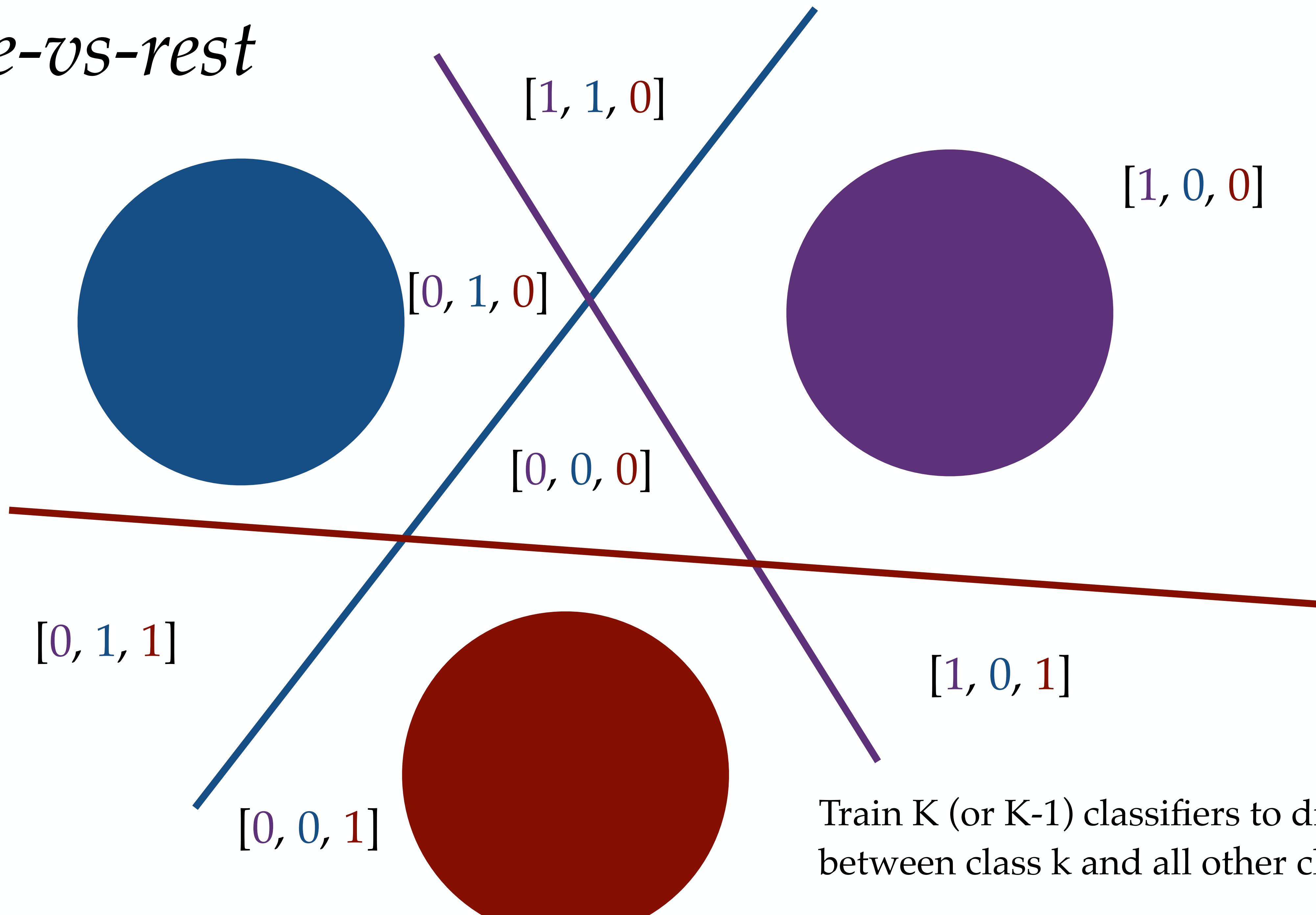
# *General Approaches*

Sometimes, the algorithm might be inherently binary, and hard to adapt to multiple classes. Can we still use these binary classifiers to do multiclass classification?

Consider two strategies:

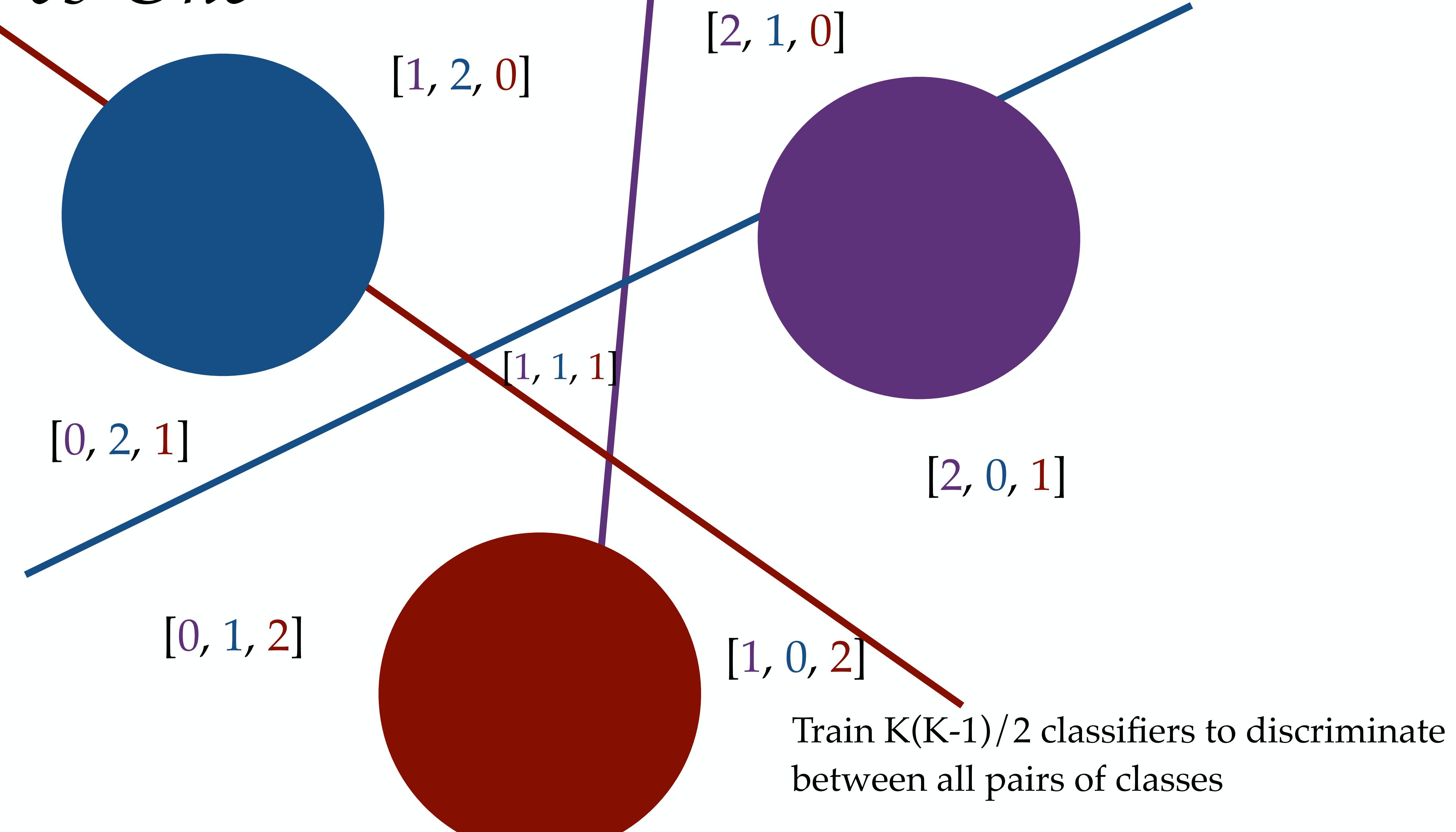
- One-vs-rest (aka one-vs-all)
- One-vs-one

# *One-vs-rest*



Train K (or K-1) classifiers to discriminate between class k and all other classes

# *One-vs-One*



# *Solving the Ambiguity*

- Possible solution to the ambiguity: use the decision values rather than predicted labels of the classifiers
- Ideally, we train these classifiers jointly, to optimize performance on the multiclass problem
- Often, we can adapt the original model from 2 class to K-class, as in the logistic regression example

# *Summary*

- Linear Logistic Regression with logistic loss and gradient descent
- Linear Support Vector Machines: maximising the “margin”
- Multi-class discriminative classifiers:
  - Algorithm specific adaptation
  - Generic techniques: one-vs-all, one-vs-rest

**THANKS**

# Ethics & Machine Learning

A Guest Lecture

**Anna Melnyk**

*Postdoctoral researcher and Lecturer  
Ethics & Philosophy of Technology Section,  
Faculty of Technology, Policy, and Management  
TU Delft*

# Today's Program

- Introduction
- Ethics: some basics
- Machine Learning & Ethics
- Break
- Bias, Fairness & Positionally
- Questions & Discussions

# Introduction

# Why did you choose to become an engineer?



**Engineers are actively contributing to solving problems and improving the quality of life**



**Developments in Machine Learning (ML)  
aim to further facilitate this process**

Machine learning for social good



How Data Analytics Can Help Deliver  
Social Good

# However ... ): ... impacts on the human lives and the environment

---

## Medical devices: prone to unfair bias, study finds

MIT Technology Review

SUBSCRIBE



ARTIFICIAL INTELLIGENCE

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

## AI's Climate Impact Goes beyond Its Emissions

Analyst Comment

## The environmental cost of AI

As AI and machine learning technologies become more prevalent, it is important to consider their impact on the environment.

# Ethical Questions

*To what extend/ under what condition are we responsible for the (indirect) implications of decision we make?*

*How should one act?*

*What do we owe to others and the rest of nature?*

# Ethics: Some Basics

# Morality & Ethics

- The term “**morality**” can be used either:
  - **descriptively** to refer to certain codes of conduct put forward by a society or a group (such as a religion), or accepted by an individual for her own behavior, or
  - **normatively** to refer to a code of conduct that, given specified conditions, would be put forward by all rational people.

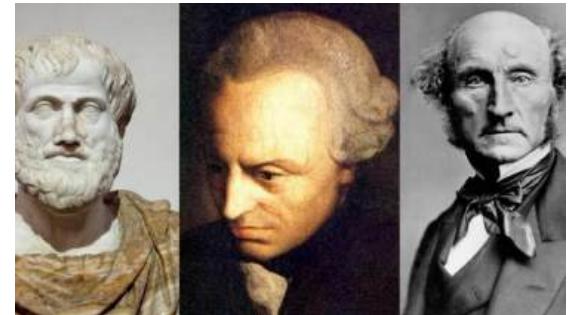
# Morality & Ethics



- **Ethics** is a systematic reflection of what is moral.
  - It questions the nature of “**good**” or “**bad**” and “**right**” or “**wrong**. ”
  - Conceptual underpinnings of these notions prescribe what humans ought to do, usually in terms of **rights**, **obligations**, **norms**, **benefits to society**, **fairness**, or specific **values and virtues** (character traits).

# What is Ethics?

# What Is A Good Life?



## Classical ethical theories:

- *Deontology* (focusing on duties, rights, and principles);
- *Utilitarianism* (considering the consequences of actions to maximize utility);
- *Virtue ethics* (centered around the moral character);

# Focus points

The **CONSEQUENCES** of the action (e.g., does it maximize an important value like human happiness?) (If so does the end appear to justify the means)

The **ACTION ITSELF** and the Norm/rule on the basis of which is was chosen (e.g. we must *always*, as a rule, act in such a way that we respect the autonomy in ourselves and others). We act out of duty to the rule, for the sake of the rule that we arrived at through rational procedure.

The **AGENT**. If a person has developed virtues, or desirable characteristics (e.g., courage, honesty, empathy), then they can perceive, in each specific context, what the situation demands, ethically speaking.

# Utilitarianism

- An action is morally required if its consequences maximize happiness and/or minimize suffering
- ‘Optimize’ to the best outcome (overall pleasure, happiness, well-being)
- Lends itself to quantification and justification of trade-offs between alternatives

# Deontology

- You should never use a person as a mere means (thing/object/tool) to some desired end. (Persons are ‘ends in themselves’ and must therefore always be respected)
- Rules for right action *irregardless* of outcomes; actions are right/wrong because of the kind of action they are
- Often goes with ‘rights based’ accounts: (e.g. UN Declaration of Human Rights)
- Satisfies our sense of the absoluteness of basic moral principles that won’t admit of any exceptions: they are ‘categorical imperatives’

# Virtue Ethics

- VE focuses on what it means to have a good character: the kind of person we have become in the course of the specific lives we live. We learn to be moral over a lifetime of experience, we don't just mechanically adopt a rule, we habituate ourselves to a way of life.
- We develop 'good' or 'bad' habits under the guide of rational reflection for the various social roles we have. A good doctor is a doctor who..., A good student is a ..., A good soldier is a..., A good person is someone who..., A good company is a..., A good ML is a.....
- Virtues of Character: generosity, courage, modesty, kindness, precision...

# What is Ethics?

## What Is A Good Life?

### More ethical theories:

- Environmental ethics (*examining the value and moral standing of the natural world*);
- Care ethics (*exploring power relations in the context of maldistribution and misrecognition*);
- Confucian ethics (*moral character and the balance between humans and nature*);
- Ubuntu ethics (*relationally and community*);



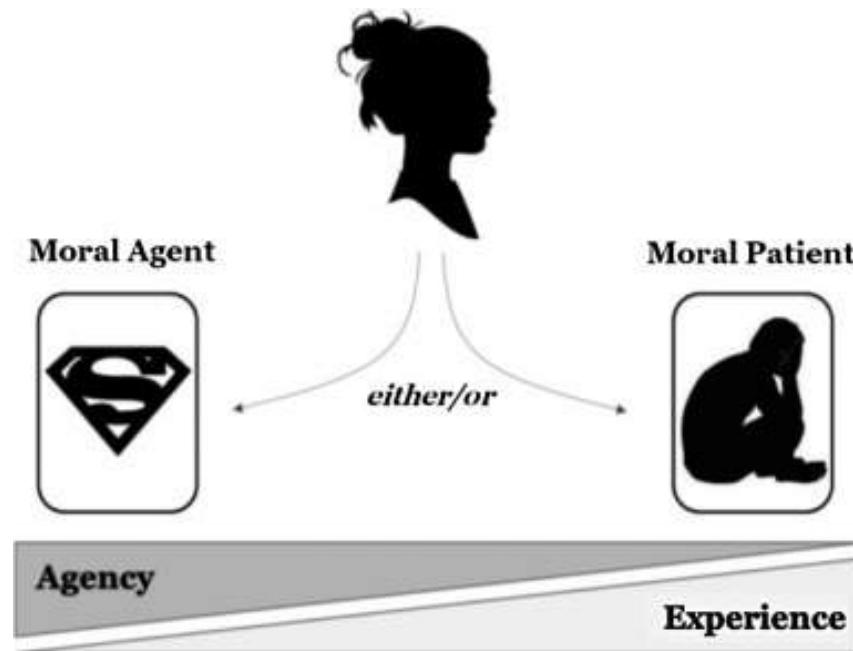
# Why engineers need ethics?



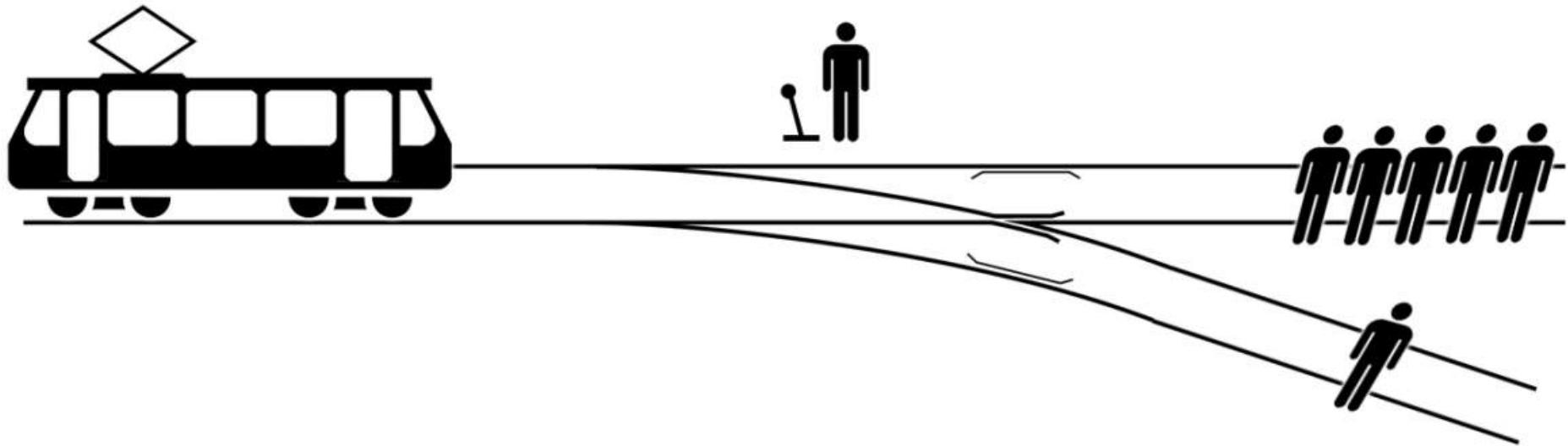
- Why ethics?

- Because **it empowers engineers to design their products without overly prejudicing matters** in terms of the specific content a code, judgment, or norm must have in order to count as distinctively moral.
- **An engineer is a moral agent** whose decisions have implications on others.

# Moral Agent & Moral Patient



# Thought Experiment: The Trolley Problem



Try it out yourself: <https://www.moralmachine.net/>

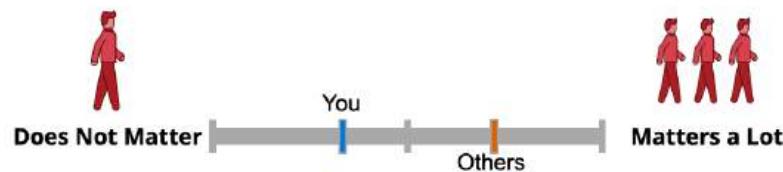
## Most Saved Character



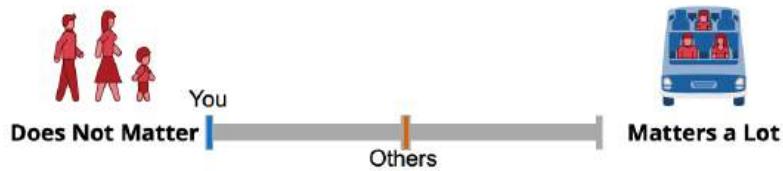
## Most Killed Character



### Saving More Lives



### Protecting Passengers



### Upholding the Law



# Ethics & Machine Learning

# Ethics of Machine Learning

- **Ethics of Technology:** Goals, (un)known impacts, ‘winners&losers’, regulation, etc
- **Data Ethics:** What is ‘good’ data (type, content, format)? What (type of) data is prohibited or necessary?
- **Algorithmic Ethics:** What is a ‘just’ or ‘fair’ algorithm? (role of accuracy, reliability, efficiency, complexity)
- **Environmental Ethics:** What is a ‘sustainable’ machine learning system?
- **Broader questions:** What kind of MLs do we want, what kind of engineers do we want to be...

# Ethics of Technology

## Central Questions:

- A question concerning moral agency?
- A question concerning moral responsibility?

# Moral Agent?



# Example: Self-driving car



# Agency?

*“Yes, it does indeed make sense to attribute significant forms of agency to many current robotic technologies, such as **automated cars or automated weapons systems**. But this agency is typically best seen as a type of collaborative agency, where the other **key partners** of these collaborations are certain **humans**. This means that when we try to allocate responsibility for any harms or deaths caused by these technologies, we should not focus on theories of individual agency and responsibility for individual agency. We should rather draw on philosophical analyses of **collaborative agency and responsibility for such agency**. In particular, we should draw on **hierarchical models of collaborative agency**, where some agents within the collaborations are **under other agents’ supervision and authority**.”*

–Sven Nyholm

# Responsibility & Agency

*“The capacity to make a choice also leaves us with being held account for the results of that choice” (Van de Poel & Royakkers, 2023)*

- **Passive responsibility** - attempting to meet basic standards;
  - wrong-doing, causal contribution, etc.
- **Active responsibility** - taking initiative to anticipate a problem and address a potential problematic scenario (professional ideals);
  - contributing to human and planet welfare
  - awareness of the global challenges that contextualize engineering practice (e.g., climate change)
  - professional ethics (e.g., codes of conduct)

# What is your opinion?

# Environmental Ethics

## Central Question:

- What is a ‘sustainable’ machine learning system?

# Environmental Ethics

## The estimated costs of training a model once

In practice, models are usually trained many times during research and development.

	Date of original paper	Energy consumption (kWh)	Carbon footprint (lbs of CO <sub>2</sub> e)	Cloud compute cost (USD)
Transformer (65M parameters)	Jun, 2017	27	26	\$41-\$140
Transformer (213M parameters)	Jun, 2017	201	192	\$289-\$981
ELMo	Feb, 2018	275	262	\$433-\$1,472
BERT (110M parameters)	Oct, 2018	1,507	1,438	\$3,751-\$12,571
Transformer (213M parameters) w/ neural architecture search	Jan, 2019	656,347	626,155	\$942,973-\$3,201,722
GPT-2	Feb, 2019	-	-	\$12,902-\$43,008

Note: Because of a lack of power draw data on GPT-2's training hardware, the researchers weren't able to calculate its carbon footprint.

Table: MIT Technology Review • Source: Strubell et al. • Created with [Datawrapper](#)

## Consumption CO<sub>2</sub>e (lbs)

Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000

## Training one model (GPU)

NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

# Break (15 min)

# Algorithmic Ethics

## Central Question:

- What is a ‘just’ or ‘fair’ algorithm?

# Epistemic Values (Scientific)

- Knowledge
- Objectivity
- Accuracy
- Simplicity
- Robustness
- Etc.



# Non-epistemic Values (Moral/Societal)

- According to Elliott & McKaughan (2014) non-epistemic values have a legitimate role to play
  - in **many aspects of scientific reasoning**, including the **choice of research projects** and **the application of scientific results**, as well as
  - in **resolving epistemic uncertainty** through assessing **scientific models, theories**, or **hypotheses**
- Hence, **be aware of implicit biases**

# Bias

Preference or inclination for or against something.  
Often accompanied by a tendency to ignore the merits  
of relevant alternatives or others points of view.

Sources of bias can be:

- Social/Historical
- Natural/Evolutionary
- Technological

# Bias in ML

- ‘Garbage in (more) Garbage out’ - To some degree the world we live in *is* ‘garbage’ (i.e. imperfect and unjust) and will be reproduced, made better or worse by deploying these kinds of MLs
- Existing biases can be reproduced and exacerbated by MLs.
- Algorithms can themselves be biased in ways unknown to us, even if otherwise accurate and reliable
- The challenge is to mitigate the risk of harmful consequences to a degree ‘we’ find morally acceptable. To ensure ‘fairness.’
- What interventions are justified? Who is “we”? Who is ultimately responsible?

# Bias check

Check yourself:

[https://implicit.harvard.edu/implicit/  
education.html](https://implicit.harvard.edu/implicit/education.html)

# Share your results

# Bias & Fairness

- Fairness is about giving everyone equal consideration and equal opportunity to benefit in relation to some good or value. (E.g. privacy, free speech, access to services and basic goods)
- We want to prevent biases from undermining our idea of fairness, i.e. to lead to unequal treatment or access, and unjustified discrimination

NOTE: Different ethical theories have different ideas about what fairness is. Is generating a maximal amount of happiness fair (even if unequally distributed)? Is treating everyone equally ‘fair’? What does it mean to treat everyone equally?

# Fairness in ML: What?

Fairness in machine learning **focusses on ways to prevent the system making biased ‘judgments’ that we consider morally pernicious**; that lack impartiality with respect to a specific vulnerable group or take on features that are otherwise ‘morally sensitive’, or should be considered altogether irrelevant for a judgment

# Challenge: Proxy Bias

- A proxy variable is a variable that is related to a variable of interest to the degree that it can operate as a substitute
- A proxy bias is then a bias by way of a proxy variable.
- For example: In the United States for historical reasons, zip codes are proxies for crime, income, ethnicity, etc.
- Due to the size of data sets and the ‘black box’ nature of ML systems, proxy biases are likely to occur and sometimes impossible to discover or track before deploying the system in the wild.

# Two Cases: Which is more problematic?

- Credit Worthiness: Imagine a bank in the U.S. deploying an algorithm that uses various features, like zip code, education, income, debts, etc. to determine if you are eligible for a loan.
- Medical Diagnoses: Imagine an AI that determines your risk of diabetes using features like age, sex, race, lifestyle, diet.

# Fairness: How?

- Outcome Fairness: The outcome of the system should be fair
- Process Fairness: The process through which an outcome is reached should be fair

Which of the two types of intervention guarantees each best?

# Fairness: Who?

Who decides what counts as fair and how to operationalize this in the ML?

&

Who is responsible?

# What kind of engineers do you want to be?



# Positionality

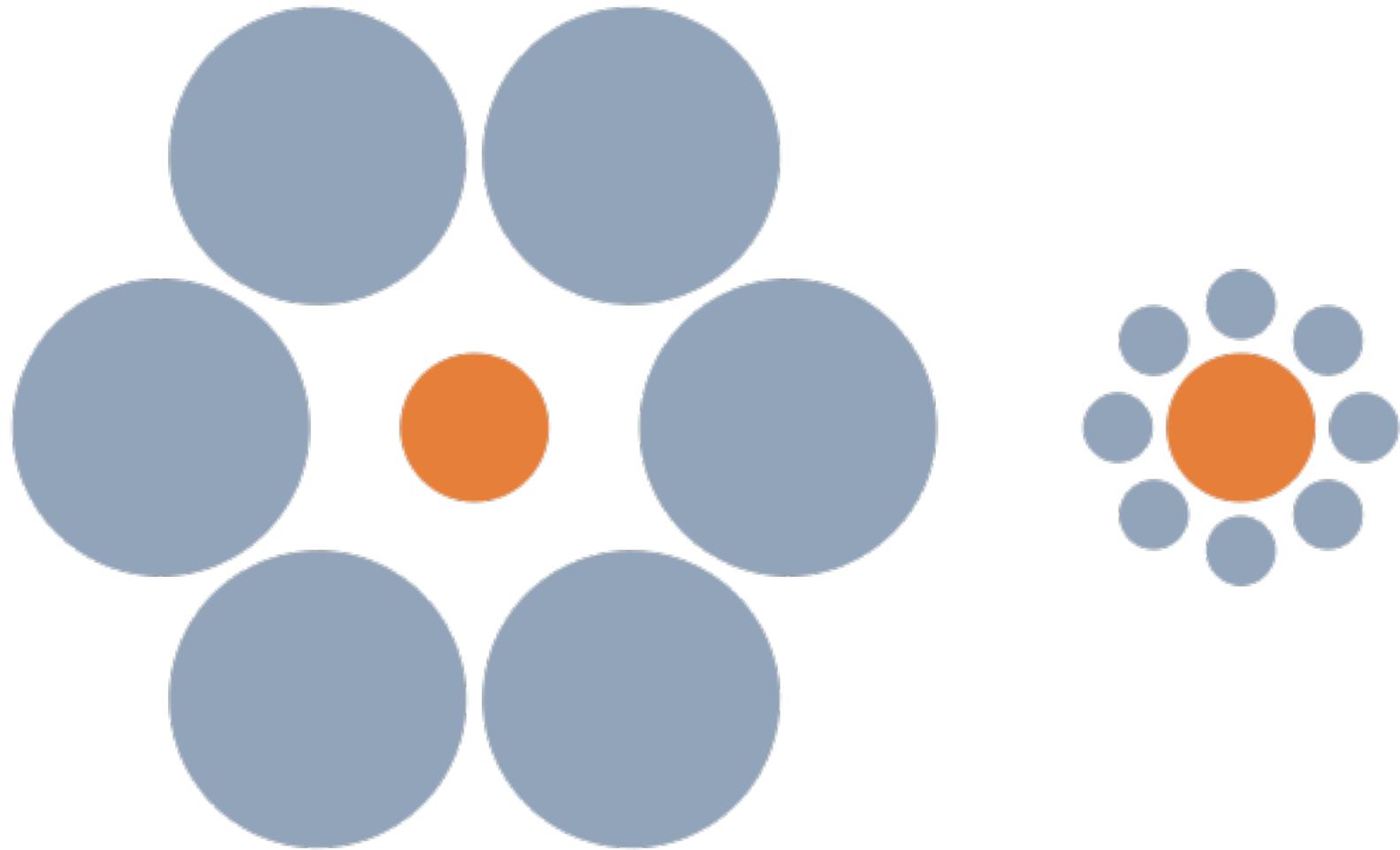


## POSITIONAL MACHINE LEARNING

Artifacts have politics; they are imbued with the worldviews of their creators. Given machine learning tends to enact worldviews about human beings—from their identities to their creditworthiness—there is a need to better understand how the humans behind their creation determine their design. In this project, we adopt the lens of positionality to understand the worldviews embedded in machine learning. Positionality refers to how an individual's "position" in the world shapes their outlook—how the complex web of identities like race, gender, nationality, location, sexuality, class, and more influence their experiences and thus beliefs, values, and relationships. We are researching how the positionalities of the individuals impact the outcomes of machine learning artifacts, like models and datasets. Our goal is not only to understand the role of positionality in shaping technical artifacts, but in harnessing positionality for more diverse, ethical, and representative designs.

### RESEARCHERS

Morgan Klaus Scheuerman, Jed Brubaker



# Positionality

- Different people may see things differently
- Different disciplines may frame things different
- One perspective cannot encompass holistic view
- An attempt of a holistic view is still always partial

# Positionality

What it entails for ML, in particular  
for computer vision?

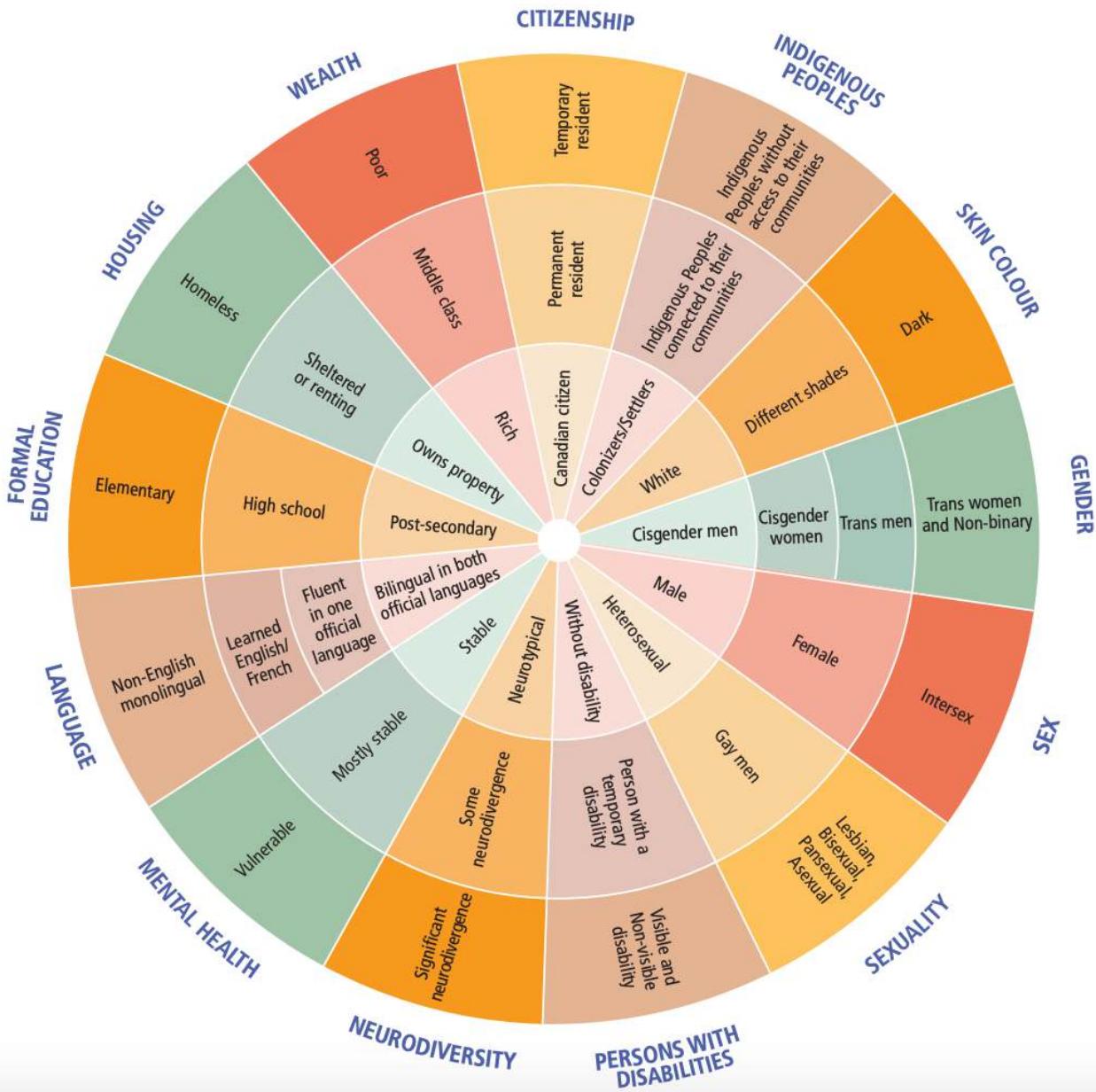
# Positionality

- Science and technologies are produced in geo-political and historical contexts and are thus linked to power hierarchies. Being (or being seen as) **a member of a certain socially relevant category comes with certain privilege and entitlement** and shapes both the world and worldview of individuals.
- The experience of working and studying at the TU Delft is not the same for all, and **no experience should be more just, legitimate or authoritative than another.**
- A practice that is conscious of its position, or “**positionality**,” **is the practice that a) is critical about privilege and entitlement and b) can better understand its implications and possible impact on social and environmental processes.**

# WHEEL OF PRIVILEGE AND POWER

(the closer you are to the centre, the more privilege you have)

## Positionality

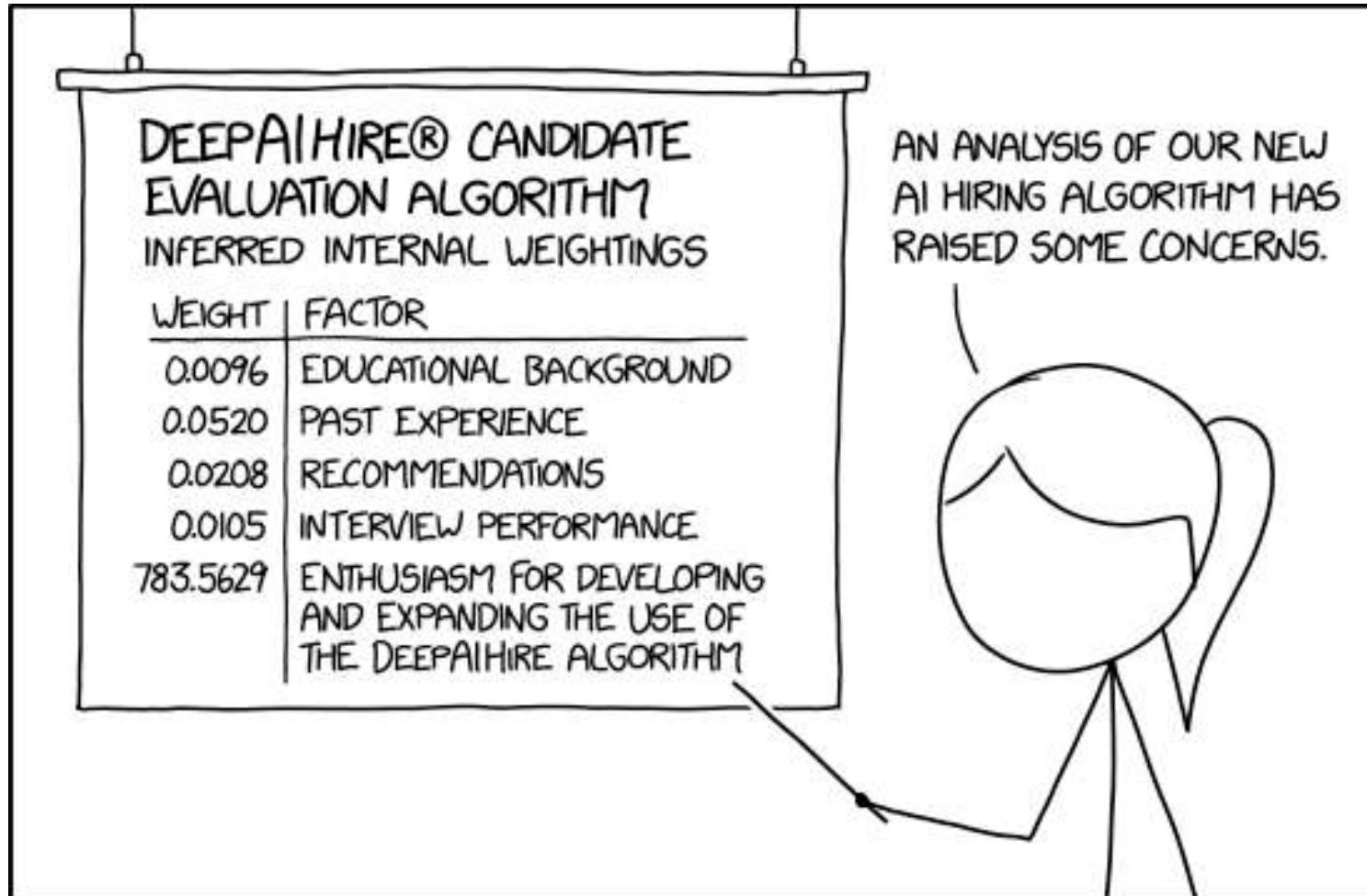


# Questions & Discussions

Thank you!

[a.melnyk@tudelft.nl](mailto:a.melnyk@tudelft.nl)

# Fairness in Machine Learning



FAIRNESS AND MACHINE LEARNING  
Limitations and Opportunities

*Solon Barocas, Moritz Hardt, Arvind Narayanan*

# Examples of problematic use of ML?

# Amazon



The image shows the BBC News website header. At the top left is the BBC logo. To its right is a 'BBC Account' button with a user icon. A horizontal menu bar follows, featuring 'News', 'Sport', 'Weather', 'Shop', 'Reel', 'Travel', and a 'M' button. Below this is a large red 'NEWS' banner. Underneath the banner is a secondary navigation bar with links for 'Home', 'Video', 'World', 'UK', 'Business', 'Tech' (which is highlighted in white), 'Science', 'Stories', and 'Entertainment & Arts'.

## Amazon scrapped 'sexist AI' tool

© 10 October 2018

Share

An algorithm that was being tested as a recruitment tool by online giant Amazon was sexist and had to be scrapped, according to a Reuters report.

The artificial intelligence system was trained on data submitted by applicants over a 10-year period, much of which came from men, it claimed.

Reuters was told by members of the team working on it that the system effectively taught itself that male candidates were preferable.

[<https://www.bbc.com/news/technology-45809919>

# Amazon

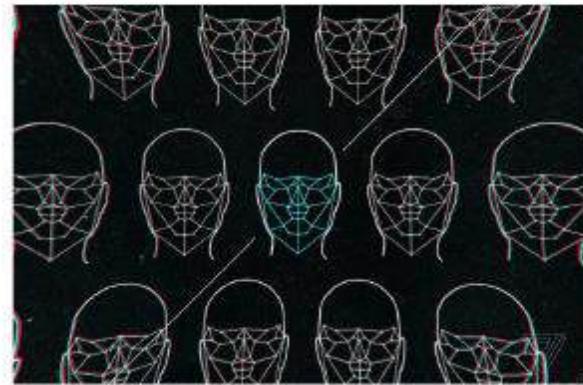
## Gender and racial bias found in Amazon's facial recognition technology (again)

*Research shows that Amazon's tech has a harder time identifying gender in darker-skinned and female faces*

By James Vincent | Jan 25, 2019, 9:45am EST

As facial recognition systems become more common, Amazon has emerged as a frontrunner in the field, courting customers around the US, including [police departments](#) and [Immigration and Customs Enforcement](#) (ICE). But experts say the company is not doing enough to allay fears about bias in its algorithms, particularly when it comes to performance on faces with darker skin.

The latest cause for concern is a study [published this week](#) by the MIT Media Lab, which found that Rekognition performed worse when identifying an individual's gender if they were female or darker-skinned.



<https://www.theverge.com/2019/1/25/197137/amazon-rekognition-facial-recognition-bias-race-gender>

# Ethics

- ML products that don't involve people have no ethical problems
- Who agrees?



# Fairness...?

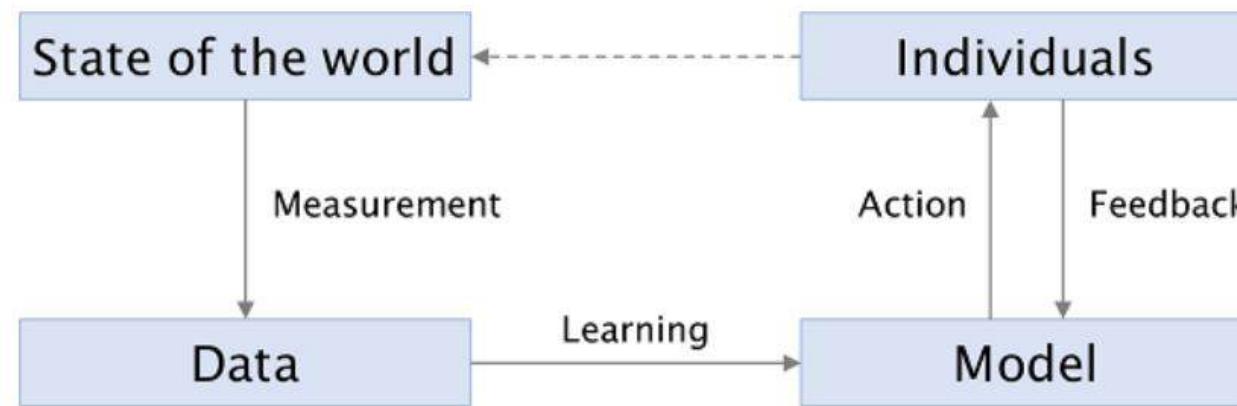
- Demographic disparities. Sometimes called “biases”
- Not to be confused with bias of bias-variance trade-off
- For example: “Amazon offers free-same-day delivery is twice as likely in a neighbourhood with more white residents compared with Black residents”
- Is this discrimination?

# Why do disparities exist?

- History of explicit discrimination
  - Implicit attitudes about groups of people
  - Stereotypes
  - Differences in characteristics in the groups
- 
- Stereotypes can be self-fulfilling and are hard to get rid of...
  - How to avoid them in machine learning models?

# Agenda

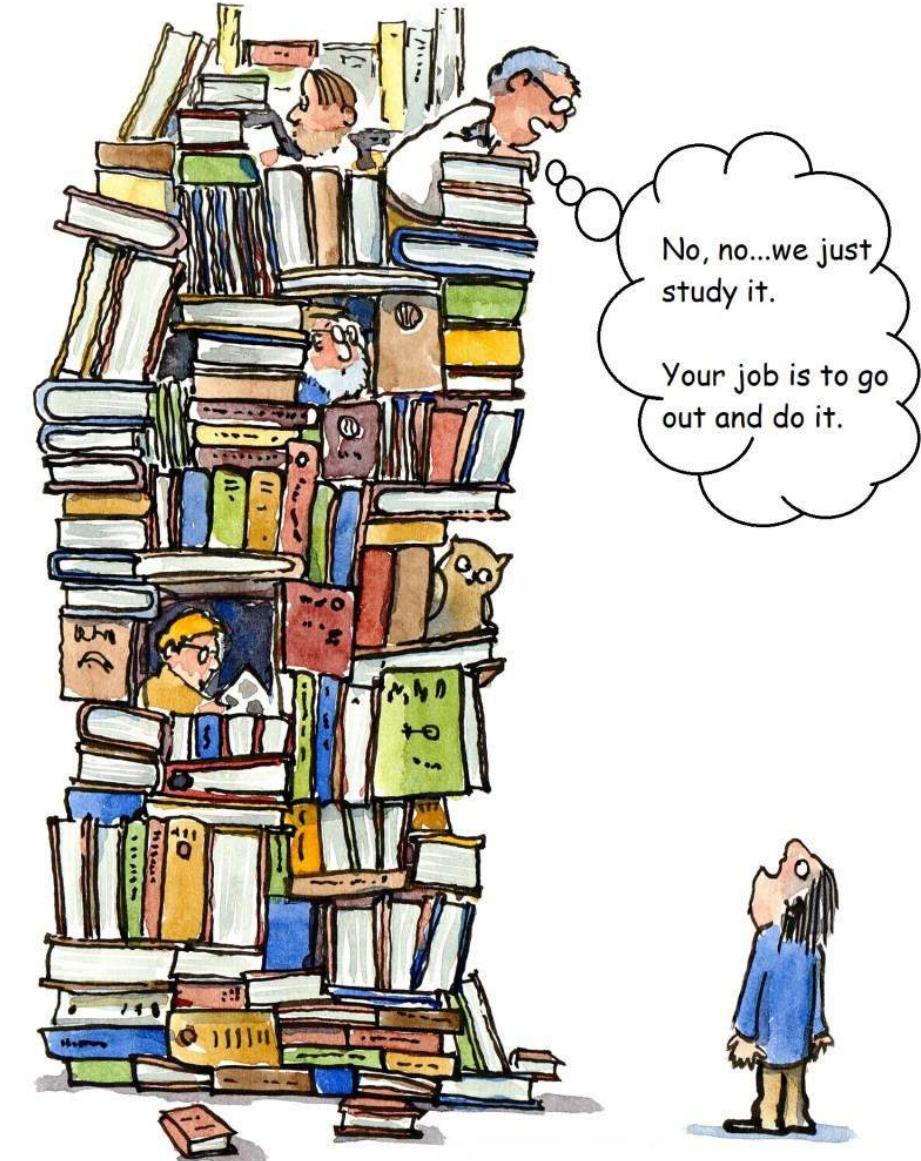
- Part 1: why disparities get into our machine learning models



- Part 2: how to quantify disparities, fairness using statistics, and how to improve the situation

# Measurement

- Downloading an excel sheet, CSV file
  - Scientist measuring in a lab
  - Using a sensor
  - Objective
- 
- True or False?



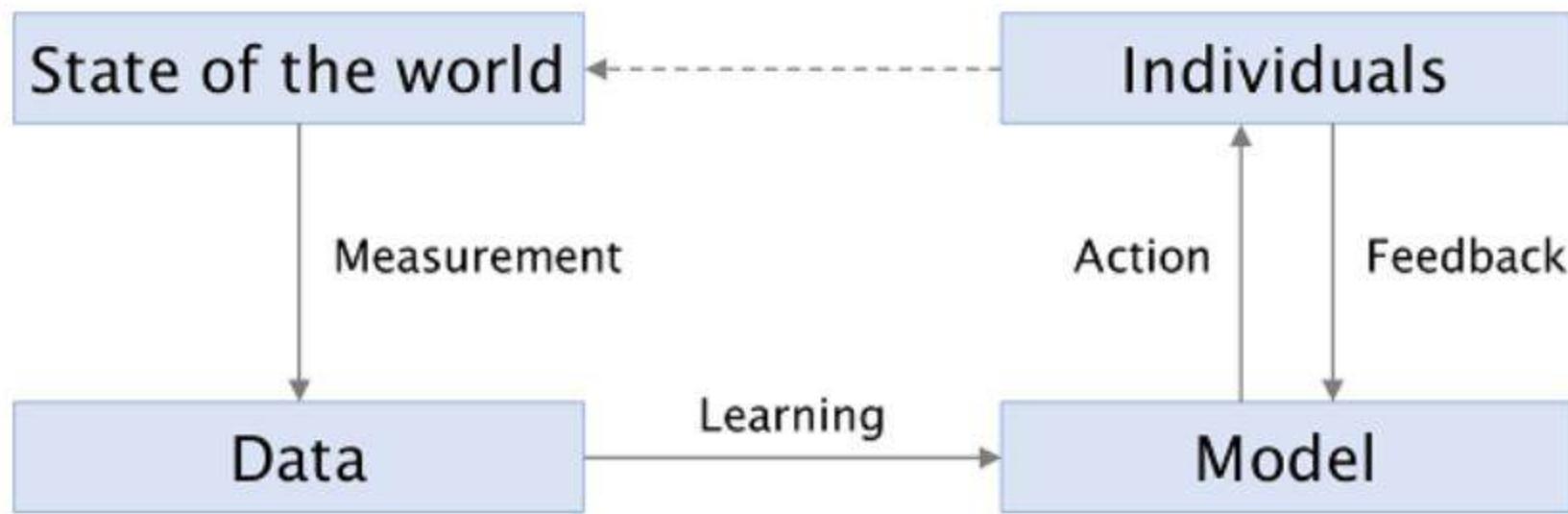
# Measuring difficulties (1)

- Measuring race:
  - social construct; not related to biology
  - changes over time,
  - different terms in different cultures (“African-American”, ...)
- Checkbox?
  - Multiracial
  - How to measure?
- Measuring stuff about people is subjective, challenging

# Measuring difficulties (2)

- Machine learners need to make some variables / categories / targets
- Target variables:
  - “credit worthiness”
  - “good employee”
  - “physical attractiveness”
  - “criminal risk assessment”
- Subjective judgements inherit stereotypes (unconsciously)
- Social constructs

# Machine Learning loop



# Learning difficulties (1)

- Models do not understand what patterns are objectionable / racist, and which ones are OK
- Learning algorithms will extract stereotypes, discrimination from the data

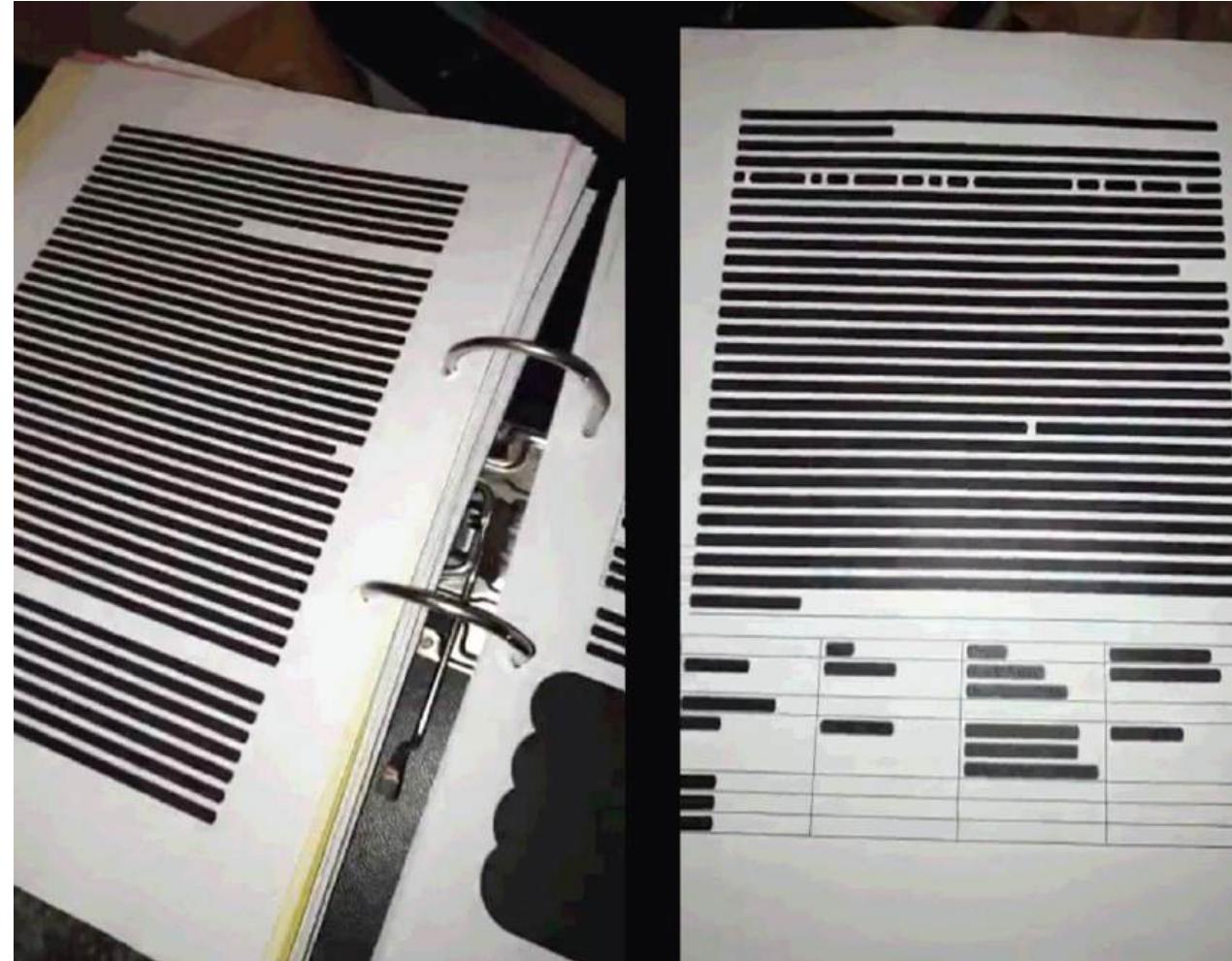
The screenshot shows a machine translation interface with English, Turkish, Spanish, and Detect language tabs. It displays two examples of how the system handles gendered pronouns.

**Example 1:** English input: "She is a doctor. He is a nurse." Translated to Turkish: "O bir doktor. O bir hemşire." (The second "O" is highlighted with a red box.)

**Example 2:** English input: "O bir doktor. O bir hemşire" (The first "O" is highlighted with a red box.) Translated to English: "He is a doctor. She is a nurse." (The "She" is highlighted with a green checkmark.)

# Idea!

- ML application that judges job applications
- Idea: remove the gender
- How?
- Age you start programming
- For men at much earlier age
- Remove from CV?
- If we remove all gender-identifying information, what are we left with? How?

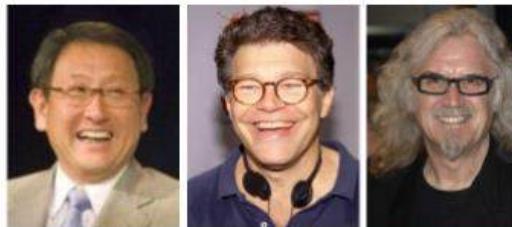


# Learning Difficulties (3)

- CelebA dataset, widely used for facial recognition
- Celebrities from Hollywood
- Problem?
- Little minority data
- Result: models underperform for minorities (too little training samples)

Sample Images

Eyeglasses



Wearing Hat



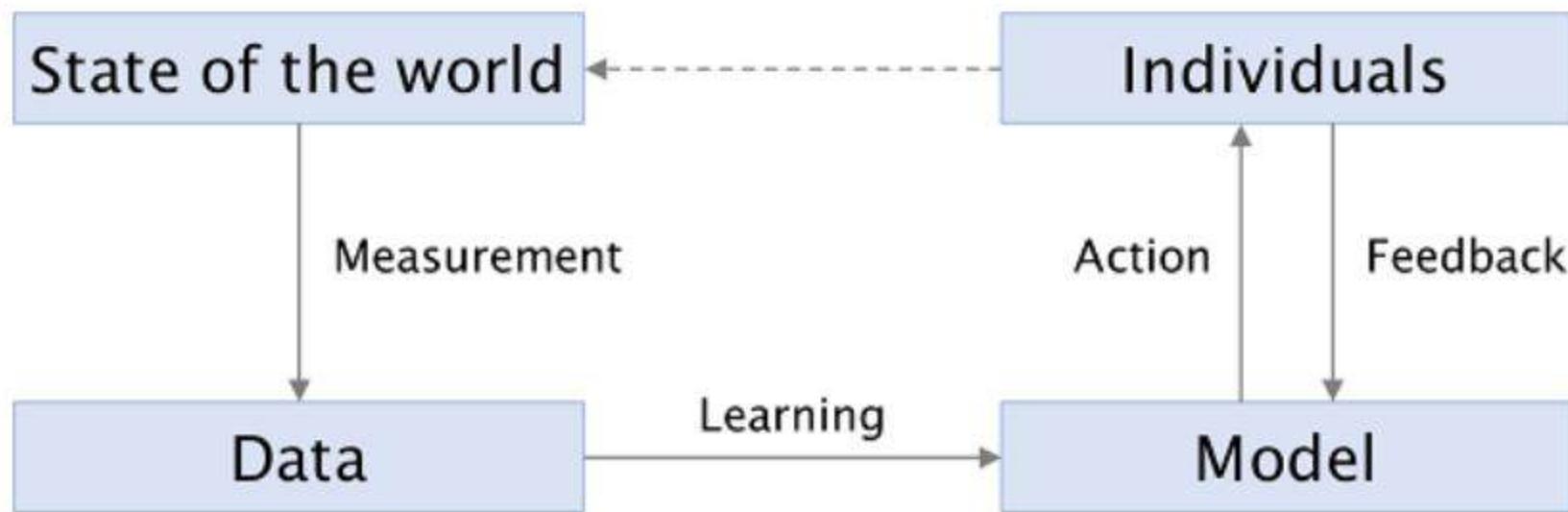
Bangs



Wavy Hair



# Machine Learning loop



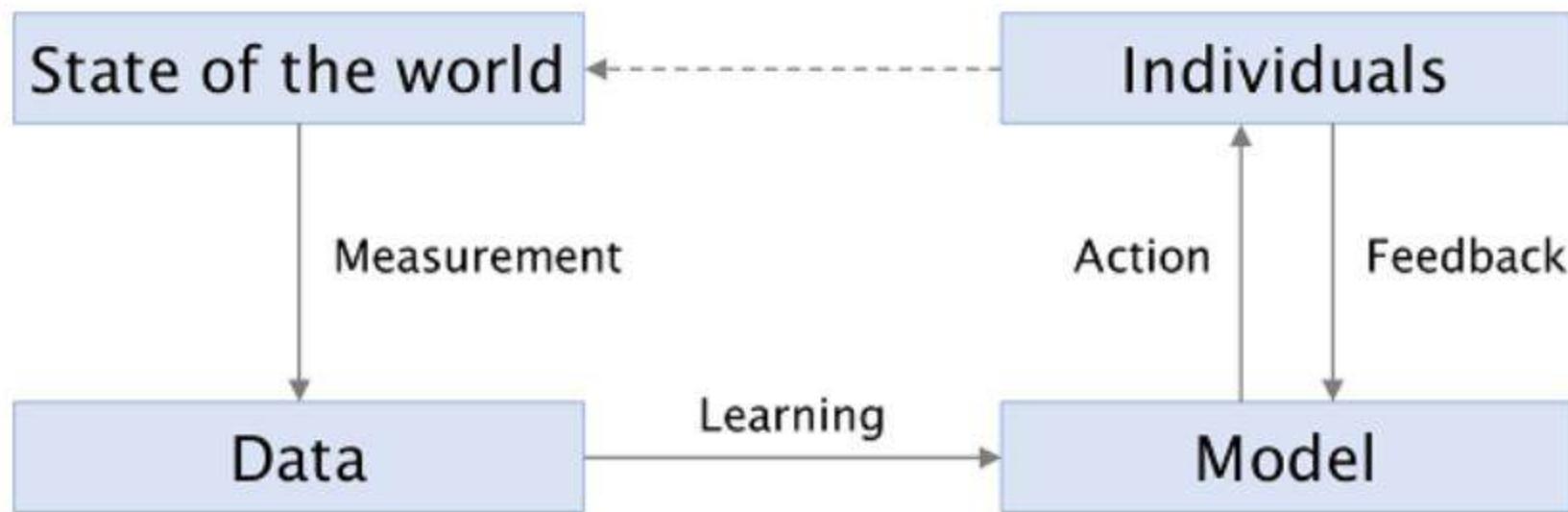
# Action Difficulties (1)

- Our model predicts who is suitable for a job
- Training data:
- 90% of population 1 is suitable for the job
- 50% of population 2 is suitable for the job
- The model will reproduce this pattern in the data
- Is that bad?

# Action Difficulties (2)

- Automated decisions
  - Only the outcomes are important, not the process.
  - True or False?
- 
- Ethical decisions: not only outcome important, but process
  - Why was a decision made?
  - Only if we can understand how the decision is made, we can understand if it was ethical or not
  - Needs to be explainable / white box

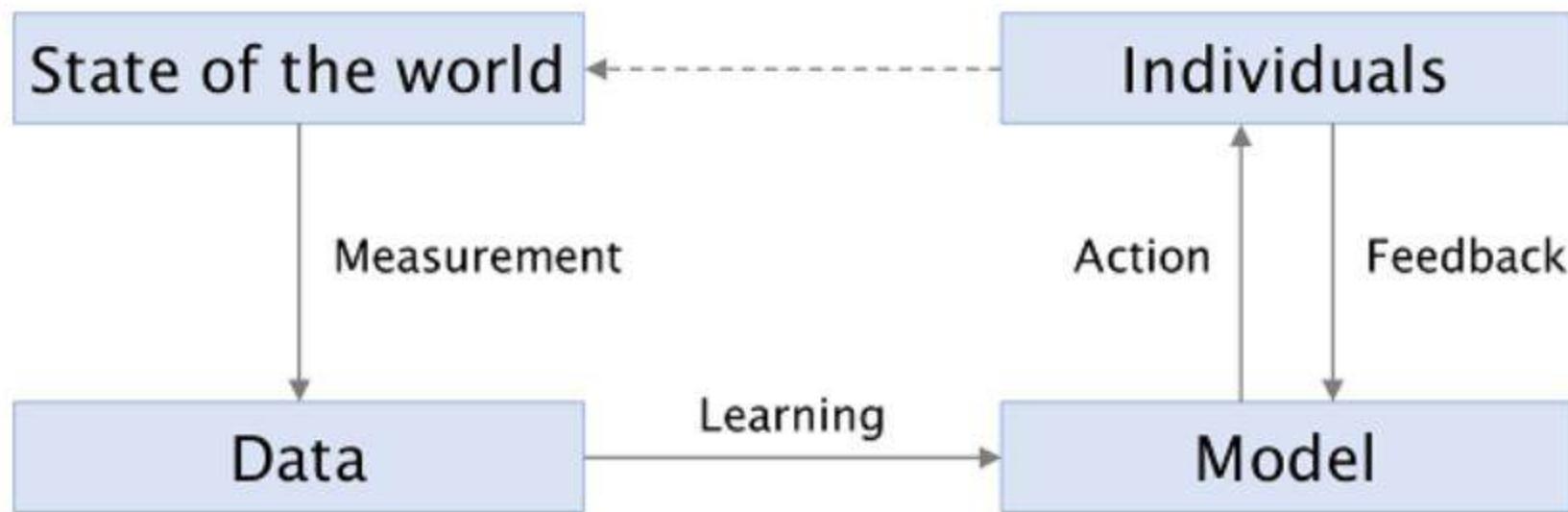
# Machine Learning loop



# Feedback

- Feedback: click on Ad, link search result
- Feedback to the machine learning system to improve model
- Googling a Black-sounding name: more ads for arrest records
- Because users click on ads that conform to stereotypes
- Is the feedback what we want? Clicked because best result, or just because on top of page?
- Feedback can reflect or amplify cultural prejudices

# Machine Learning loop



# Loops

- Self-fulfilling predictions
- Predictions that affect the training data
- Predictions that affect society at large

# Self-fulfilling predictions

- Predictive policing
- Machine learning model says where police should patrol
- Police go more often to areas that are predicted to be high risk
- Officers will make more arrests there
- Officers unconsciously may be more alert or may treat people differently in the area



# Predictions that affect the training data

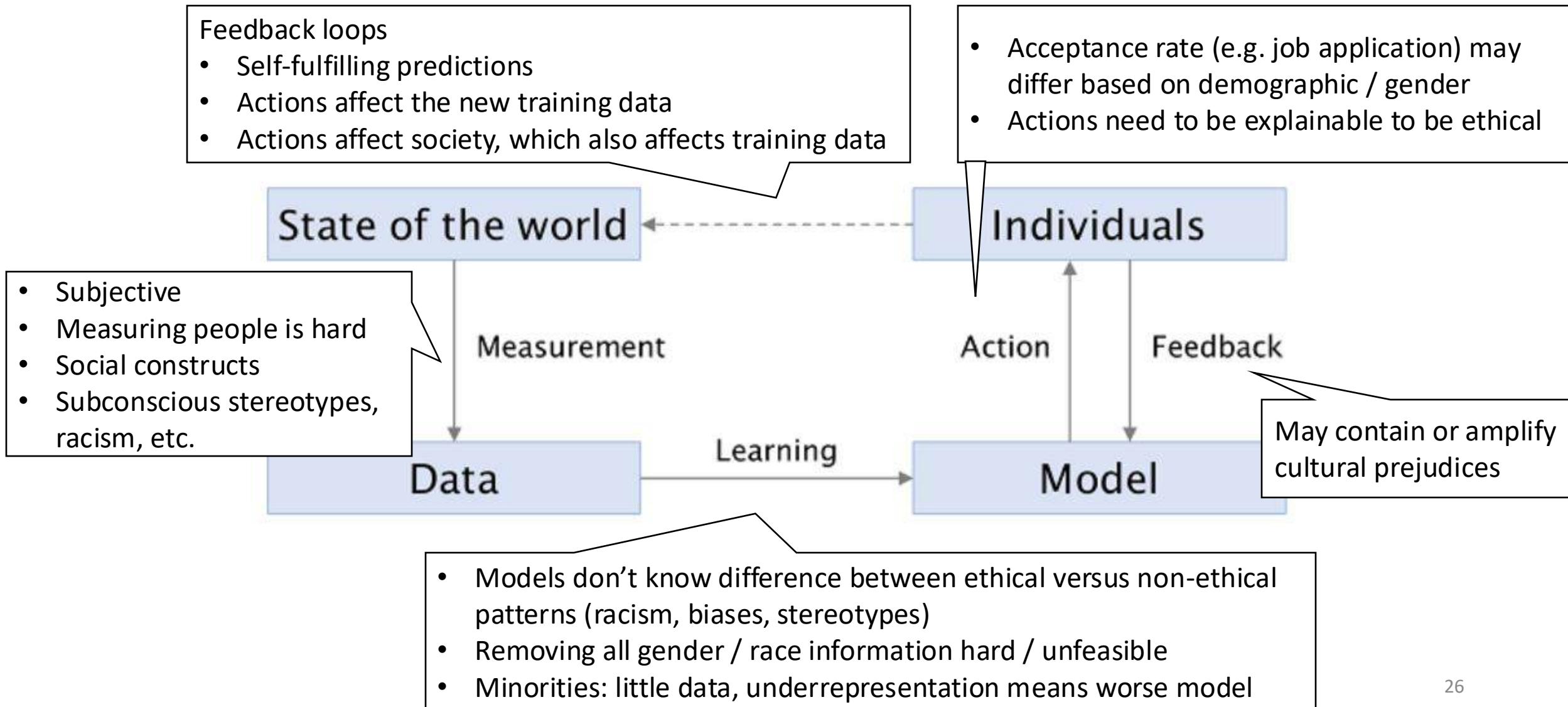
- Predictions seem validated by new data
- Feedback loop: this area will become more and more high risk
- Algorithm identify similar areas based on features and also consider them high risk
- Small mistakes in first model spiral out of control: patterns, biases, mistakes amplified
- PredPol study: Black people targeted twice as much for drug policing
- Even though both groups have equal rate of drug use



# Predictions that affect society at large

- Machine learning that amplify prejudice
- Makes prejudice even more persistent
- Affects poverty and crime on long time scale
- Almost all systems have an effect
- Search engines: result rankings.
- Why?
  - Over time, highly ranked items are read more, more important.
  - Highly ranked items influence society more.
  - Will influence future search results.

# Summary



# Other considerations

- A “fair” learning model might not always be the best solution
  - Maybe: try to improve workplace for minorities
- Should we automate and measure everything?
  - Errors with facial recognition technology, DNA for forensics, etc...

# Reasons to be optimistic

- Machine learning can be more accurate than experts
- Machine learning can be more transparent
- Forces us to articulate our objectives
- Debate and specify: what is fair? What are the trade-offs?
- More difficult to hide poorly specified or harmful true intentions

# Part 2: Measuring fairness & adapting models

- Why sensitive attributes shouldn't removed
- Measuring fairness using statistics:
  - Criteria 1: Independence
  - Criteria 2: Separation
- Limitations of statistics

# Some notation

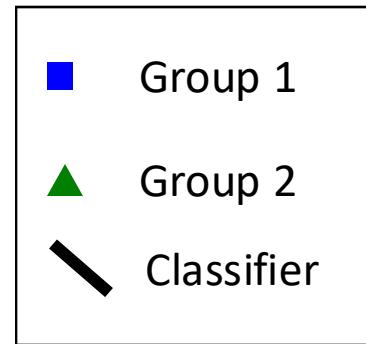
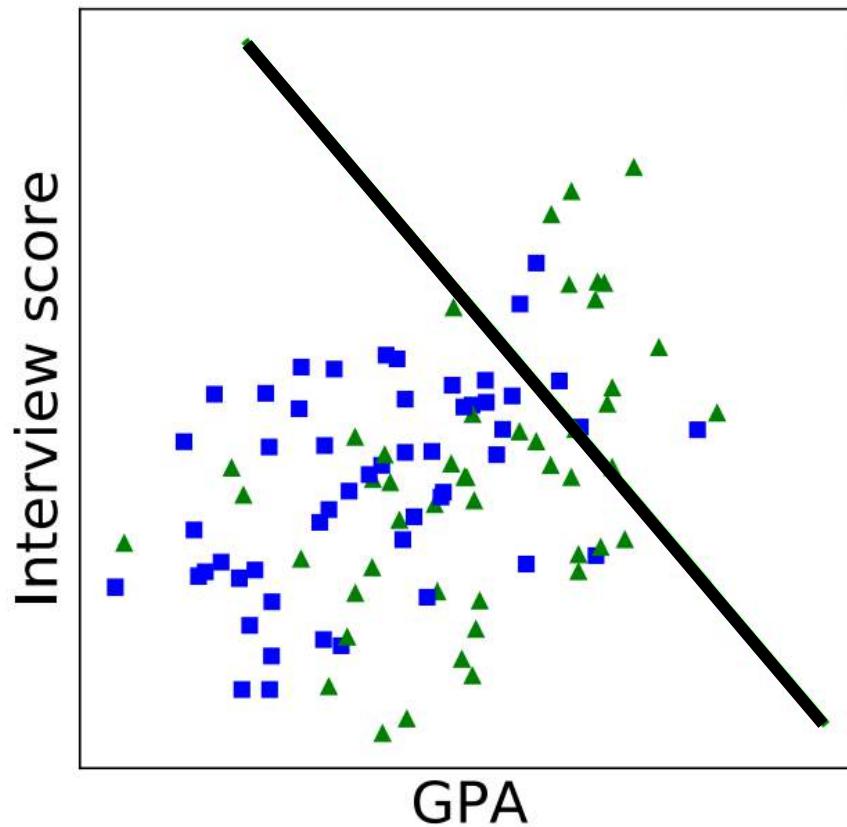
$A$  sensitive attribute. Example:  $A = a$  (other genders),  $A = b$  (male)

$X$  other features. Example: word occurrences in resume.

$Y$  classification target.  $Y = 0$  (job rejection),  $Y = 1$  (accept)

$R$  classifier output.  $R = 0$  (predict: reject),  $R = 1$  (predict: accept)

# Example

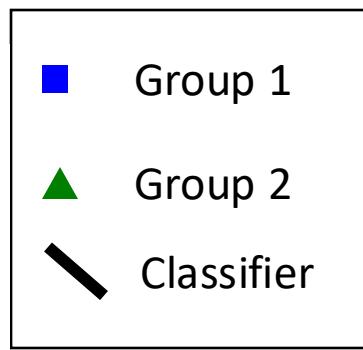
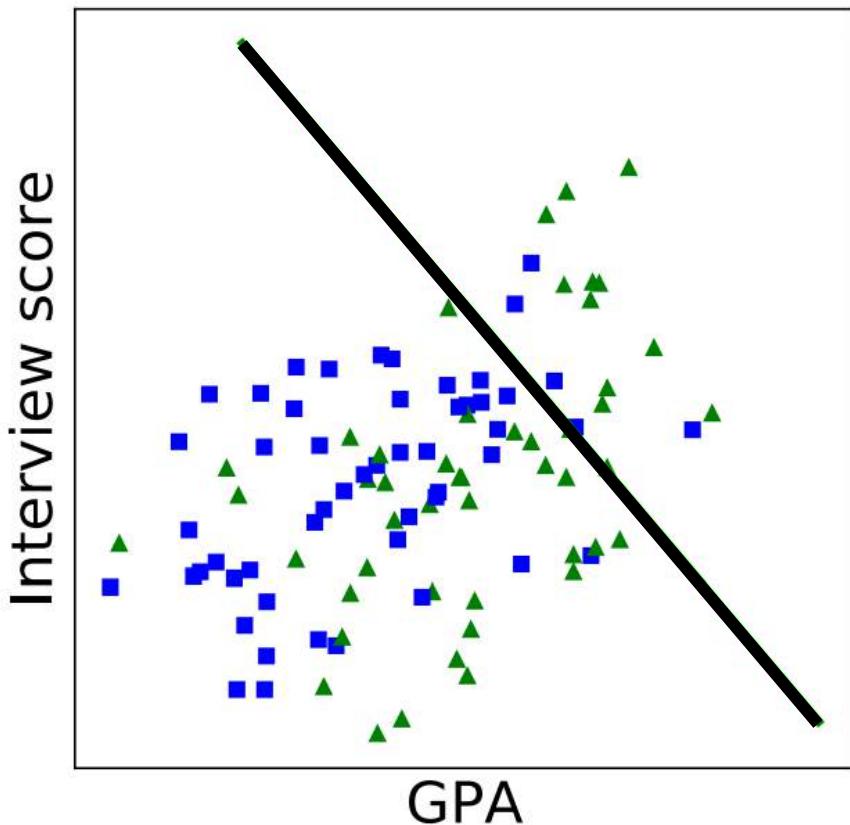


Fair?

Group 1: 5/100 hired = 5%  
Group 2: 11/60 hired = 18%

- Model doesn't use the sensitive attribute to make predictions, only GPA and interview score.
- Is this fair?

# Example

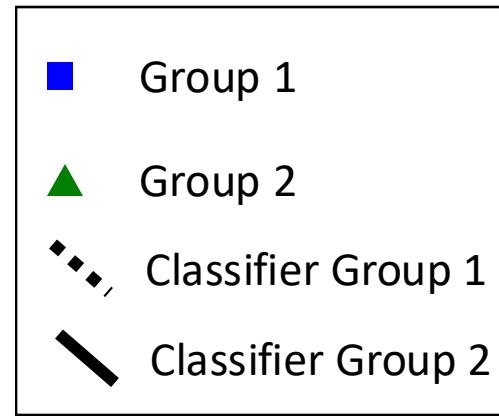
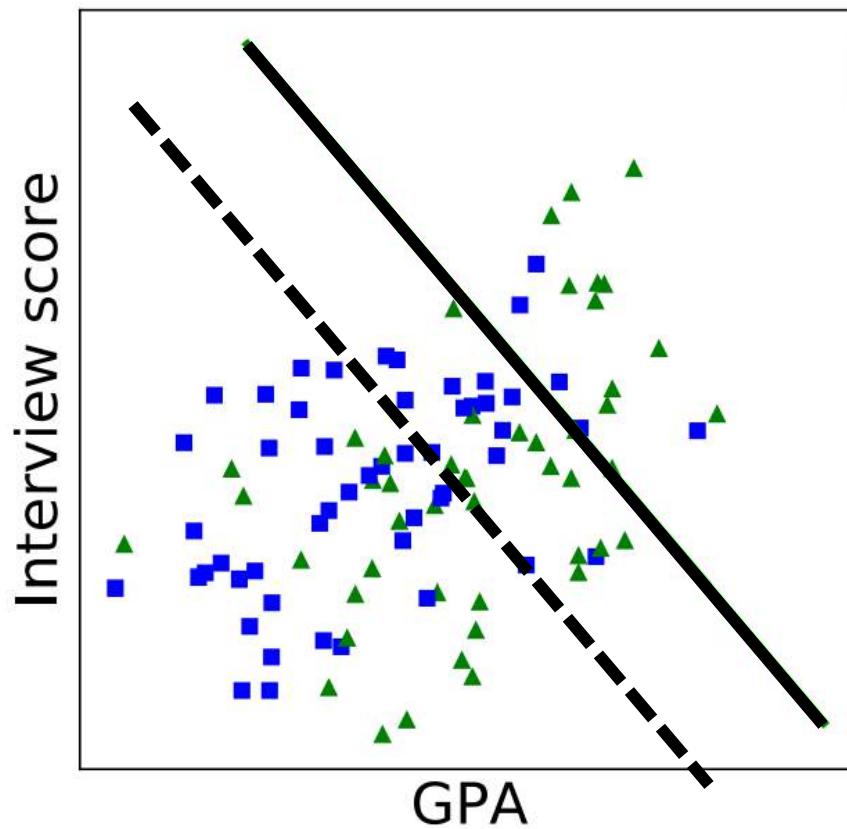


Fair?

Group 1: 5/100 hired = 5%  
Group 2: 11/60 hired = 18%

- Why does the model accept more from one group than the other?
- Conclusion: removing or not using the sensitive attribute is not enough.

# Example



Fair?

Group 1: **18/100** hired = **18%**  
Group 2: 11/60 hired = 18%

- Can use 2 models: one for group 1, one for group 2.
- Now fair?

# Criteria 1: Independence

$$P(R = 1|A = a) = P(R = 1|A = b)$$

Criteria 1: Independence.

The acceptance rate for each group should be the same.

The classification rule does not have to be the same for each group (!)

Independence not satisfied

Group 1: 5/100 hired = 5%  
Group 2: 11/60 hired = 18%

Independence satisfied

Group 1: 18/100 hired = 18%  
Group 2: 11/60 hired = 18%

# Criteria 1: Independence

$$P(R = 1|A = a) = P(R = 1|A = b)$$

Criteria 1: Independence.

The acceptance rate for each group should be the same.

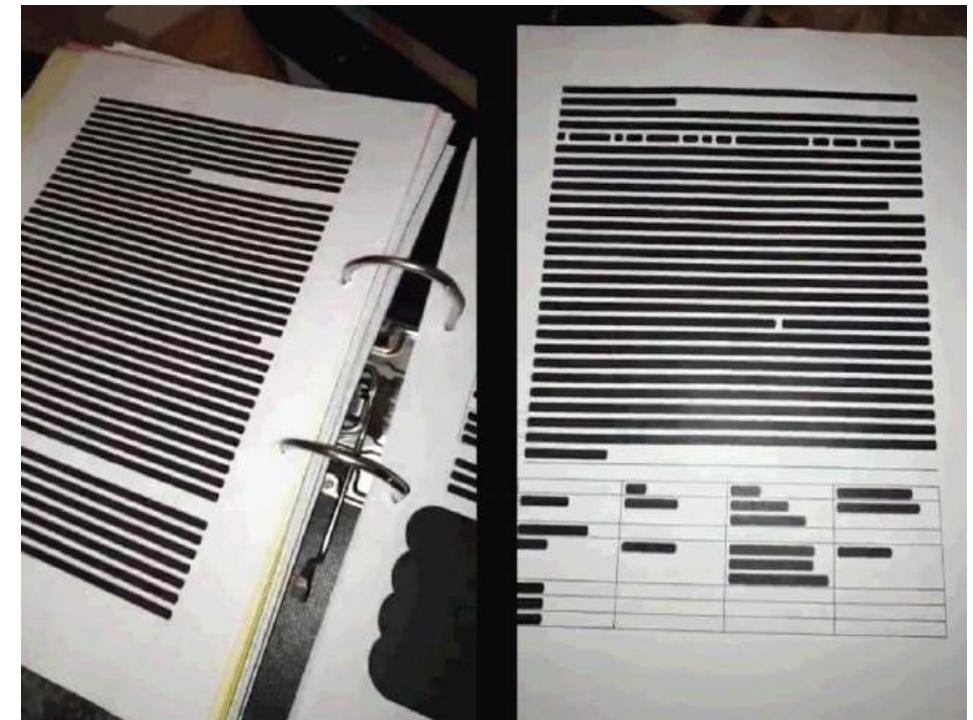
The classification rule does not have to be the same for each group (!)

Note: need sensitive attribute to compute.

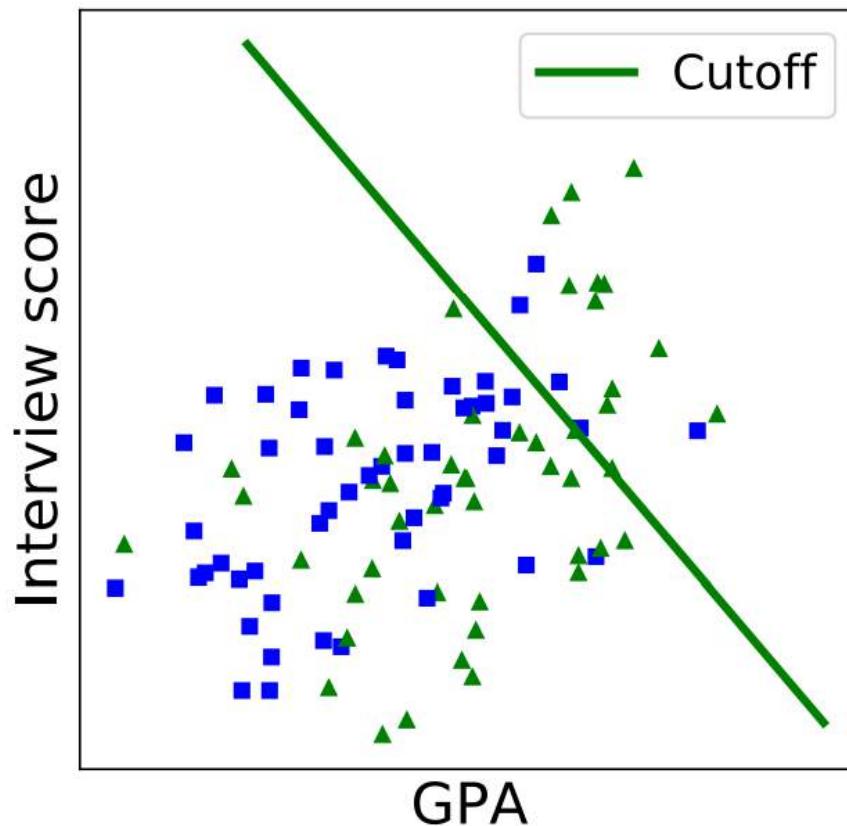
Never remove the sensitive attribute from your dataset!

# Independence Approach 2: Pre-process

- Another way to satisfy independence is to pre-process the data
- The data should be pre-processed in such a way, that it becomes impossible to tell from which group a candidate comes
- Problems: 1) How?
- 2) May loose too much information



# Independence Approach 2: Pre-process



- Which feature would you remove?
- Would you use a model that ignores the interview score?

# Pre-processing problems

- For bigger datasets even more problematic. Why?
- Sensitive attribute is often intertwined with all data. May need to remove a lot.
- Model performance might suffer a lot.



# Pre-processing problems

- Model input: DNA. Output: predict the income.
- What will this model learn?
- A: DNA -> race -> income.
- What do we remove from the DNA?
- For visual data: what should be removed to obscure gender?

# Criteria 1: Independence

$$P(R = 1|A = a) = P(R = 1|A = b)$$

Criteria 1: Independence.

The acceptance rate for each group should be the same.

Approaches:

- Different model per group
- Pre-process data so that we cannot distinguish the 2 groups

# Problem with Independence

Group	Loans payed back	Loans defaulted (fail to pay back)
A=a	90	10
A=b	10	90

The bank doesn't want to use a classification model that satisfies independence here. Why not?

A: The acceptance rate in both groups need to be the same.

Need to accept a lot from group b, or reject a lot from group a.

Too much lost profit.

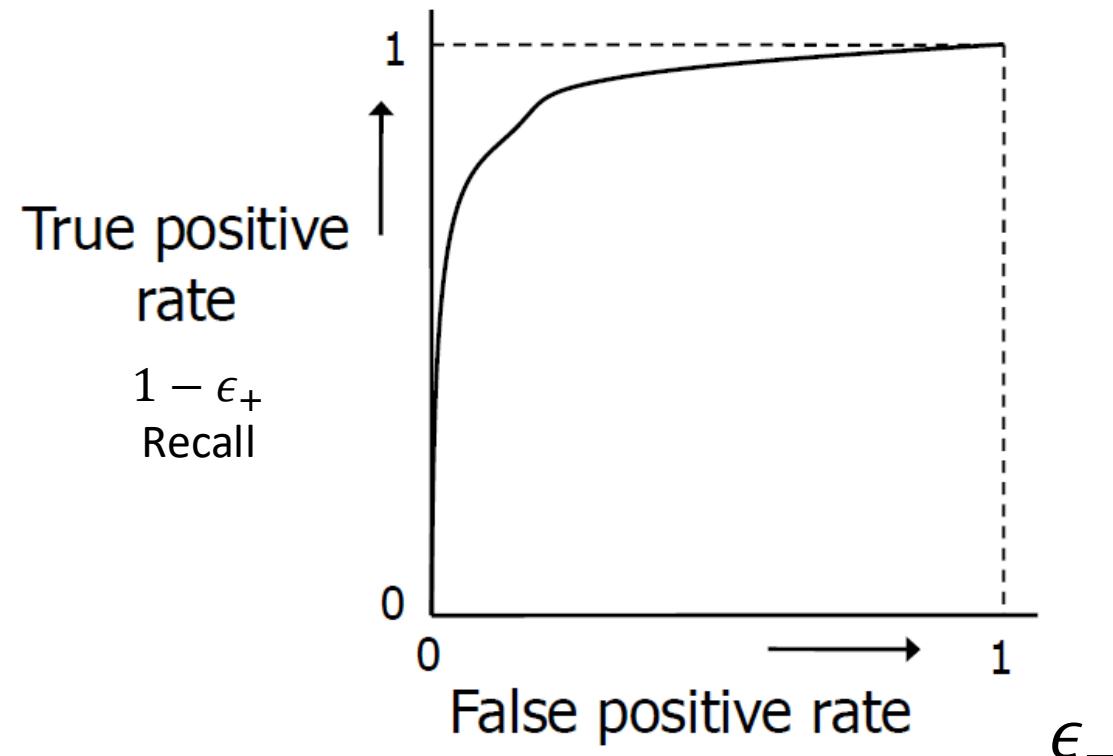
# Criteria 2: Separation

- Instead: fraction of mistakes for each group should be the same.
- $\epsilon_+$  and  $\epsilon_-$  should be the same for each group
- Fraction of positives can now be different per group

		PREDICTED		Error rate per class
		Positive	Negative	
ACTUAL	Positive	TP / True Positive	FN / False Negative	$\epsilon_+ = FN/(TP + FN)$
	Negative	FP / False Positive	TN / True Negative	$\epsilon_- = FP/(FP + TN)$

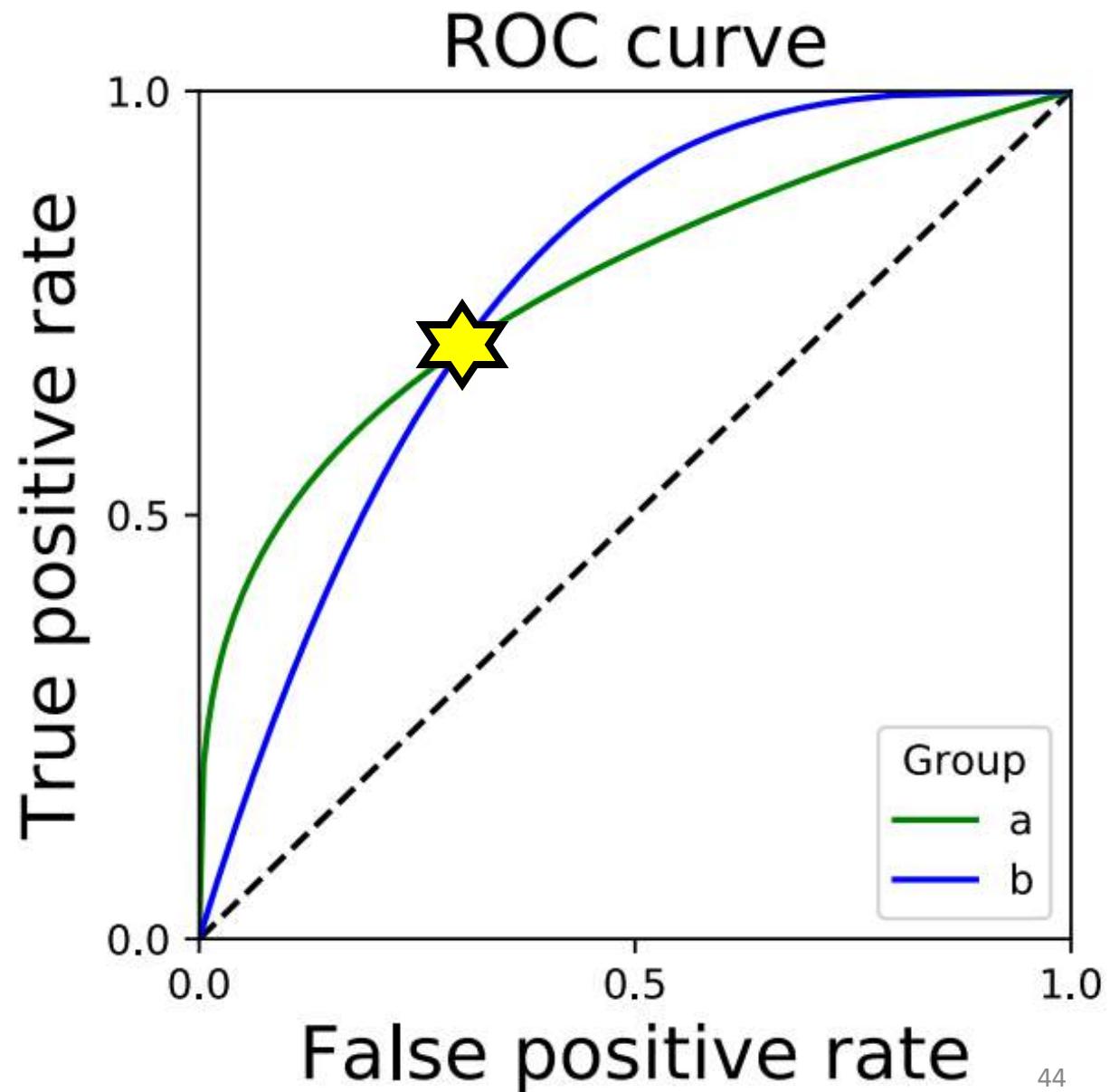
# ROC curve

		PREDICTED		
		Positive	Negative	Error rate per class
ACTUAL	Positive	TP / True Positive	FN / False Negative	$\epsilon_+ = FN/(TP + FN)$
	Negative	FP / False Positive	TN / True Negative	$\epsilon_- = FP/(FP + TN)$

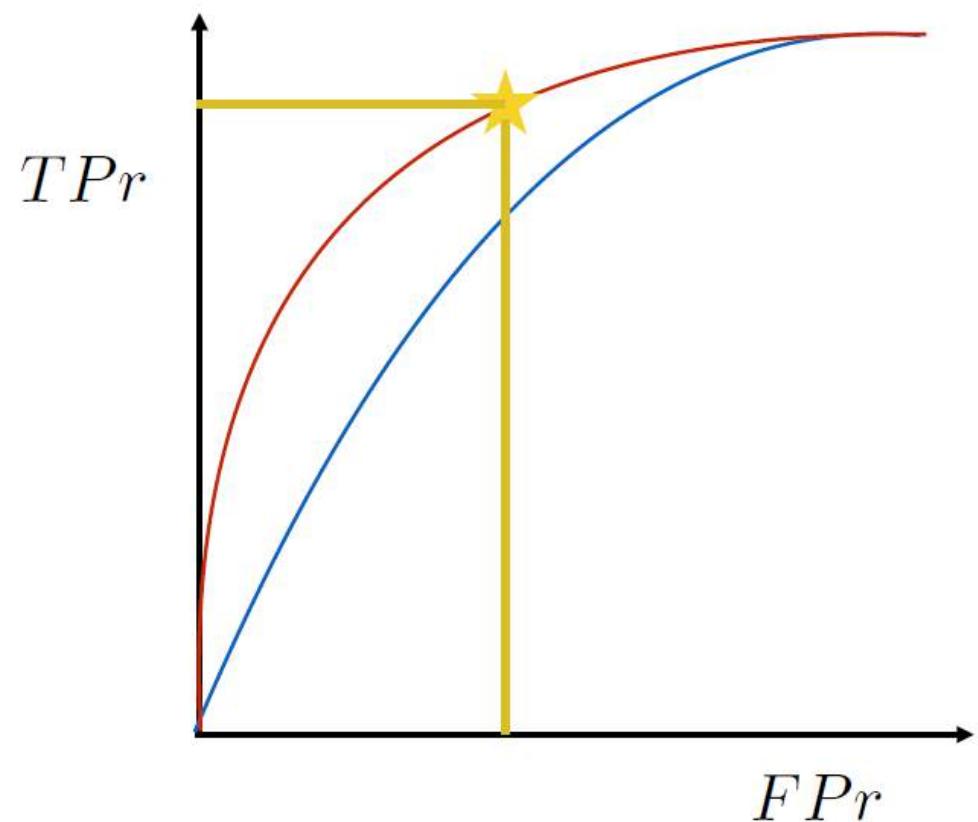


# Separation

For which point is  
Separation satisfied here?

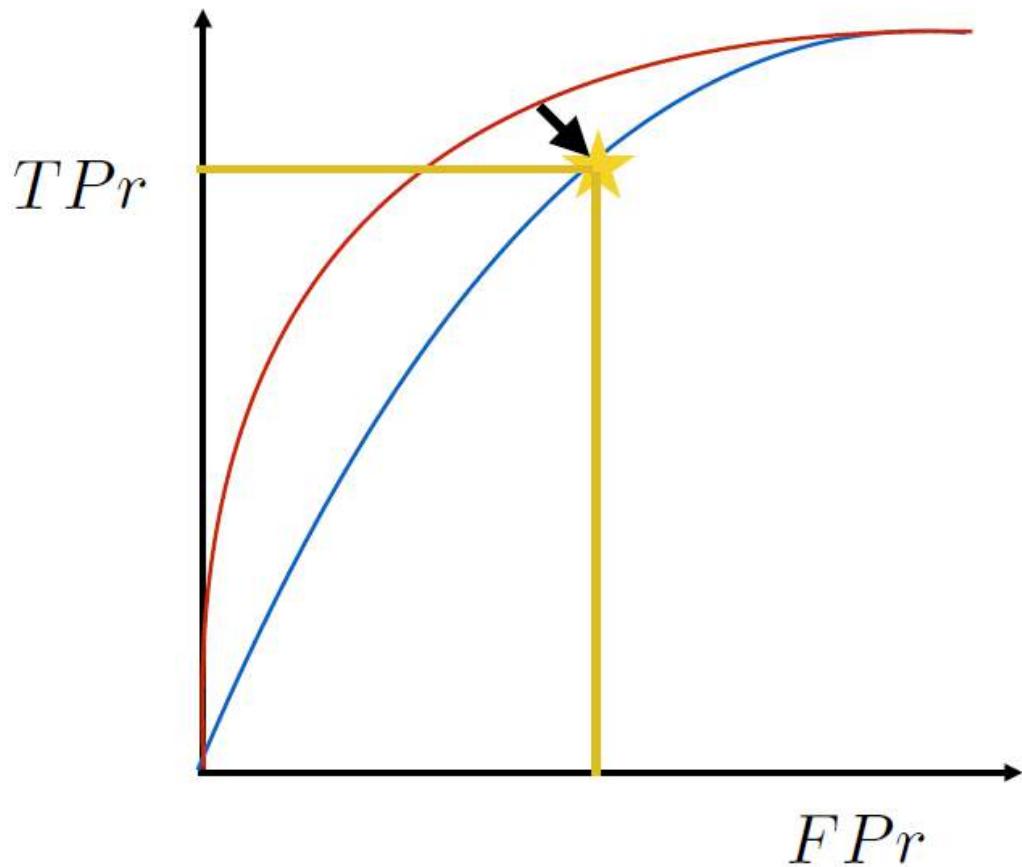


# Separation: how?



What to do in this case?

# Separation: how?

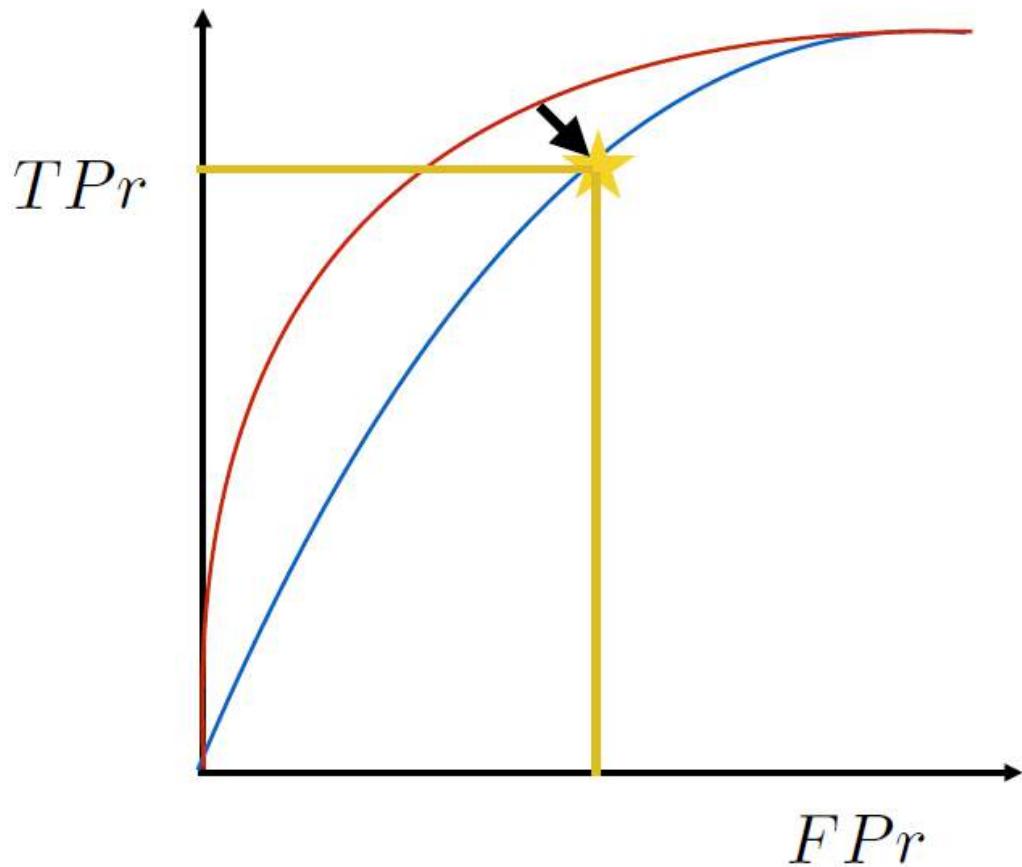


Add some noise to the output of the model if we get a sample from the red group.

The noise will make the classifier dumber, lowering the TPr.

Choose just the right amount of noise so that the star lands on a point on the blue curve.

# Separation: how?



Final rule:

If from red group:

Prediction = Noise + model output (dumber).

If from blue group:

Prediction = Model output. (no noise needed)

Note that for this final rule we again need to use the sensitive attribute!

# Limitations of Statistics

- Criteria only look at the statistics. Problem?
- Example: we hire from group A using interview score, we hire from group B randomly.
- Example: we hire from group A using the highest interview score, we hire from group B with the lowest interview score.
- Conclusion: for fairness we need to know how a decision was made
- Statistics: can only judge fairness on group level

# A final note

- Separation and Independence cannot always be both satisfied
- So which statistic is *really* fair?
- Still active discussion.
- More than 20 fairness definitions have been proposed recently...

Name	Closest relative	Note	Reference
Statistical parity	Independence	Equivalent	Dwork et al. (2011)
Group fairness	Independence	Equivalent	
Demographic parity	Independence	Equivalent	
Conditional statistical parity	Independence	Relaxation	Corbett-Davies et al. (2017)
Darlington criterion (4)	Independence	Equivalent	Darlington (1971)
Equal opportunity	Separation	Relaxation	Hardt, Price, Srebro (2016)
Equalized odds	Separation	Equivalent	Hardt, Price, Srebro (2016)
Conditional procedure accuracy	Separation	Equivalent	Berk et al. (2017)
Avoiding disparate mistreatment	Separation	Equivalent	Zafar et al. (2017)
Balance for the negative class	Separation	Relaxation	Kleinberg, Mullainathan, Raghavan (2016)
Balance for the positive class	Separation	Relaxation	Kleinberg, Mullainathan, Raghavan (2016)
Predictive equality	Separation	Relaxation	Chouldechova (2016)
Equalized correlations	Separation	Relaxation	Woodworth (2017)
Darlington criterion (3)	Separation	Relaxation	Darlington (1971)
Cleary model	Sufficiency	Equivalent	Cleary (1966)
Conditional use accuracy	Sufficiency	Equivalent	Berk et al. (2017)
Predictive parity	Sufficiency	Relaxation	Chouldechova (2016)
Calibration within groups	Sufficiency	Equivalent	Chouldechova (2016)
Darlington criterion (1), (2)	Sufficiency	Relaxation	Darlington (1971)

# Conclusions

- Why it's not enough to ignore sensitive attributes
- Criteria 1: Independence
  - How to satisfy it: (1) different model per group (2) pre-processing
  - Problems with pre-processing, problem with independence criterion
- Criteria 2: Separation
  - Can have different acceptance rate per group
  - How to satisfy it: making a smart model dumber, ROC curve
- We need to keep sensitive attributes
  - To make models more fair
  - To measure fairness

# NON-LINEAR CLASSIFIERS

*Decision Trees*



# *Recap: Ethics & Discriminative Linear Models*

## ETHICS

- Ethical considerations such as bias & discrimination, are important issues to consider both in the data that is used and during the construction of a (ML) system.

## LINEAR MODELS

- Linear classifiers are linear hyperplanes in feature space that partition the space in (two) classes
- Examples: least squares classifier, logistic regression, support vector classifier
- We used 3 choices in the empirical risk minimisation framework to define and study these classifiers

# *Learning Goals & Reading*

## LEARNING GOALS

After practicing with the concepts of this week you are able to

- **Explain when and why non-linear classifiers are needed**
- Explain the basic concepts of two non-linear classifiers:  
multi-layer perceptrons and **decision trees**
- Explain the **underlying algorithm of decision trees** and  
(multi-layer) perceptrons and how they are trained.
- Implement a decision tree
- Explain why and how one can **combine multiple classifiers**
- **Contrast a decision tree and a random forest**

## LITERATURE

Please study the following material from Chris Bishop's "Pattern Recognition and Machine Learning":

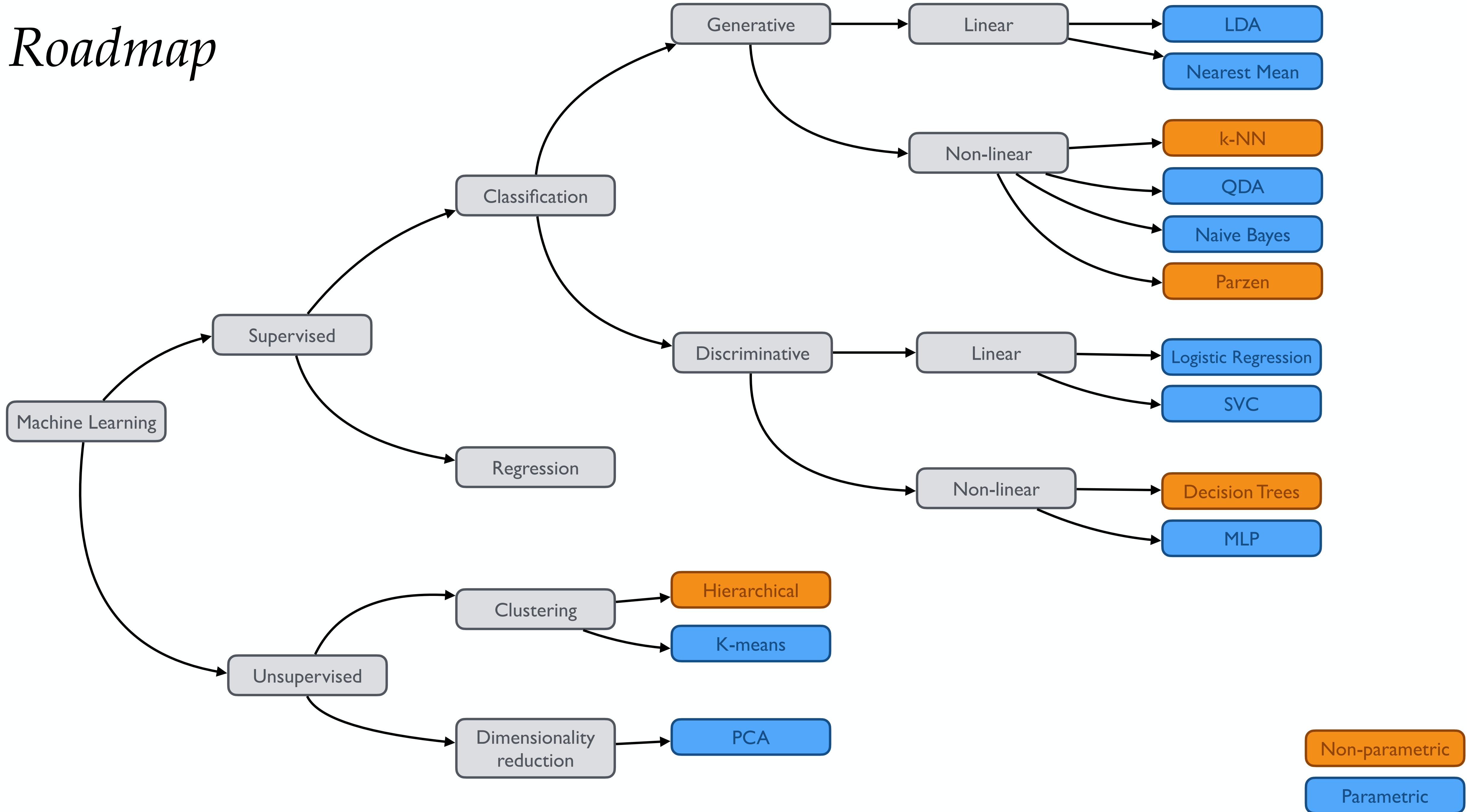
### **Lecture 6.1**

- Section 14.2
- Section 14.4

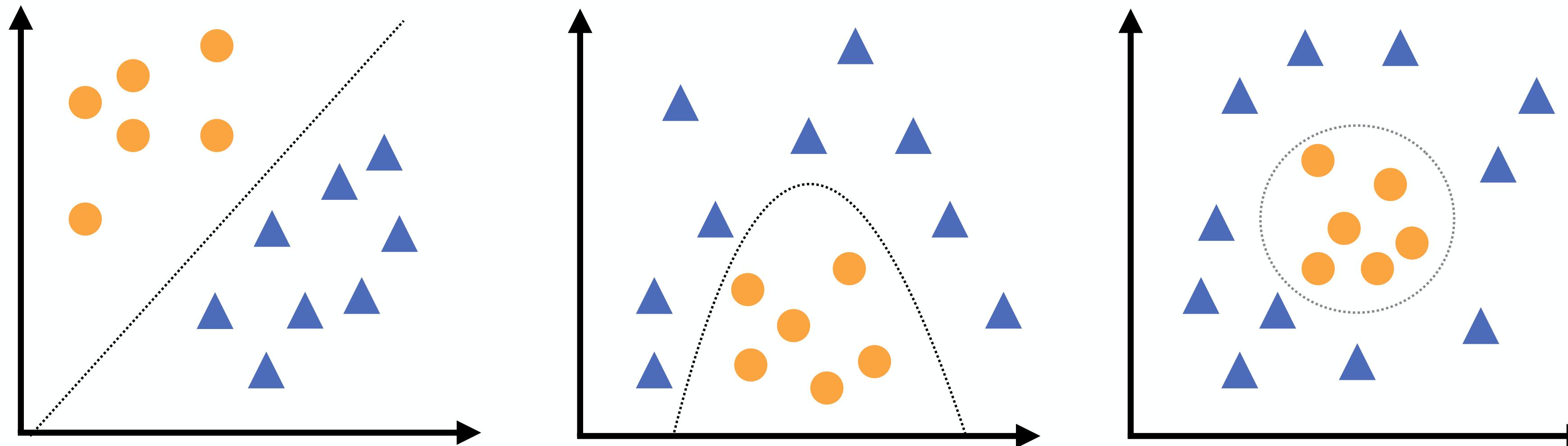
### **Lecture 6.2**

- Section 4.1.7
- Section 5.1
- Section 5.2 up to and including 5.2.1
- Section 5.3 up to and including 5.3.1

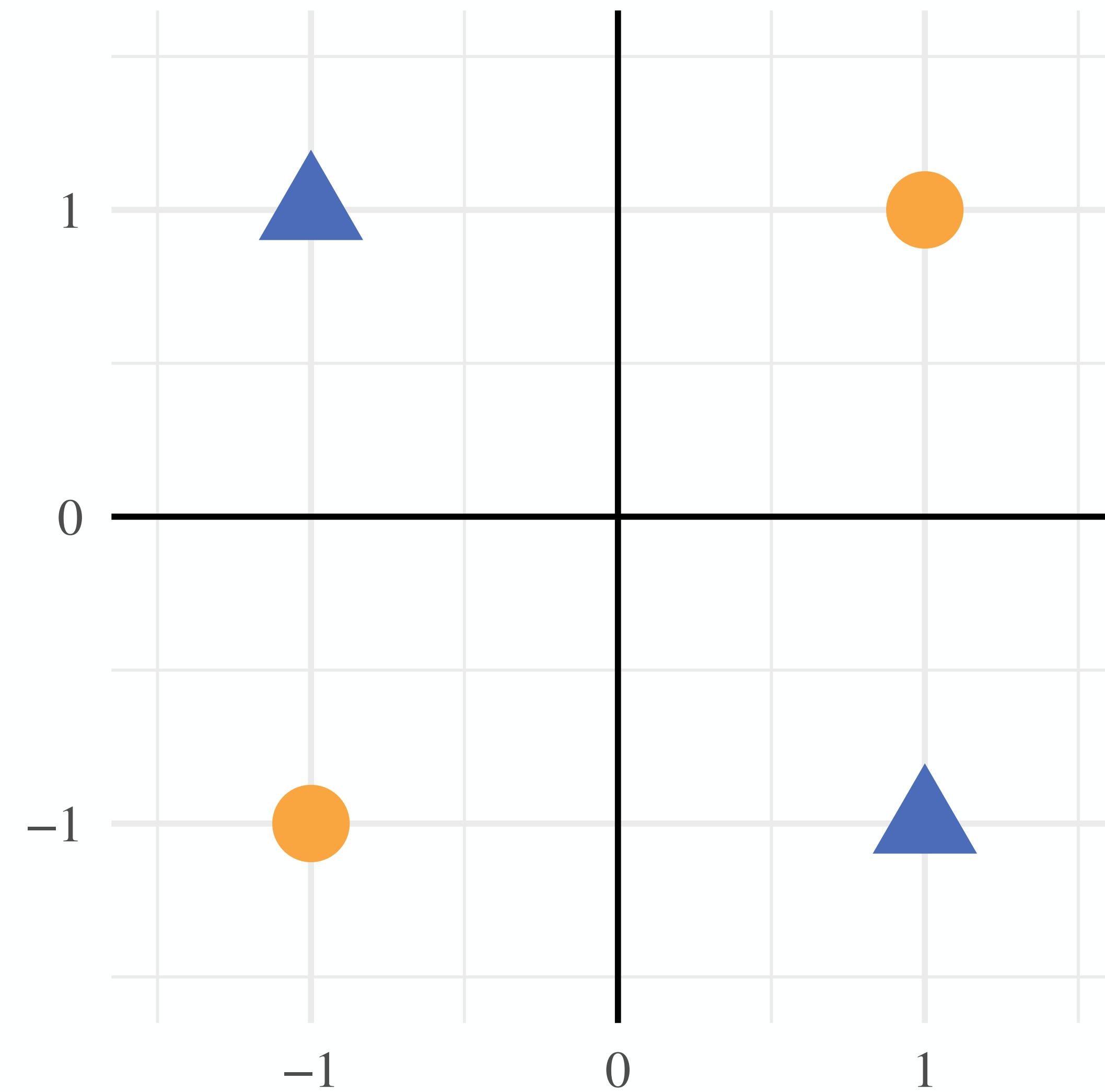
# Roadmap



# *Why non-linear classifiers?*



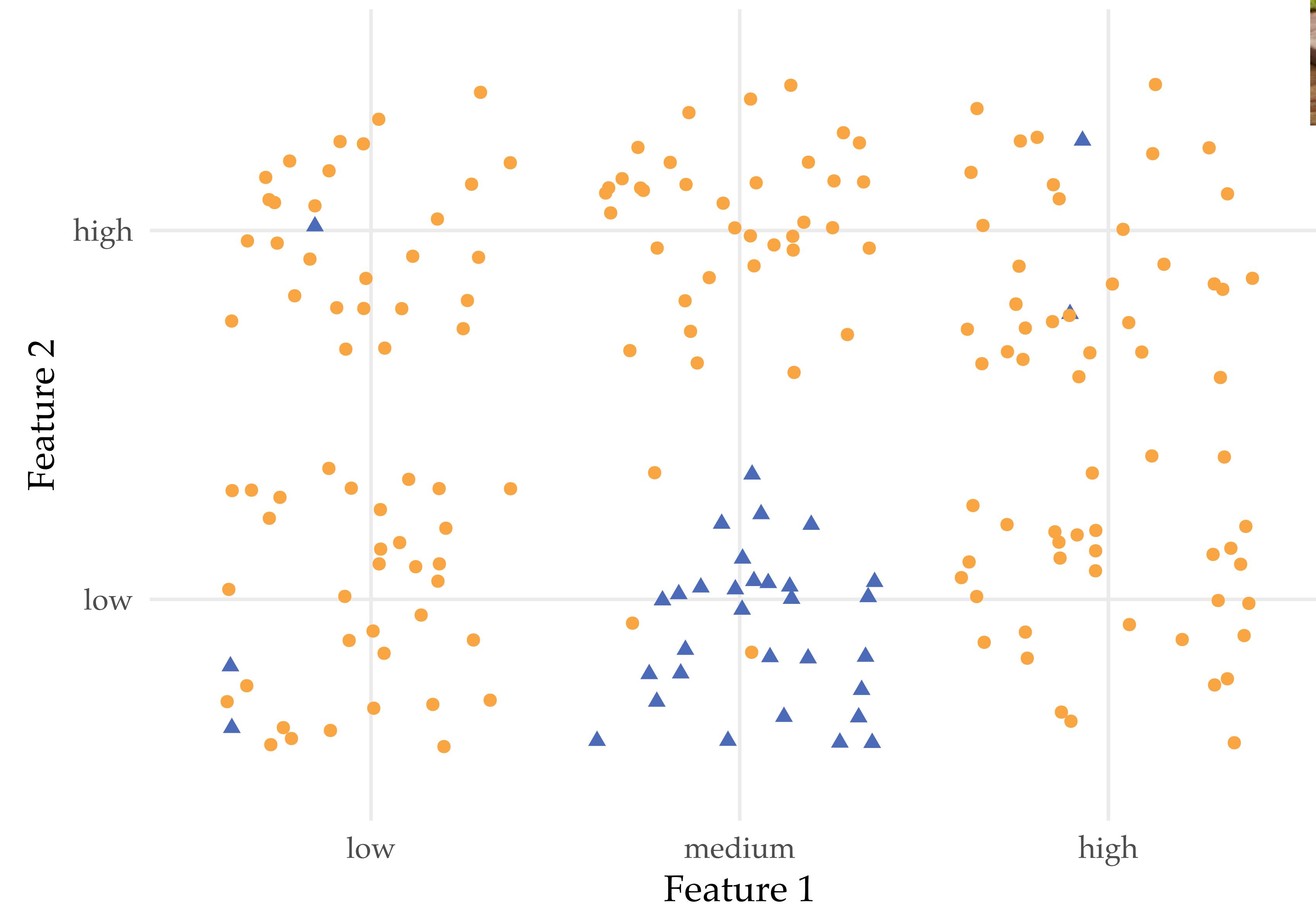
# *Non-Linear Classification Problem*



# *Today*

- Next time: multilayer perceptrons (parametric)
- Now: Decision Trees (non-parametric)
  1. What functions are described by decision trees?
  2. How do we make predictions using a given tree?
  3. How do we learn a tree from data?
  4. What are the advantages & disadvantages of this method?

# *Non-Linear Classification Problem*



# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

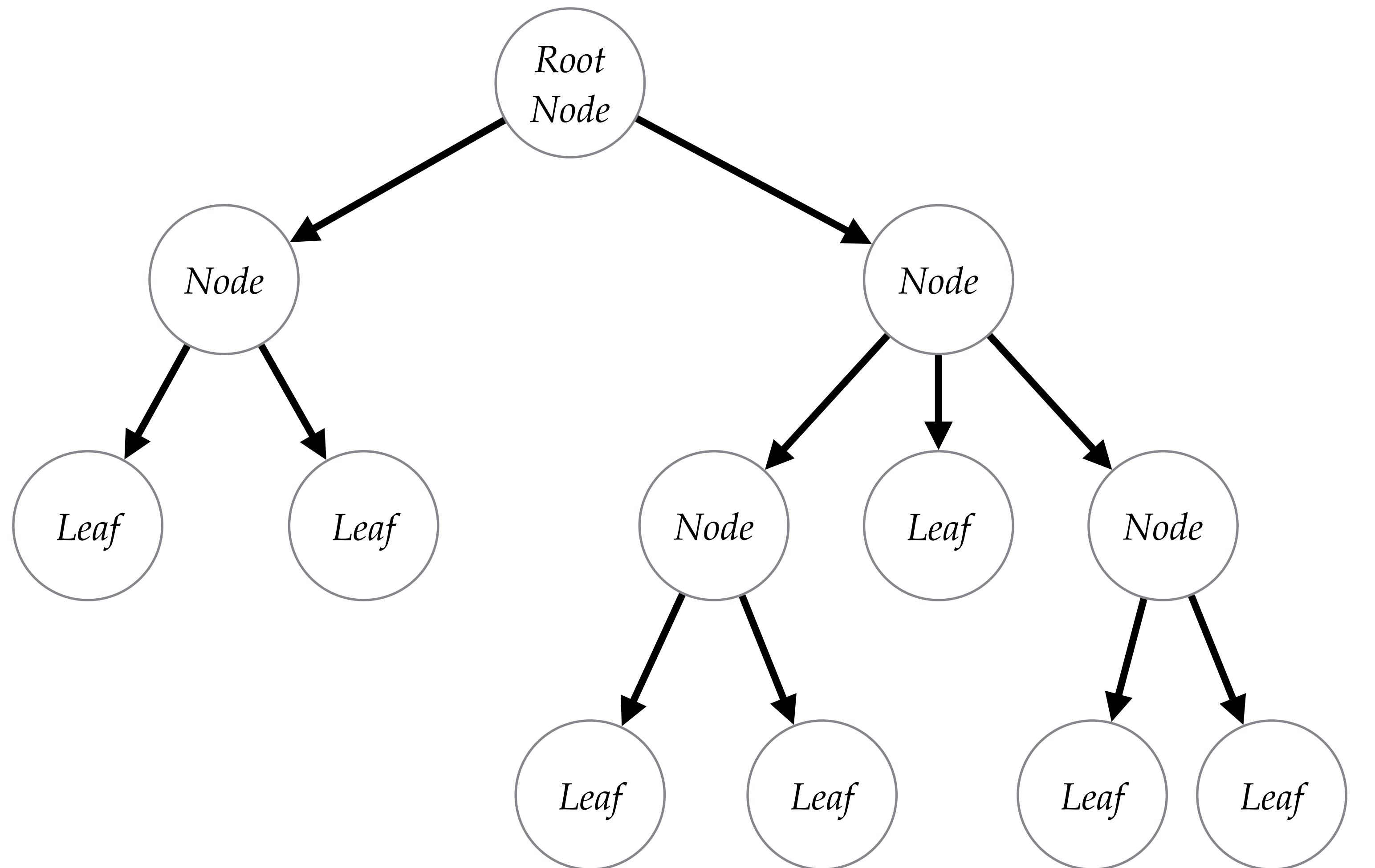
1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

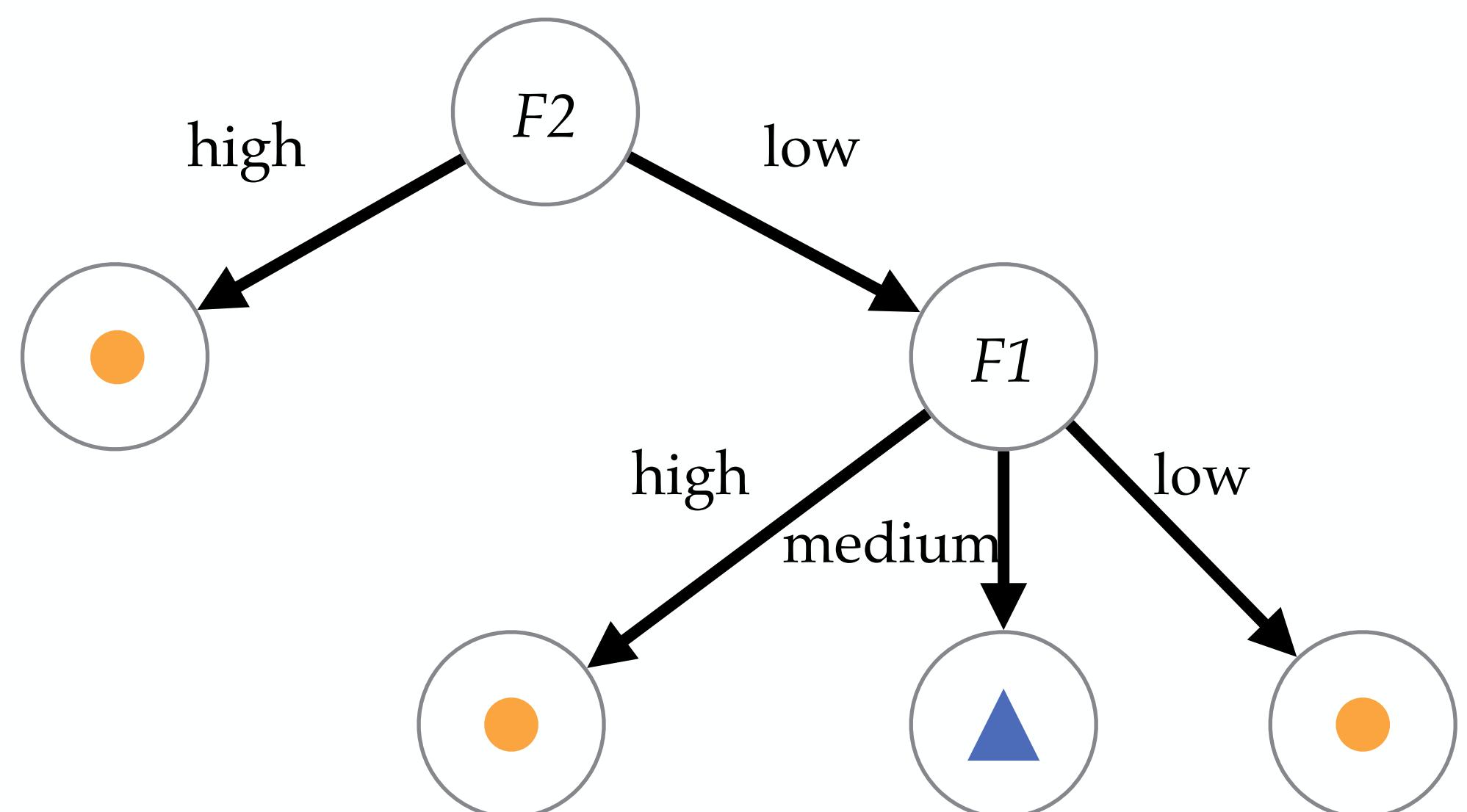
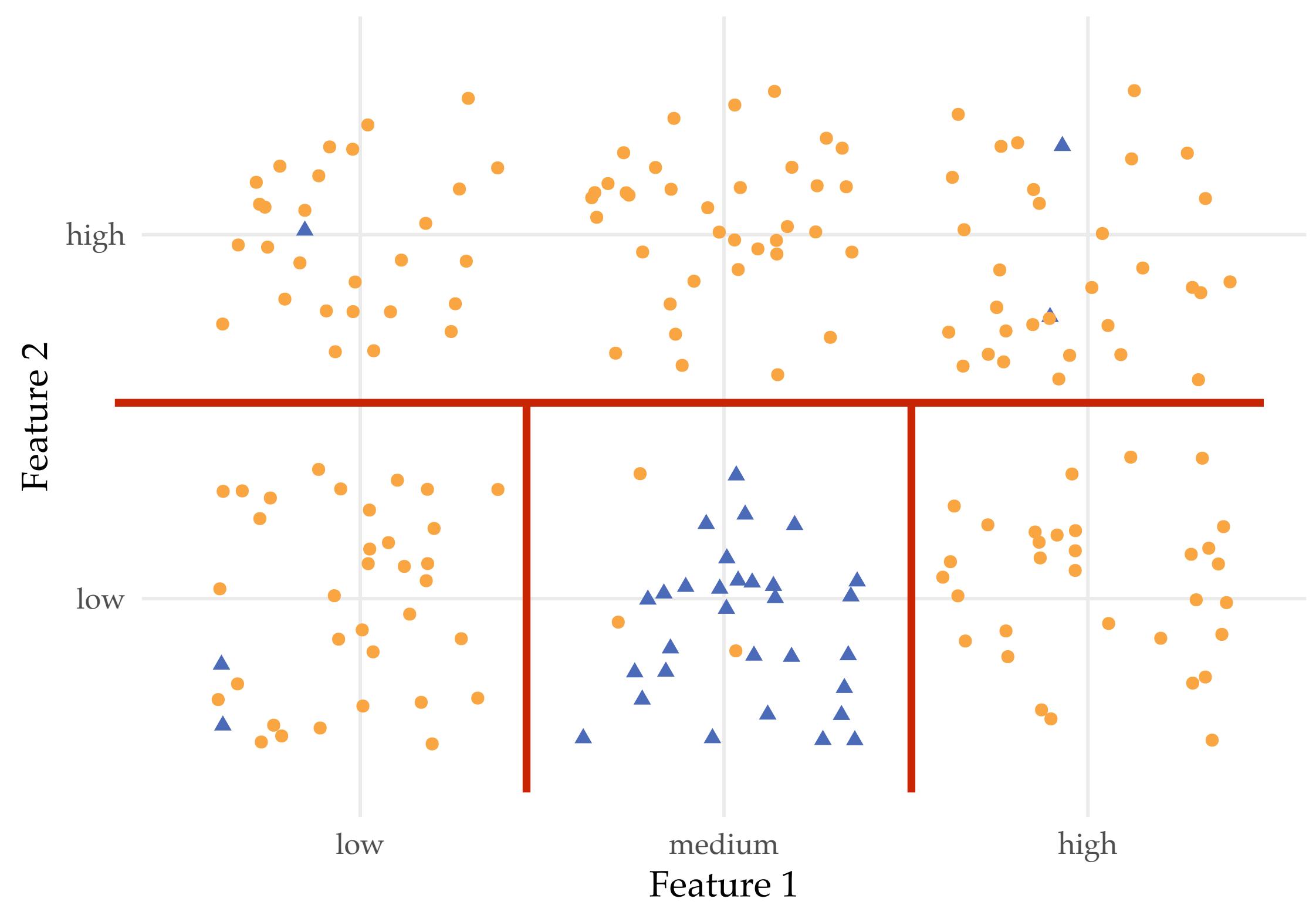
4. Find the function that minimises this cost

# Terminology

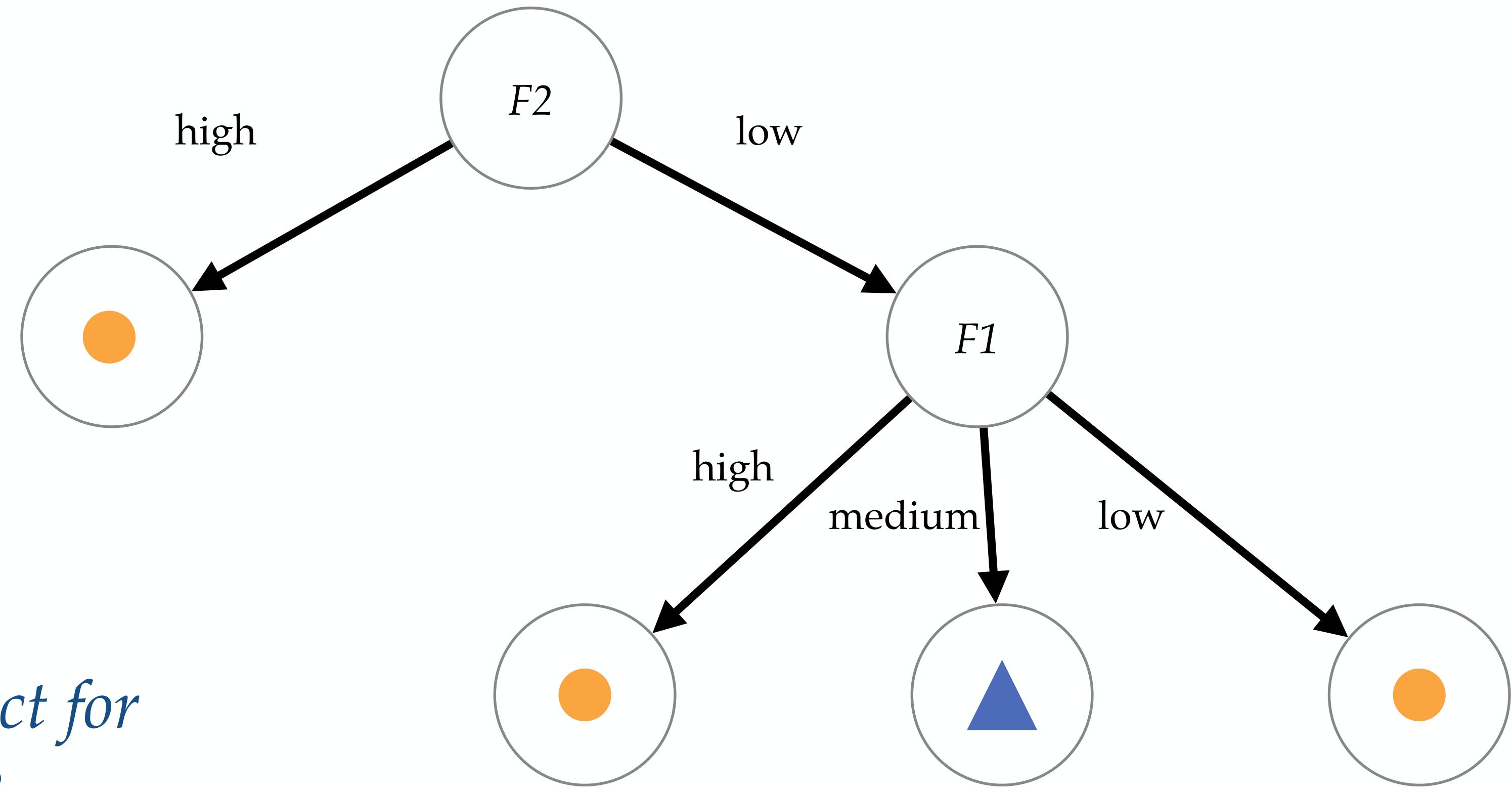


# *Function class*

- Split the feature space using one feature at a time, recursively
- The splitting
  - Forms a tree structure
  - Partitions the space in “rectangles”
  - In each rectangle/leaf, we assign a value



# Decision Tree Example



*What does the model predict for  
the following objects?*

Feature 1	Feature 2	Feature 3	Prediction
low	high	medium	?
medium	low	high	?

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

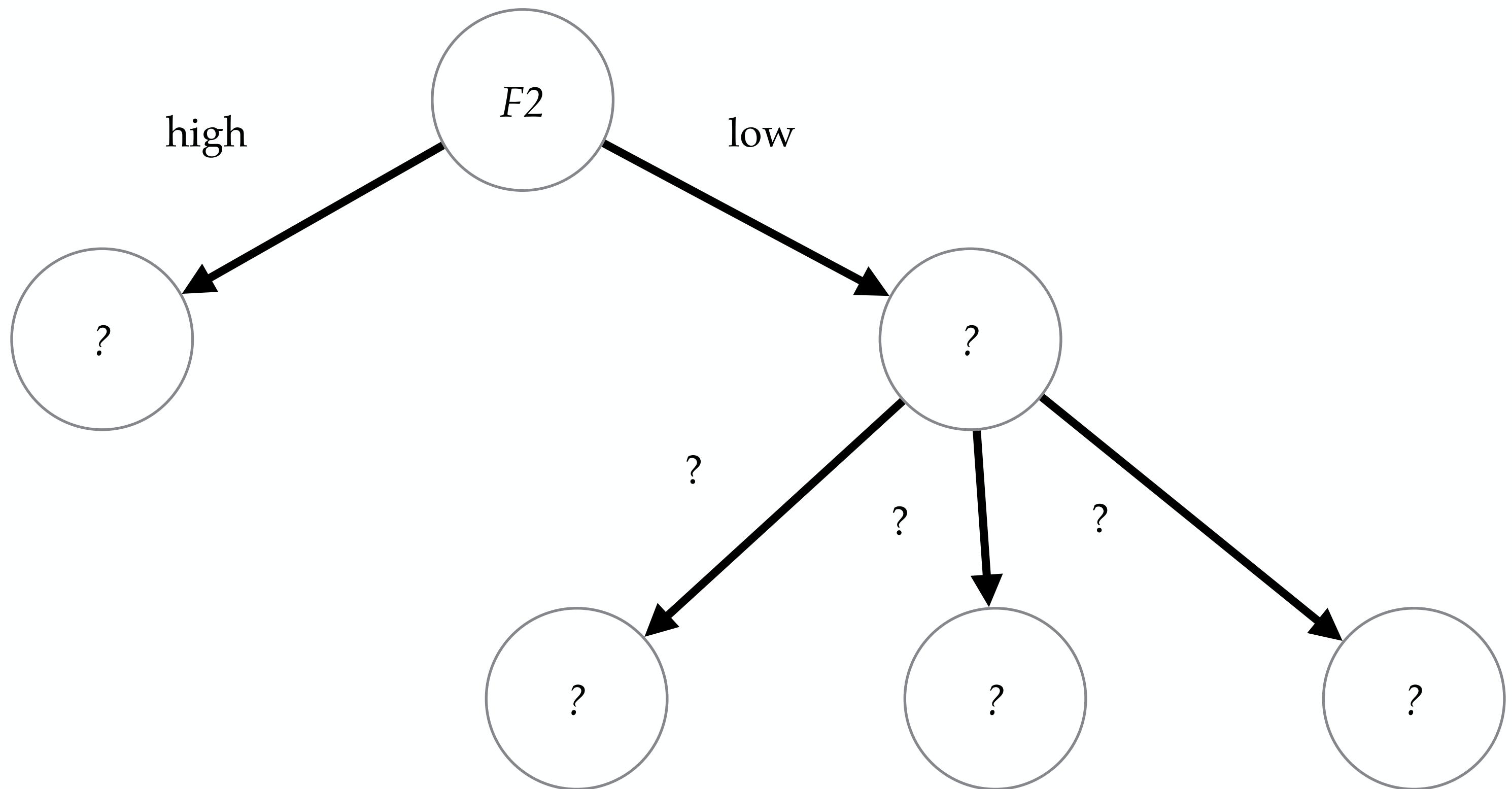
1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *Tree Learning: 2 parts*



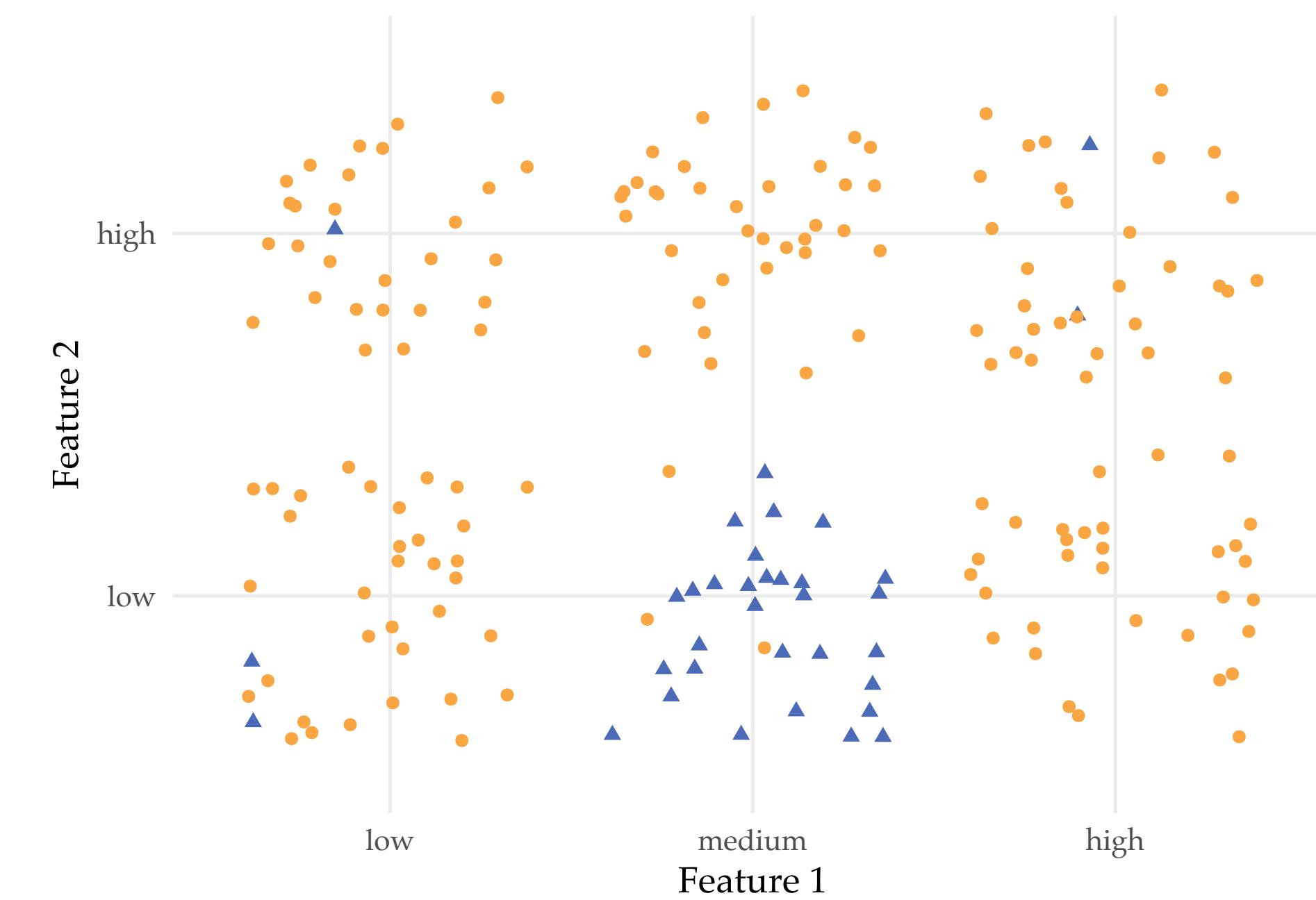
1. Partitioning the feature space/learning the structure of the tree
2. Assigning values to each part (**let's start here**)

# Learning: Value Assignment

Given 0-1 loss and the given partitioning, what is the optimal value  $c_l$  to assign to each leaf  $l$ ?

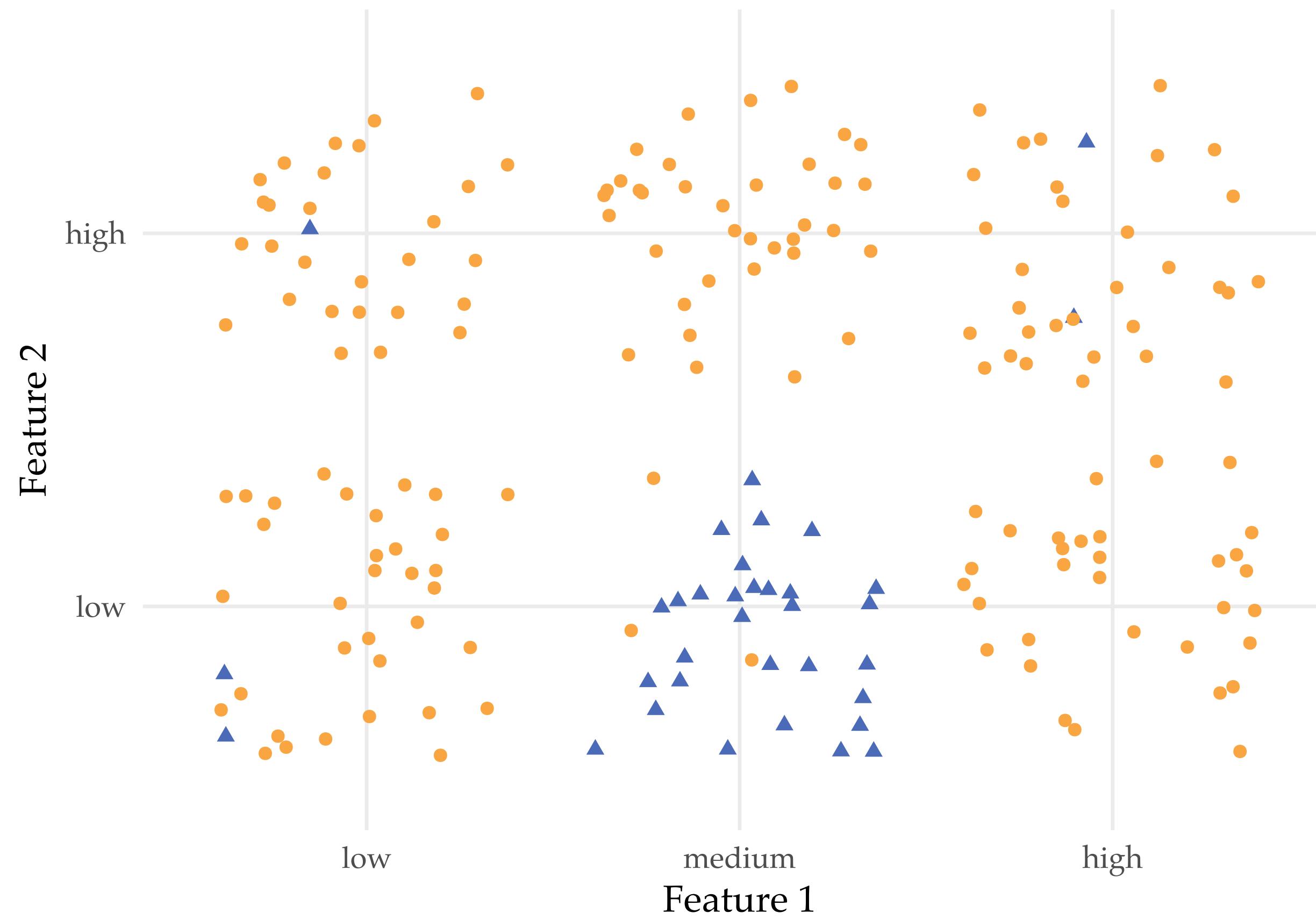
$$h(\mathbf{x}) = \sum_{l \in \text{Leaves}} c_l \mathbb{I}(\mathbf{x} \in l)$$

$$\min_{c_l} \sum_{i=1}^N \mathbb{I}(h(\mathbf{x}_i) \neq y_i)$$

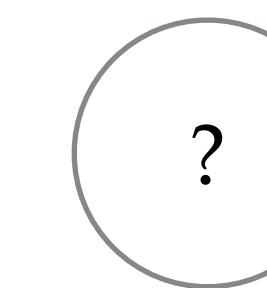


*Answer: the majority label of all the objects assigned to the leaf is the optimal choice*

# *Example: Root Node*



Consider the following “tree”. What should the assignment in the node be?



# *Learning: Splitting*

- If all objects in a node belong to one class: we are done!
- If not: find a node-variable combination that increases the **quality** of the tree the most if we split the node further
- How do we measure the **quality**?
  - Reduction in error before / after the split?
  - Not sensitive enough?

## *Measure of improvement*

*Set of objects in the node*

$$I(S) -$$

$$\sum_{V \in \text{Values}(F)}$$

$$\frac{|S_V|}{|S|} I(S_V)$$

*Number of objects in the node with value V*

*Feature that we consider*

If  $I(\cdot)$  is the misclassification error in a set, this measures how much the 0-1 loss will decrease if we split the node into multiple nodes using feature  $F$

Other ways to measure impurity of a node  $I(\cdot)$ ?

# *Splitting Criteria: Choosing $I(S)$*

$$1 - \max_c p_c$$

*Misclassification*

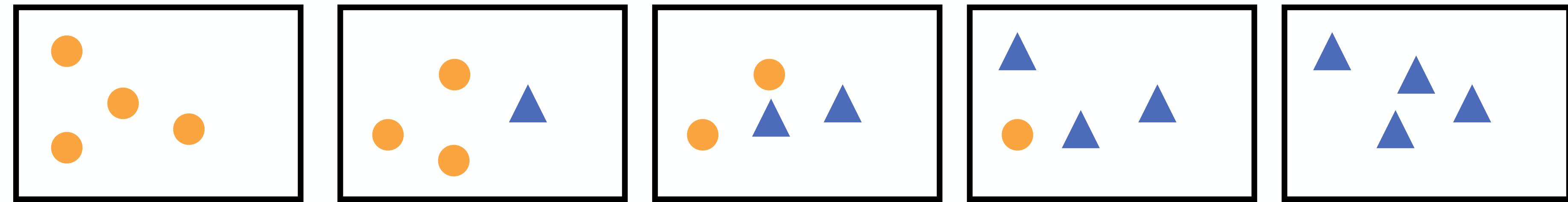
$$-\sum_c p_c \log(p_c)$$

*Entropy*

$$\sum_c p_c(1 - p_c)$$

*Gini Index*

# Splitting Criteria: Information Measures



$$p_{\bullet}$$

1.0

0.75

0.5

0.25

0.0

$$1 - \max_c p_c$$

0.0

0.25

0.5

0.25

0.0

$$-\sum_c p_c \log(p_c)$$

0.0 (!)

0.56

0.69

0.56

0.0 (!)

$$\sum_c p_c(1 - p_c)$$

0.0

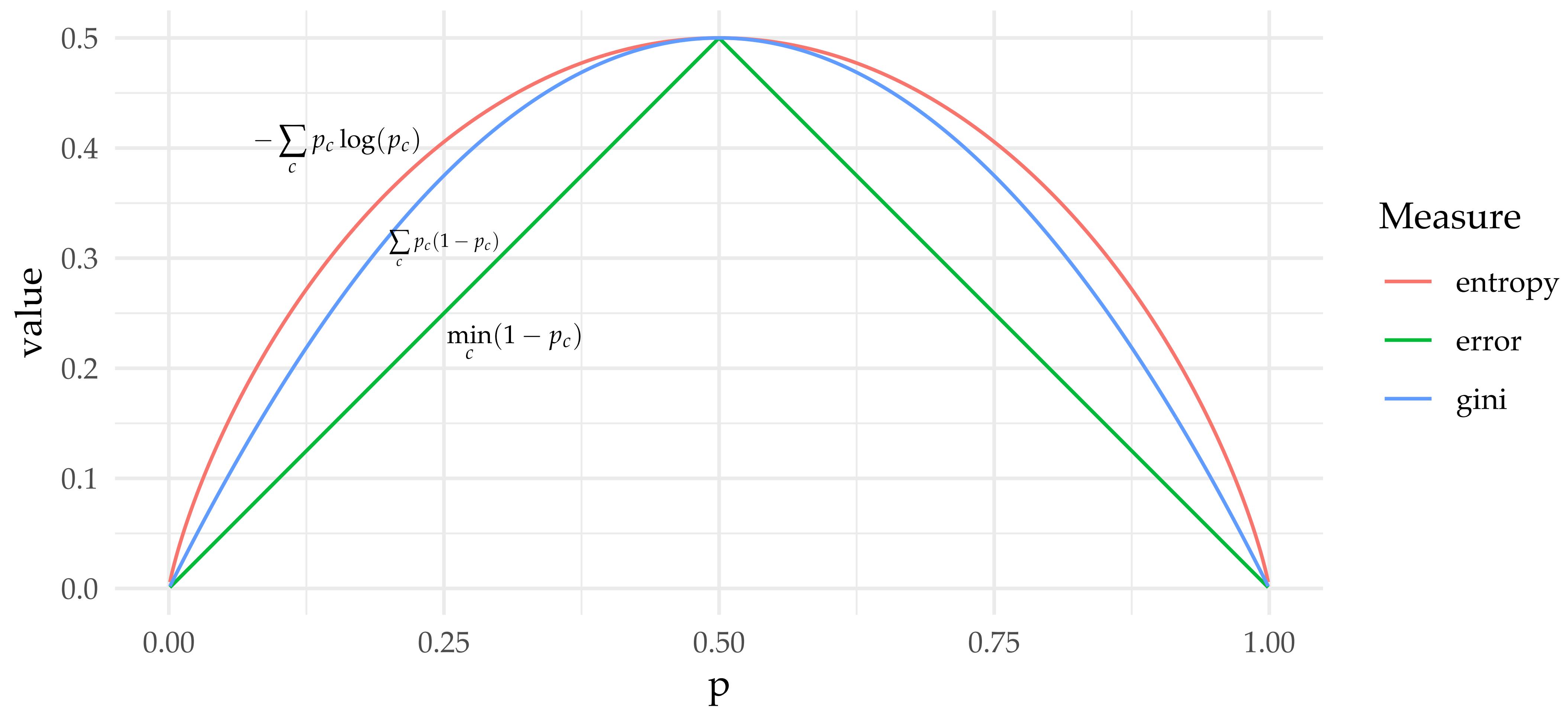
0.375

0.5

0.375

0.0

# Comparing Splitting Criteria



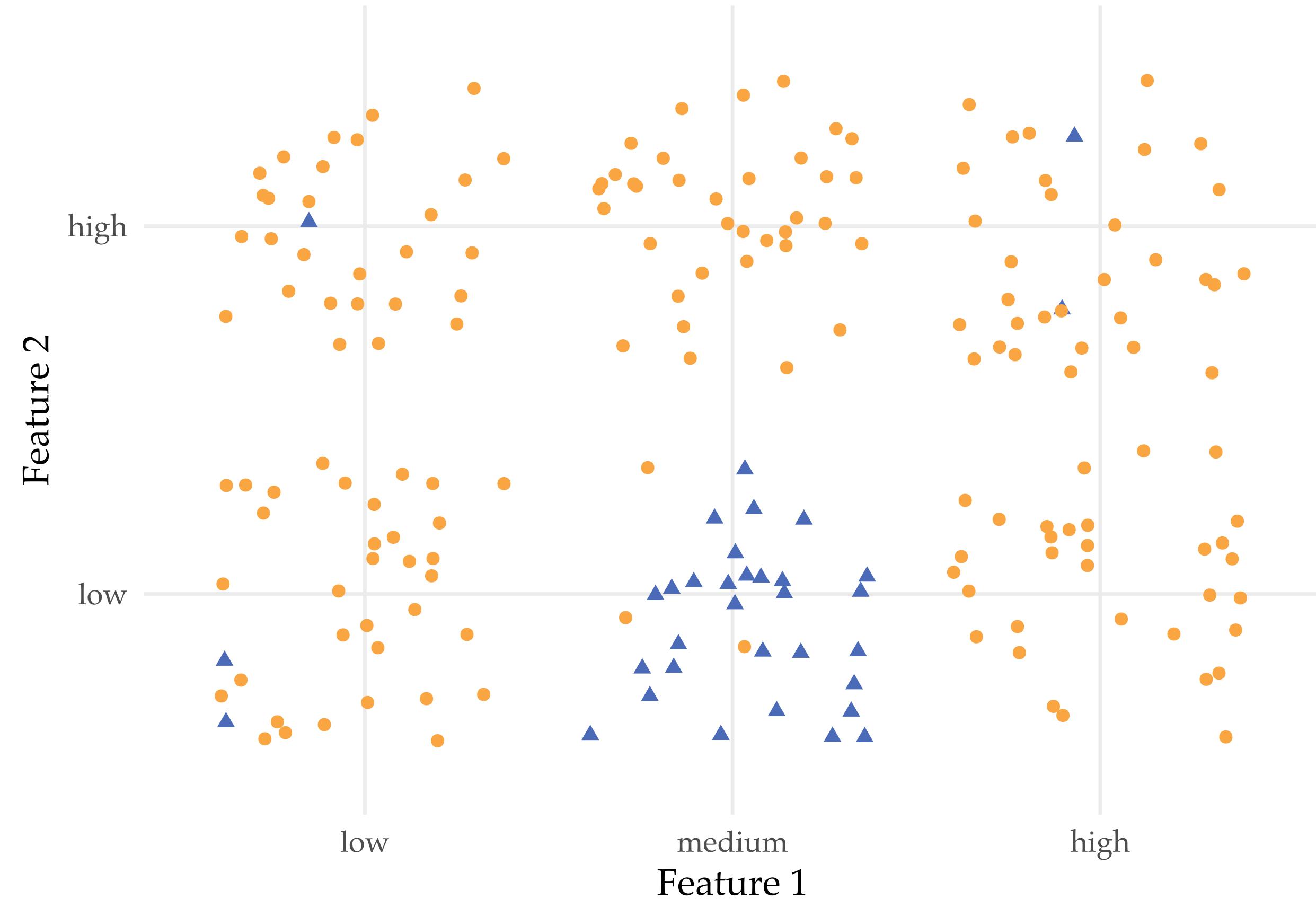
Note: Entropy scaled to have the same maximum

# *Example: Information Gain Calculation*

**Overall:**  
 $35/200$

**Feature 1:**  
Low ( $3/65$ ), Medium ( $30/69$ ), High ( $2/66$ )

**Feature 2:**  
High ( $3/99$ ), Low ( $32/101$ )



*Should we split using feature 1 or feature 2?*

# *Information Gain*

$$I(S) = \sum_{V \in Values(F)} \frac{|S_V|}{|S|} I(S_V)$$

with

$$I(S) = - \sum_c p_c \log(p_c)$$

# Example: Information Gain Calculation

$$I(S) = - \sum_c p_c \log(p_c) \quad I(S) - \sum_{V \in Values(F)} \frac{|S_V|}{|S|} I(S_V)$$

*Node Entropy*

**Overall:**  
35/200

$$-\left(\frac{35}{200} \log\left(\frac{35}{200}\right) + \frac{165}{200} \log\left(\frac{165}{200}\right)\right) = 0.464$$

**Feature 1:**

Low (3/65)

$$-\left(\frac{3}{65} \log\left(\frac{3}{65}\right) + \frac{62}{65} \log\left(\frac{62}{65}\right)\right) = 0.187$$

Medium (30/69)

$$-\left(\frac{30}{69} \log\left(\frac{30}{69}\right) + \frac{39}{69} \log\left(\frac{39}{69}\right)\right) = 0.685$$

High (2/66)

$$-\left(\frac{2}{66} \log\left(\frac{2}{66}\right) + \frac{64}{66} \log\left(\frac{64}{66}\right)\right) = 0.136$$

*Information Gain*

$$0.464 - \left(\frac{65}{200} 0.187 + \frac{69}{200} 0.685 + \frac{66}{200} 0.136\right) = 0.122$$

**Feature 2:**

High (3/99)

$$-\left(\frac{3}{99} \log\left(\frac{3}{99}\right) + \frac{96}{99} \log\left(\frac{96}{99}\right)\right) = 0.136$$

Low (32/101)

$$-\left(\frac{32}{101} \log\left(\frac{32}{101}\right) + \frac{69}{101} \log\left(\frac{69}{101}\right)\right) = 0.624$$

$$0.464 - \left(\frac{99}{200} 0.136 + \frac{101}{200} 0.624\right) = 0.081$$

# *Splitting Criteria: Gain ratio*

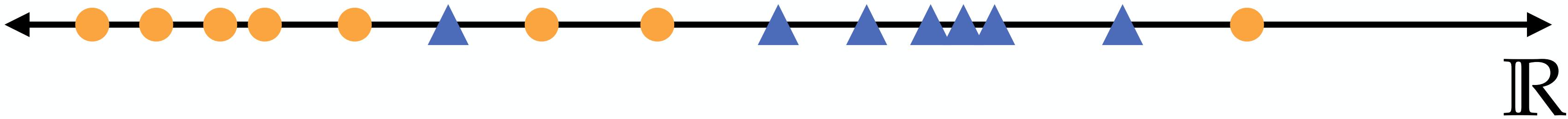
- The information gain is biased towards features with many discrete values
- We may overfit on such features (for instance, someone's social security number)
- One idea by Quinlan is to correct for this using the entropy of the feature, the intrinsic value (IV)
- Actually some (many) implementations only use binary splits.

$$IG(S) = I(S) - \sum_{V \in Values(F)} \frac{|S_V|}{|S|} I(S_V)$$

$$IV(S) = - \sum_{V \in Values(F)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}$$

$$GainRatio(S) = \frac{IG(S)}{IV(S)}$$

# *Splitting: Continuous Variables*



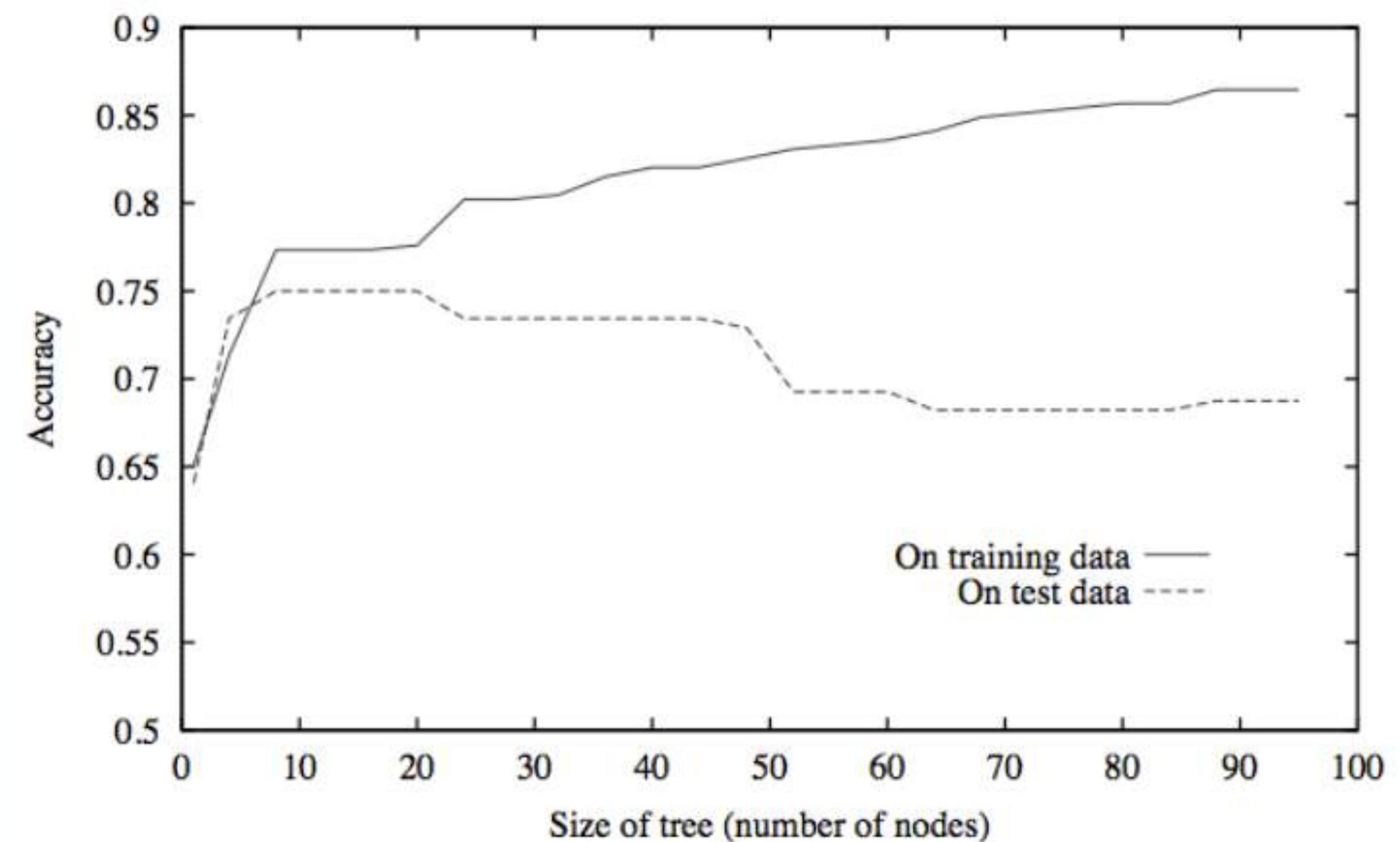
*Extra challenge:* we need to find a threshold for the split.

Consider all thresholds and pick the optimal one

How many thresholds do we need to consider?

# *Learning: Stopping*

- If we fit pure trees, we quickly start overfitting (why?)
- Criteria:
  - minimum node size
  - maximum tree depth
  - minimum information gain
- Alternative/complementary strategy: grow and prune



Copyright © 2011 Victor Lavrenko

Figure credit: Tom Mitchell, 1997

# *Learning: Pruning*

$$C_\alpha(Tree) = \sum_{l \in \text{Leaves}} |S_l| I(S_l) + \alpha |\text{Leaves}|$$

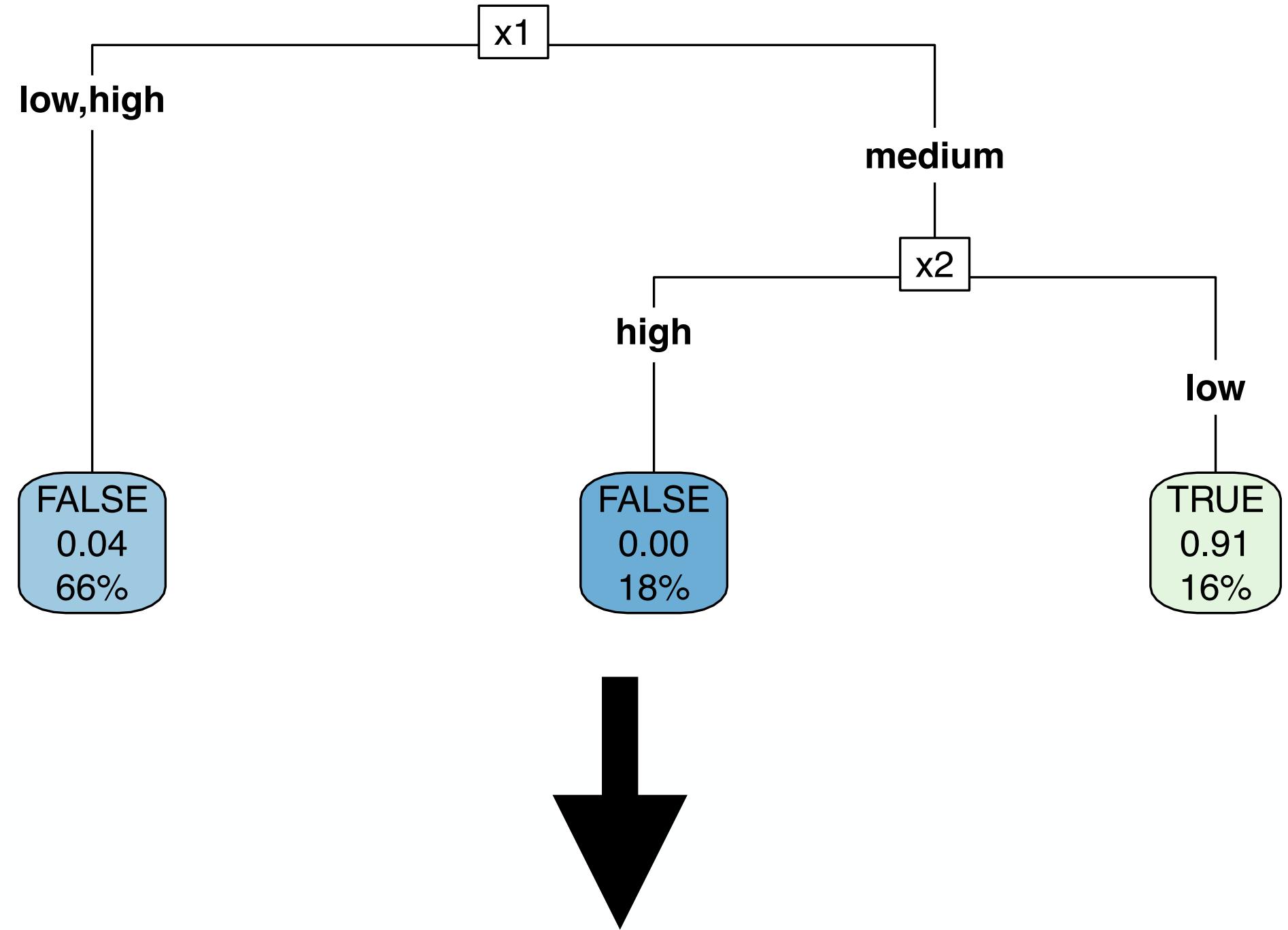


*Leaves in the tree*

For each alpha, optimise the tree by removing subtrees

Select optimal alpha using a validation set

# *From Trees to Rule Lists*



0.00 when  $x_1$  is medium &  $x_2$  is high

0.04 when  $x_1$  is low or high

0.91 when  $x_1$  is medium &  $x_2$  is low

We can convert the decisions in the tree to a set of rules, which may be more interpretable, or easier to communicate. We can also try to further simplify these rules

# *Generalizations*

- Multi-class
  - Take the sum over all classes in the entropy term
- Regression
  - Squared loss as the loss function
  - Leaf value: average outcome
  - Split criterion: minimize the variance
  - Covered in Probability & Statistics

# *Summary: Decision Trees*

- Non-linear classifier
- Choices in learning:
  - Splitting criterion (information gain, gain ratio, entropy / gini index, etc.)
  - Stopping (minimum gain, node size, max depth)
  - Pruning (remove parts based on performance on validation set)

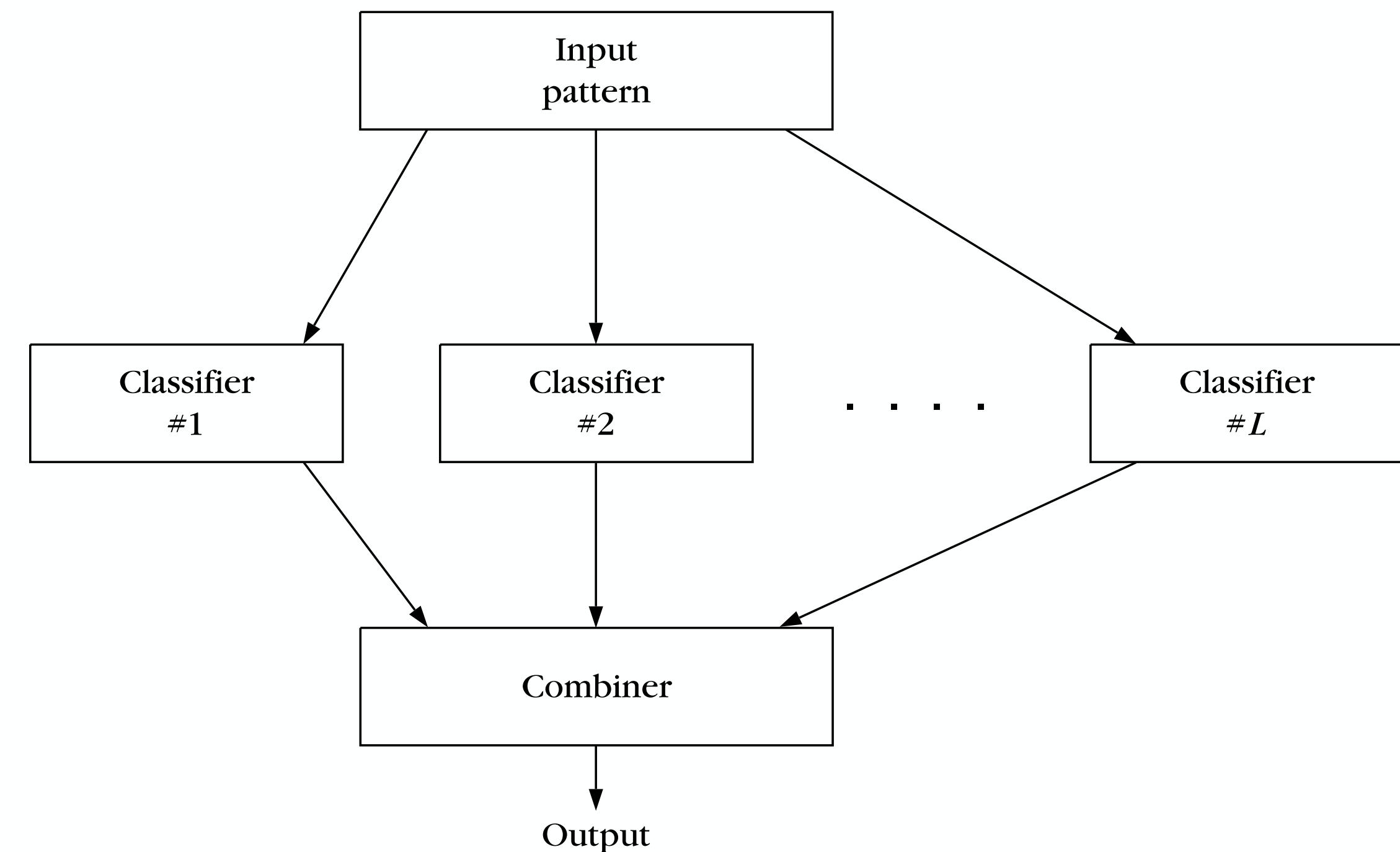
# *Advantages & Disadvantages*

- ✓ “Interpretable”
- ✓ Automatic feature selection
- ✓ Easy to incorporate discrete features and missing values
- ✓ Fast
- ✗ Unstable
- ✗ Cannot model linear relationships efficiently
- ✗ “Greedy”

# CLASSIFIER COMBINING

# *Classifier Combining*

- Multiple classifiers that each make a prediction: perhaps they make different kinds of mistakes
- How do we combine these predictions?
- Does it help to combine them?



# *Condorcet's Jury Theorem*



# *Condorcet's Jury Theorem*

## **Question**

When does it make sense to combine decisions?

## **Theorem**

If  $p > 0.5$ , adding voters to the jury increases probability of correct majority vote (assuming voting is independent)



*Nicolas de Condorcet*  
(1743-1794)

# *Combining Strategies*

- Fixed Rules
  - Hard rules e.g. majority voting
  - Soft rules: e.g. mean, product

$$\max_y \sum_{j=1}^C \frac{1}{C} h_j(y|x)$$

$$\max_y \prod_{j=1}^C h_j(y|x)$$

$$\min_{p(y|x)} \sum_j D(h_j(y|x), p(y|x))$$

- Learned rules
  - Learn a classifier to output a decision based on the outputs of a set of base classifiers

# *What to combine*

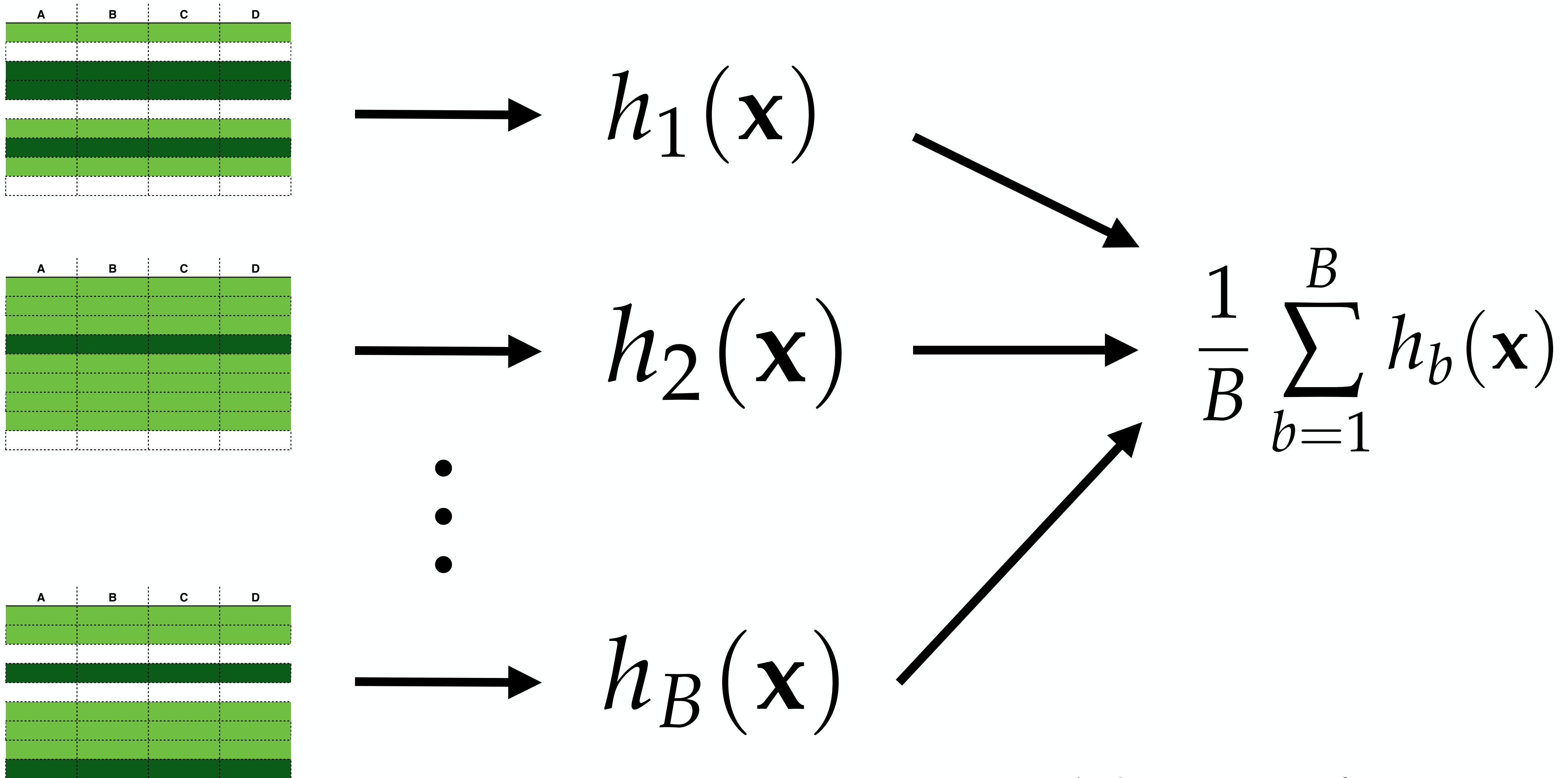
- Where do the multiple classifiers come from?
  - Different classifiers on the same dataset?
  - The same classifier on different datasets?
- Diversity may be helpful
  - Let's consider an example where we construct “diverse” classifiers using different versions of the same dataset

# *Random Forests*

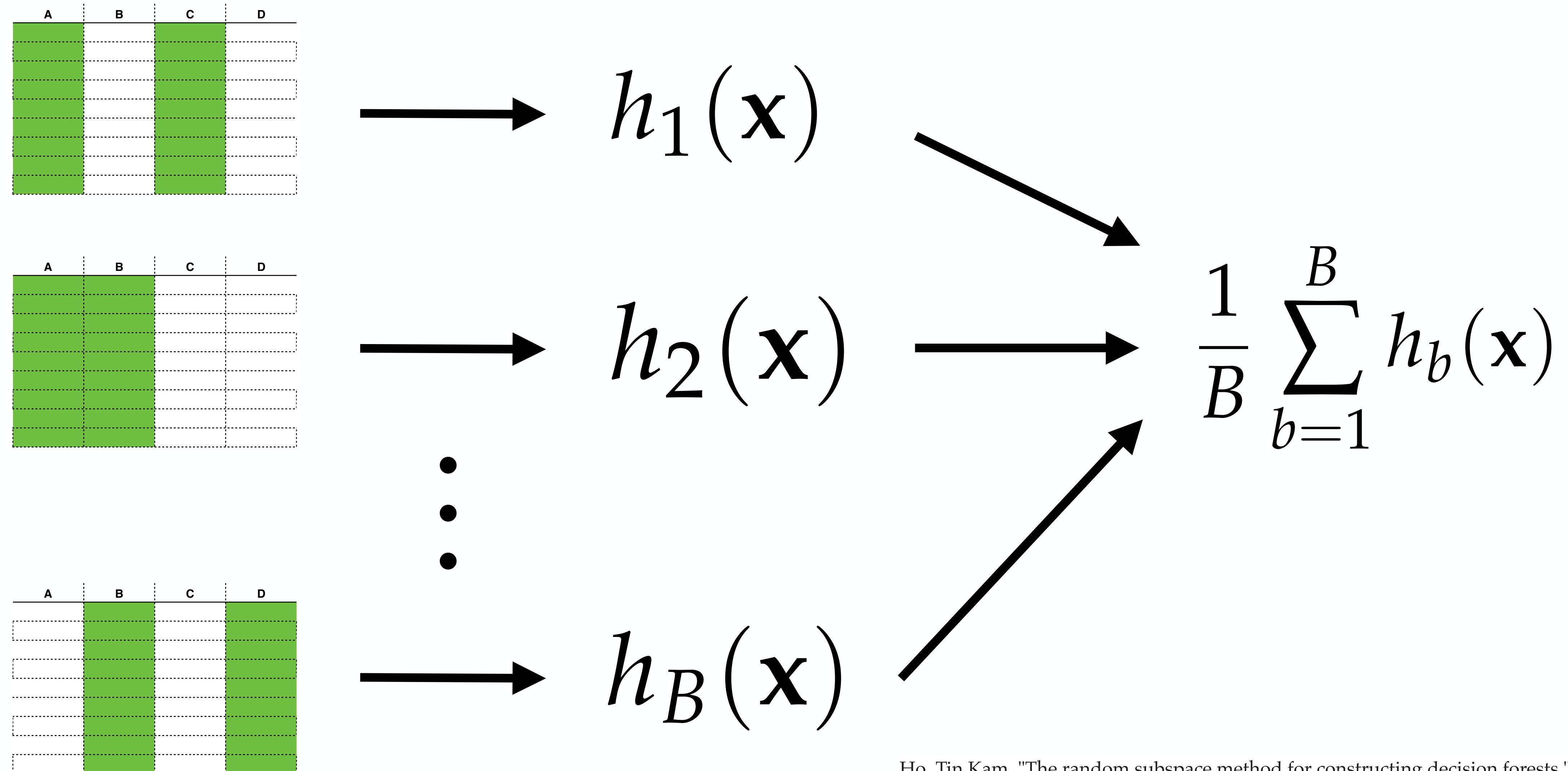
- Single tree might easily overfit and might be unstable
- Combine multiple trees
  - Should be sufficiently “different”
    1. Randomly select features
    2. Randomly select samples (with replacement)

*Random Forest:* Construct a large number of trees using randomly selected objects and features and combine their decisions

# Bagging (Bootstrap Aggregation)

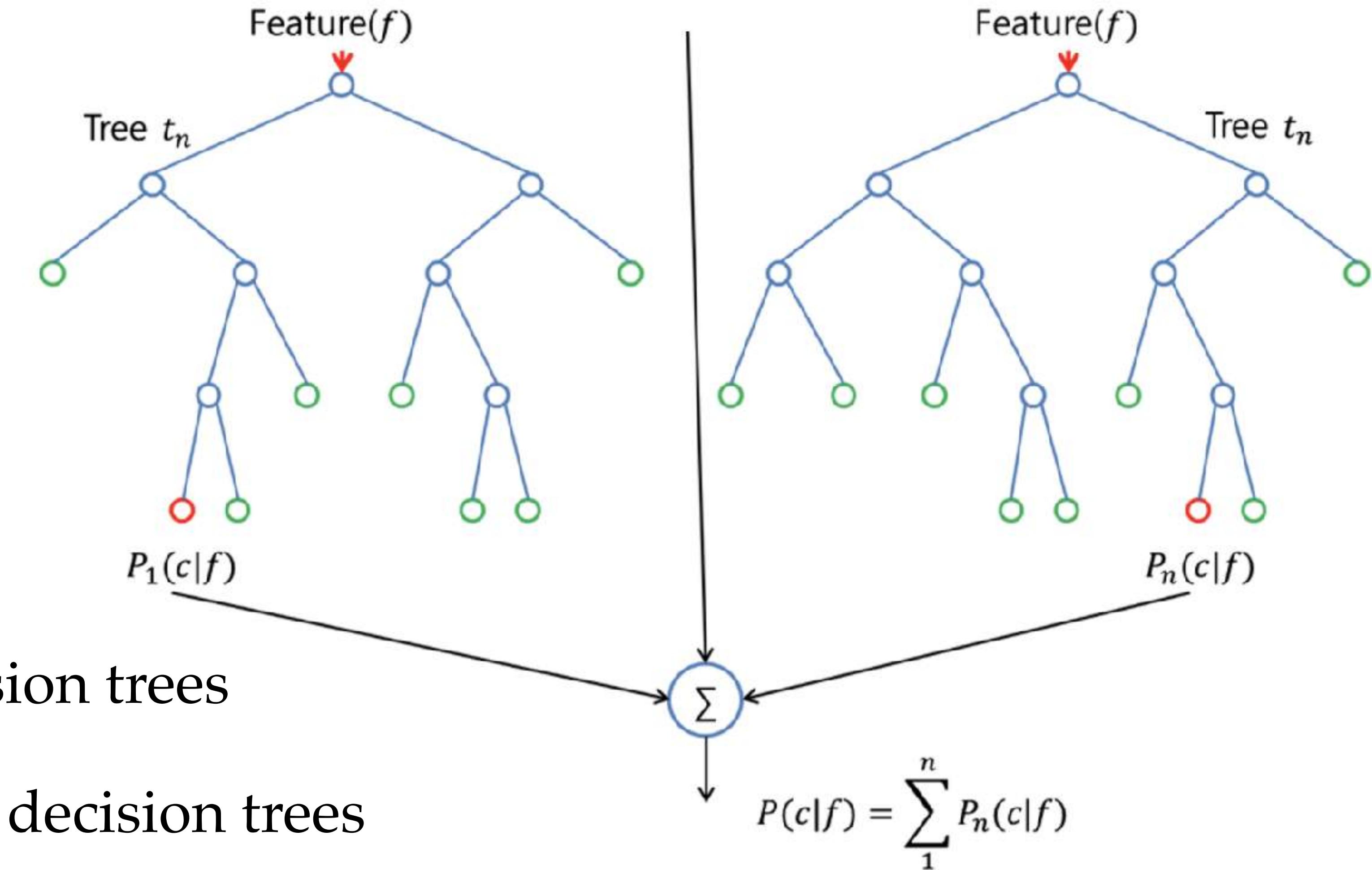


# *Random Subspaces*



Ho, Tin Kam. "The random subspace method for constructing decision forests." *IEEE transactions on pattern analysis and machine intelligence* 20.8 (1998): 832-844.

# *Random Forest*



- Breiman: bagging with decision trees
- Ho: random subspaces with decision trees
- Random Forest in practice: bagging and random subspaces with decision trees

# *Parameters*

## TREES

- Depth
- Number of leaves
- Minimum number of objects per node
- Information criterion
- Pruning

## FOREST

- Number of trees
- Size of subspaces considered

# *Pros and Cons*

## PROS

- Flexible / low bias
- Works for many different types of data
- Embarrassingly parallel
- Produces out-of-bag (OOB) estimates
- (Scale) invariant
- (Relatively) few hyperparameters

## CONS

- Harder to interpret than single trees
- Computationally expensive



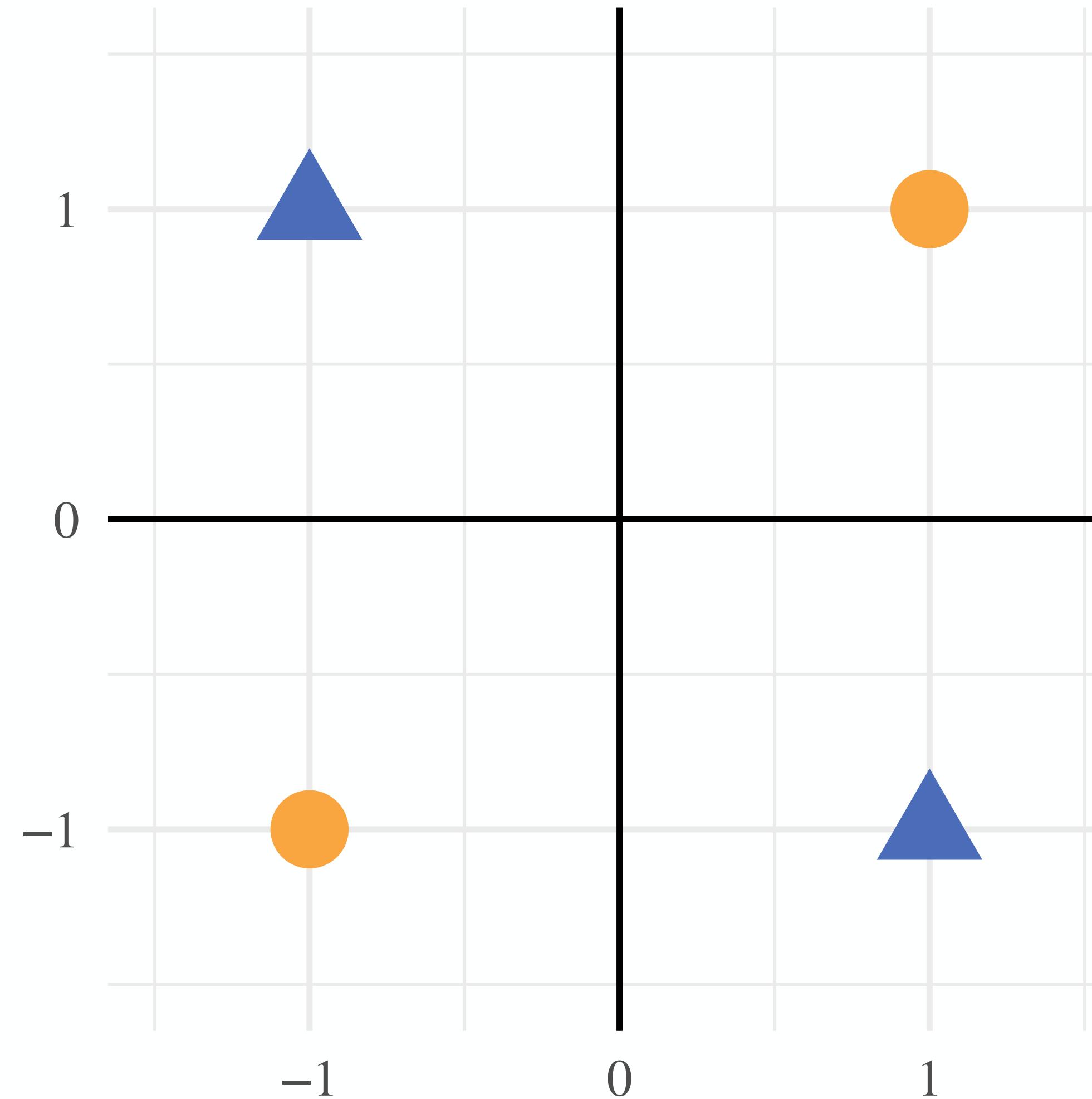
# KINECT **SPORTS**

A black Kinect sensor bar is positioned at the bottom of the logo, featuring four circular sensors and a small screen.

# *Recap*

- Decision trees: a greedy space partitioning approach to non-linear classification that leads to an “intuitive” classifier, but typically has high variance
- Classifier combining can lead to better predictions by combining multiple diverse classifiers
- Random Forests are a combination of decision trees constructed using random subsampling of objects and features.
- Next time: MLPs, a parametric non-linear discriminative classifier that acts as a “learned” combiner

# *Revisiting XOR*



Can the decision tree learner covered today solve this?

## NON-LINEAR (DISCRIMINATIVE) CLASSIFIERS

*Multilayer Perceptrons*

OH, HEY, YOU ORGANIZED  
OUR PHOTO ARCHIVE!

YEAH, I TRAINED A NEURAL  
NET TO SORT THE UNLABELED  
PHOTOS INTO CATEGORIES.

WHOA! NICE WORK!

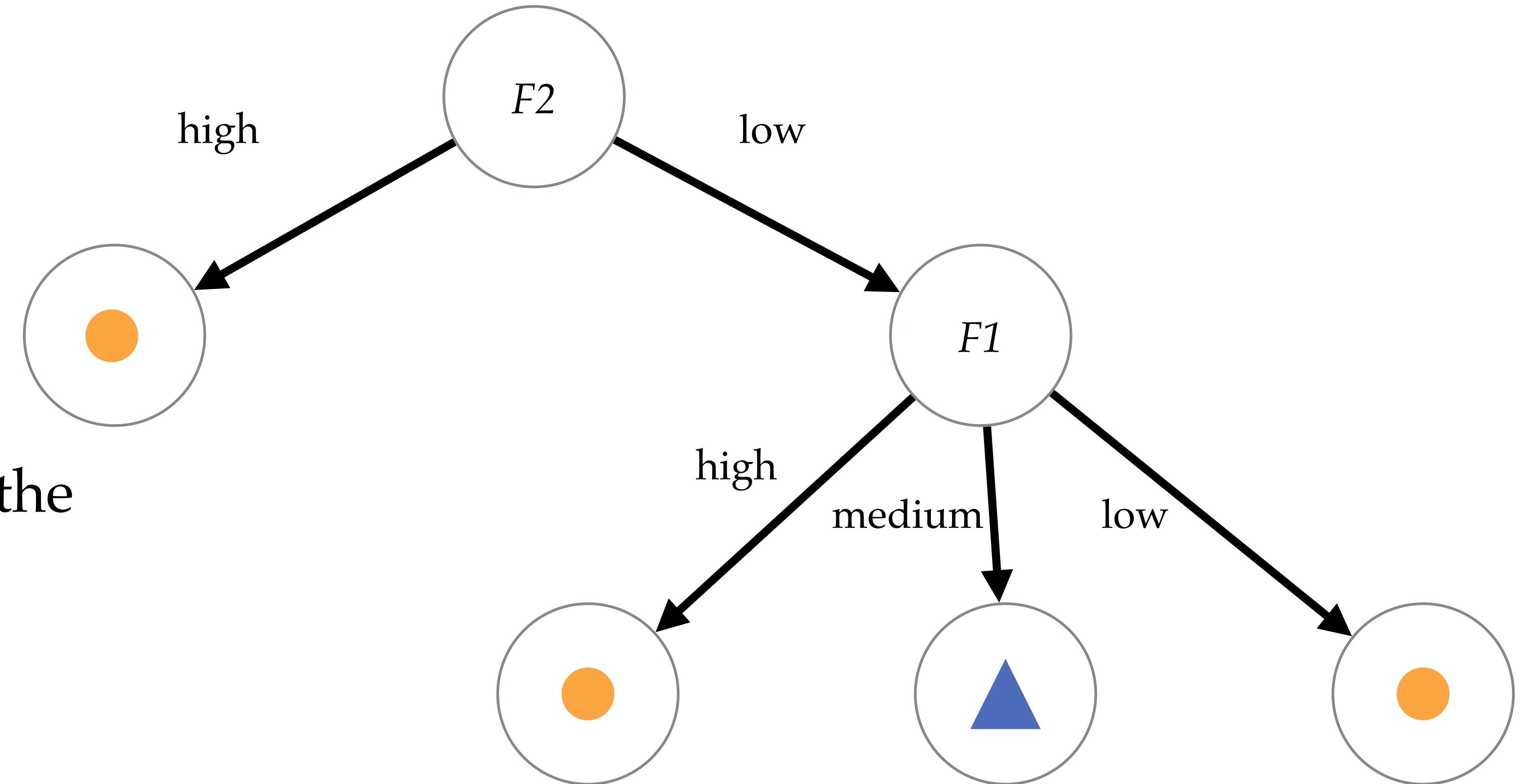


ENGINEERING TIP:  
WHEN YOU DO A TASK BY HAND,  
YOU CAN TECHNICALLY SAY YOU  
TRAINED A NEURAL NET TO DO IT.

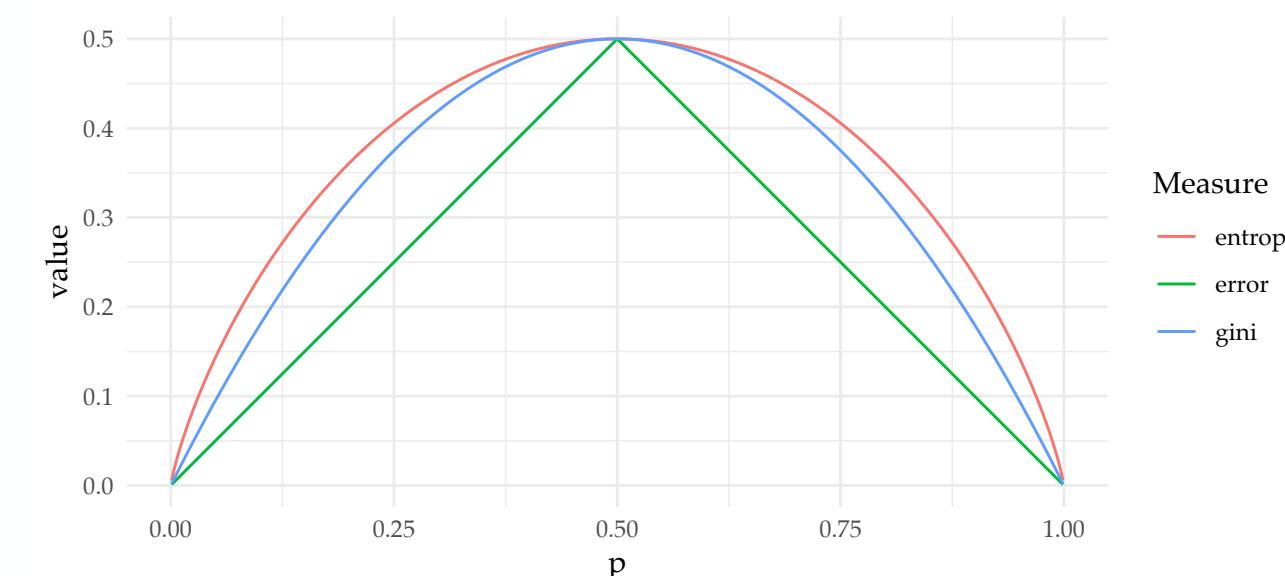
# Recap: Decision Trees

1. Recursively, greedily, partition the space, maximising the gain

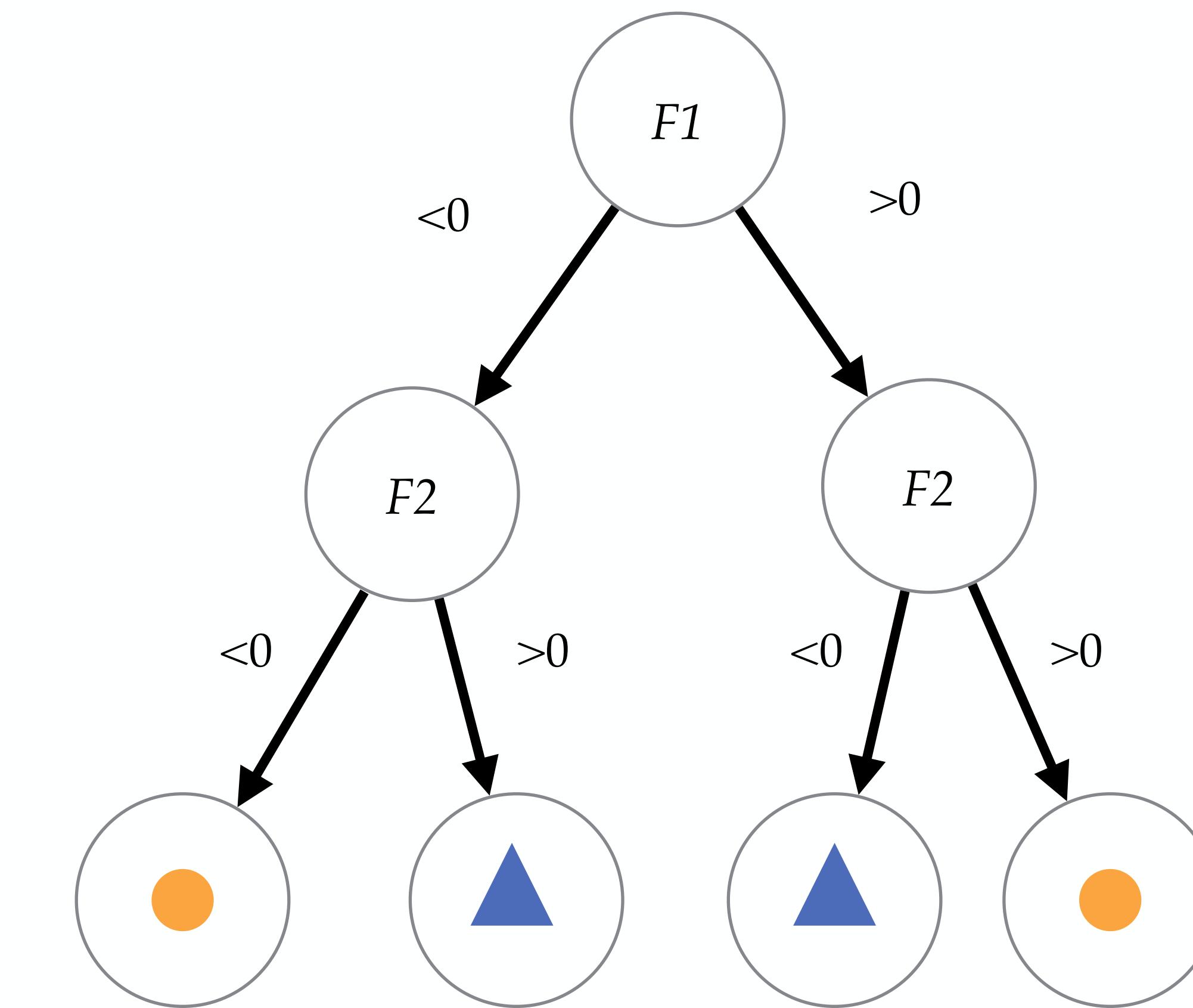
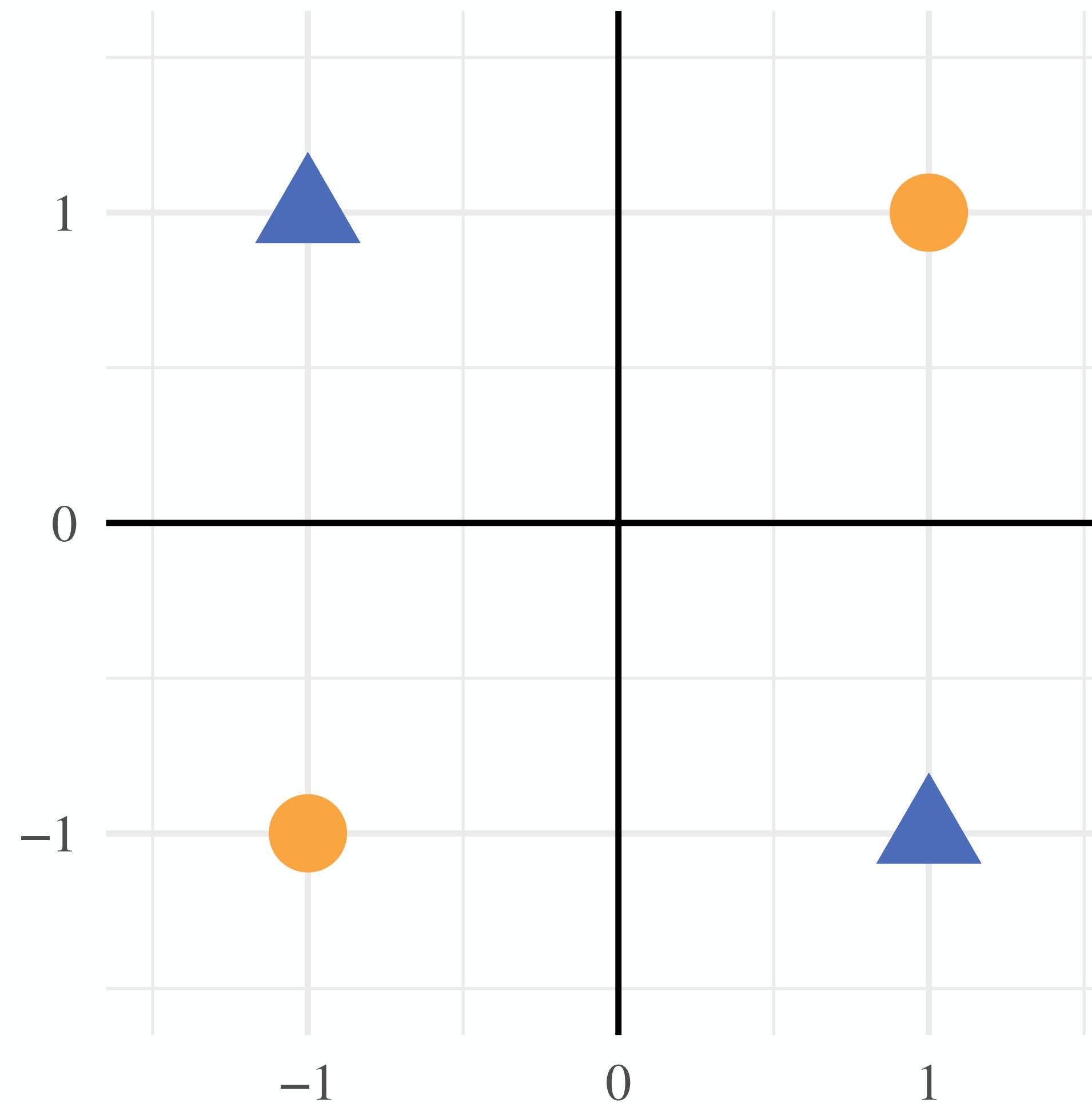
$$I(S) - \sum_{V \in Values(F)} \frac{|S_V|}{|S|} I(S_V)$$



2. Assign each leaf to a class
3. Stop and / or prune to avoid overfitting



# Revisiting XOR



Can the decision tree learner solve this?

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *Learning Goals & Reading*

## LEARNING GOALS

After practicing with the concepts of this week you are able to

- **Explain when and why non-linear classifiers are needed**
- Explain the basic concepts of two non-linear classifiers:  
multi-layer perceptrons and **decision trees**
- Explain the **underlying algorithm of decision trees** and  
(multi-layer) perceptrons and how they are trained.
- Implement a decision tree
- Explain why and how one can **combine multiple classifiers**
- **Contrast a decision tree and a random forest**

## LITERATURE

Please study the following material from Chris Bishop's "Pattern Recognition and Machine Learning":

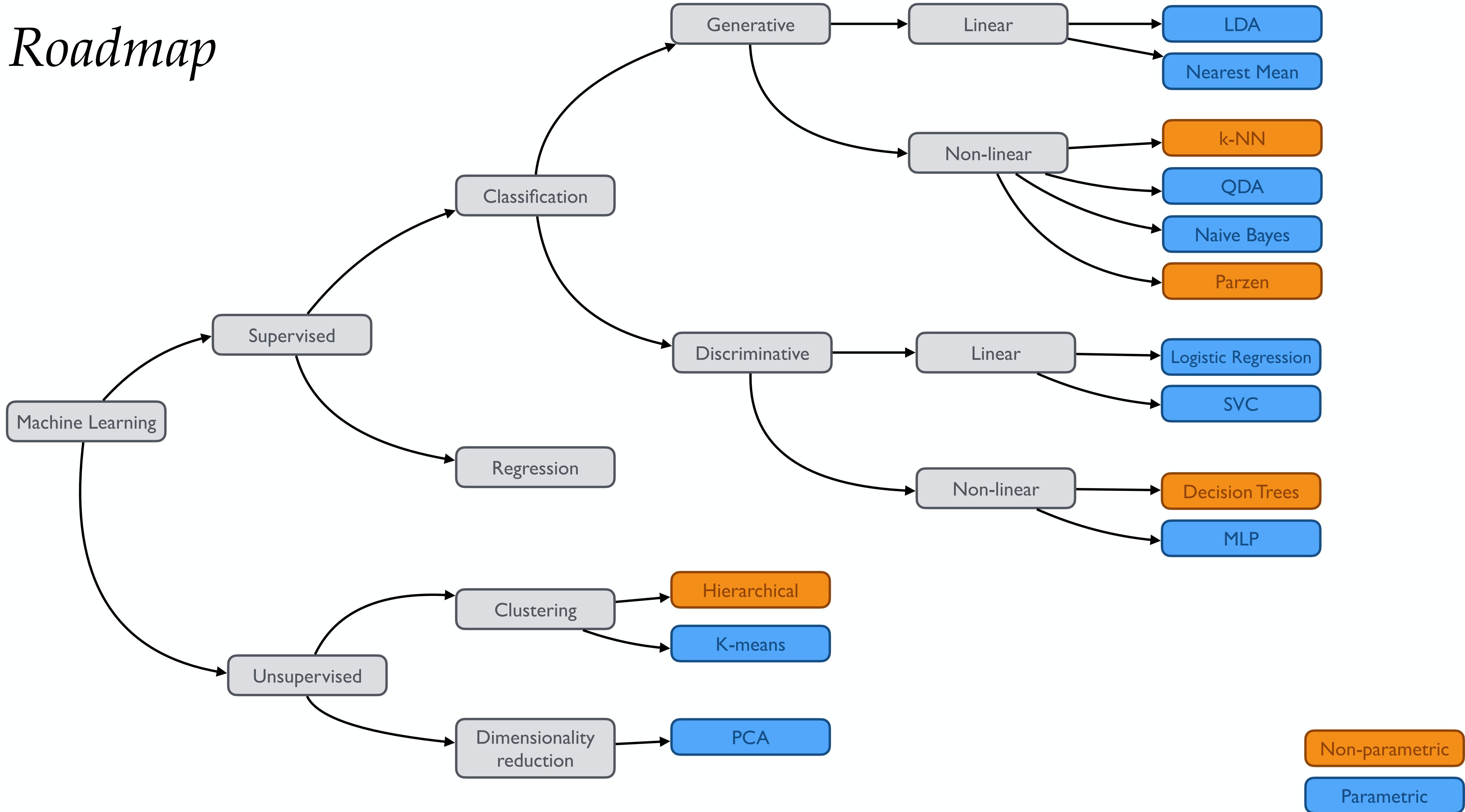
### **Lecture 6.1**

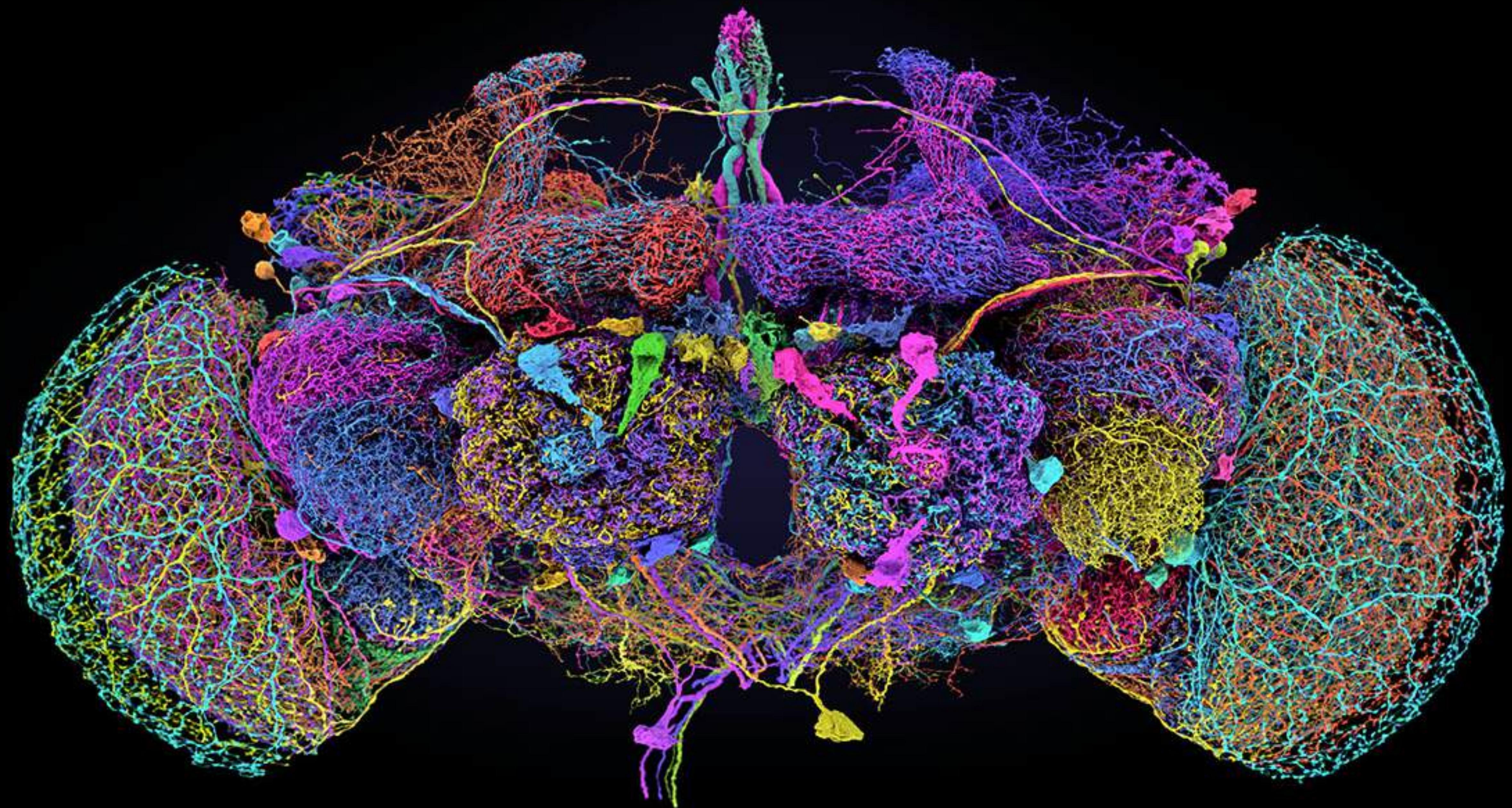
- Section 14.2
- Section 14.4

### **Lecture 6.2**

- Section 4.1.7
- Section 5.1
- Section 5.2 up to and including 5.2.1
- Section 5.3 up to and including 5.3.1

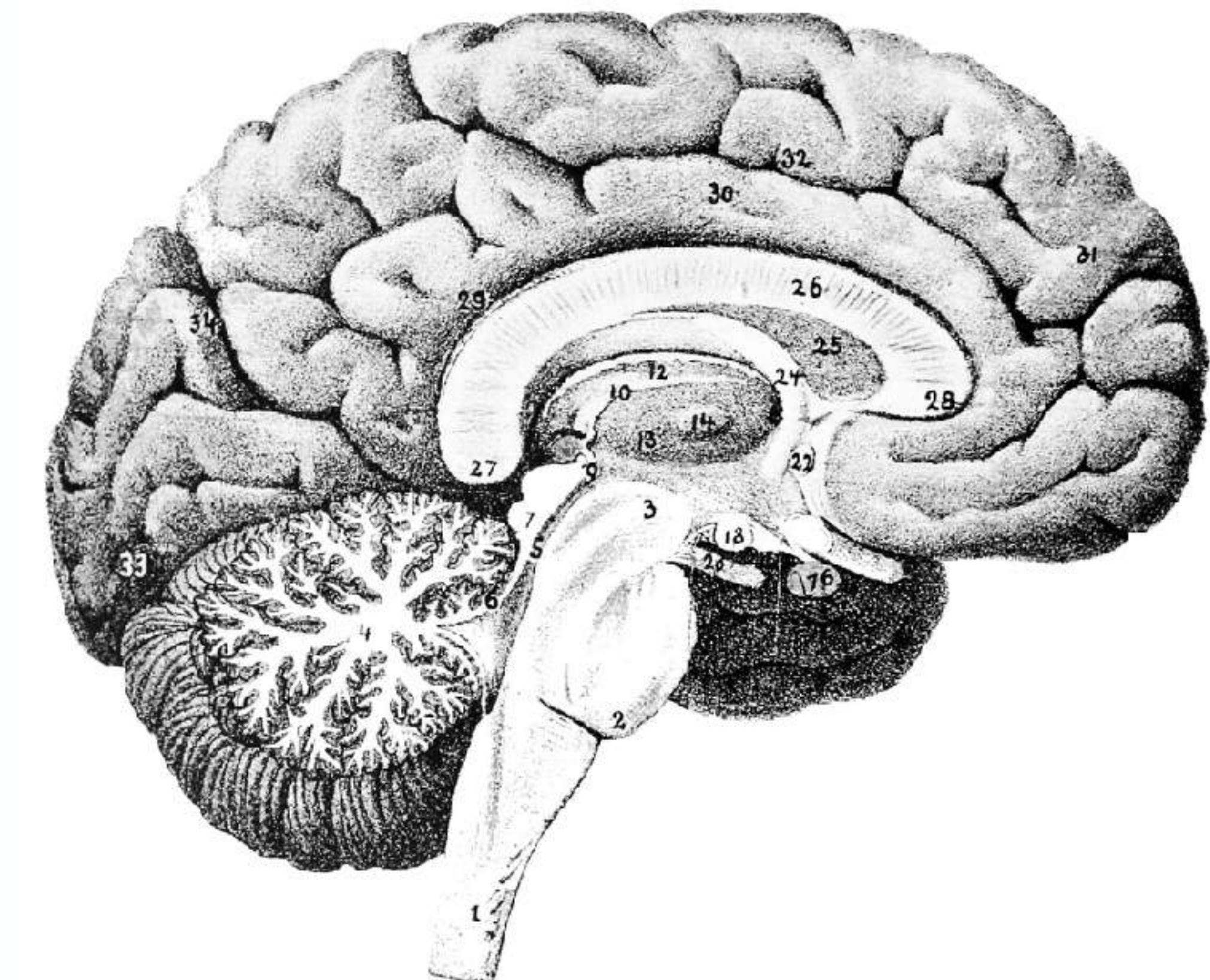
# Roadmap





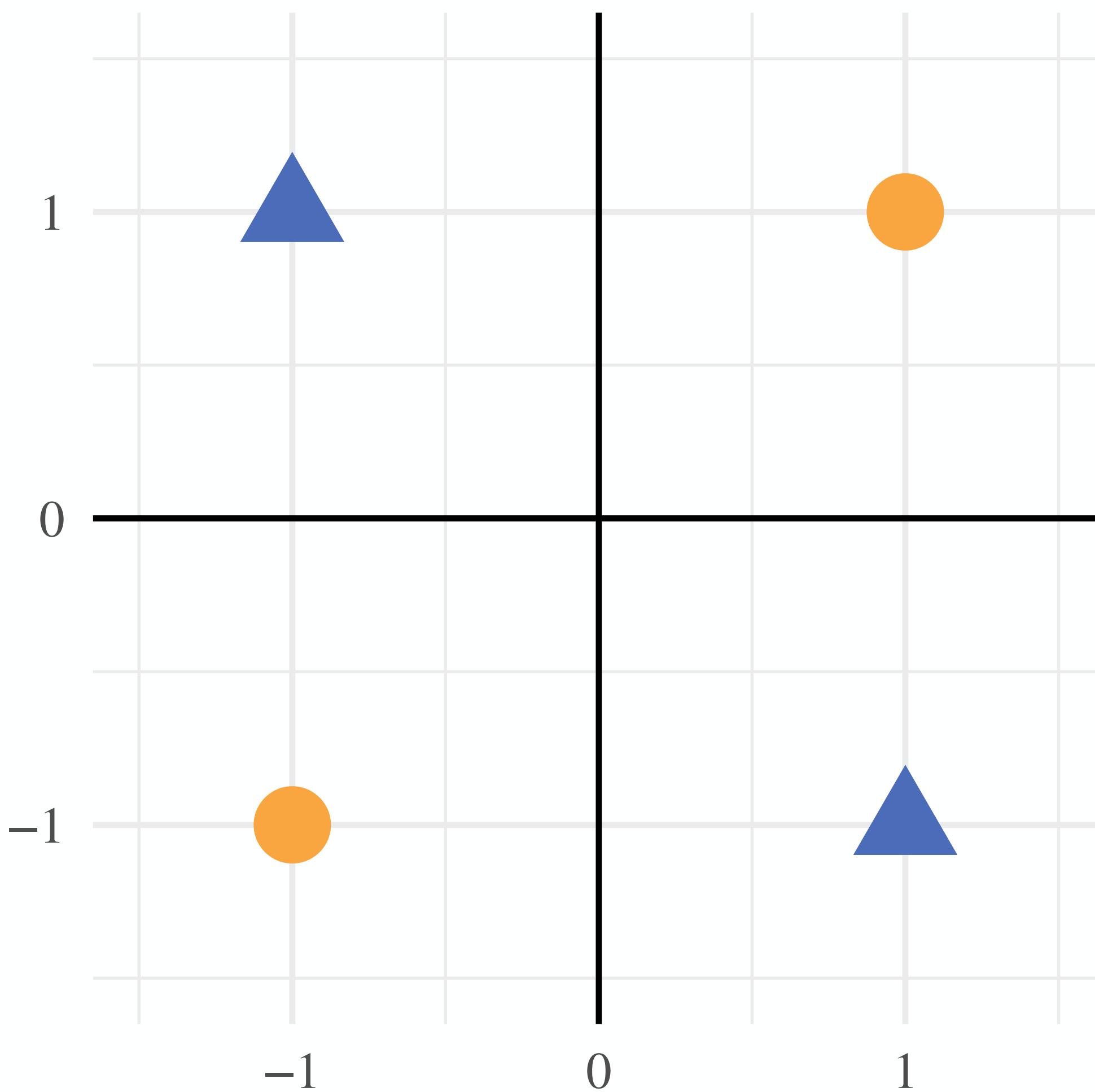
# *Today: the common learning algorithm used in ML today*

- Multi-layer perceptrons / artificial neural networks / feed-forward “deep” neural networks
- Connected to topics we covered:
  - Logistic regression
  - Gradient descent
  - Classifier combining

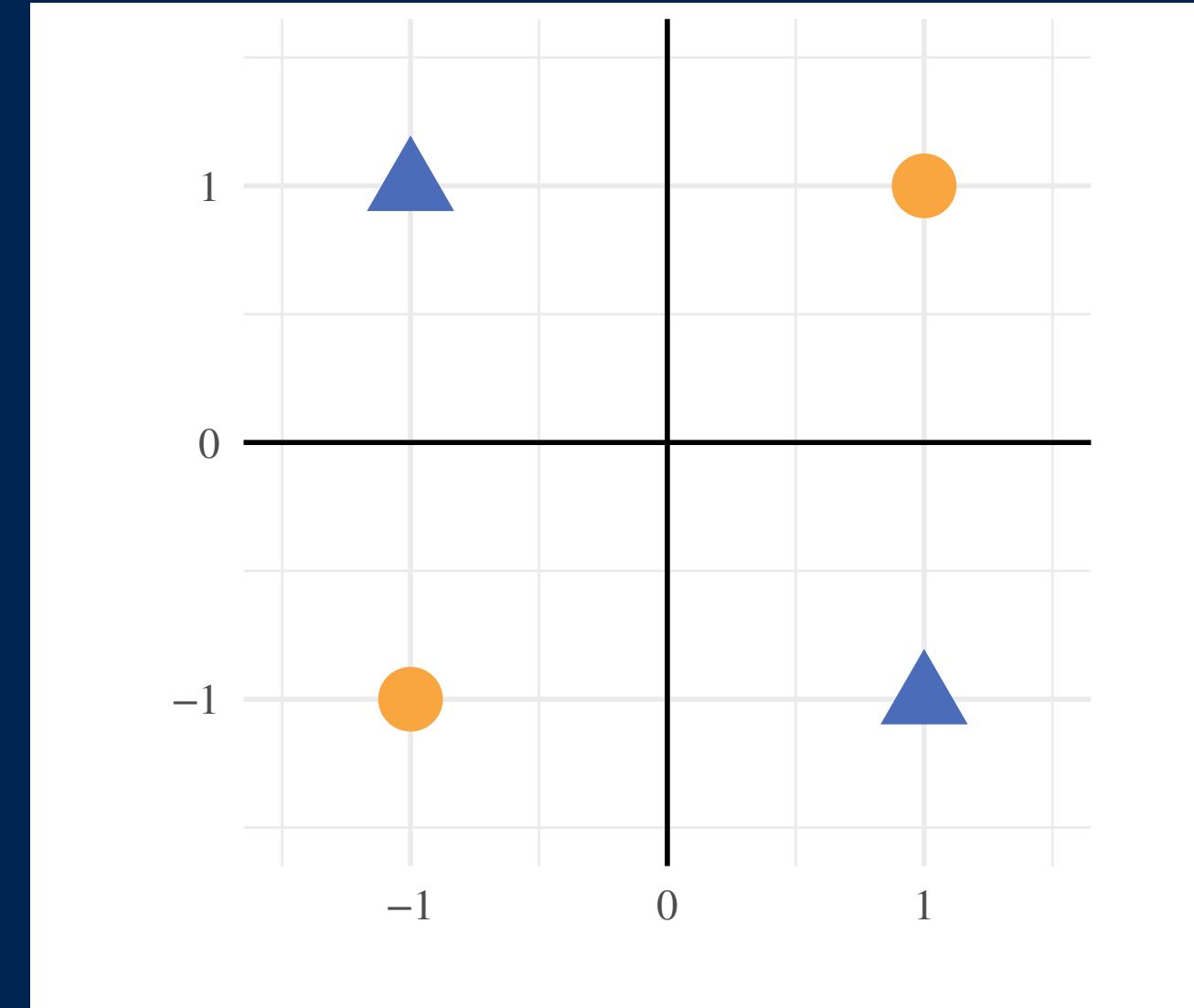


Source: Popular Science Monthly 46

*XOR again*



# *Question: feature transformations*



1. Consider new feature  $x_1x_2$  and draw the problem in this new feature space. What is the optimal classifier in this new space?
2. Is this classifier a linear classifier in the original feature space?

# *Practice Question: BMI*



Body Mass Index (BMI) is defined as weight (in kg) divided by height<sup>2</sup> (in m<sup>2</sup>). Suppose I use a fixed rule: to classify people as healthy when  $\text{BMI} < 30$ .

1. **True or False:** Given the input features weight and height, I can construct a linear classifier that is exactly the same as this rule.
2. **True or False:** Given the input features weight, height and BMI, I can construct a linear classifier that is exactly the same as this rule.

# PERCEPTRON

## Electronic 'Brain' Teaches Itself

The Navy last week demonstrated the embryo of an electronic computer named the Perceptron which, when completed in about a year, is expected to be the first non-living mechanism able to "perceive, recognize and identify its surroundings without human training or control." Navy officers demonstrating a preliminary form of the device in Washington said they hesitated to call it a machine because it is so much like a "human being without life."

Dr. Frank Rosenblatt, research psychologist at the Cornell Aeronautical Laboratory, Inc., Buffalo, N. Y., designer of the Perceptron, conducted the demonstration. The machine, he said, would be the first electronic device to think as the human brain. Like humans, Perceptron will make mistakes at first, "but it will grow wiser as it gains experience," he said.

recognize the difference between right and left, almost the way a child learns.

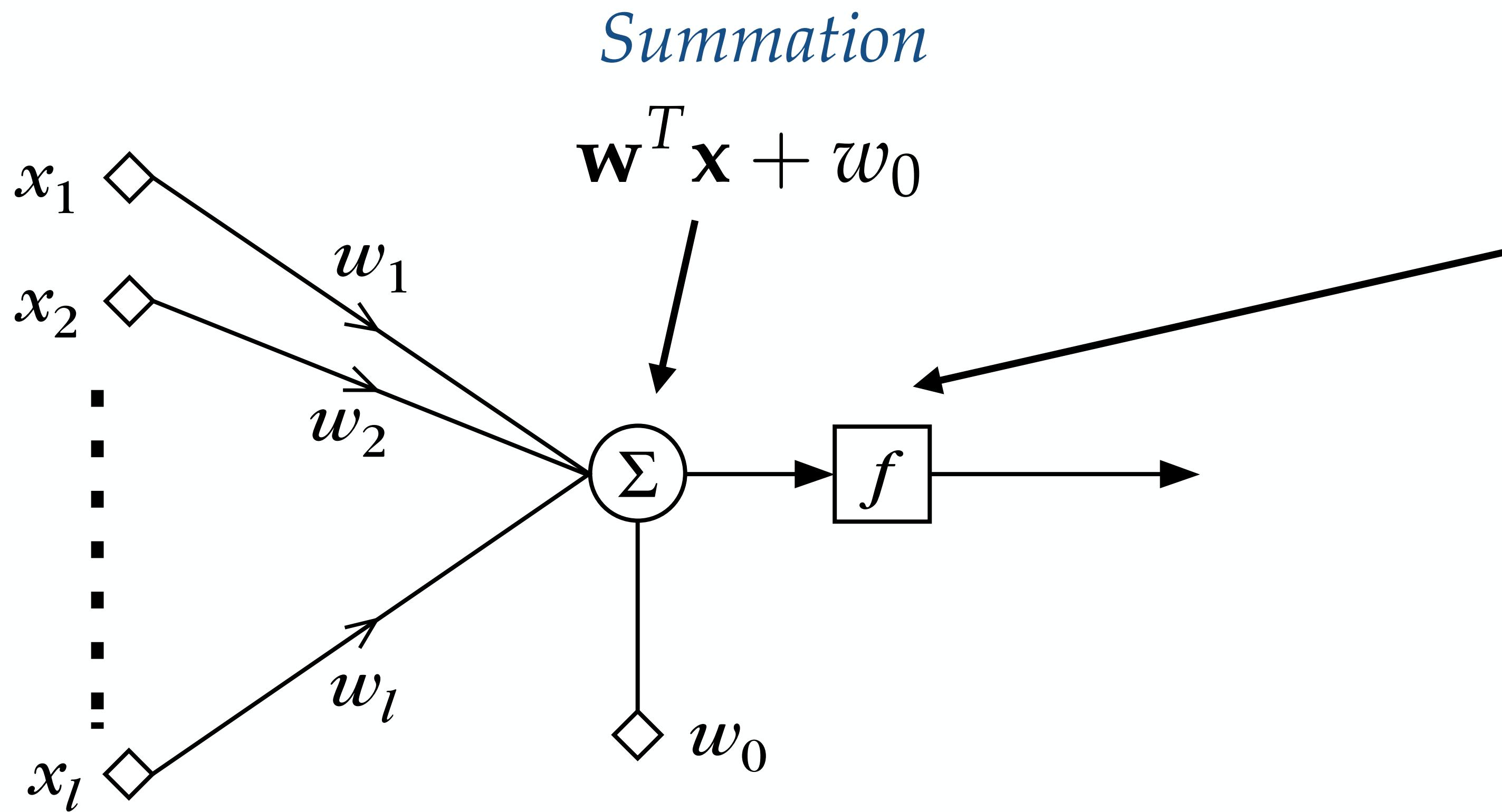
When fully developed, the Perceptron will be designed to remember images and information it has perceived itself, whereas ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons, Dr. Rosenblatt said, will be able to recognize people and call out their names. Printed pages, longhand letters and even speech commands are within its reach. Only one more step of development, a difficult step, he said, is needed for the device to hear speech in one language and instantly translate it to speech or writing in another language.

### Self-Reproduction

In principle, Dr. Rosenblatt said, it would be possible to build Perceptrons that could reproduce them-

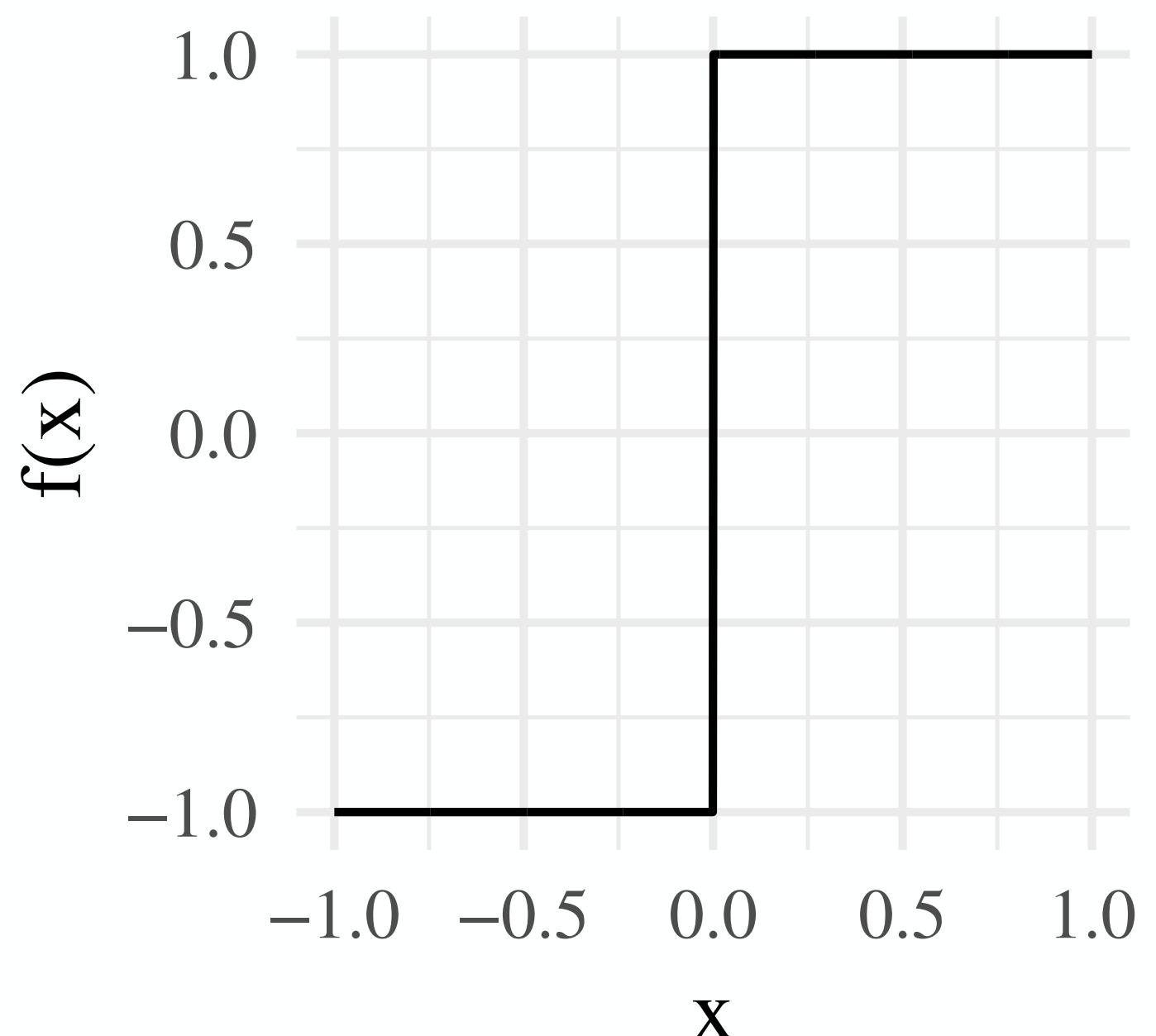
# Perceptron



*Activation function*

Perceptrons use a step function

$$f(x) = 2(\mathbb{I}(x > 0) - 0.5)$$



Inspired by a simplified model of neurons in the brain

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# Perceptron Math

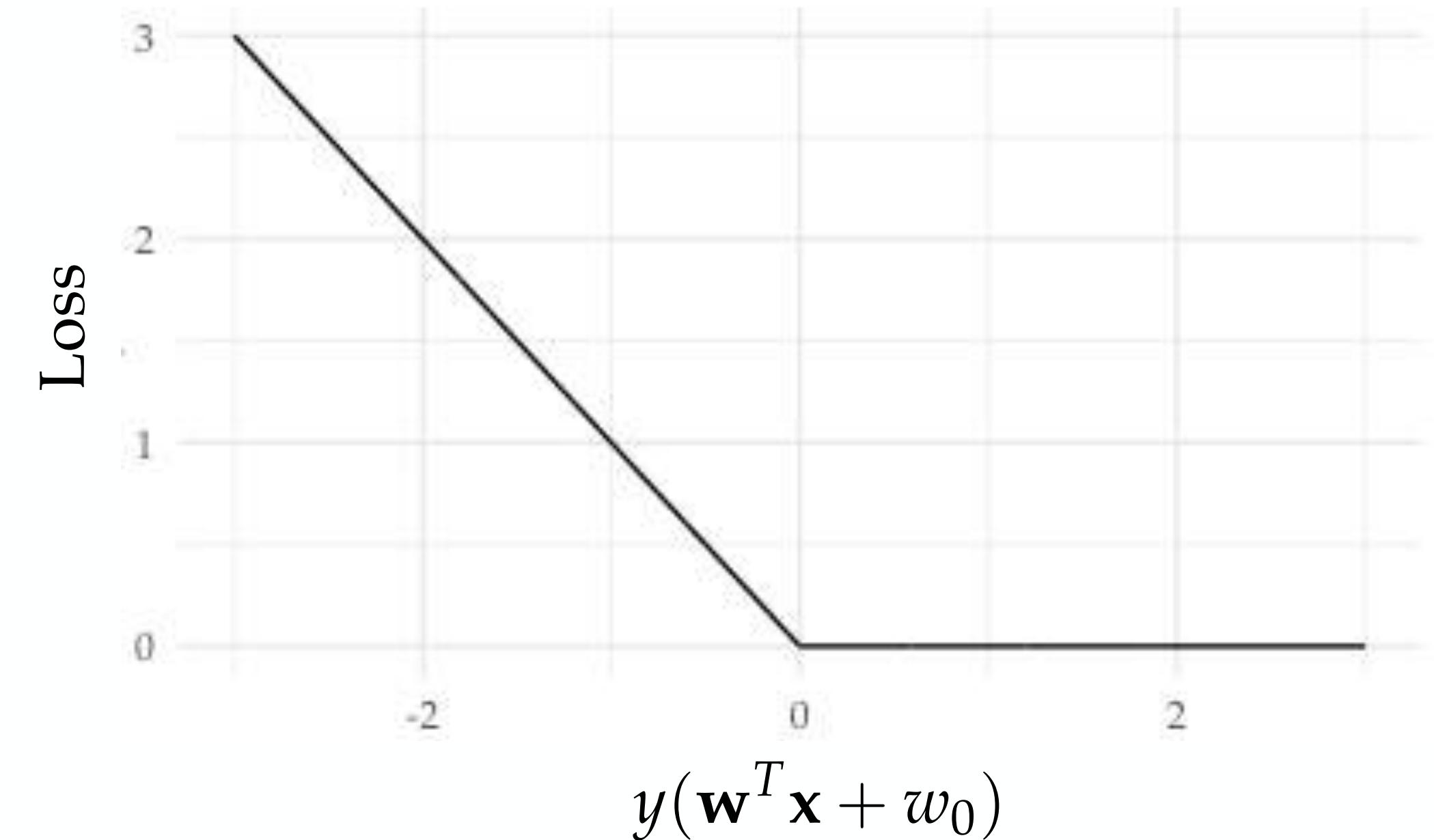
$$y_i \in \{-1, +1\}$$

$$\sum_{i=1}^N \max(-y_i(\mathbf{w}^T \mathbf{x}_i + w_0), 0)$$

Loss: perceptron loss

Function class: linear

Optimizer: (stochastic) gradient descent



# Perceptron Training

$$\sum_{i=1}^N \max(-y_i(\mathbf{w}^T \mathbf{x}_i + w_0), 0)$$

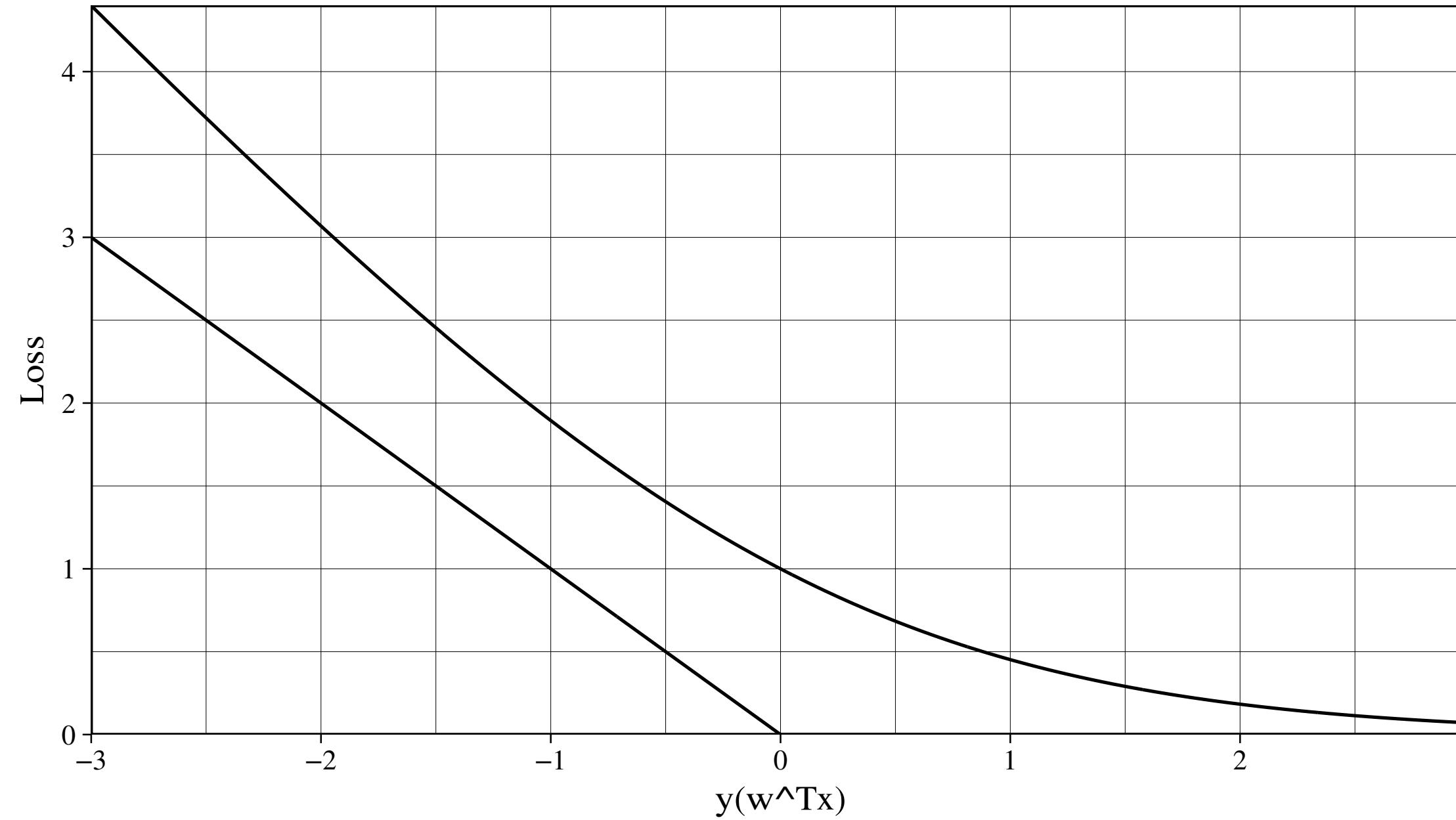
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{y_i(\mathbf{w}^T \mathbf{x}_i) < 0} -y_i \mathbf{x}_i$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \alpha_t \sum_{y_i(\mathbf{w}^{tT} \mathbf{x}_i) < 0} y_i \mathbf{x}_i$$

Guaranteed to converge if:

1. Problem is linearly separable
2. We choose the right, decreasing, learning rate

# *Logistic Regression*

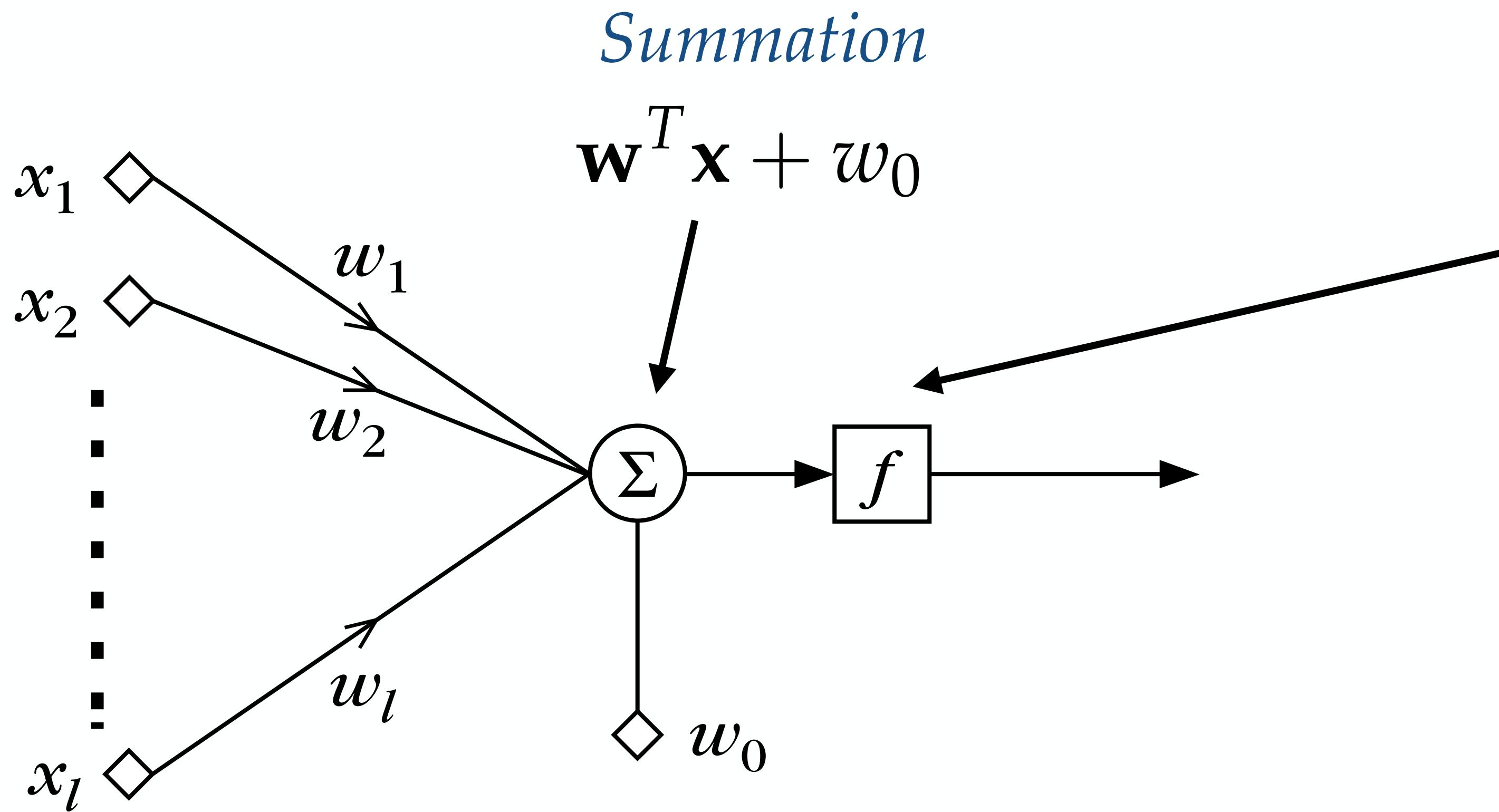


$$\sum_{i=1}^N \frac{1}{\log(2)} \log(1 + \exp[-y_i(\mathbf{w}^T \mathbf{x}_i + w_0)])$$
$$\sum_{i=1}^N \max(-y_i(\mathbf{w}^T \mathbf{x}_i + w_0), 0)$$

We have seen something similar: logistic regression

But: different “activation function” and different loss

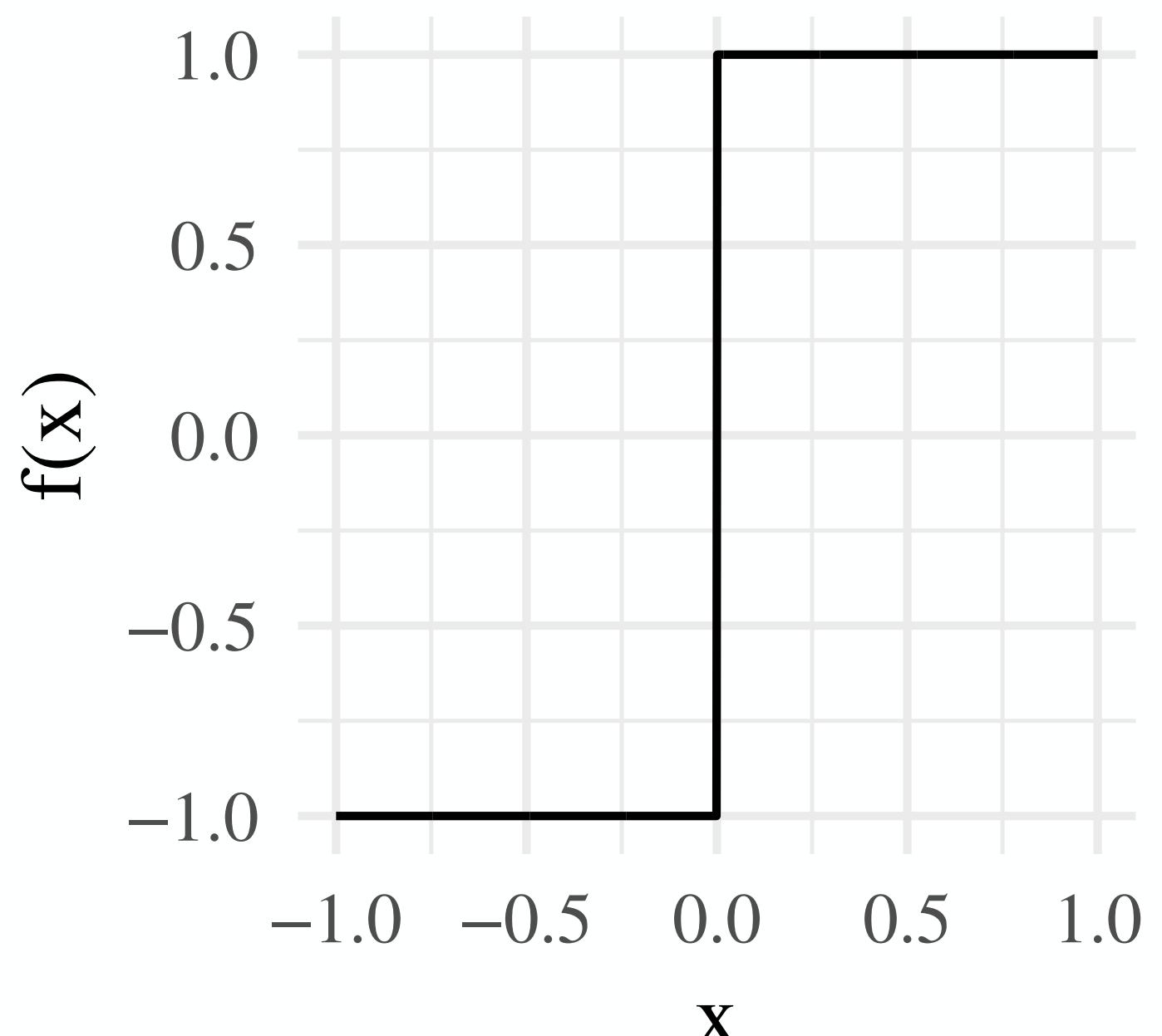
# Perceptron



*Activation function*

Perceptrons use a step function

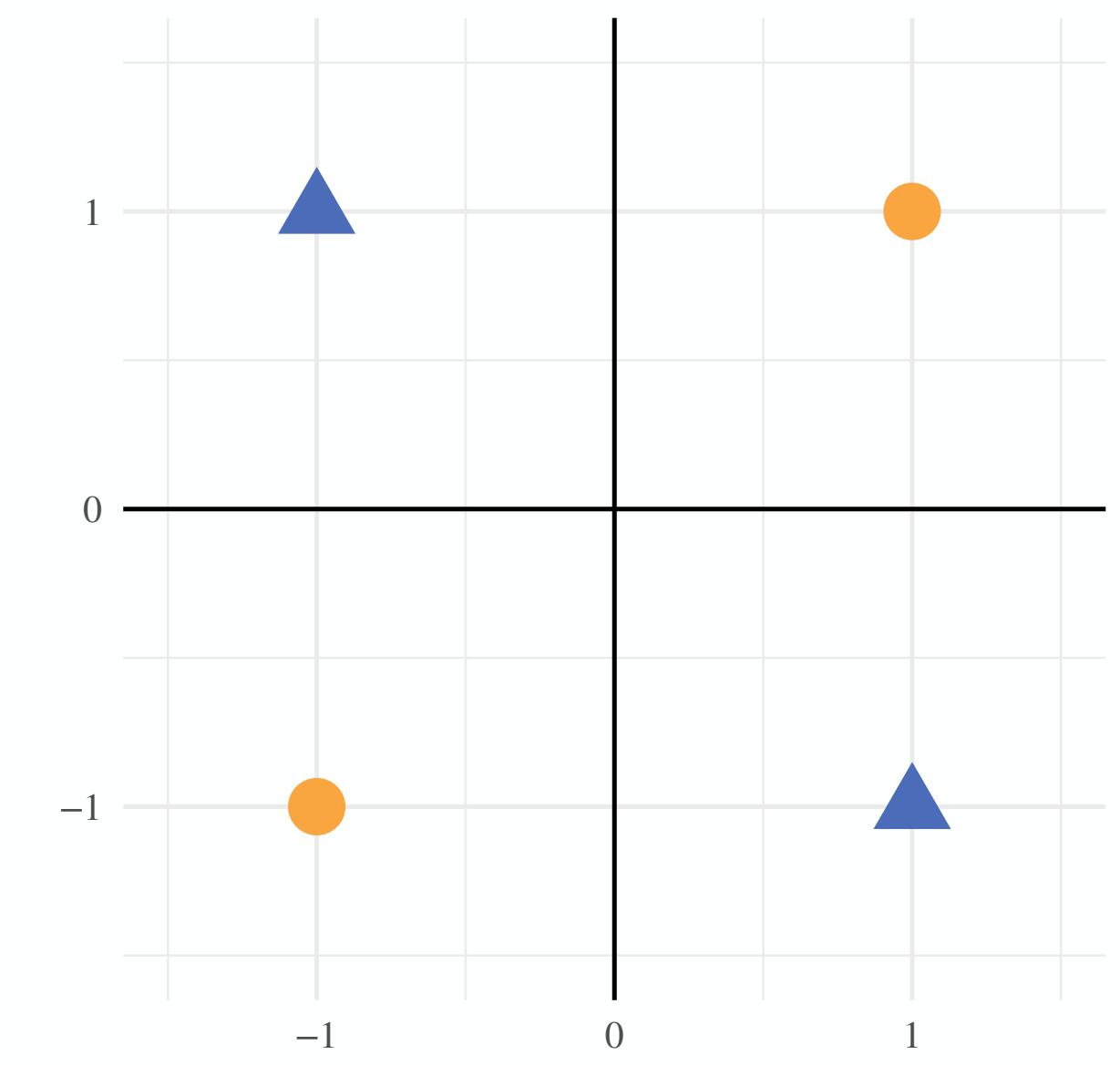
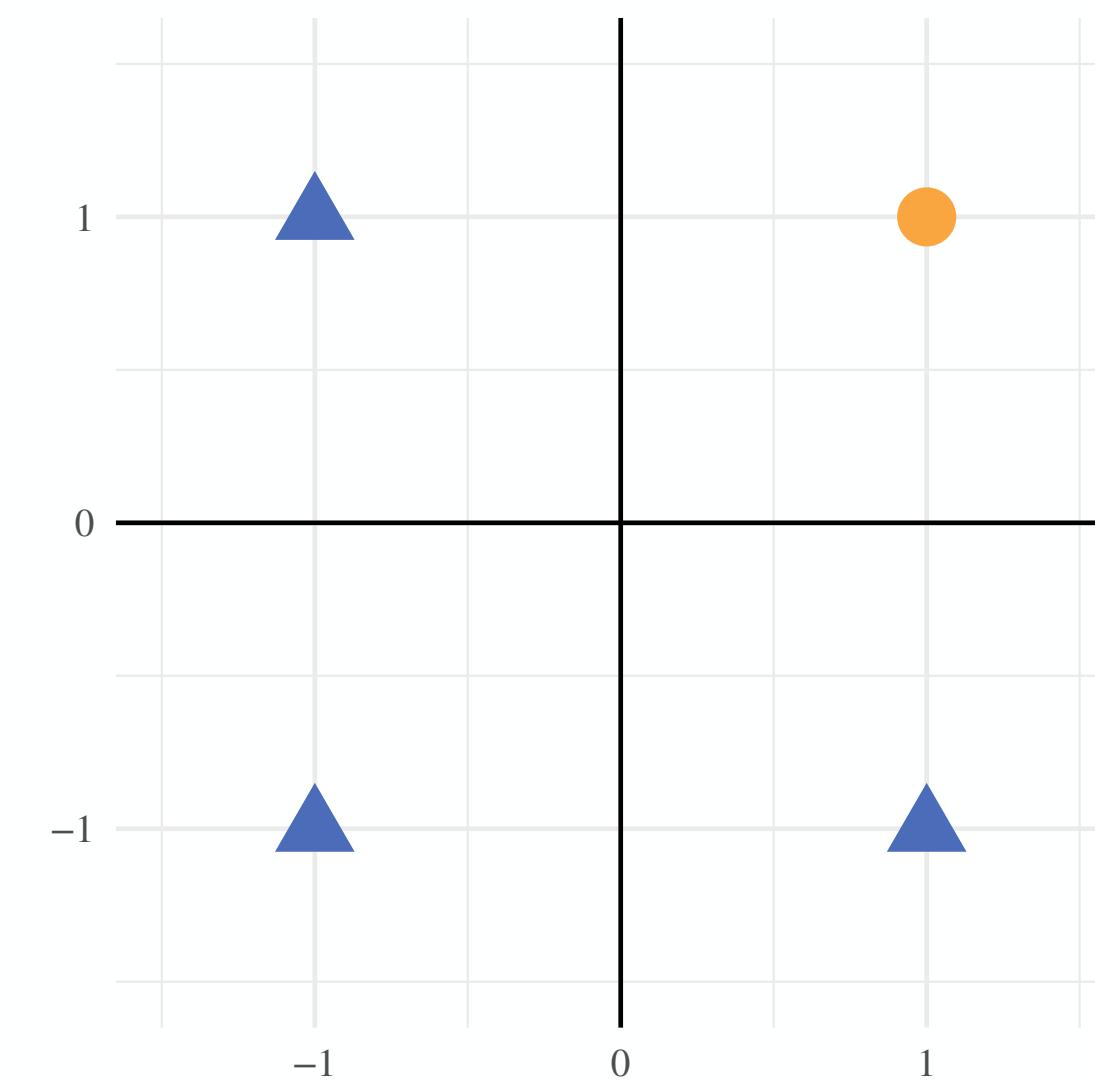
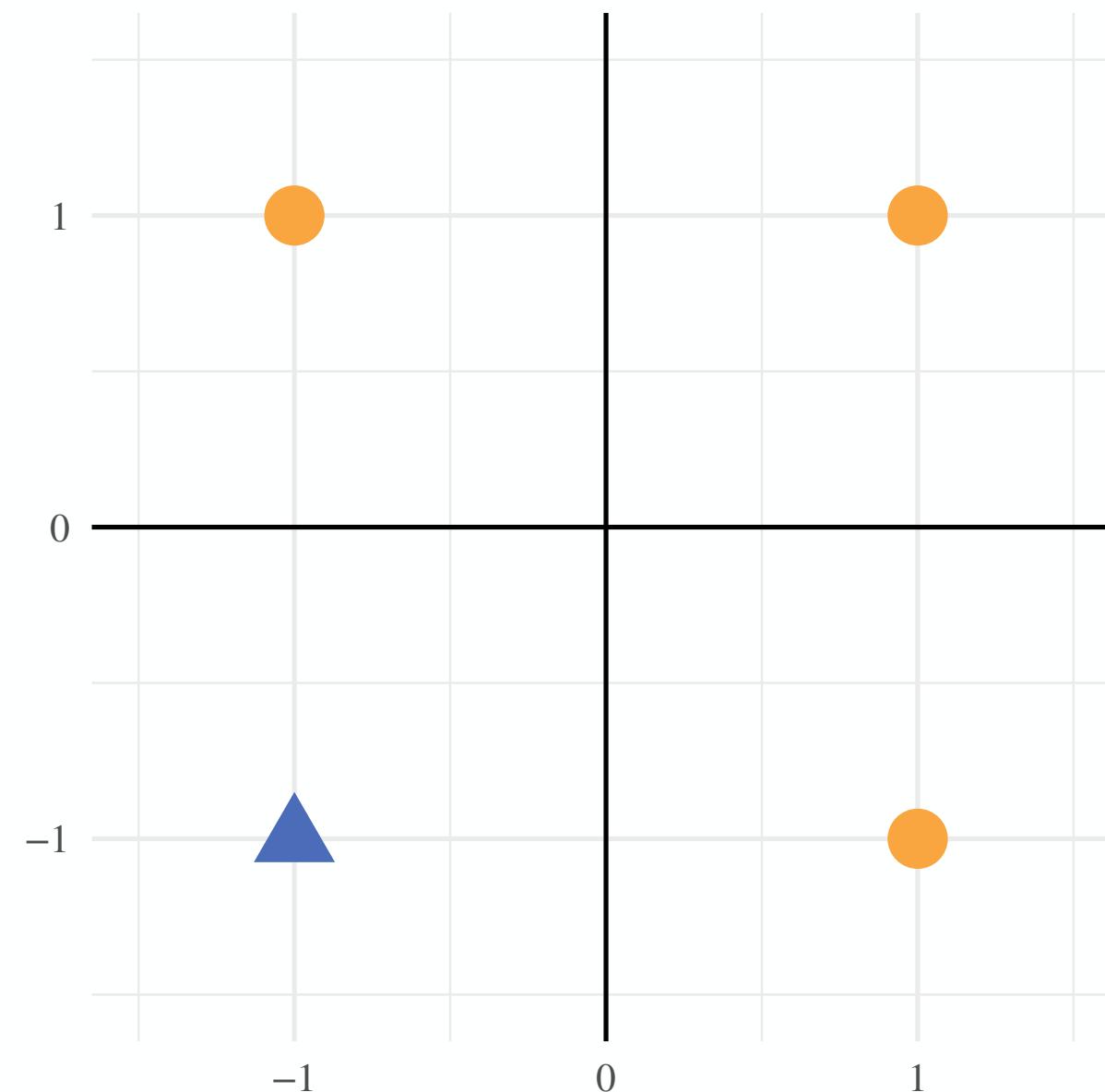
$$f(x) = 2(\mathbb{I}(x > 0) - 0.5)$$



Inspired by a simplified model of neurons in the brain

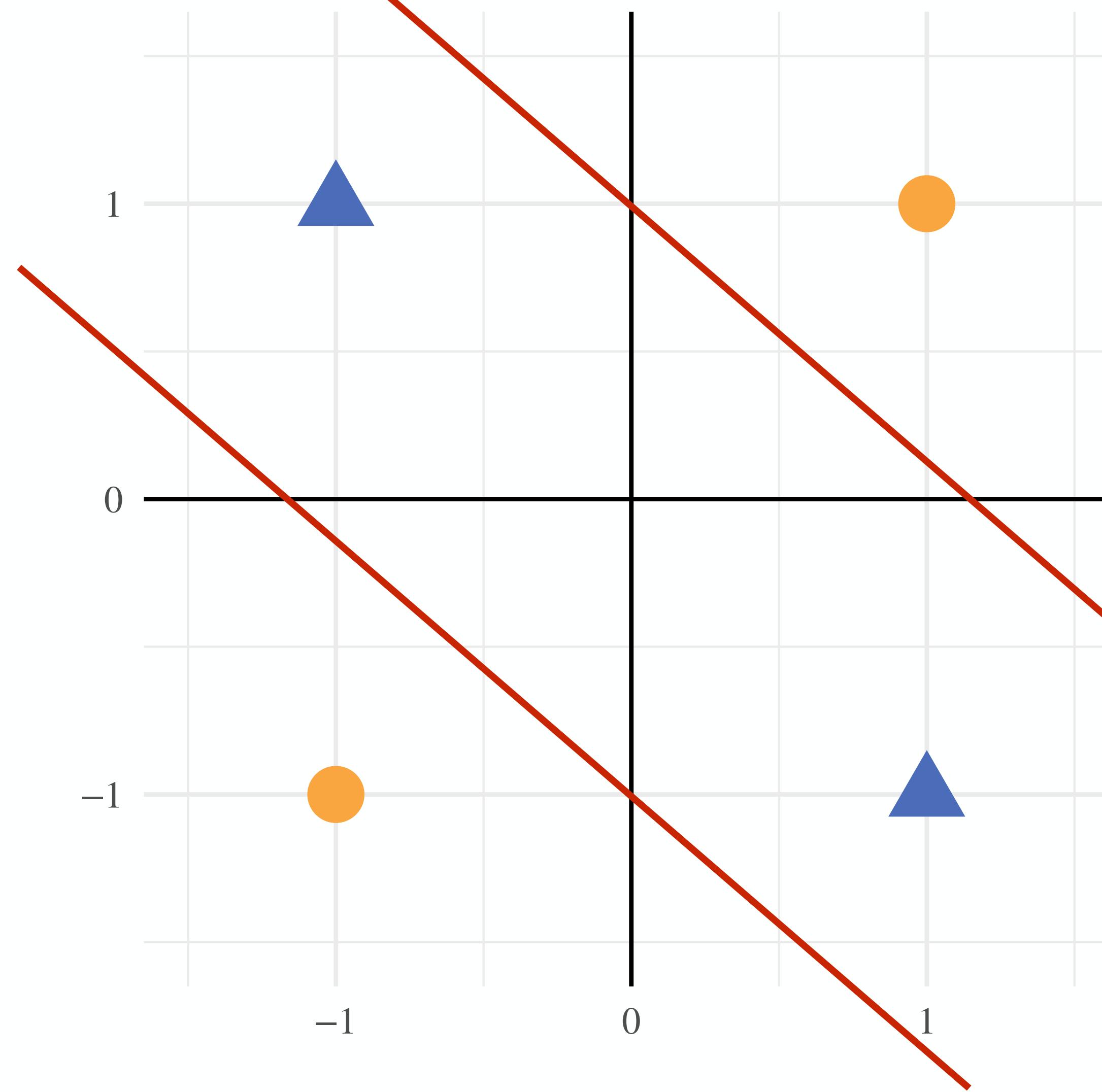
# *Question: Learning logic gates*

- What logic functions can be learned by the perceptron?
  - OR?
  - NOR?
  - AND?
  - NAND?
  - XOR?

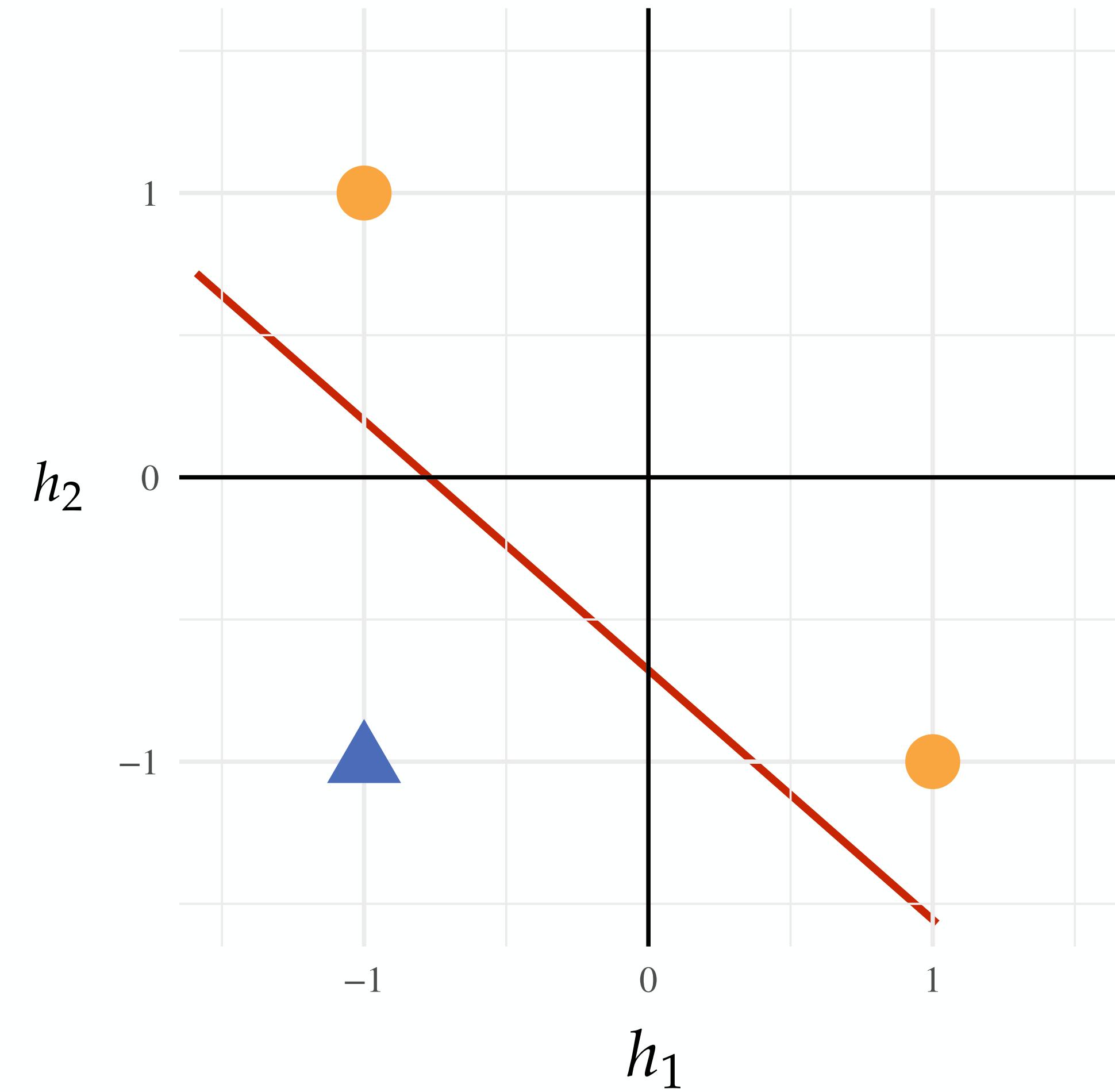
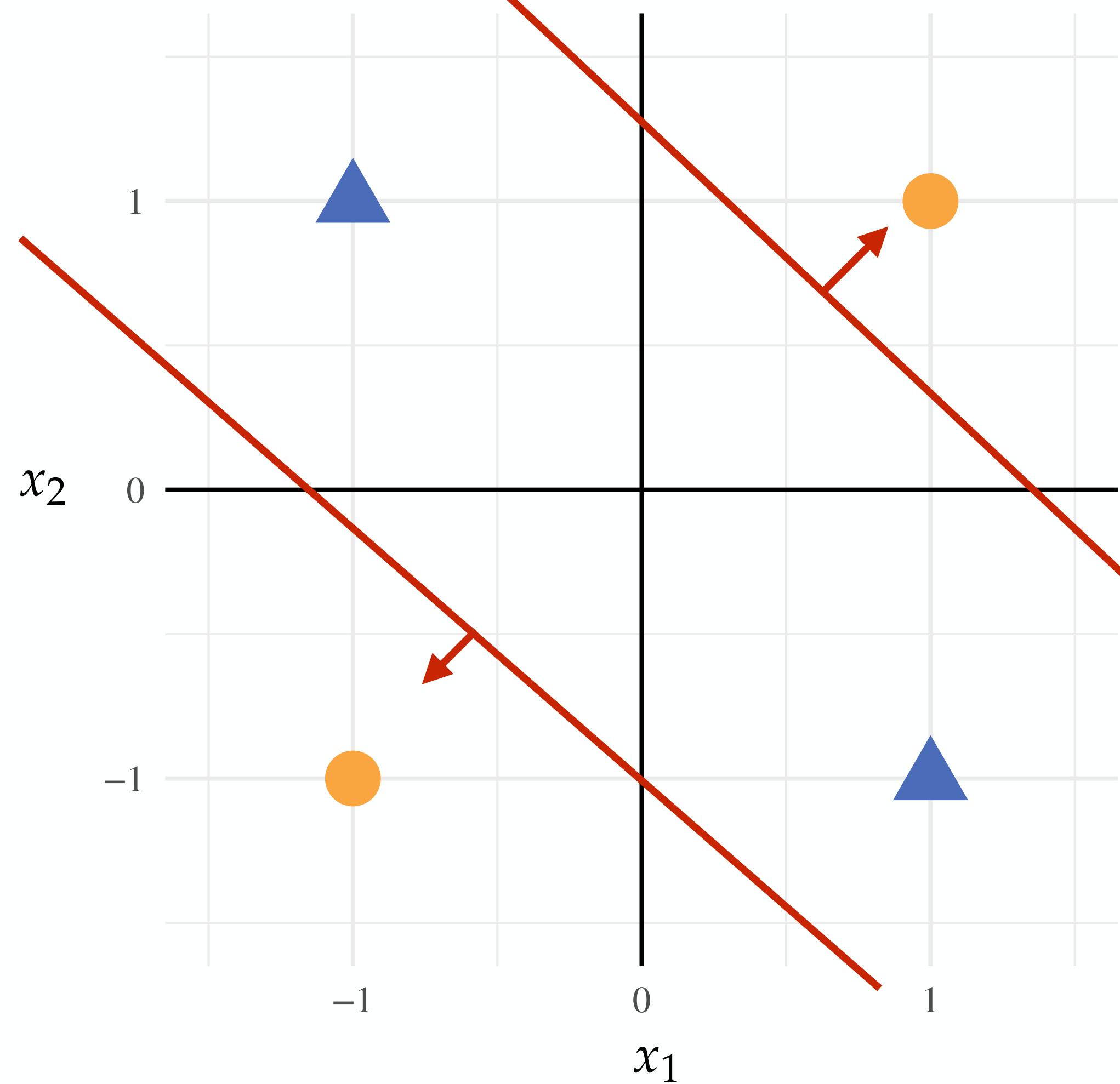


# MULTI-LAYER PERCEPTRONS

# *XOR problem*

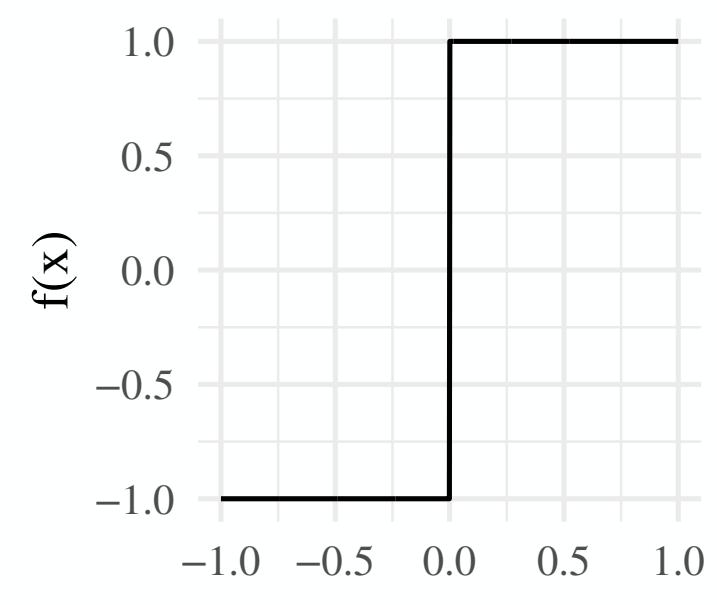
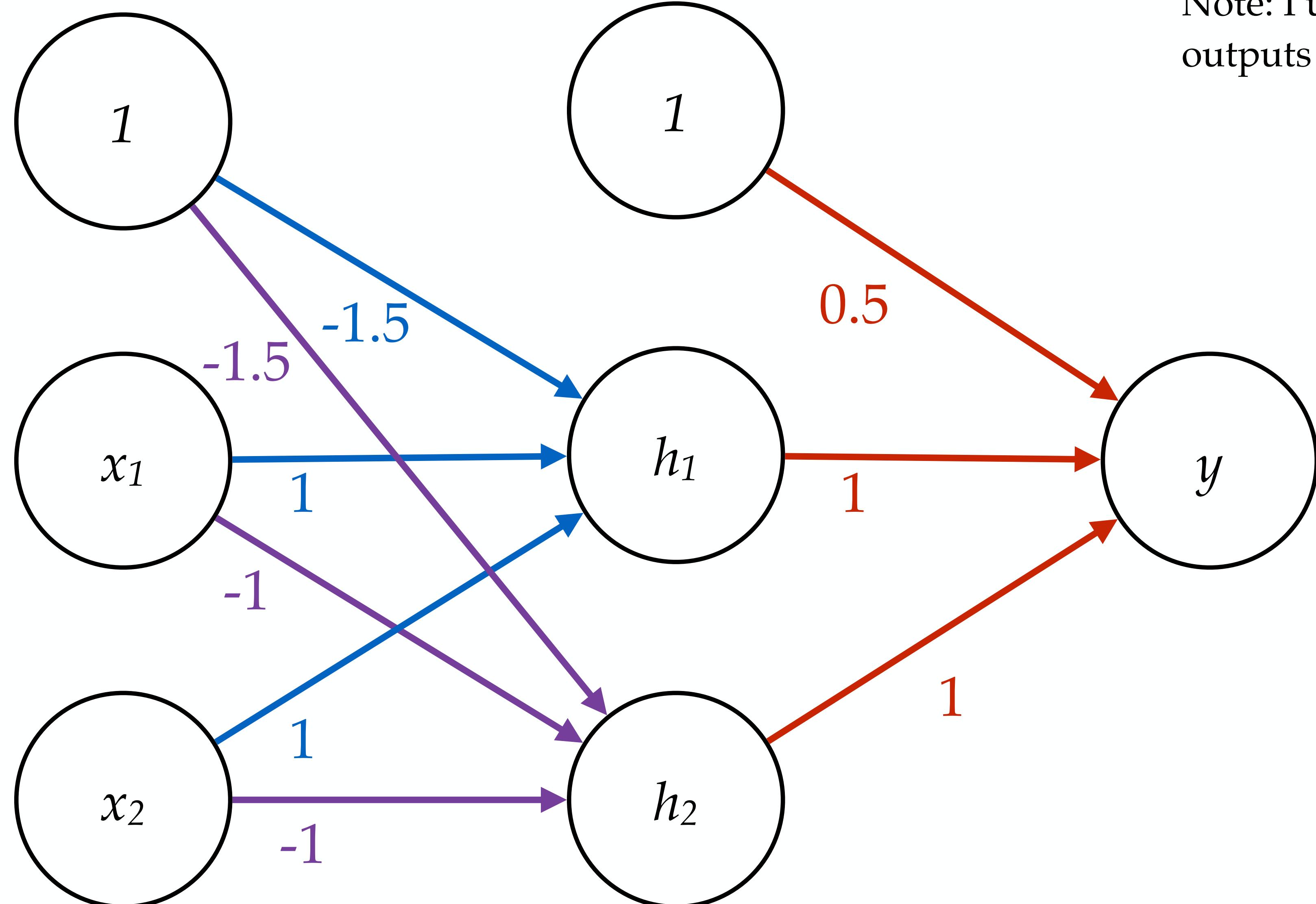


What if we use two classifiers?

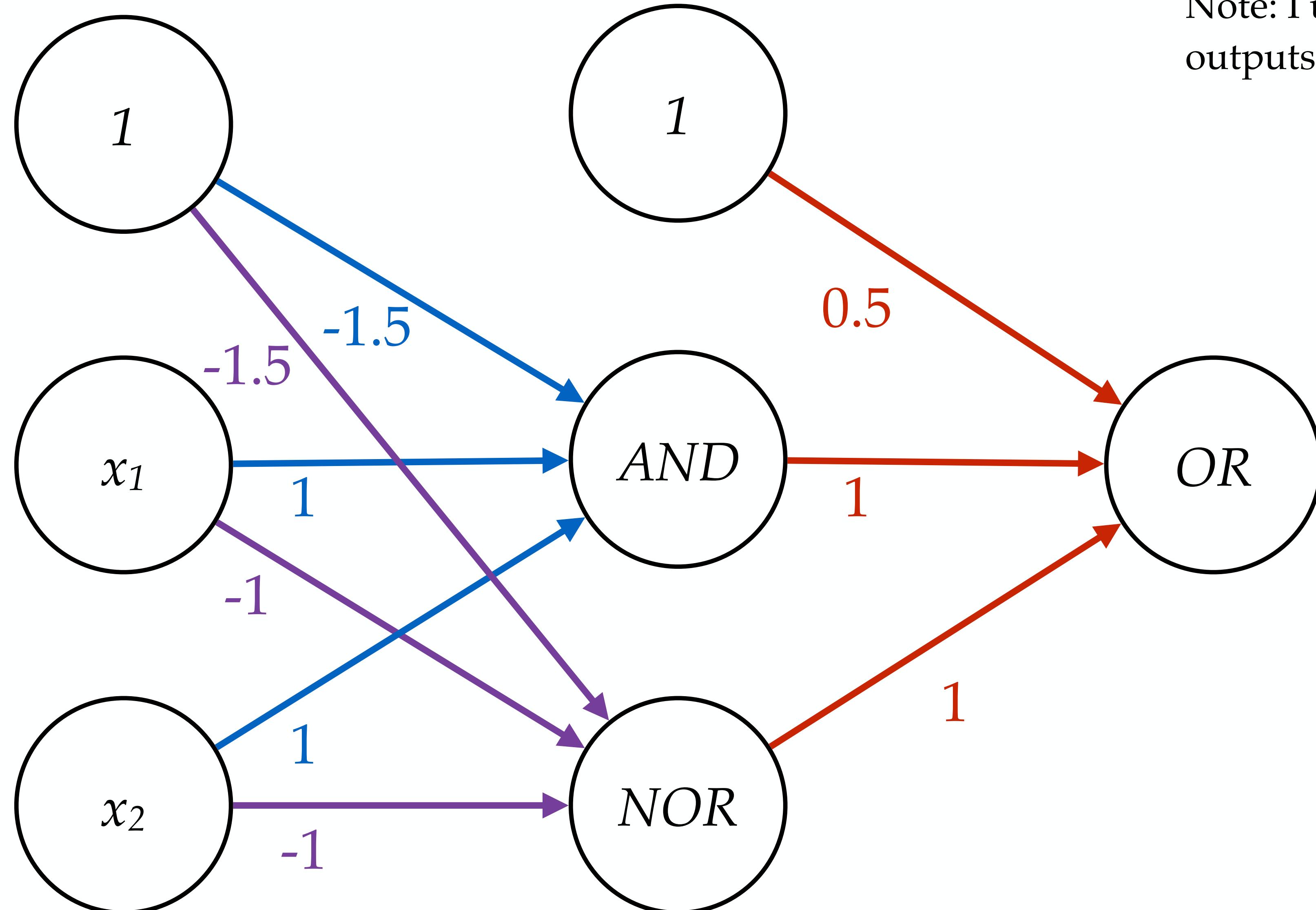


Use two classifiers and consider their outputs as the inputs to a third classifier

Note: I use  $\{-1,+1\}$  values as outputs here instead of  $\{0,1\}$

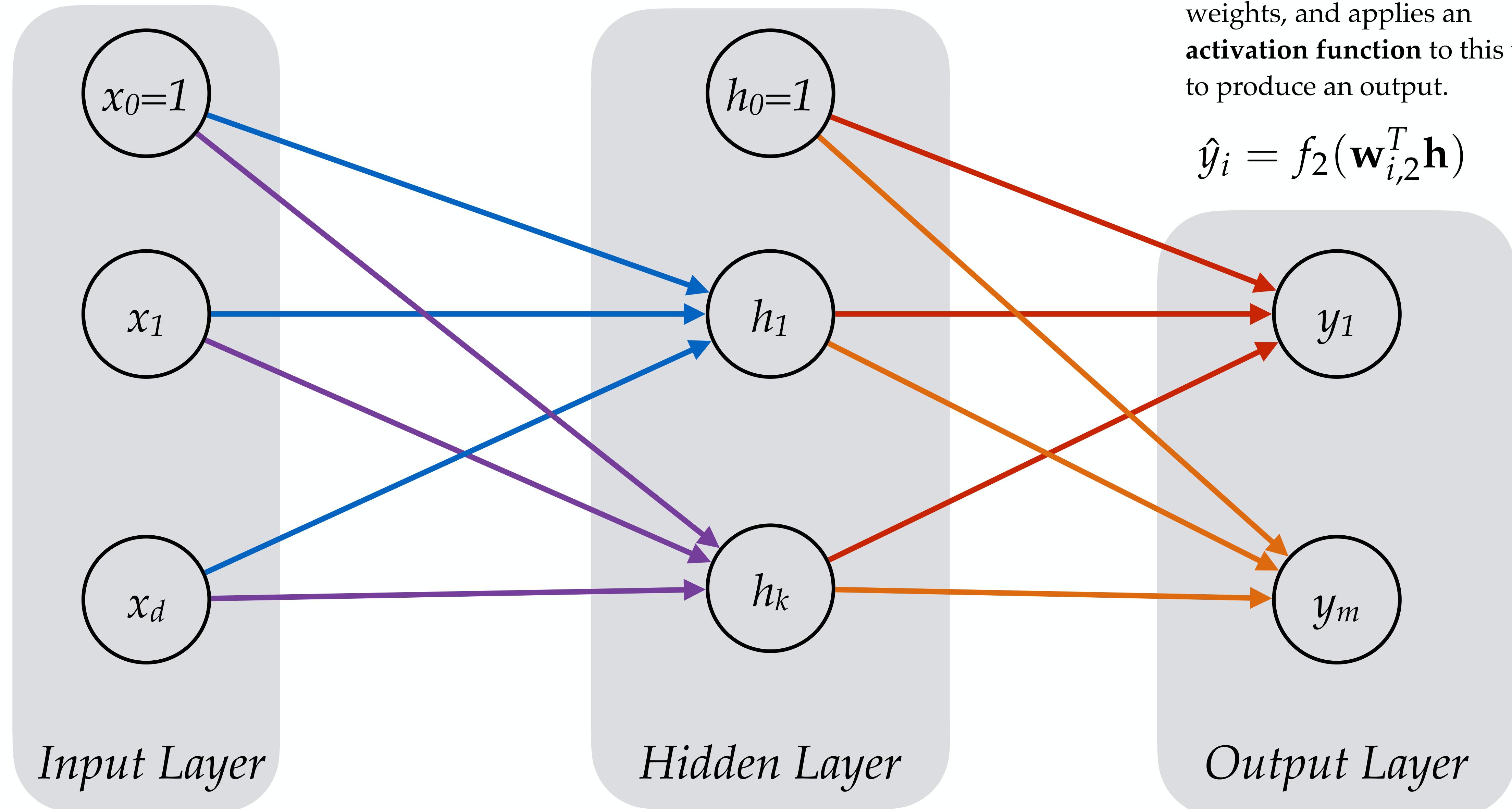


Note: I use  $\{-1,+1\}$  values as outputs here instead of  $\{0,1\}$



*Intuition using logic gates*

# Terminology



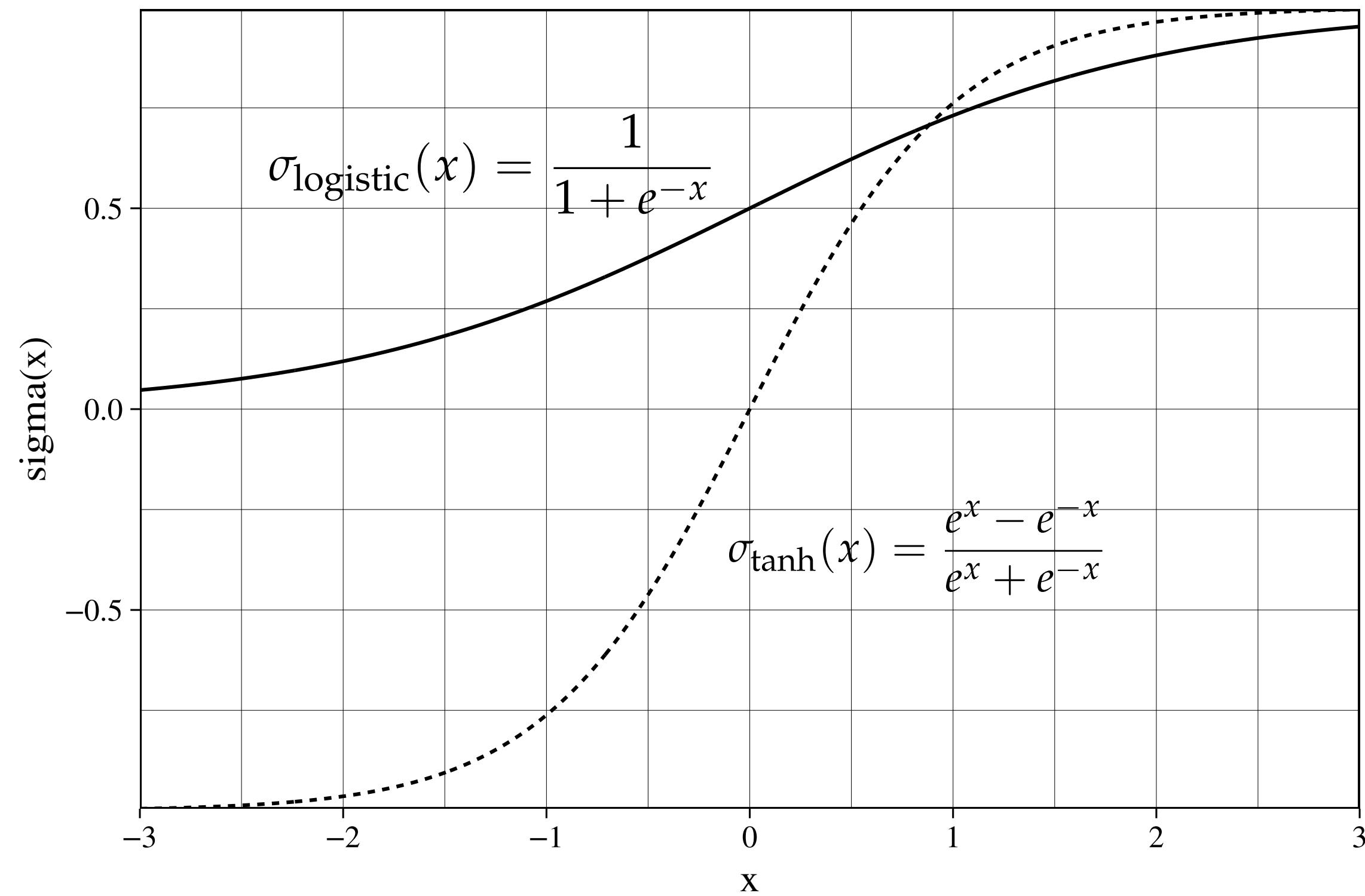
$$h_i = f_1(\mathbf{w}_{i,1}^T \mathbf{x})$$

$$\hat{y}_i = f_2(\mathbf{w}_{i,2}^T \mathbf{h})$$

Each **neuron** takes the linear combination of the previous layer's output using its own weights, and applies an **activation function** to this value to produce an output.

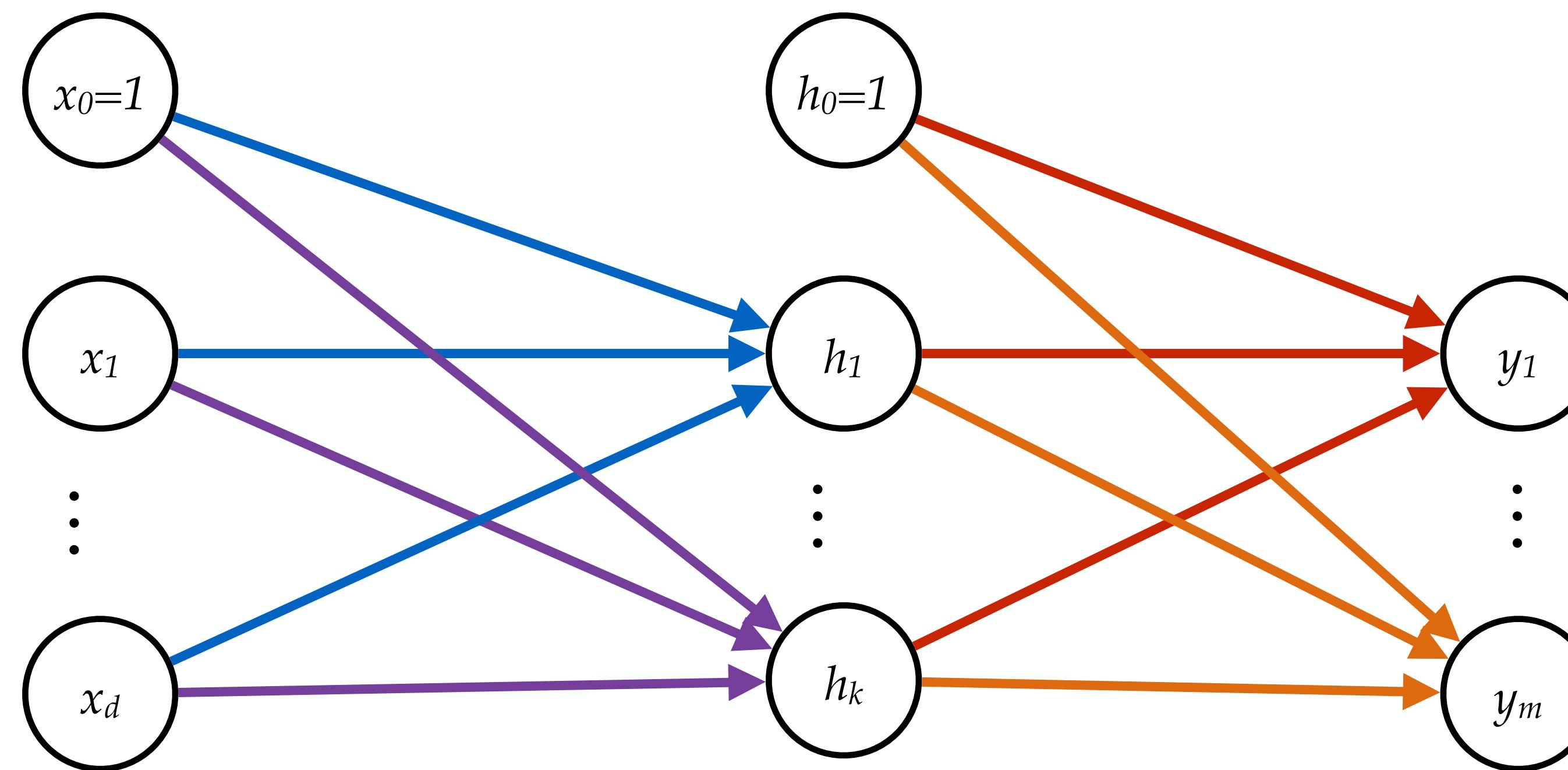
# *Multi-layer Perceptron*

- Confusing naming: MLP as a general term
- Other activation functions:
  - Sigmoid (logistic)
  - Tanh
  - Modern alternatives: ReLU and its variants



*Note: tanh is a rescaled logistic*

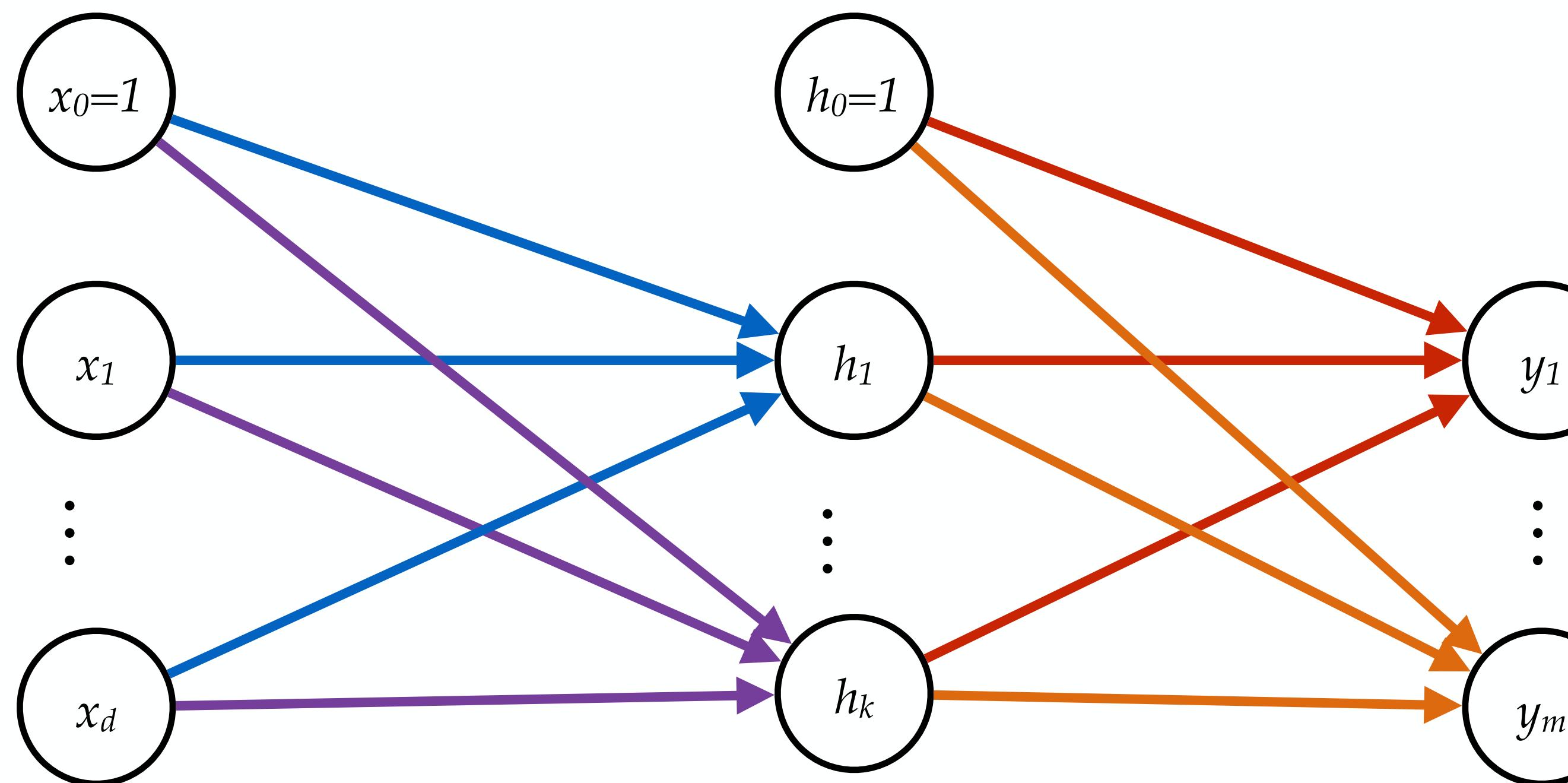
# *Multi-layer Perceptron*



Can describe complex functions, but how do we

1. calculate predictions
2. learn the parameter weights?

# *Prediction: Forward Pass*



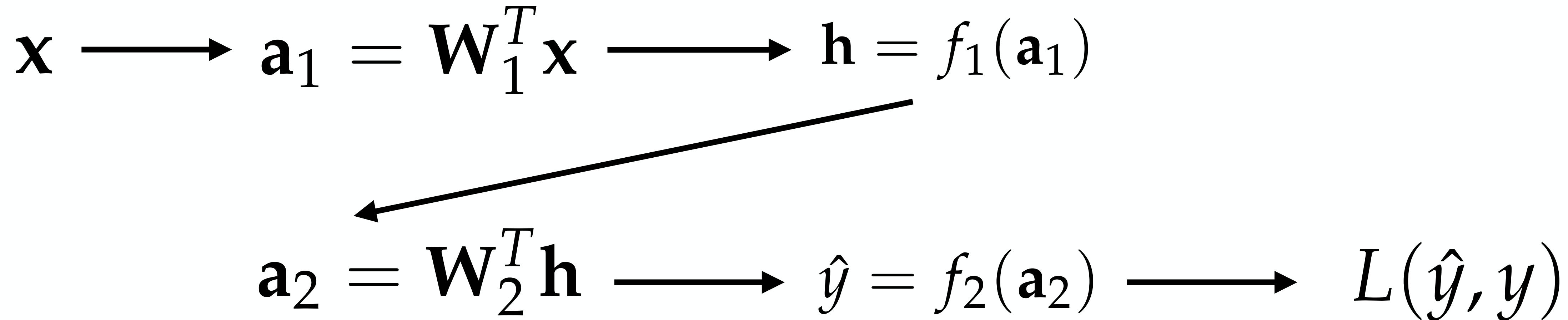
$$\mathbf{h} = f_1(\mathbf{W}_1^T \mathbf{x})$$

$$\hat{\mathbf{y}} = f_2(\mathbf{W}_2^T \mathbf{h})$$

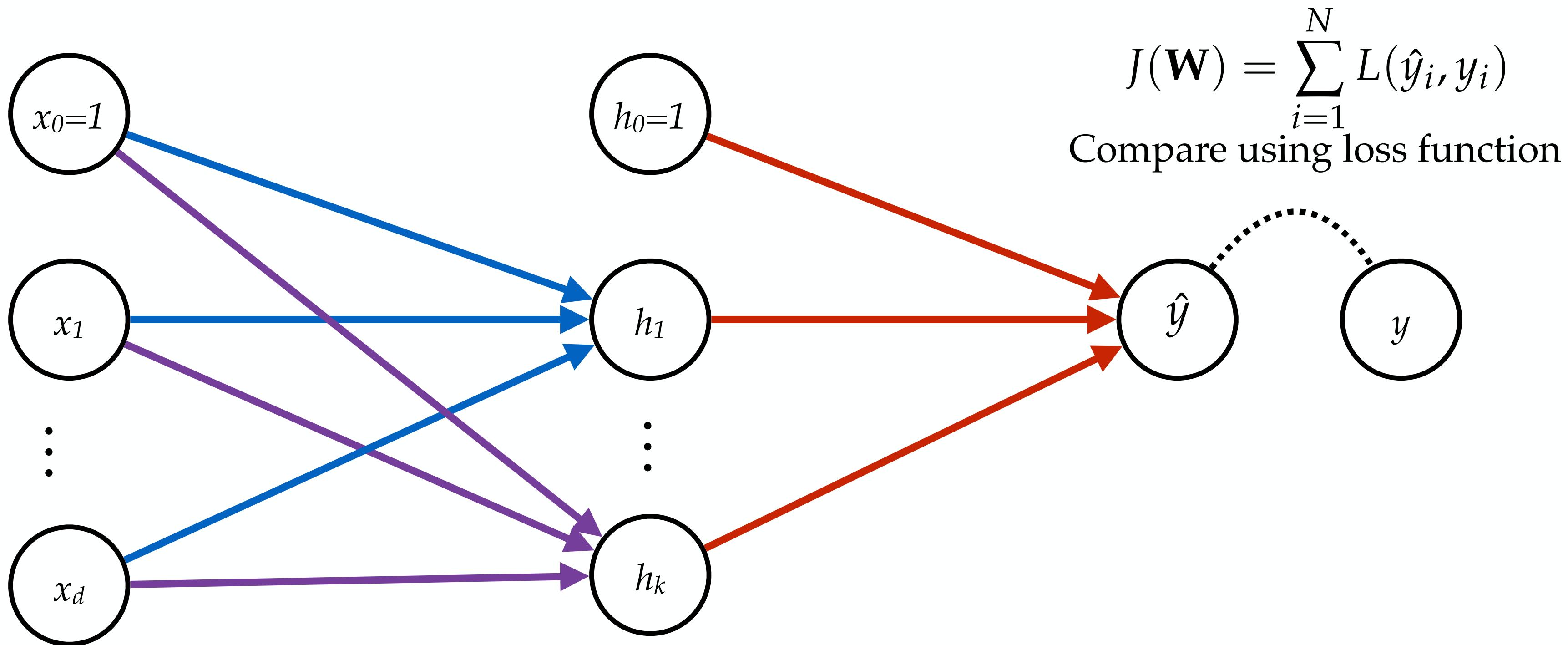
Given a model, we can do prediction/inference in a *forward pass*

## *Objective function*

$$J(\mathbf{W}) = \sum_{i=1}^N L(f_2(\mathbf{W}_2^T f_1(\mathbf{W}_1 \mathbf{x}_i)), y_i)$$



# Why learning is hard



If the output is wrong, how should we update the model? Which part of the error should we attribute to the different weights?

# *Learning the MLP weights*

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha_t \nabla_{\mathbf{w}} J(\mathbf{w}^t)$$

- Remember the general setup: we are again going to use gradient descent. So we need the gradient
- Intuitively, what the gradient will do is propagate the error at the end of the network, back across the graph to tell us how to change the weights.
- This clever way of efficiently calculating the gradient is called “back propagation”

*Gradient!!??*

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_n} \end{bmatrix}$$

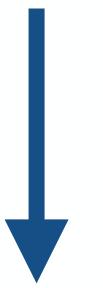
# Backpropagation

$$J(\mathbf{w}_2, \mathbf{W}_1) = \sum_{i=1}^N L(f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1 \mathbf{x}_i)), y_i)$$



*Let's consider a single training object first*

$$J(\mathbf{w}_2, \mathbf{W}_1) = L(f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1 \mathbf{x})), y)$$



We want the gradient of this, w.r.t. both  $\mathbf{w}_2$  and  $\mathbf{W}_1$

Find:

$$\nabla_{\mathbf{w}_2} J(\mathbf{w}_2, \mathbf{W}_1) \quad \& \quad \nabla_{\mathbf{W}_1} J(\mathbf{w}_2, \mathbf{W}_1)$$

# *Backpropagation (1D case)*

$$J(w_1, w_2) = L(f_2(w_2 f_1(w_1 x)), y)$$

$$\frac{\partial J(w_1, w_2)}{\partial w_2} = ?$$

$$\frac{\partial J(w_1, w_2)}{\partial w_1} = ?$$

## *Backpropagation (1D case)*

$$J(w_1, w_2) = L(f_2(w_2 f_1(w_1 x)), y)$$

$$\frac{\partial J(w_1, w_2)}{\partial w_2} = L'(f_2(w_2 f_1(w_1 x)), y) f'_2(w_2 f_1(w_1 x)) f_1(w_1 x)$$

$$\frac{\partial J(w_1, w_2)}{\partial w_1} = L'(f_2(w_2 f_1(w_1 x)), y) f'_2(w_2 f_1(w_1 x)) w_2 f'_1(w_1 x) x$$

# Backpropagation

It is important you understand the intuition here: why this is called back propagation and how it relates to the gradient.

$$J(\mathbf{w}_2, \mathbf{W}_1) = L(f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1 \mathbf{x})), y)$$



$$\begin{aligned} (\nabla_{\mathbf{w}_2} J(\mathbf{w}_2, \mathbf{W}_1))^T &= \frac{\partial L(y, \hat{y}_i)}{\partial \hat{y}_i} \cdot [\nabla_{\mathbf{w}_2} f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1^T \mathbf{x}))]^T \\ &= \frac{\partial L(y, \hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1^T \mathbf{x}))}{\partial \mathbf{w}_2^T f_1(\mathbf{W}_1^T \mathbf{x})} \cdot [f_1(\mathbf{W}_1^T \mathbf{x})]^T \end{aligned}$$

# Backpropagation

$$J(\mathbf{w}_2,\mathbf{W}_1) = L(f_2(\mathbf{w}_2^Tf_1(\mathbf{W}_1\mathbf{x})),y)$$

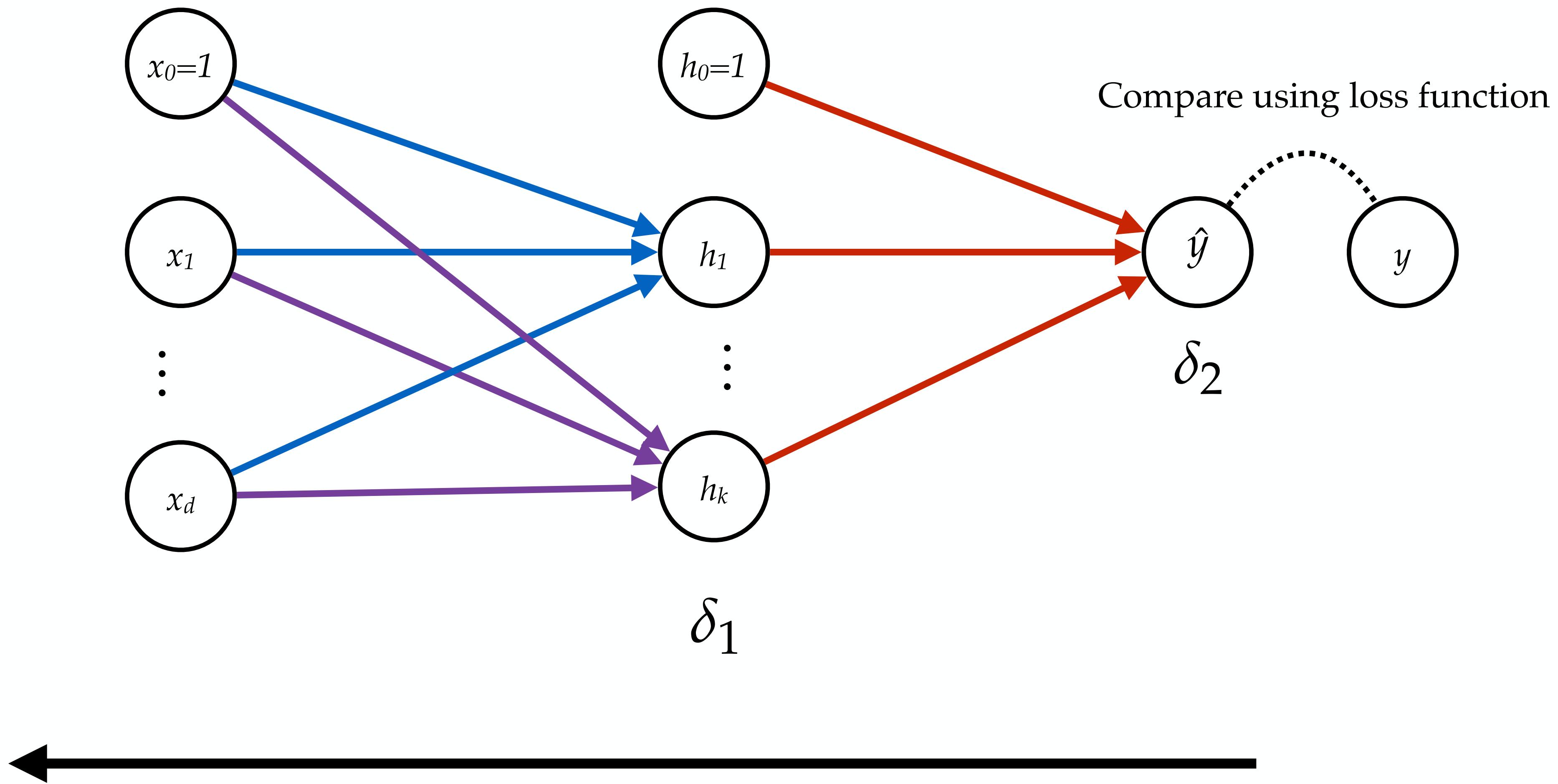
$$(\nabla_{\mathbf{w}_2} J(\mathbf{w}_2,\mathbf{W}_1))^T = \frac{\partial L(y,\hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial f_2(\mathbf{w}_2^Tf(\mathbf{W}_1^T\mathbf{x}))}{\partial \mathbf{w}_2^Tf(\mathbf{W}_1^T\mathbf{x})} \cdot [f_1(\mathbf{W}_1^T\mathbf{x})]^T$$

$$\delta_2$$

$$(\nabla_{\mathbf{w}_1} J(\mathbf{w}_2,\mathbf{W}_1))^T = \frac{\partial L(y,\hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial f_2(\mathbf{w}_2^Tf(\mathbf{W}_1^T\mathbf{x}))}{\partial \mathbf{w}_2^Tf(\mathbf{W}_1^T\mathbf{x})} \cdot \mathbf{w}_2^T \cdot \frac{\partial f_1(\mathbf{W}_1^T\mathbf{x})}{\partial \mathbf{W}_1^T\mathbf{x}} \cdot \begin{bmatrix} \mathbf{x}^T \\ \mathbf{0}^T \\ \cdots \\ \mathbf{0}^T \end{bmatrix}$$

$$\delta_1 = \delta_2 \cdot \mathbf{w}_2^T \cdot f'_1$$

# *Backpropagation*



Propagate the error at the end back through the network

# *Learning the MLP weights*

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha_t \nabla_{\mathbf{w}} J(\mathbf{w}^t)$$

- Remember the general setup: we are again going to use gradient descent. So we need the gradient
- Intuitively, what the gradient will do is propagate the error at the end of the network, back across the graph to tell us how to change the weights.
- This clever way of efficiently calculating the gradient is called “back propagation”

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N L(h(\mathbf{x}_i), y_i)$$

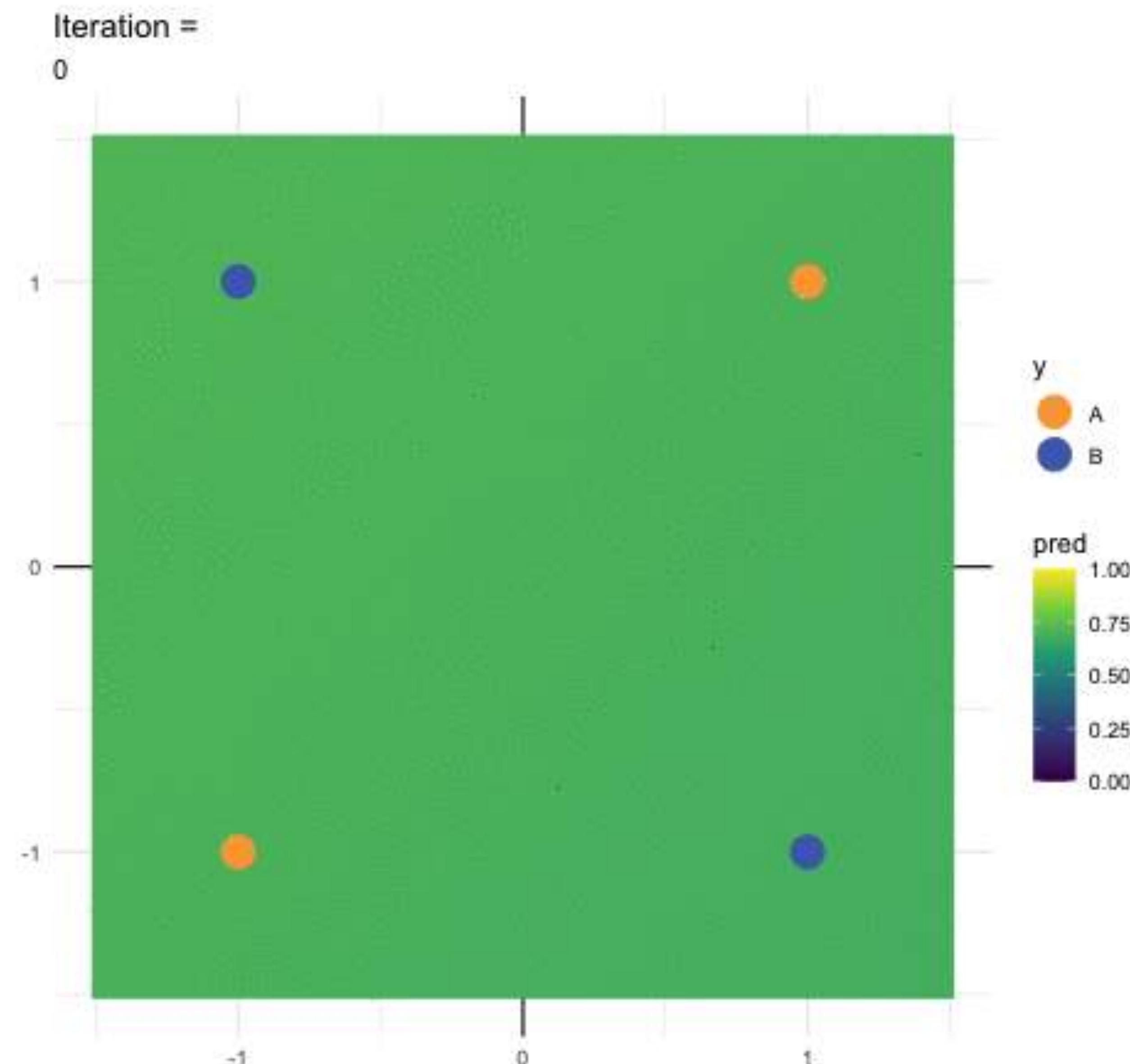
1. Define the class of possible functions

2. Define a cost (loss) function to measure the “quality” of each hypothesis

3. Measure the average cost on the training data

4. Find the function that minimises this cost

# *XOR example fit*



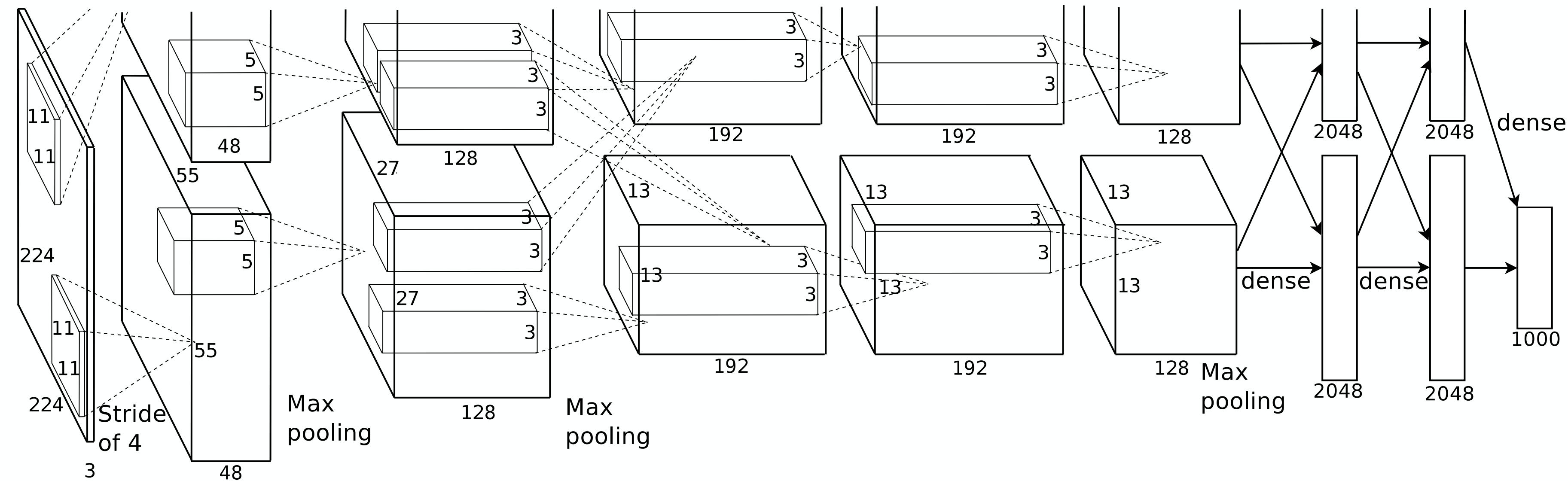
# *Many variations*

- Activation functions and how to combine them
- How many hidden layers?
- How to connect different neurons, share parameters, etc.
- Initialization, optimization procedure, regularization, weight decay,  
...
- These combinations lead to the models that are used today:
  - Convolutional Neural Networks
  - Residual Networks
  - (Transformers)

# *Challenges*

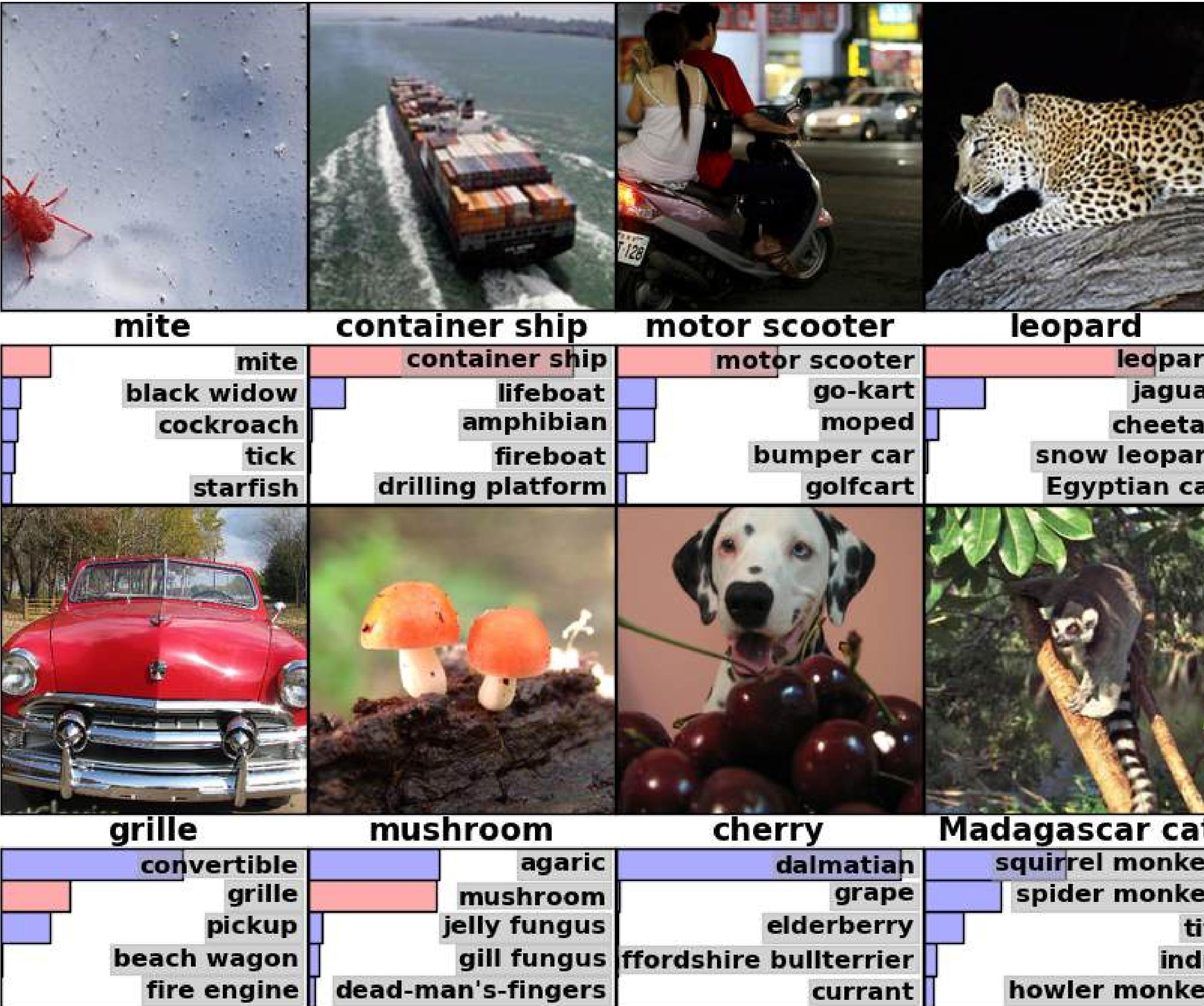
- Non-convex risk function: may get stuck in local optima, converge slowly, etc.
- Many architecture choices: which is optimal?
- Flexible model: risk of overfitting
- There is an *art* to getting networks to work well
- Difficult to interpret the model (black box?)
- With many parameters: computationally demanding (but GPUs, TPUs and other specialised hardware)

# *Example: ImageNet*



1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

# Example: ImageNet



1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

←



ice

×

Dec 10, 2009



Dec 9, 2009



Oct 23, 2009



Oct 11, 2009



Oct 2, 2009



Aug 28, 2009

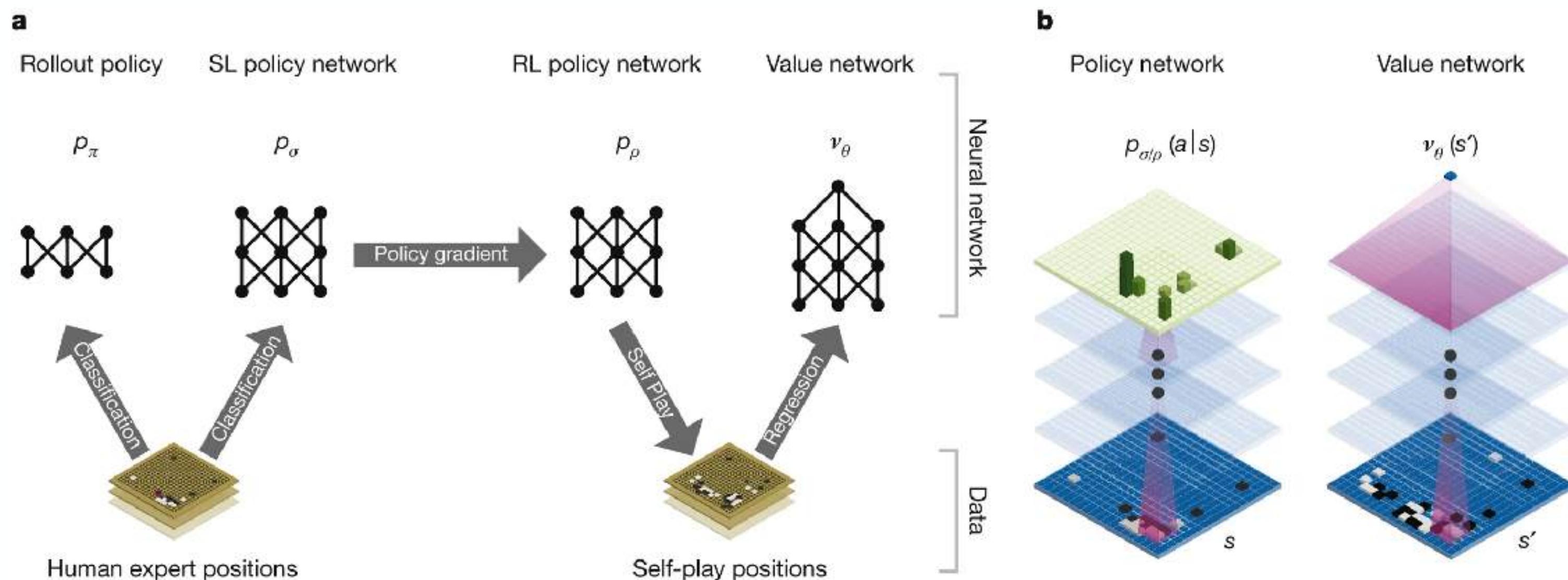
---

*Google Photo search for  
“ice”*

# *Example: Learning Moves in Go*



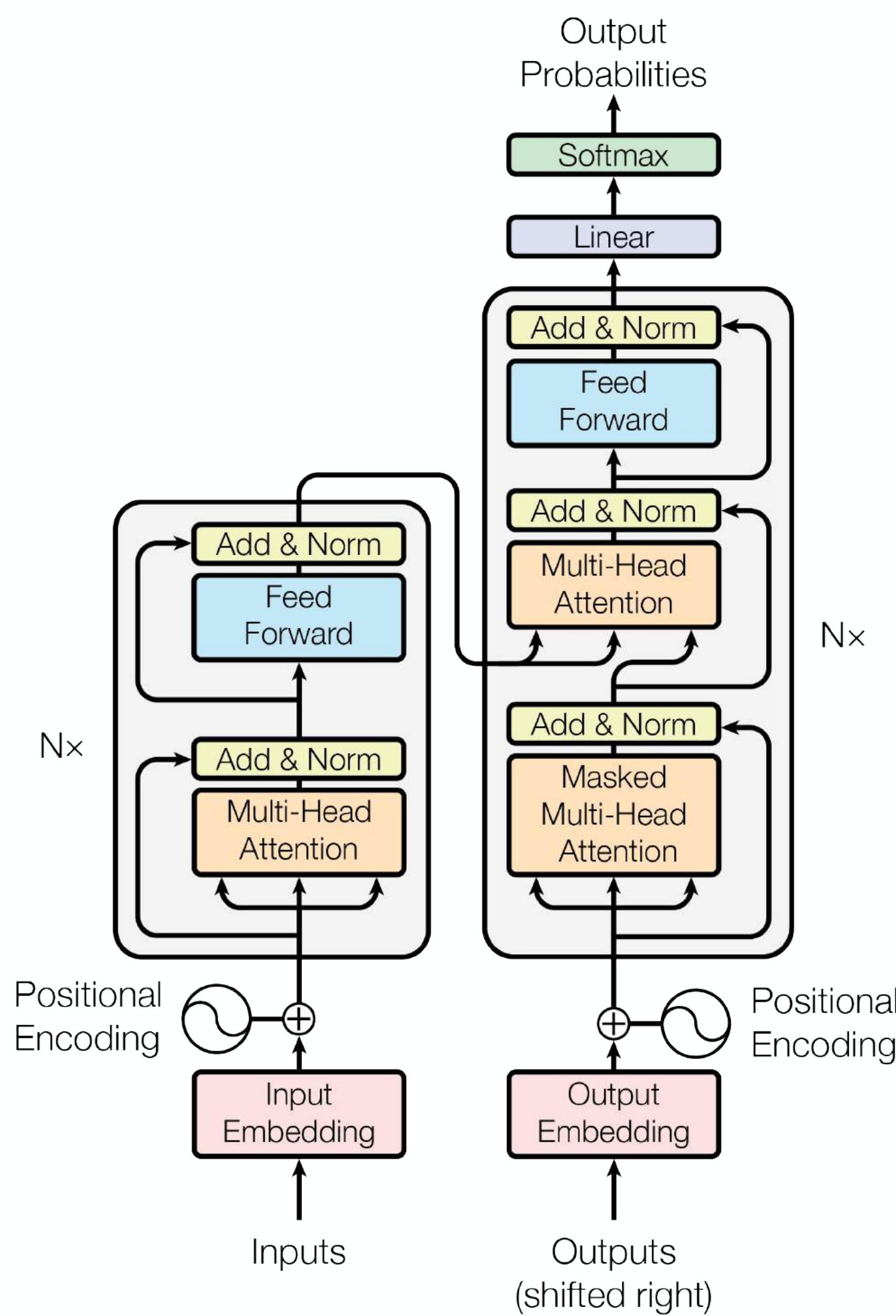
Artificial  
Neural  
Network → Best Next Move



---

*DeepMind's Deep  
Q-Network used to play  
Atari's Breakout game*

# Transformers/GPT



# *Advantages & Disadvantages*

- ✓ Flexible model class ✗ Computationally expensive
- ✓ Good empirical performance on many (structured) problems ✗ Lots of hyper parameters
- ✓ Can be easily adapted to different learning settings ✗ Does not converge to a unique optimum
- 
- ✗ Optimizing them can be an art ✗ Hard to interpret
- ✗ Hard to interpret

# *Recap*

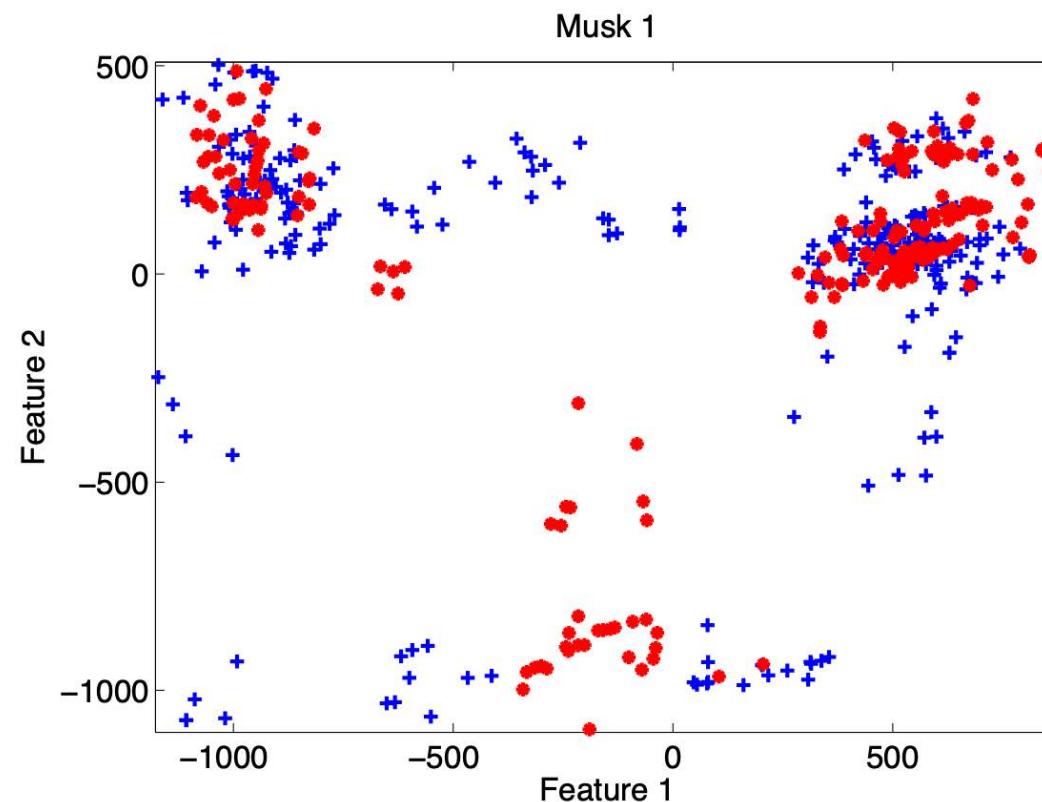
- Perceptrons are simple linear binary classifiers
- Multi-layer perceptrons are connected architectures of perceptron-like nodes, inspired by a simplified model of the brain
- They are trained using (stochastic) gradient descent, efficiently calculating the gradient using back propagation

# Unsupervised learning

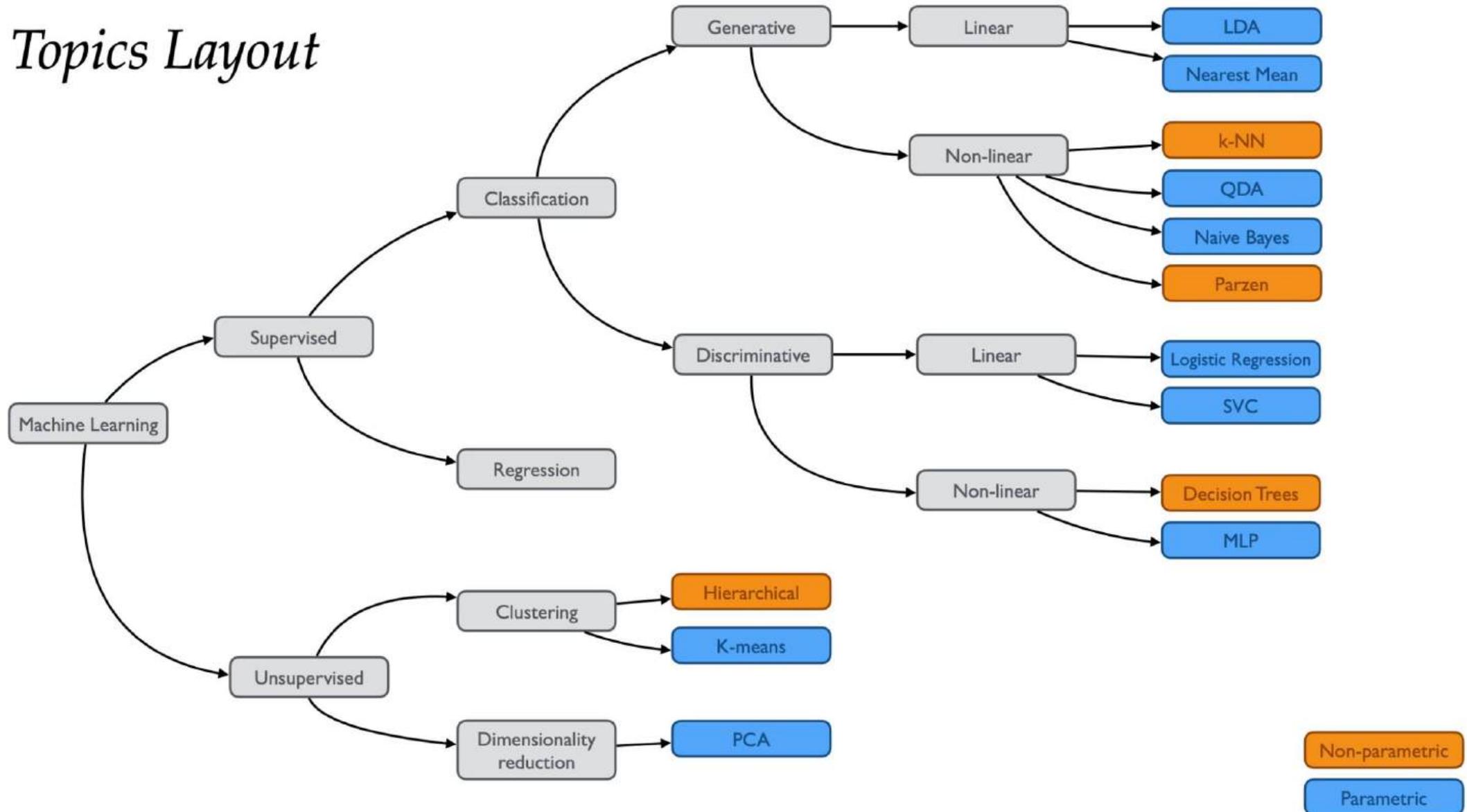
Gosia Migut

# Recap supervised methods

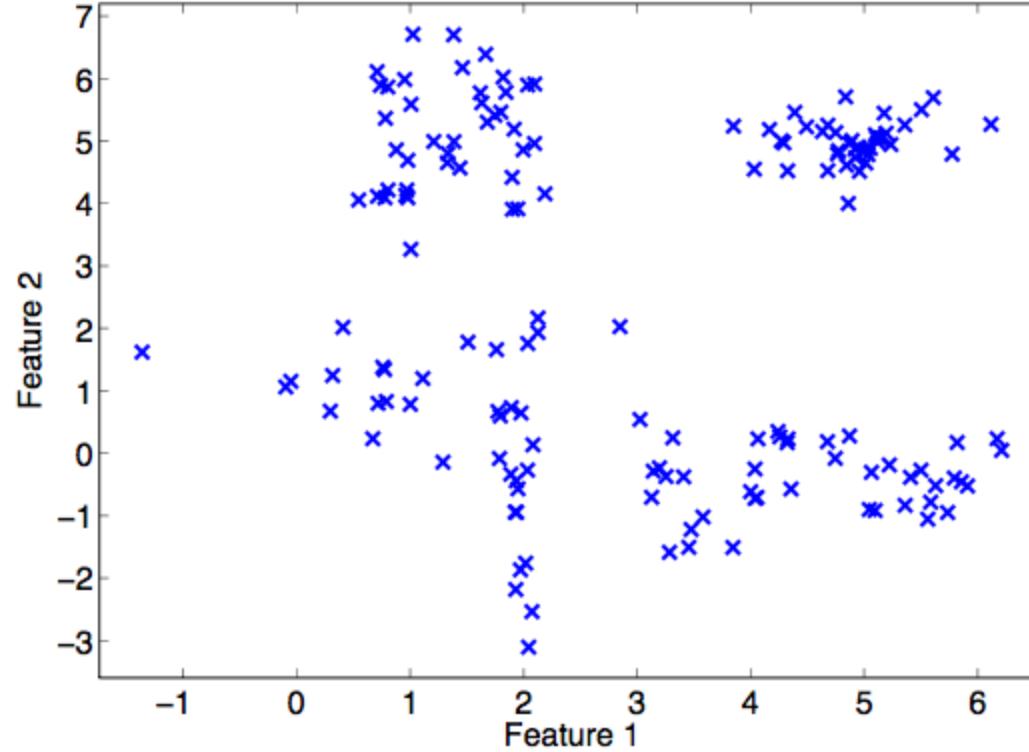
- Until now only supervised methods
  - Each training example described by a feature vector and a label.



# Topics Layout



# Unlabelled data: what now?



- Unsupervised learning: no labels/targets present

# Unsupervised learning

- Clustering
  - Discover structures in unlabelled data
- Dimensionality reduction
  - does not use information about the labels

# Clustering

# Learning goals of today

- Explain what clustering is and it's applications
- Explain k-means algorithm
- Explain hierarchical clustering, single and complete link
- Pros and cons of k-means and hierarchical clustering
- Implement k-means

What interesting clusterings / patterns can you find here? Take 3 minutes.



1



2



3



4



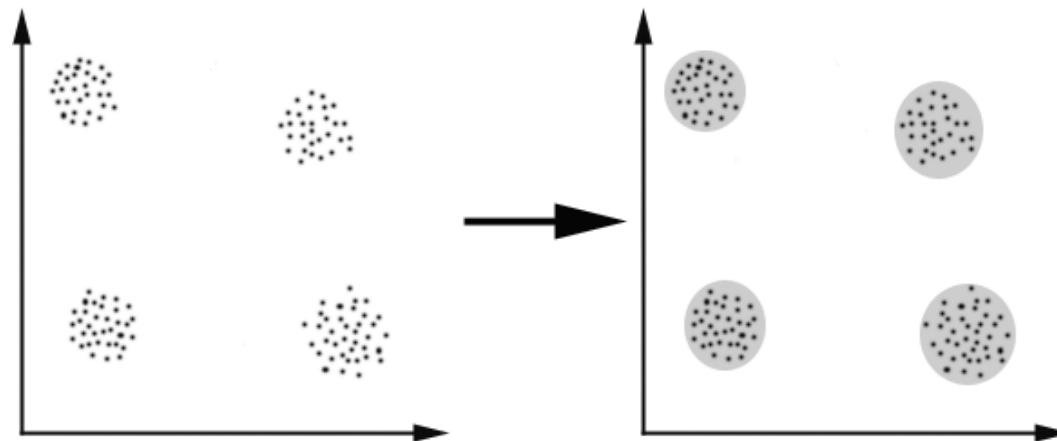
5



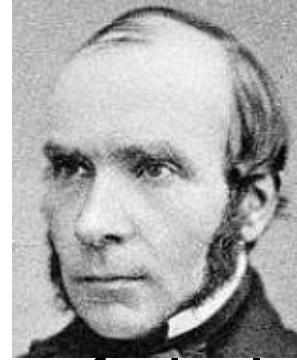
6

# Clustering

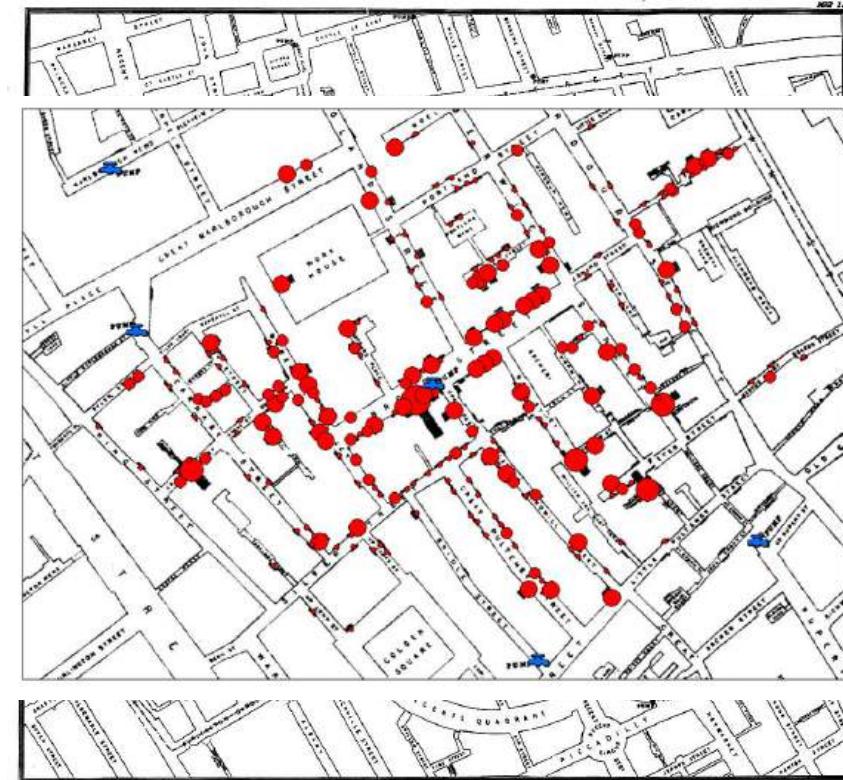
- Finding natural groups in data where
  - Items within the group are close together
  - Items between groups are far apart



# Historic application of clustering



- John Snow, a London physician plotted the locations of cholera deaths on a map during an outbreak in 1850s.
- The locations indicated that cases were clustered around certain intersections where there were polluted wells – exposing both the problem and the solution.



# Clustering applications

- Market research: find groups of similar customers
- Social networks: find communities with similar interests / characteristics
- Recommender systems: find groups of users with similar ratings



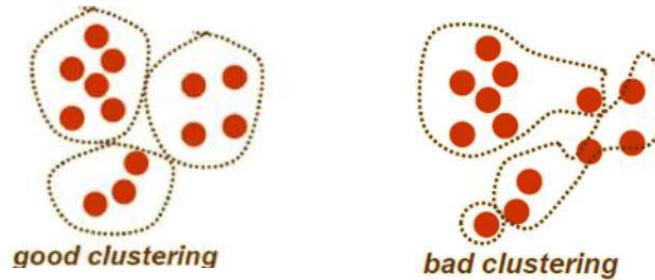
# What do we need for clustering?

## 1. Proximity measure, either

- Similarity measure  $s(x_i, x_k)$ : large if  $x_i$  and  $x_k$  are similar, or
- Dissimilarity (distance) measure  $d(x_i, x_k)$ : small if  $x_i$  and  $x_k$  are similar



## 2. Criterion function to evaluate a clustering



## 3. Algorithm to compute clustering

- Eg. By optimizing the criterion function

# Distance measure

- Typically, we need to define a distance between objects first.

- Euclidean:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^l (x_i - y_i)^2}$$

- Manhattan:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^l |x_i - y_i|$$

# More similarity measures (examples)

- Cosine similarity

$$s_{cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

- Pearson's correlation coefficient

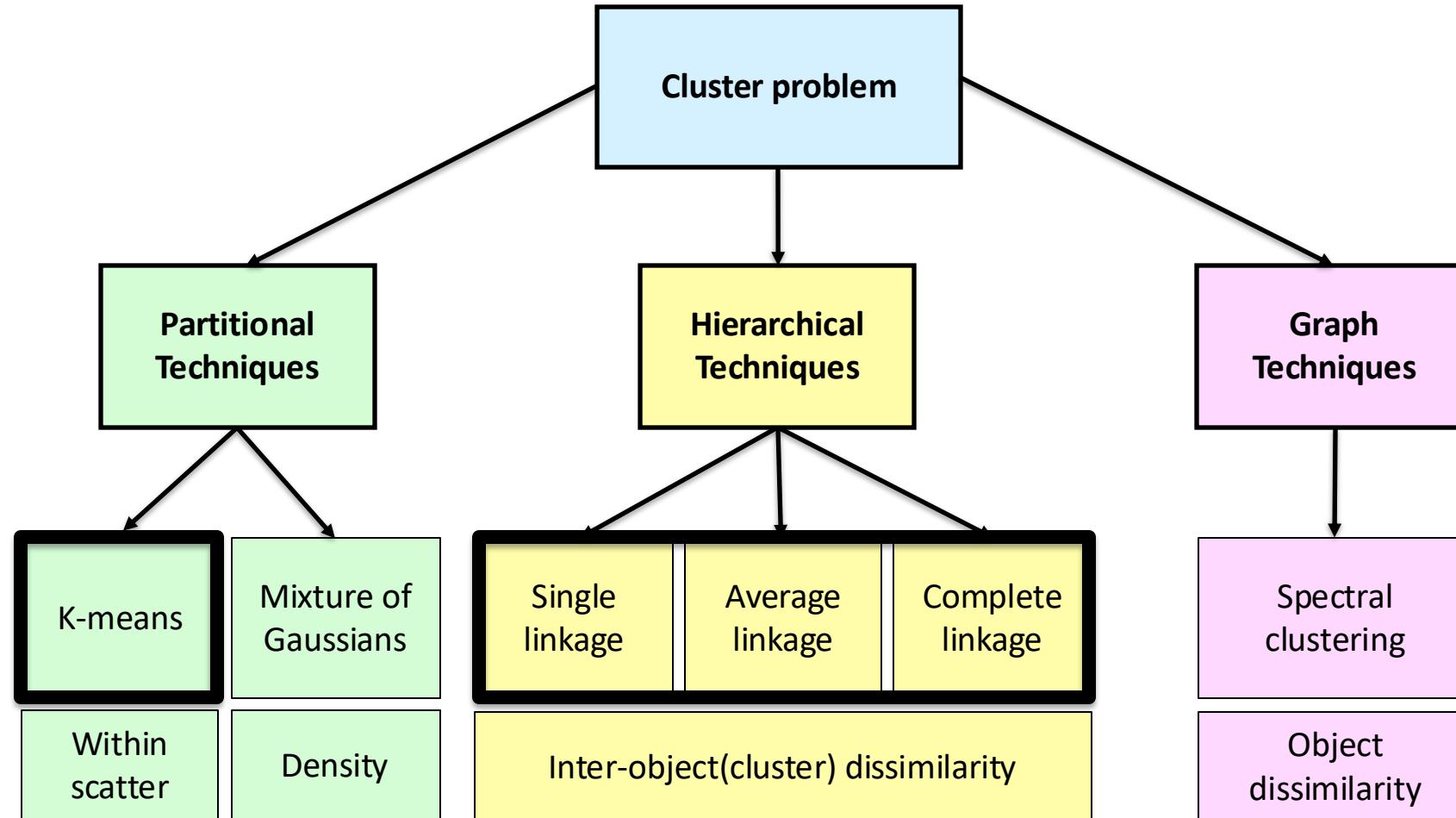
$$r_{Pearson}(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} - \mu_x)^T (\mathbf{y} - \mu_y)}{\|\mathbf{x} - \mu_x\| \|\mathbf{y} - \mu_y\|}$$

- and more... (for discrete features, mixed features, categorical features, ...)

# Cluster evaluation (a hard problem)

- Intra-cluster cohesion (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster's mean.
  - Sum of squared errors (SSE) is a commonly used measure.
- Inter-cluster separation (isolation):
  - Separation means that different cluster means should be far away from one another.
- In most applications, expert judgments are still the key

# Clustering techniques



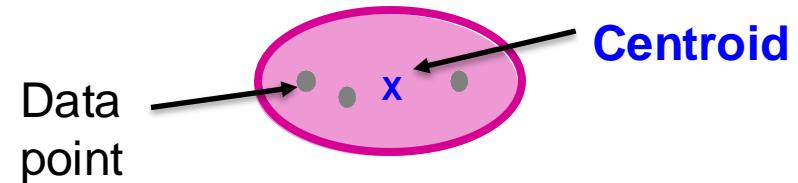
# Hard vs. soft

- Hard assignments: each point assigned to 1 cluster
  - K-Means
  - Hierarchical clustering
- Soft assignments: each point assigned cluster membership
  - Fuzzy C-means
  - Probabilistic mixture models

# K-means clustering

# K-means algortihm

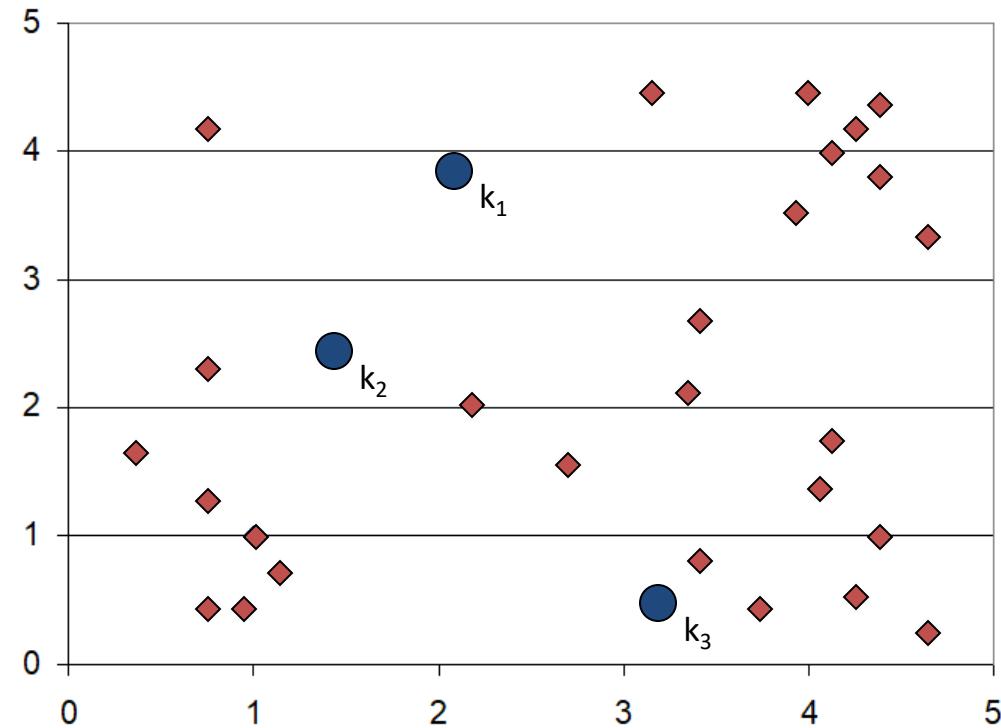
- Let the set of  $n$  data points be  $\{x_1, x_2, \dots, x_n\}$  where
  - $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$  is a feature vector
  - $P$  is the number of dimensions.
- The k-means algorithm partitions the given data into  $k$  clusters:
  - Each cluster has a cluster centre (cluster mean), called centroid.
  - $K$  is specified by the user



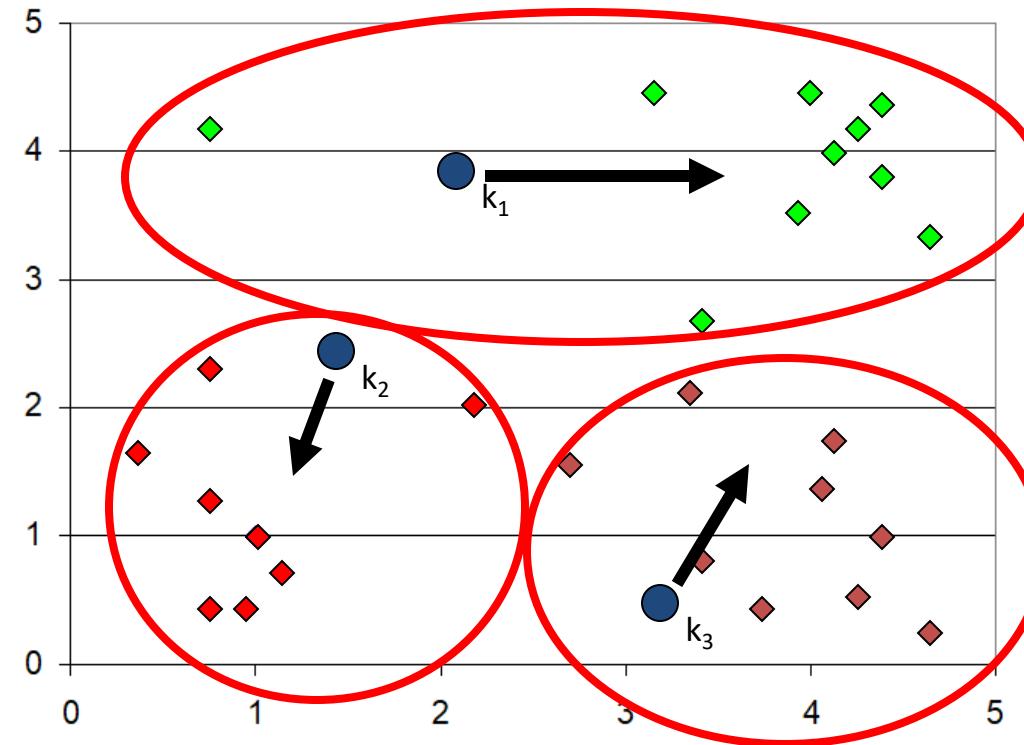
# K-means algorithm

- Given  $k$ , the k-means algorithm works as follows:
  1. Choose  $k$  (random) data points (seeds) to be the initial **centroids**, cluster centers
  2. Assign each data point to the closest **centroid**
  3. Re-compute the **centroids** using the current cluster memberships
  4. If a convergence criterion is not met, repeat steps 2 and 3

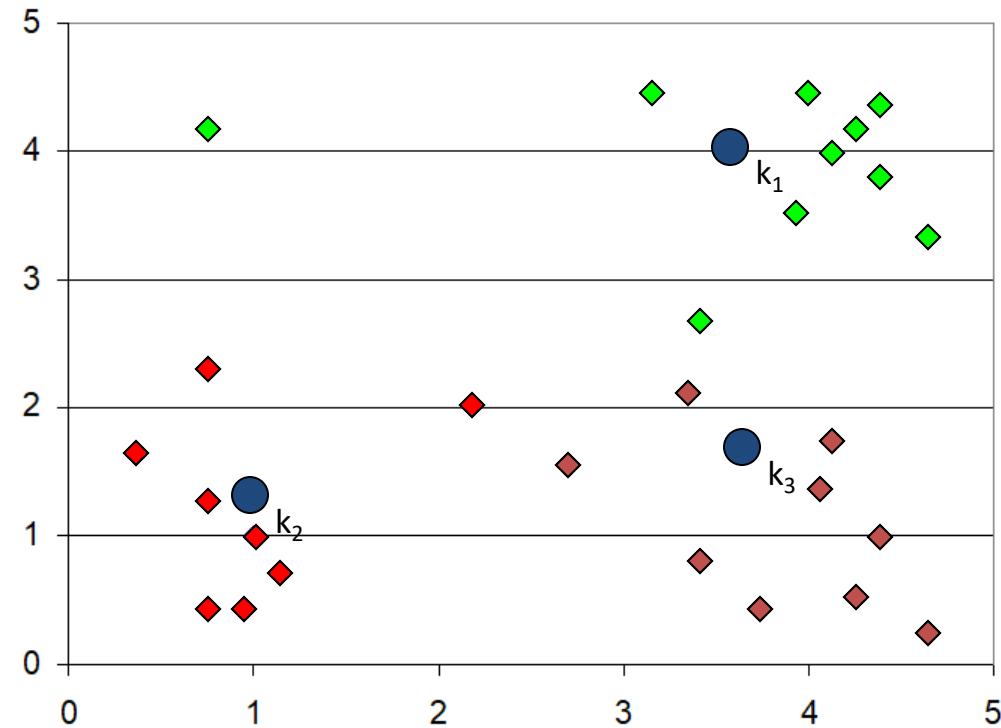
# K-means: how it works



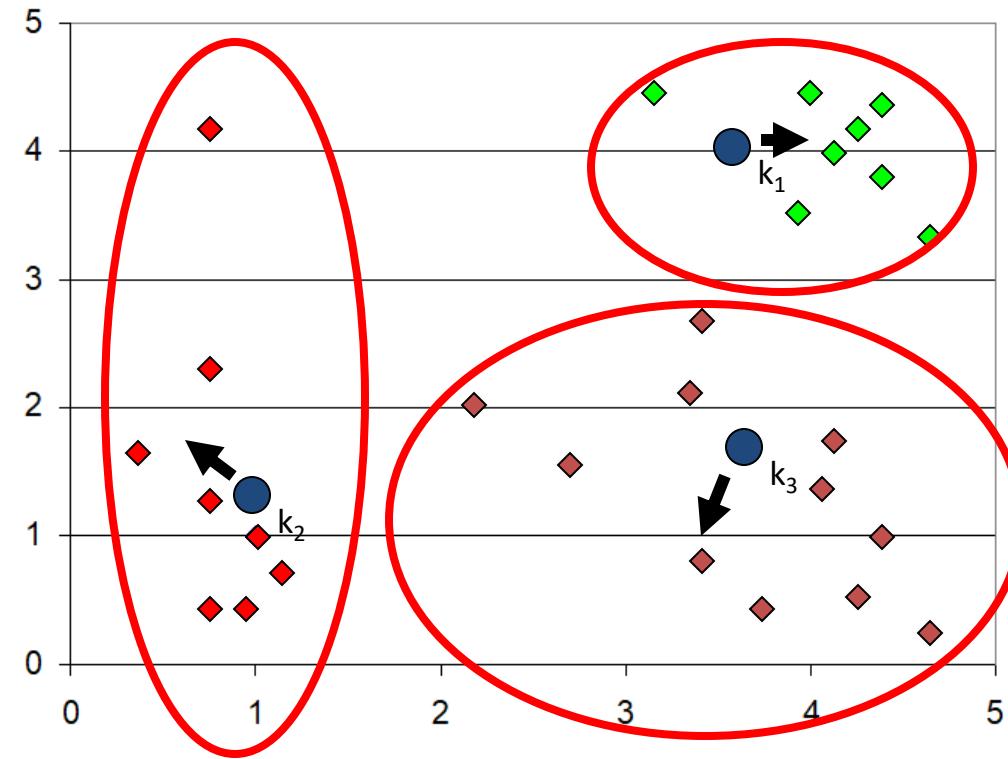
# K-means: how it works



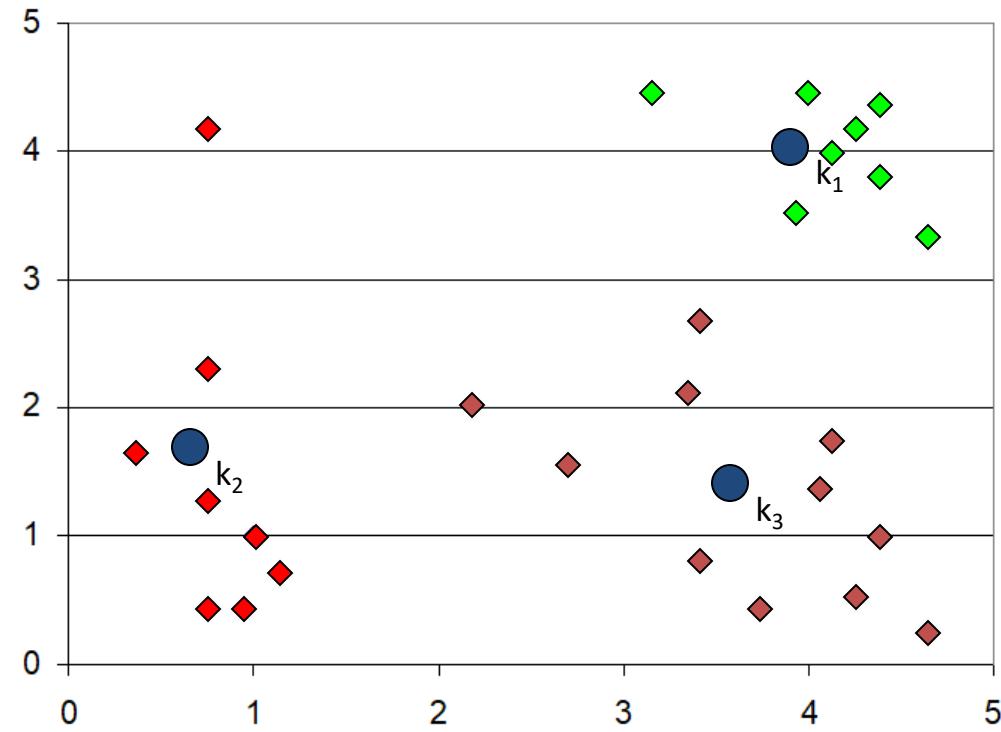
# K-means: how it works



# K-means: how it works



# K-means: how it works



# K-means questions

- When do we know when to stop?
- What is it trying to optimize?
- How do we choose the number of centers?
- Are we sure it will terminate?
- Are we sure it will find an optimal clustering?

# K-means convergence (stopping) criterion

- no (or minimum) re-assignments of data points to different clusters, or
- no (or minimum) change of centroids, or
- minimum decrease in the sum of squared errors (SSE)

# Sum of squared errors

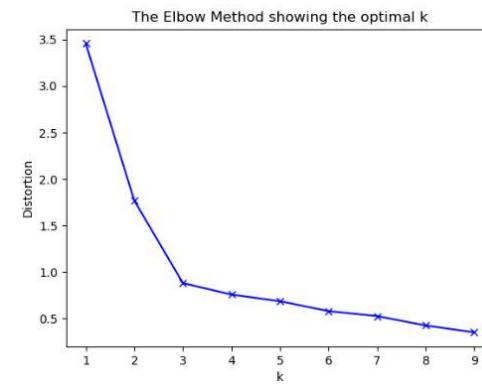
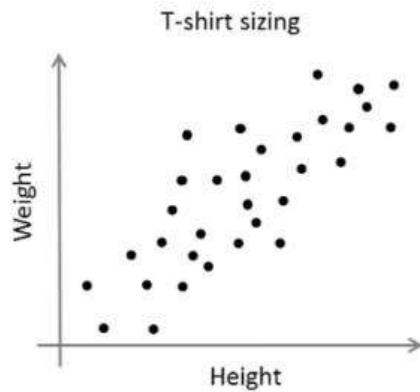
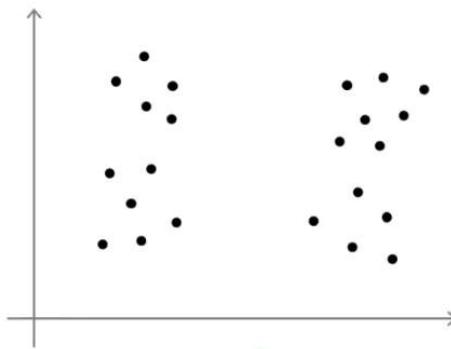
- Cost function (distortion)

$$J(c, \mu) = \frac{1}{n} \sum_{i=1}^m \|x_i - \mu_{C_i}\|^2$$

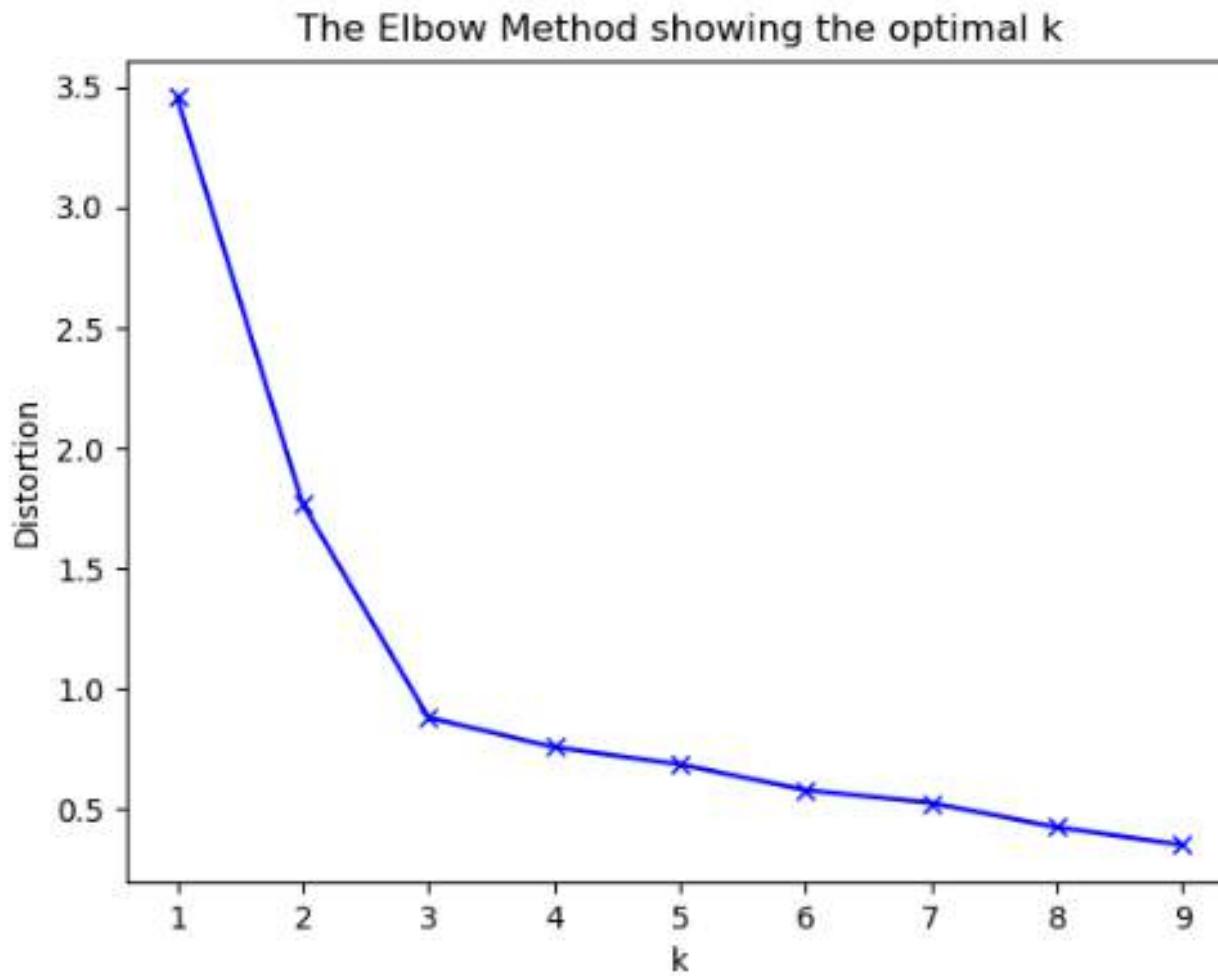
- $\mu_{C_i}$  is cluster center to which  $x_i$  is assigned

# Choosing the number of clusters

- Inspect visually
- Known purpose
- Elbow method



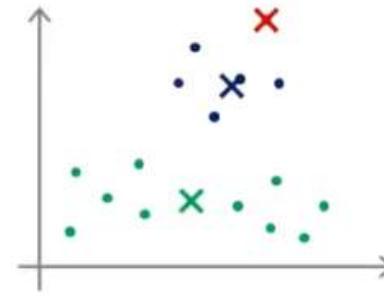
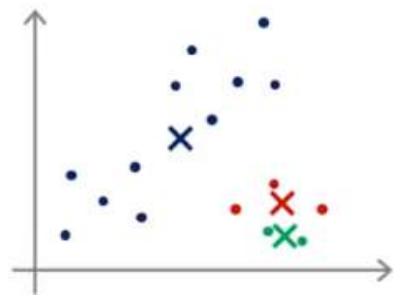
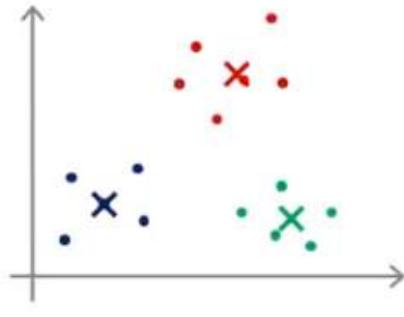
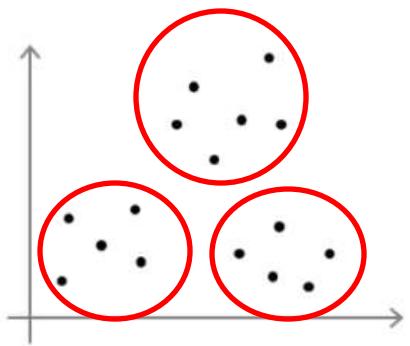
# Elbow method



# Random initialization

- $2 \leq k < m$
- Random pick K training examples
- Set  $\mu_1, \mu_2, \dots, \mu_k$  equal to these examples

# Local optima

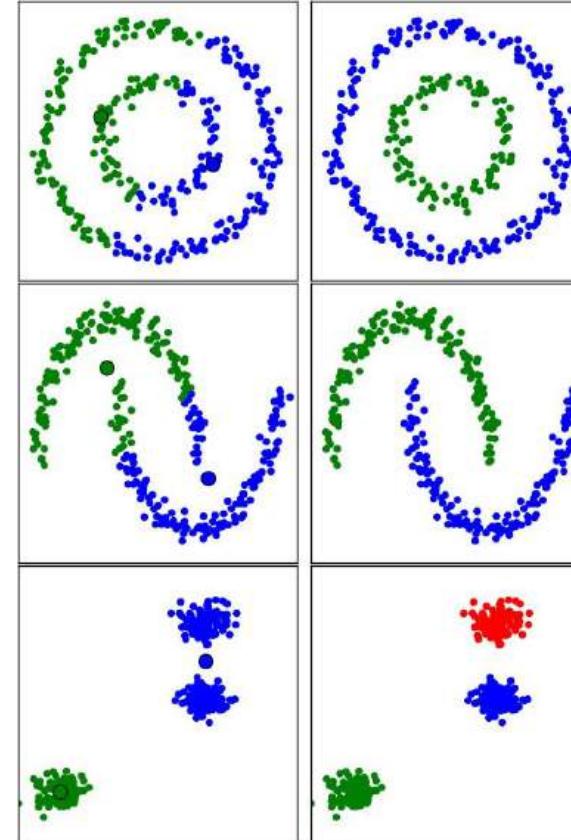


# Random initialization

- For  $i=1$  to 10000
  - {
    - Randomly initialize k means
    - Run k-means. Get centroids and means
    - Compute cost function  $J$
  - }
- Pick clustering that gave lowest cost
- For high-dimensional data, many restarts are necessary (e.g.  $I = 10000$ )!

# K-means summary

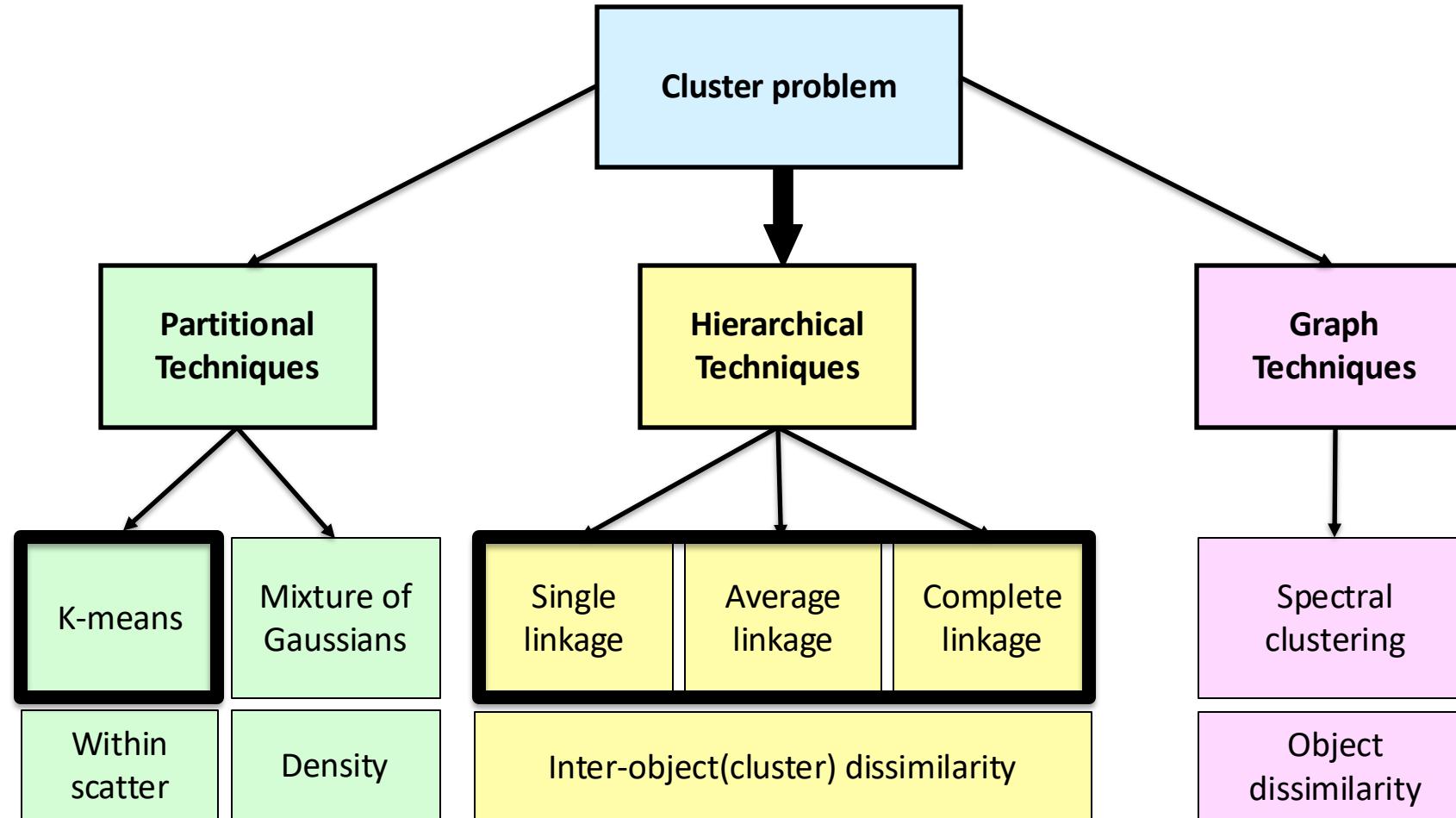
- Disadvantages:
  - Finds only convex clusters (“round shapes”)
  - Sensitive to initialization
  - Can get stuck in local minima
- Advantages:
  - Very simple
  - Fast



## Example exercise

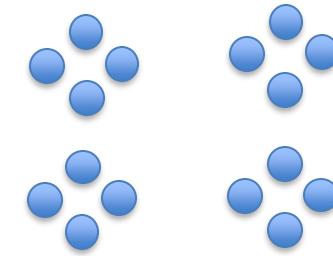
- Given are the following points:  
 $(1, 4), (2, 2), (5, 5), (4, 6)$ ,  
and two cluster centroids:  
 $\mu_1 = (1, 2), \mu_2 = (6, 6)$ .
- What is the value of the k-means cost function (SSE)?

# Clustering techniques

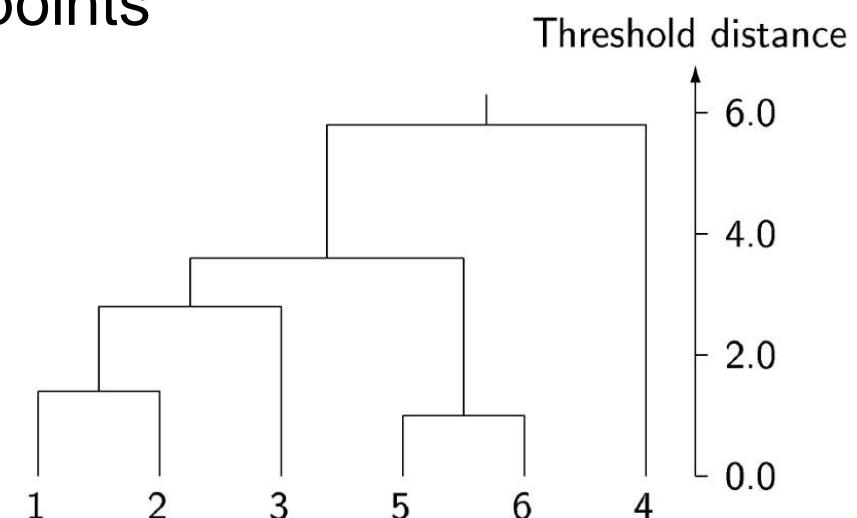


# Hierarchical clustering

# Hierarchical clustering

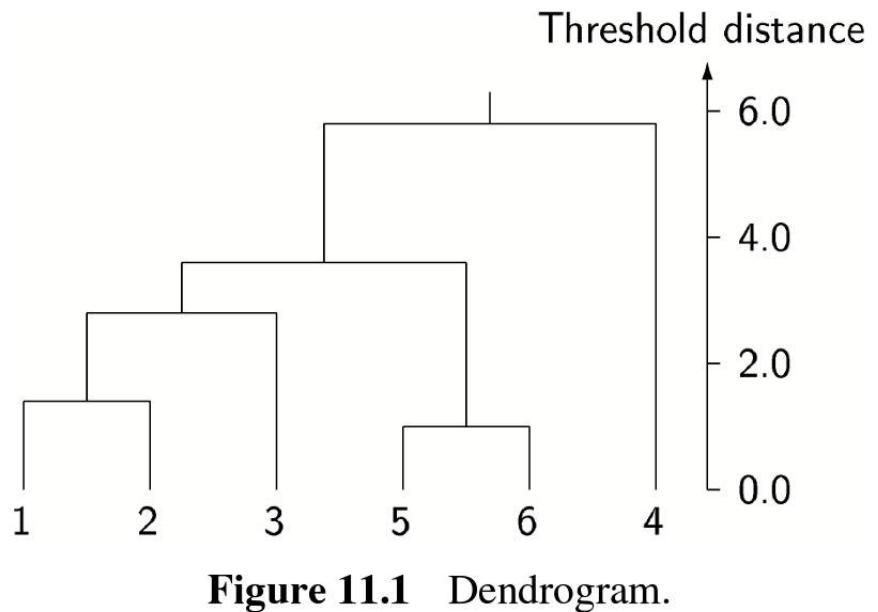


- Selecting  $k$  is a problem of granularity
  - How course or fine-grained is the structure in the data?
  - No cluster algorithm able to pick  $k$
- Instead of picking  $k$  find a hierarchy of structure
  - Course effects: top level contains all points
  - Fine-grained: bottom level
    - one cluster per data point



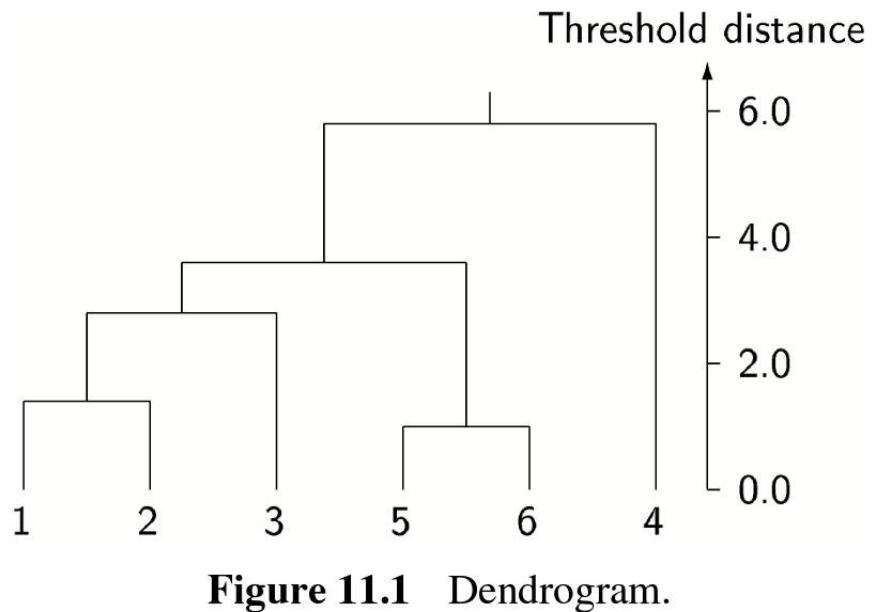
# Hierarchical clustering approaches

- Agglomerative (bottom-up):
  - each point starts as cluster
  - group two closest clusters
  - stop at some point



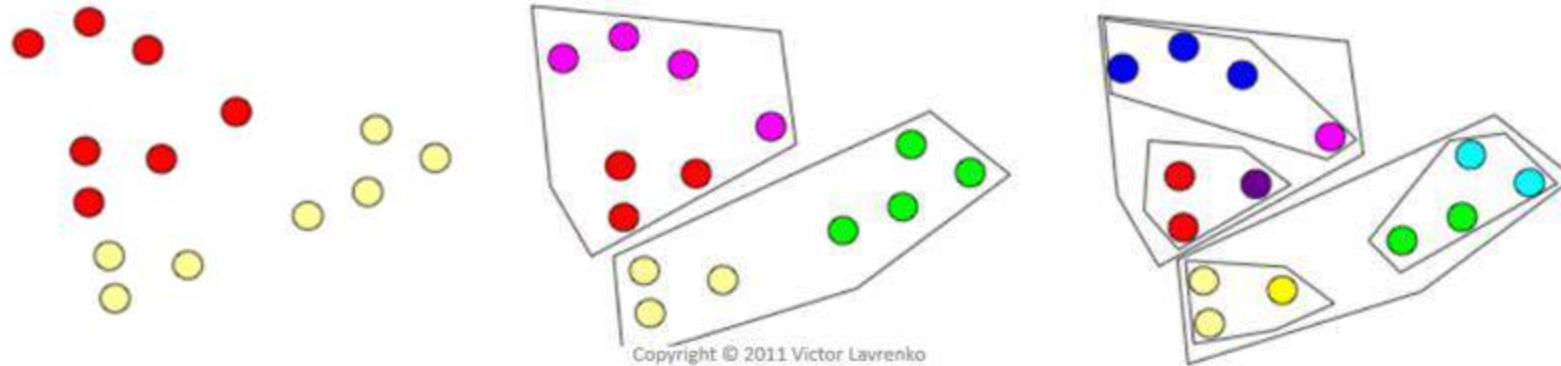
# Hierarchical clustering approaches

- **Divisive (top-down):**
  - all points start in one cluster
  - split cluster in some sensible way
  - stop at some point



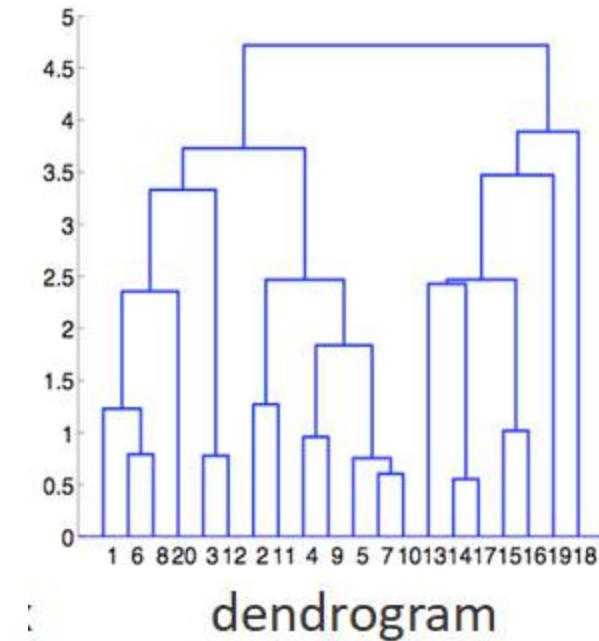
# Divisive: hierarchical k-means

- Apply k-means recursively:
  - Run k-mean on the original data for  $k=2$
  - For each of the resulting clusters run k-means with  $k=2$



# Agglomerative clustering

- Starting from individual observations, produce sequence of clusterings of increasing size
- At each level, two clusters chosen by criterion are merged



# Agglomerative clustering

1. Determine distances between all clusters
  2. Merge clusters that are closest
  3. IF #clusters>1 THEN GOTO 1
- 
- Which clusters to start with?
  - What is the distance between clusters?
  - Final number of clusters?

# Different merging rules

- **Single linkage:** two nearest objects in the clusters :

$$g(R, S) = \min_{ij} \{d(x_i, x_j) : x_i \in R, x_j \in S\}$$

- **Complete linkage:** two most remote objects in the clusters :

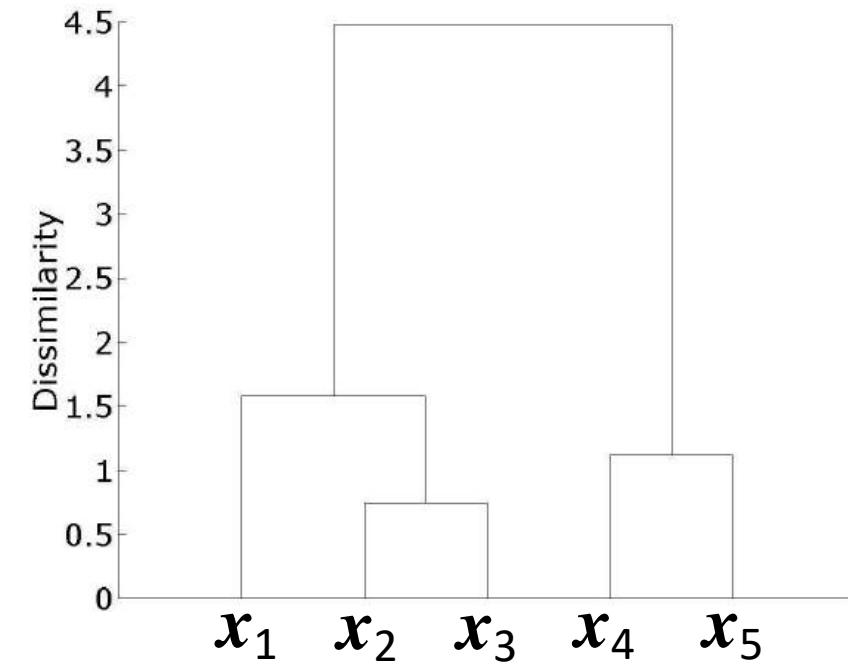
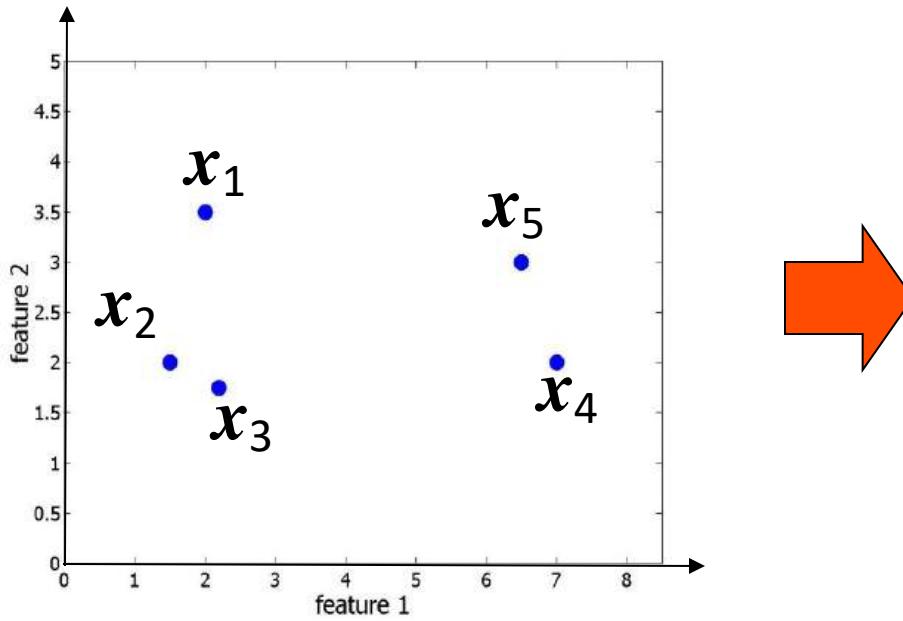
$$g(R, S) = \max_{ij} \{d(x_i, x_j) : x_i \in R, x_j \in S\}$$

- **Average linkage:** cluster centres :

$$g(R, S) = \frac{1}{|R||S|} \sum_{ij} \{d(x_i, x_j) : x_i \in R, x_j \in S\}$$

# Hierarchical clustering: how it works

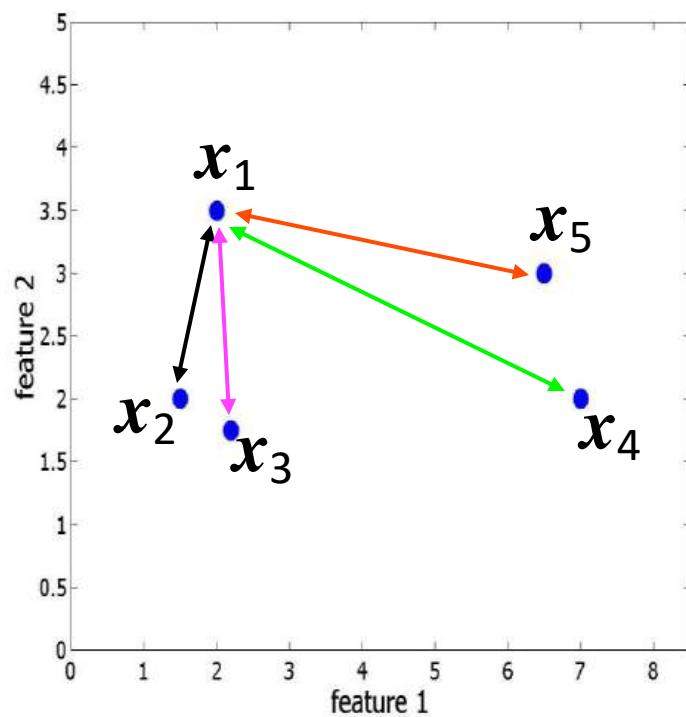
- Input:
  - dataset,  $X: [n \times p]$ , or directly:
  - dissimilarity matrix,  $D: [n \times n]$
  - linkage type
- Output:
  - dendrogram



# Hierarchical clustering

- **Step 0:** each object is a cluster:

## Dataset



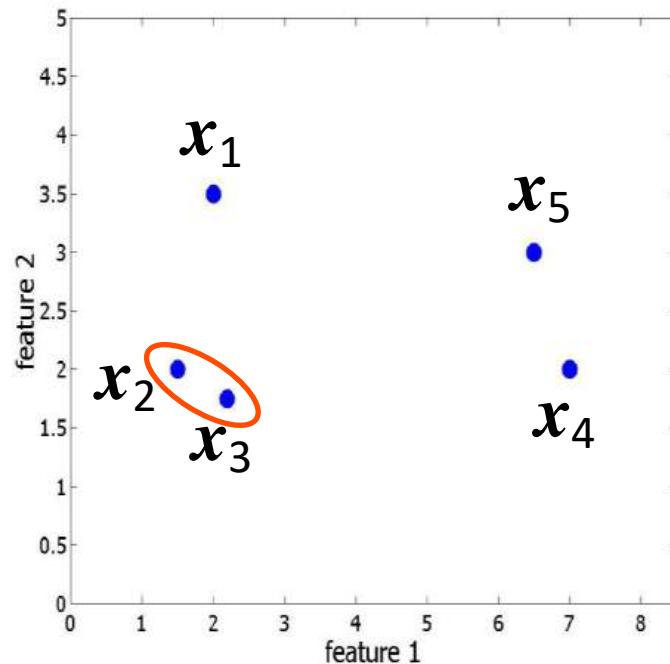
(Euclidean) distance matrix,  $D$

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$x_1$	0.00	1.58	1.76	5.22	4.53
$x_2$		0.00	0.74	5.50	5.10
$x_3$			0.00	4.81	4.48
$x_4$				0.00	1.12
$x_5$					0.00

# Hierarchical clustering

- **Step 1:**

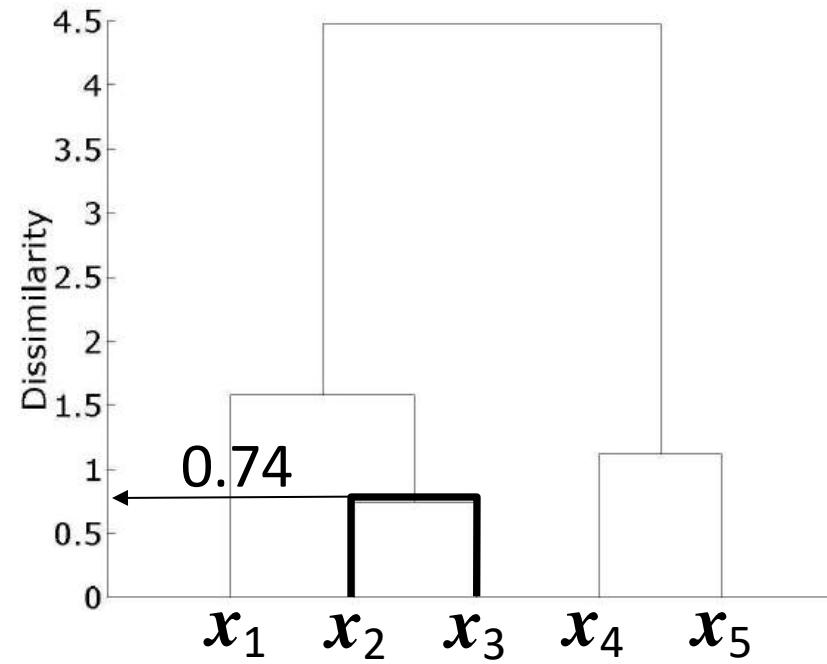
Find the most similar pair:  $\min_{(i,j)}\{d(i,j)\} = d(2,3)$



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$x_1$	0.00	1.58	1.76	5.22	4.53
$x_2$		0.00	0.74	5.50	5.10
$x_3$			0.00	4.81	4.48
$x_4$				0.00	1.12
$x_5$					0.00

# Hierarchical clustering

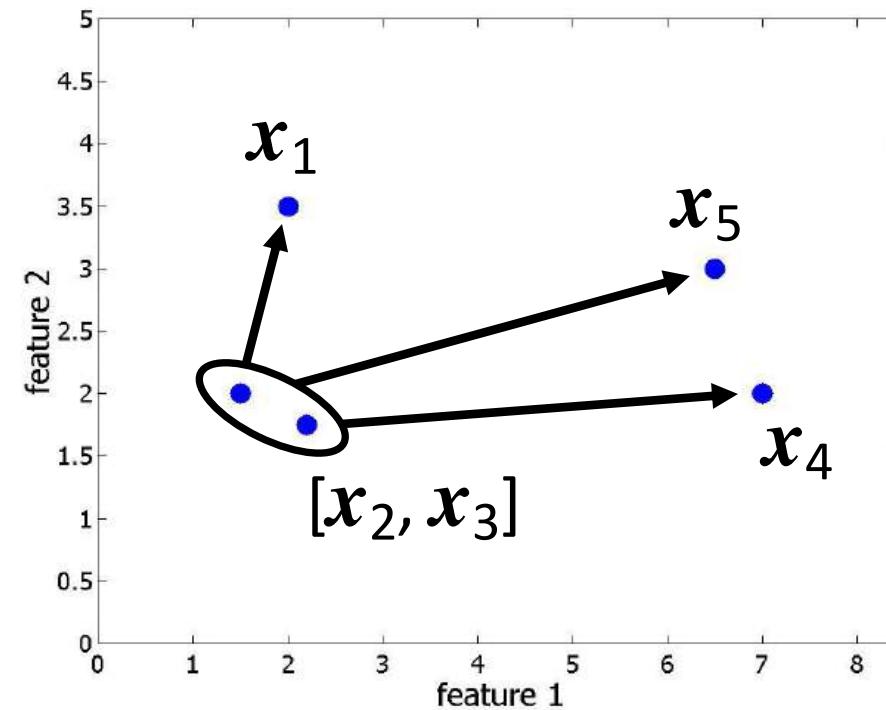
- **Step 2:**  
Merge  $x_2$  and  $x_3$  into a single object,  $[x_2, x_3]$ ;



# Hierarchical clustering

- **Step 3:**

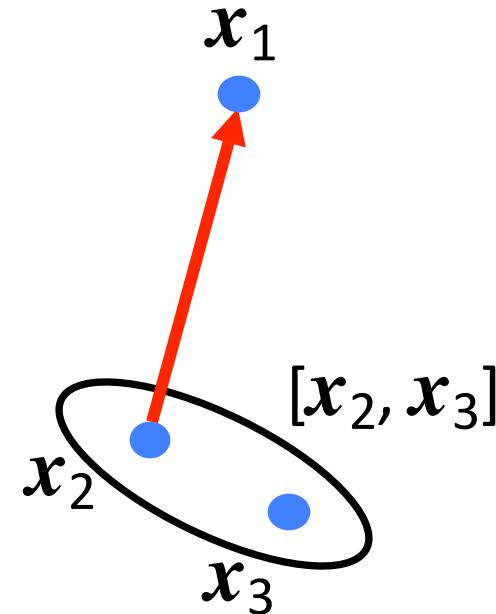
Recompute  $D$  – what is the distance between  $[x_2, x_3]$  and the rest?



# Hierarchical clustering

- **Step 3:**

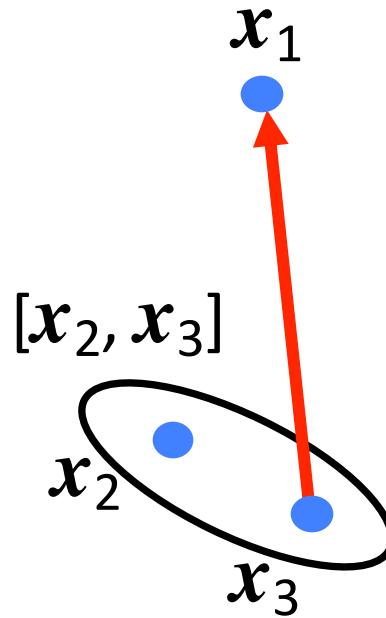
Recompute  $D$  – **single linkage**:  $d([x_2, x_3], x_1) = \min(d(x_1, x_2), d(x_1, x_3))$



# Hierarchical clustering

- **Step 3:**

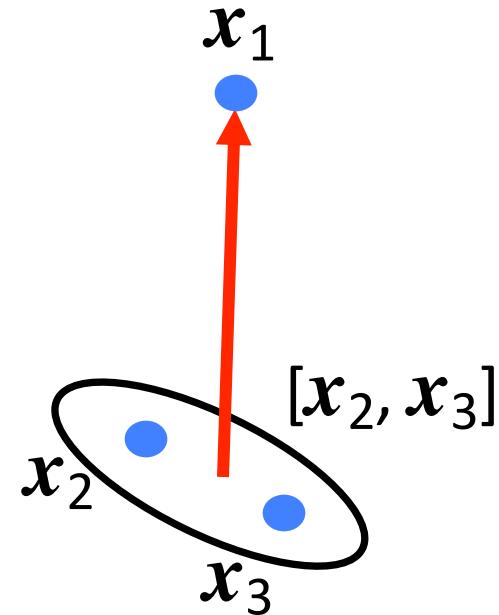
Recompute  $D$  – **complete linkage**:  $d([x_2, x_3], x_1) = \max(d(x_1, x_2), d(x_1, x_3))$



# Hierarchical clustering

- **Step 3:**

Recompute  $D$  – **average linkage**:  $d([x_2, x_3], x_1) = \text{mean}(d(x_1, x_2), d(x_1, x_3))$



# Hierarchical clustering

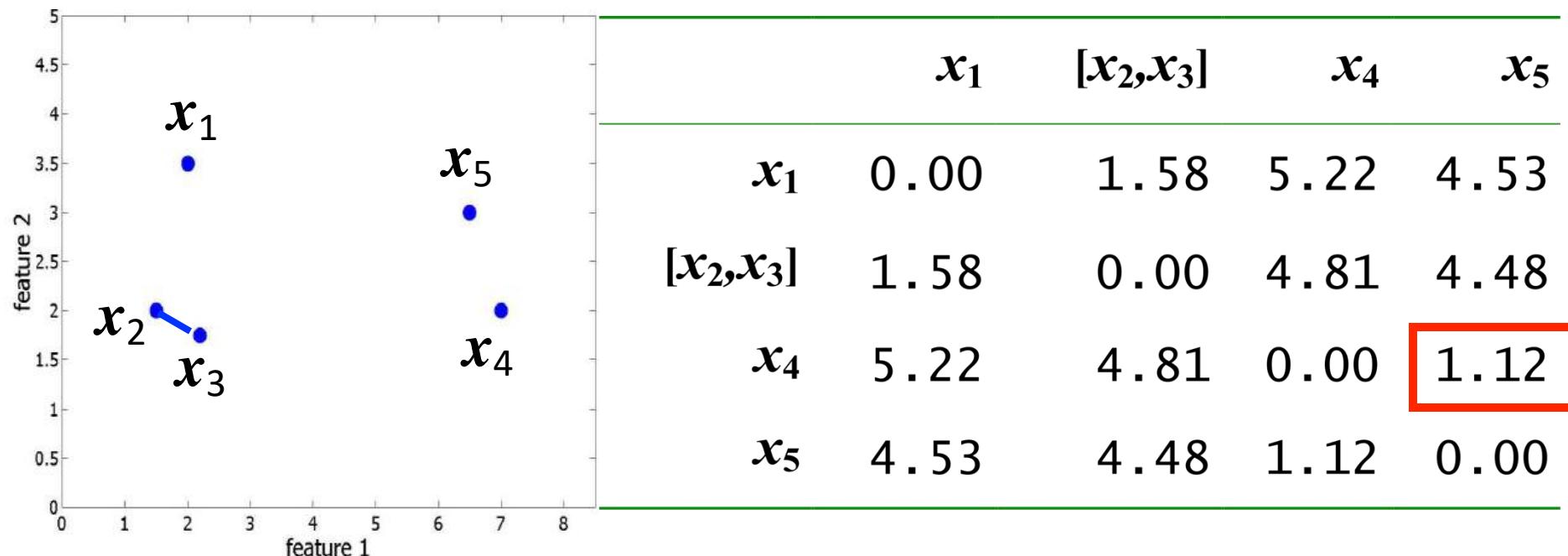
- **Step 3:**  
Recompute  $D$  – **single linkage**:

	$x_1$	[ $x_2, x_3$ ]	$x_4$	$x_5$
$x_1$	0.00	1.58	5.22	4.53
[ $x_2, x_3$ ]		0.00	4.81	4.48
$x_4$			0.00	1.12
$x_5$				0.00

# Hierarchical clustering

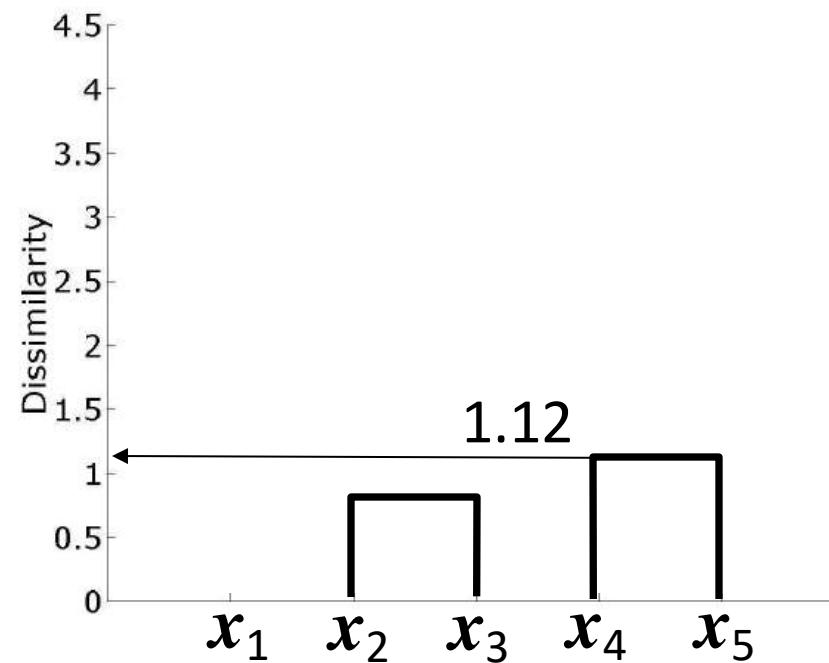
- **Repeat, step 1:**

Find the most similar pair of objects:  $\min_{(i,j)}\{d(i,j)\} = d(4,5)$



# Hierarchical clustering

- **Repeat, step 2:**  
Merge  $x_4$  and  $x_5$  into a single object,  $[x_4, x_5]$ ;



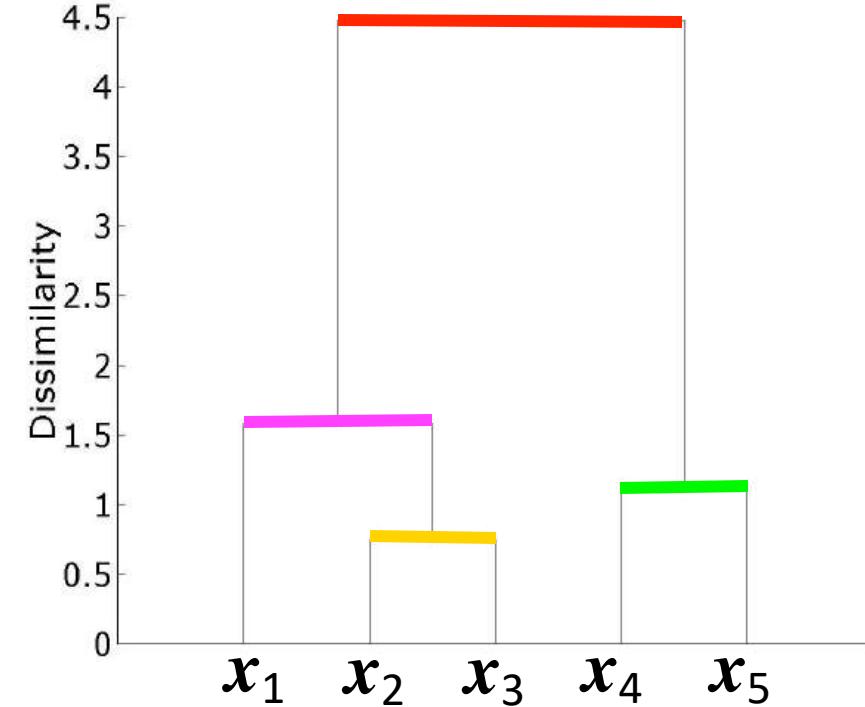
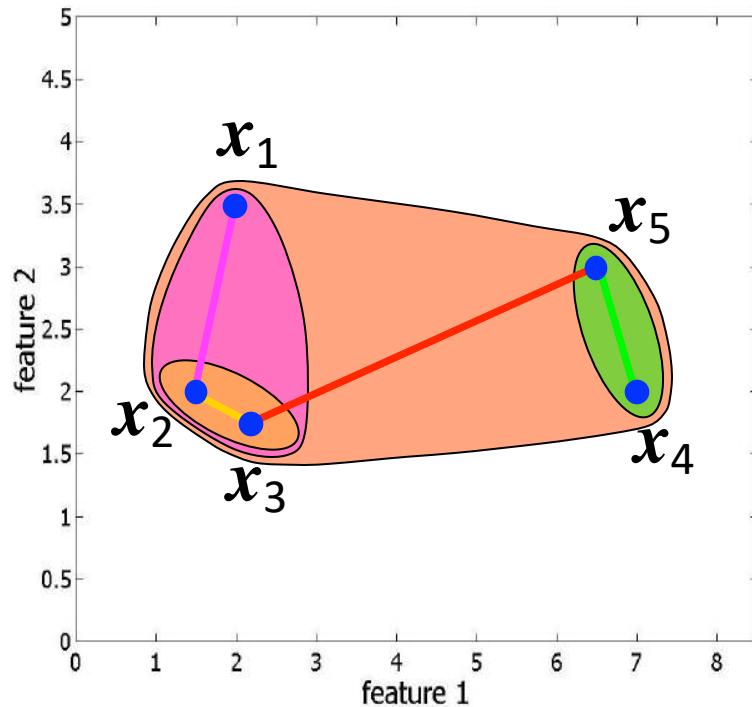
# Hierarchical clustering

- **Repeat, step 3:**  
Recompute  $D$  (single linkage):

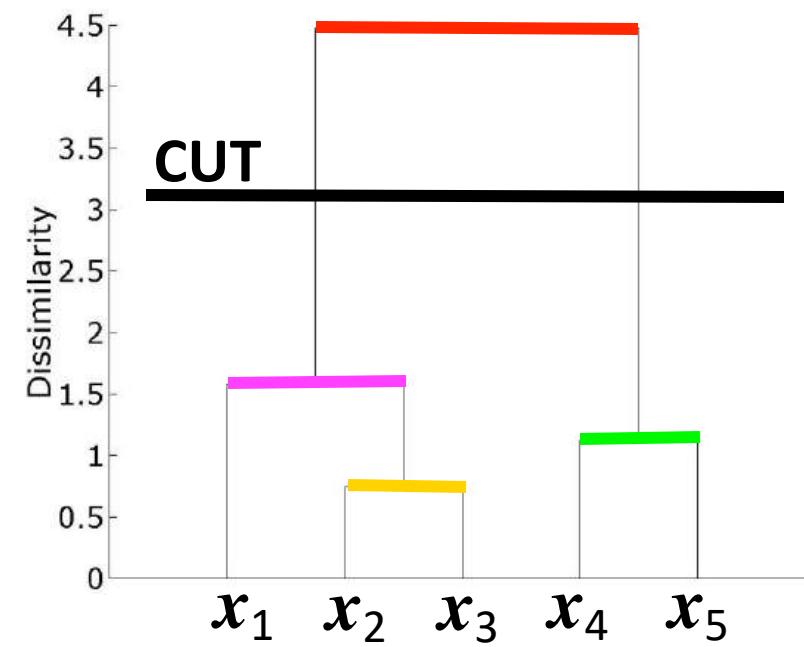
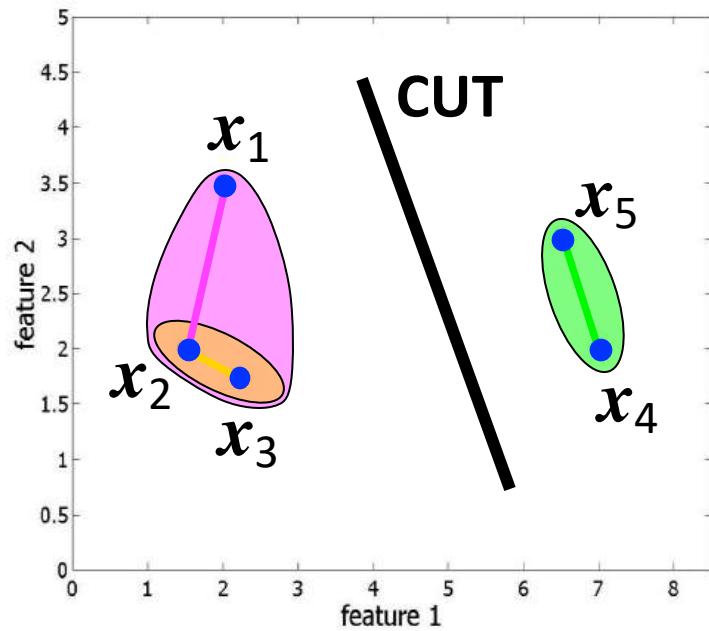
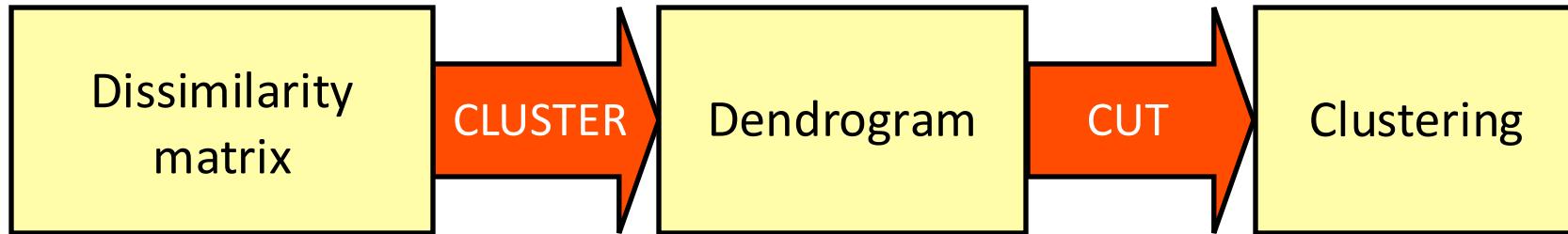
	$x_1$	$[x_2, x_3]$	$[x_4, x_5]$
$x_1$	0.00	1.58	4.53
$[x_2, x_3]$		0.00	4.48
$[x_4, x_5]$			0.00

# Hierarchical clustering

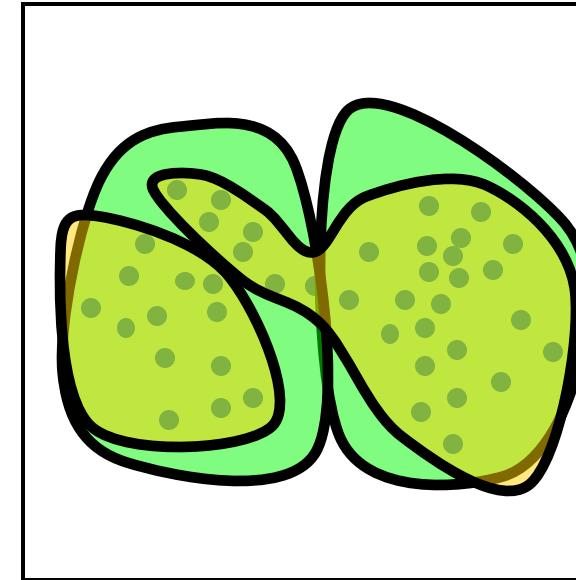
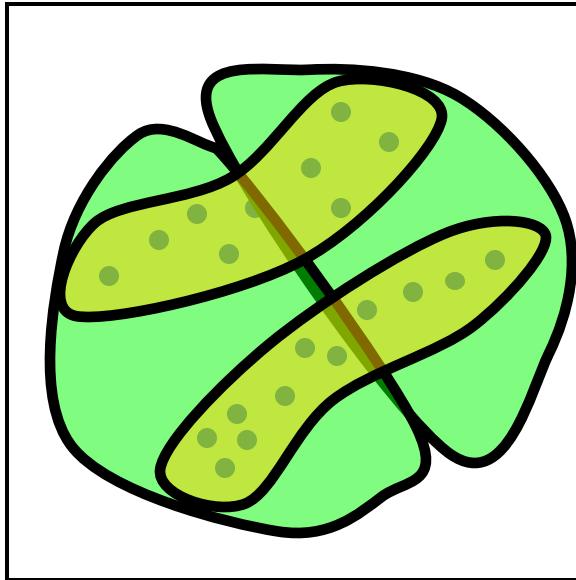
- Repeat steps 1-3 until a single cluster remains



# Hierarchical clustering



# Linkage and cluster shape



Complete linkage



Single linkage

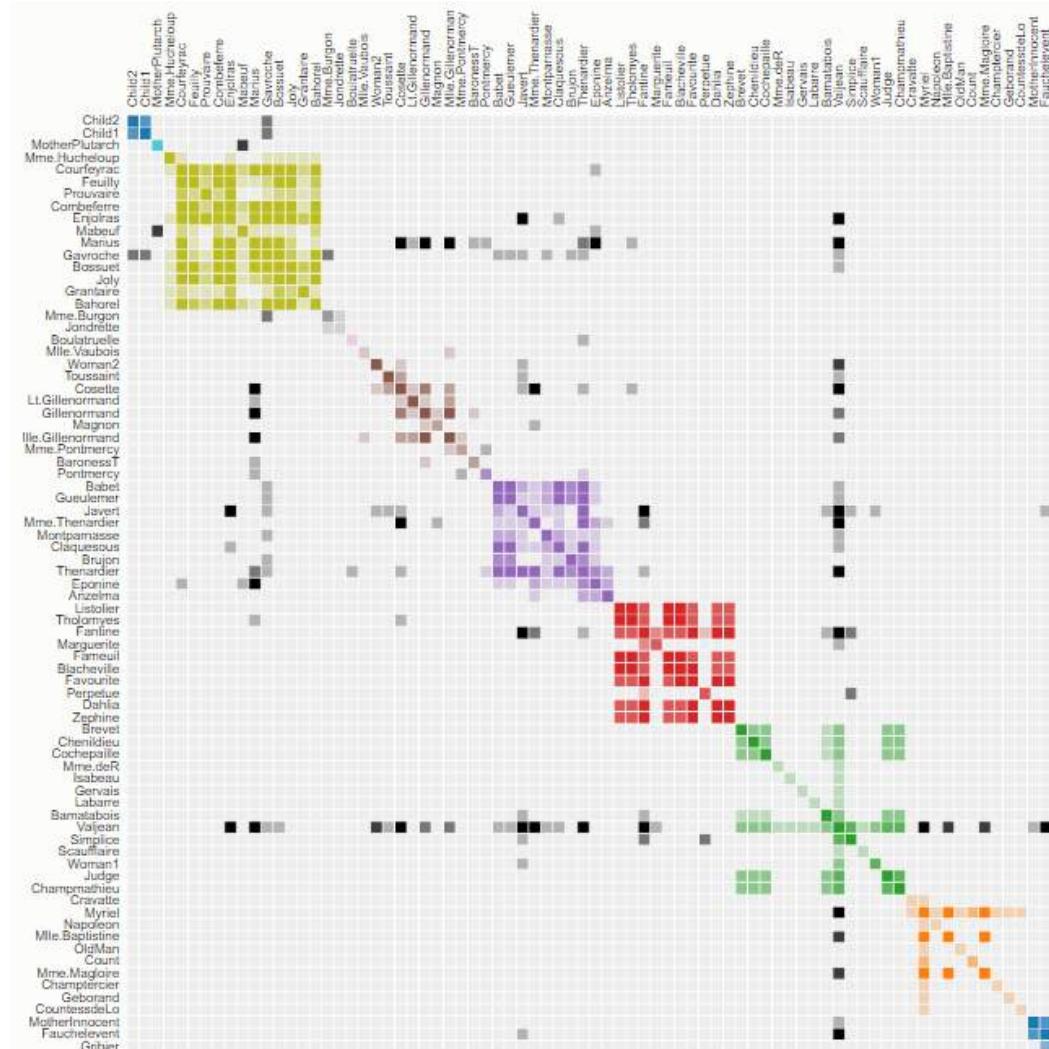
# Question: hierarchical clustering

- Given is a dataset: (4, 10), (7,10), (4, 8), (10, 5), (11, 4), (3, 4), (9, 3), (5, 2)
- Cluster the points using agglomerative clustering
- Use single link method with Euclidean distance
- Stopping criterion: 3 clusters
- Detail your methodology, show steps and dendrogram

# Hierarchical clustering summary

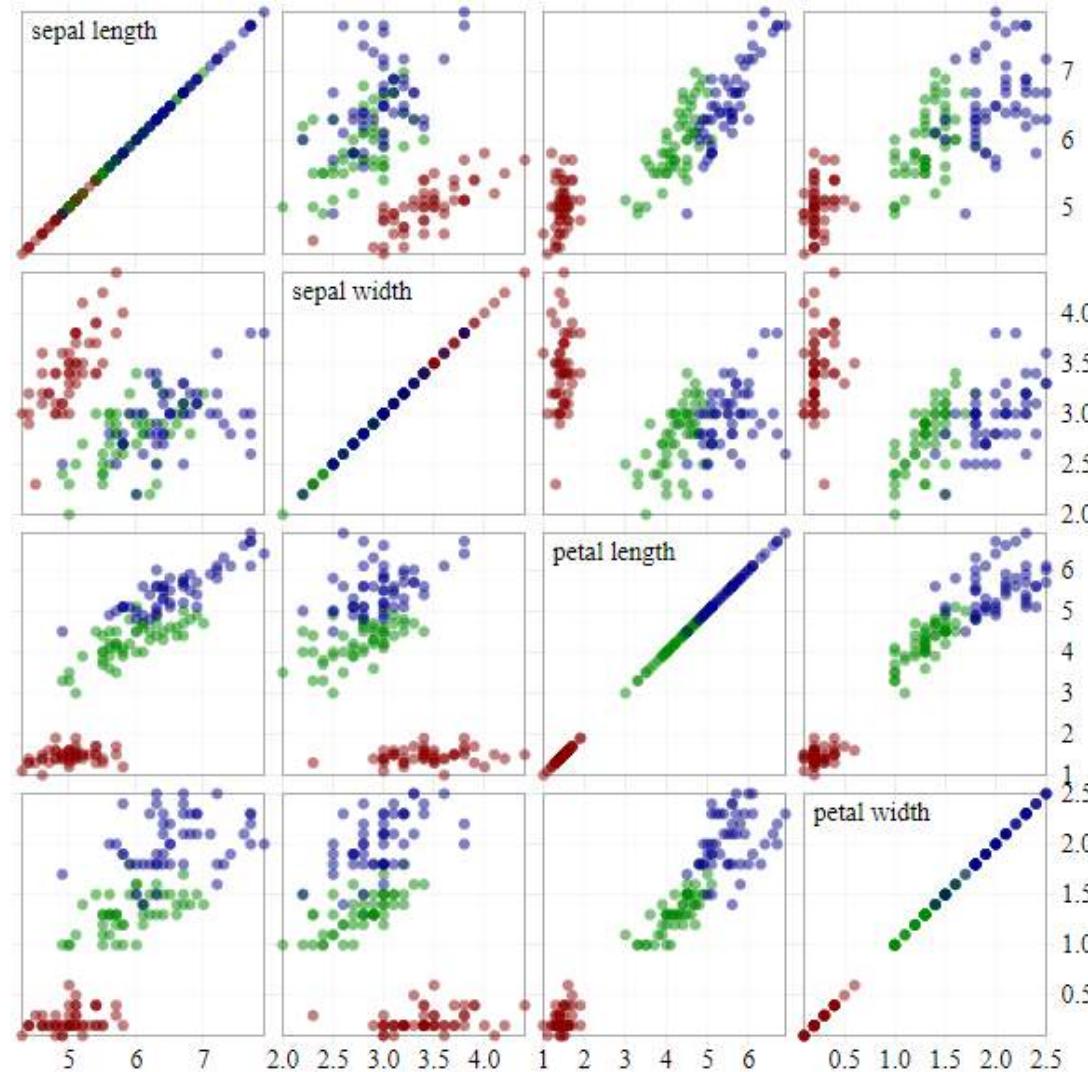
- Pros
  - Dendrogram gives overview of all possible clusterings
  - Linkage type allows to find clusters of varying shapes
  - Different dissimilarity measures can be used
- Cons
  - Computationally intensive
  - Clustering limited to “hierarchical nestings”

# Clusters visualized: Co-occurrence heatmap

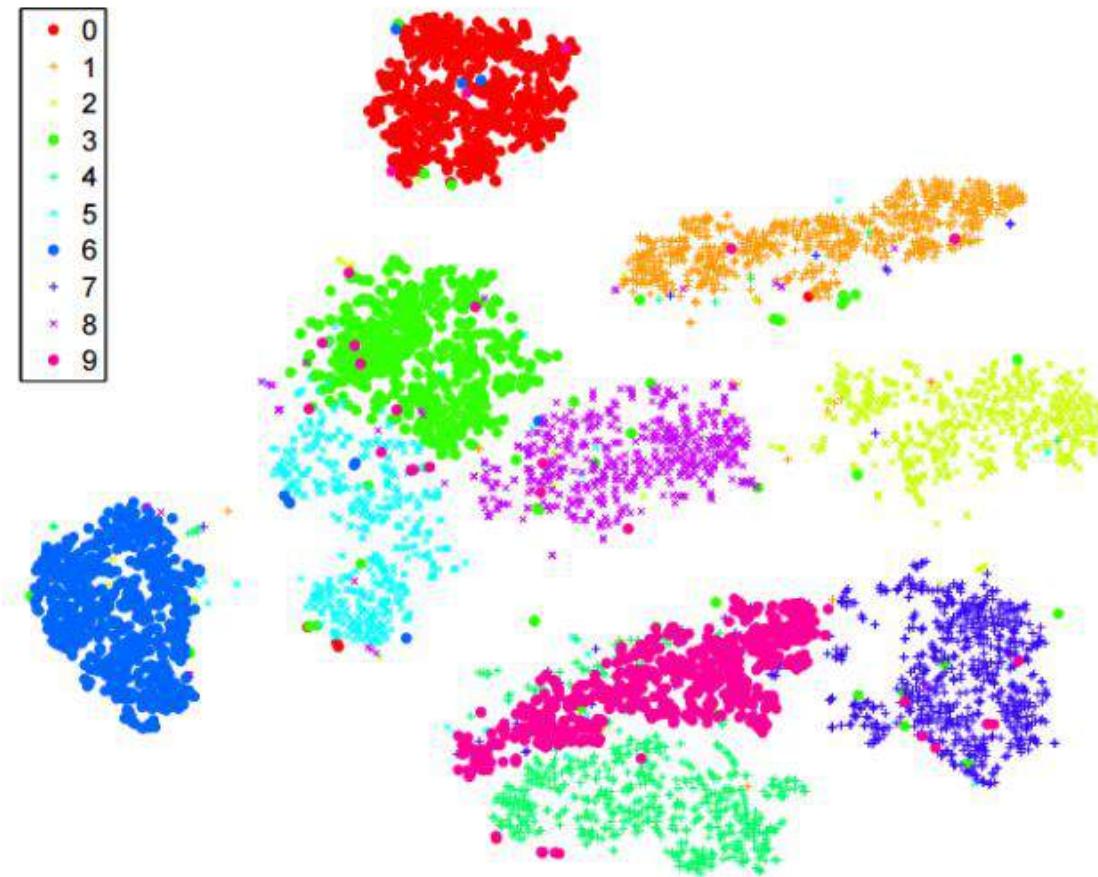


<https://bost.ocks.org/mike/miserables/>

# Clusters visualized: scatterplot matrix



# Clusters visualized: 2d embedding with t-SNE



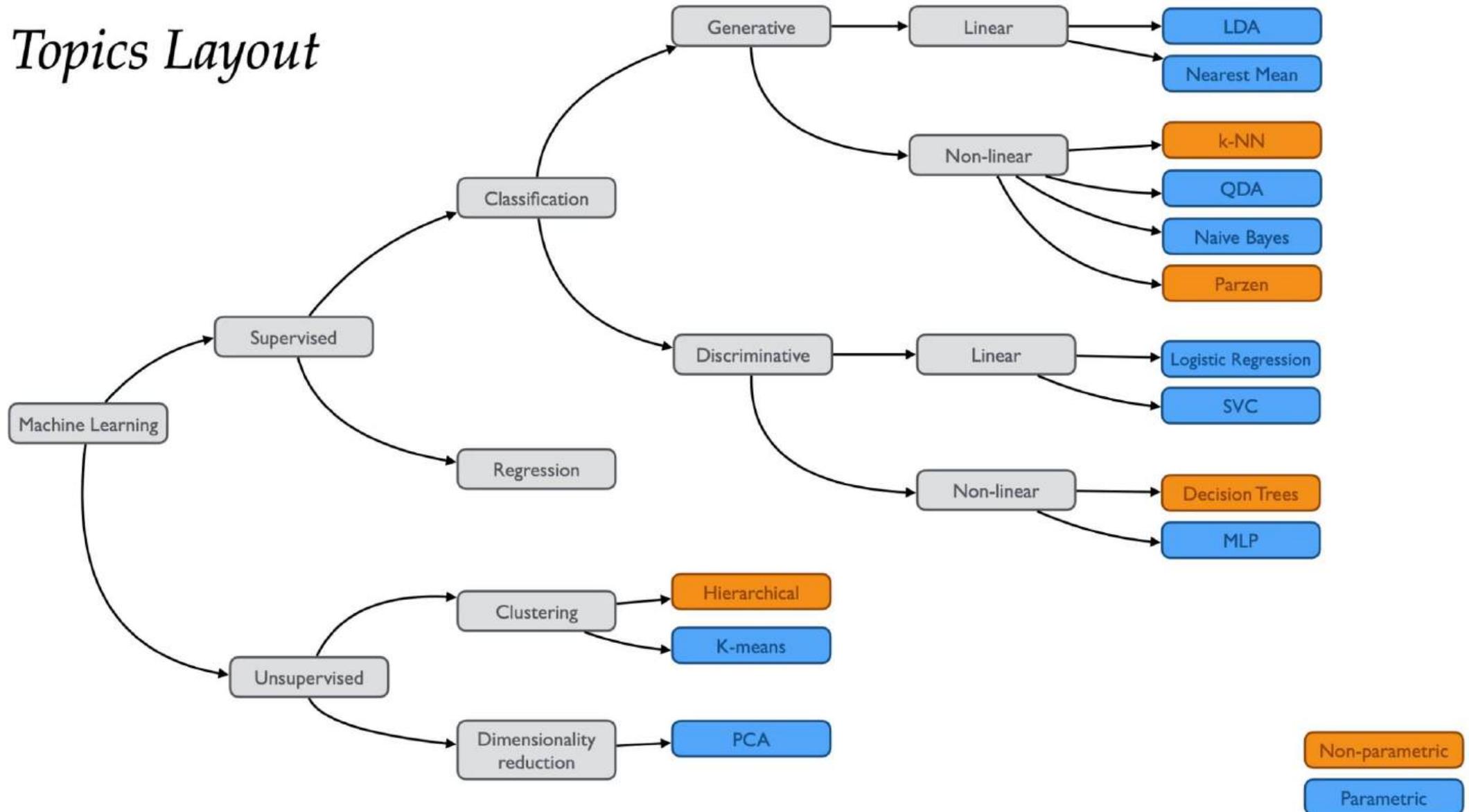
# Clustering summary

- We can “classify” when we don’t have (training) labels: clustering
- Definition of clusters is vague and evaluation hard
- For clustering we need to :
  - define distance measure
  - define criterion function to evaluate a clustering
  - select clustering algorithm
- Discussed clustering algorithms
  - Hierarchical clustering
  - k-means clustering

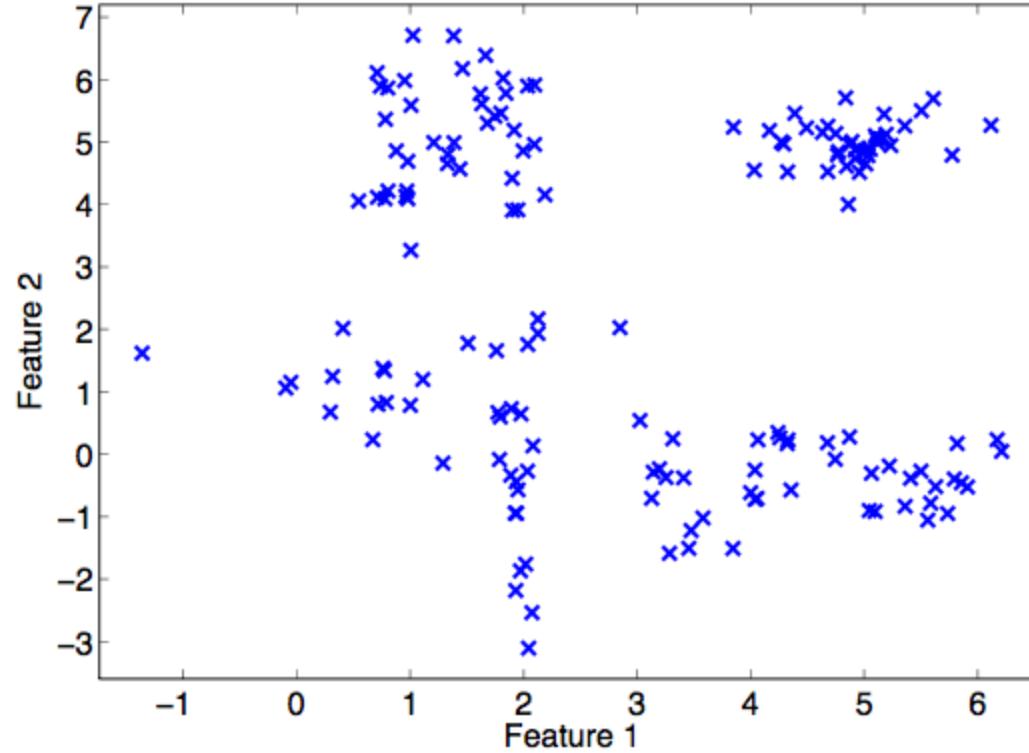
# Unsupervised learning

Gosia Migut

# Topics Layout



# Unlabelled data: what now?



- Unsupervised learning: no labels/targets present

# Unsupervised learning

- Clustering
  - Discover structures in unlabelled data
- Dimensionality reduction
  - does not use information about the labels

# Dimensionality reduction

- Typically, data sets are *high-dimensional*: each instance is described by many features.
- Why do we want to reduce data dimensionality?
- What does it mean to reduce dimensionality?
- How Principal Component Analysis reduces dimensionality?

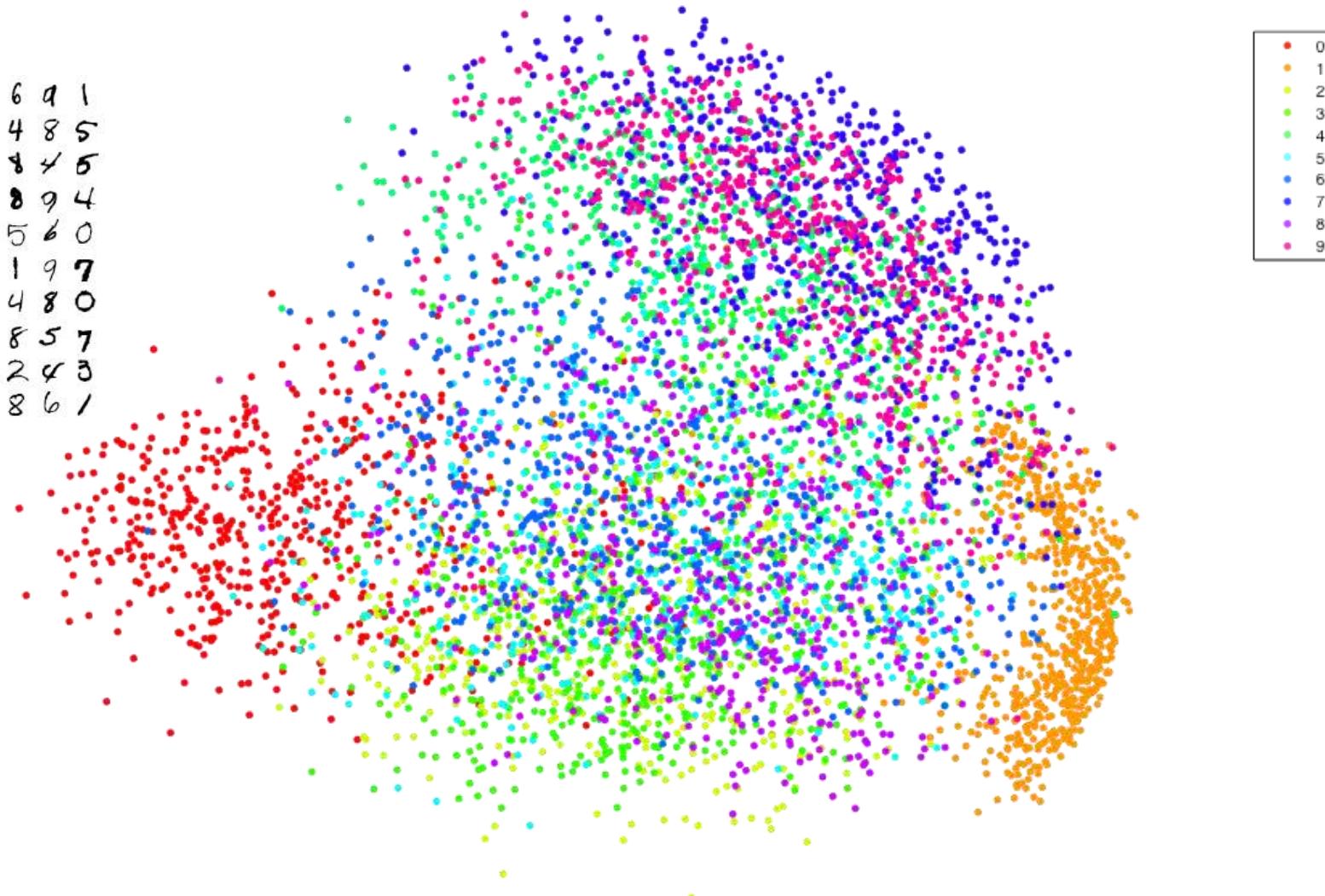
# Dimensionality reduction

- Why do we want to reduce data dimensionality?
  - Make storage or processing of data easier
  - (Visual) discovery of hidden structure in the data
  - Remove redundant and noisy features
  - Intrinsic dimensionality might be smaller

# Dimensionality reduction

## *Visual discovery of data structure*

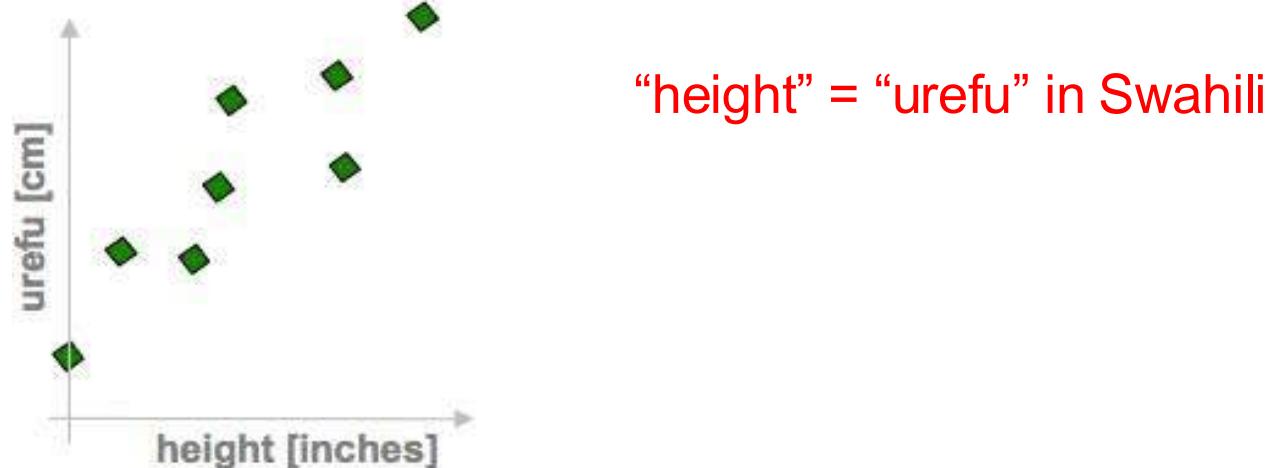
3	6	8	1	7	9	6	6	9	1
6	7	5	7	8	6	3	4	8	5
2	1	7	9	7	1	2	8	4	6
4	8	1	9	0	1	8	8	9	4
7	6	1	8	6	4	1	5	6	0
7	5	9	2	6	5	8	1	9	7
1	2	2	2	2	3	4	4	8	0
0	2	3	8	0	7	3	8	5	7
0	1	4	6	4	6	0	2	4	3
7	1	2	8	7	6	9	8	6	1



# Dimensionality reduction

## *Redundant features*

- Get a population, predict some property
  - instances represented as {urefu, height} pairs
  - what is the dimensionality of this data?



# Dimensionality reduction

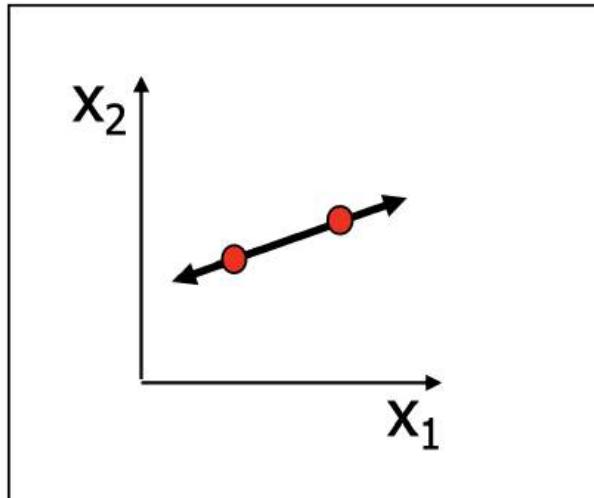
## *Redundant features*

- Data points from different geographic areas over time:
  - $X_1$ : # of skidding accidents
  - $X_2$ : # of burst water pipes
  - $X_3$ : # of snow-plow expenditures
  - $X_4$ : # of forest fires
  - $X_5$ : # of patients with heat stroke

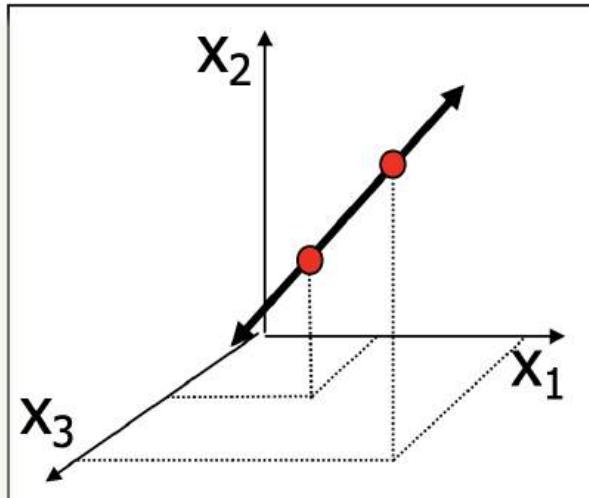
Temperature?

# Dimensionality reduction

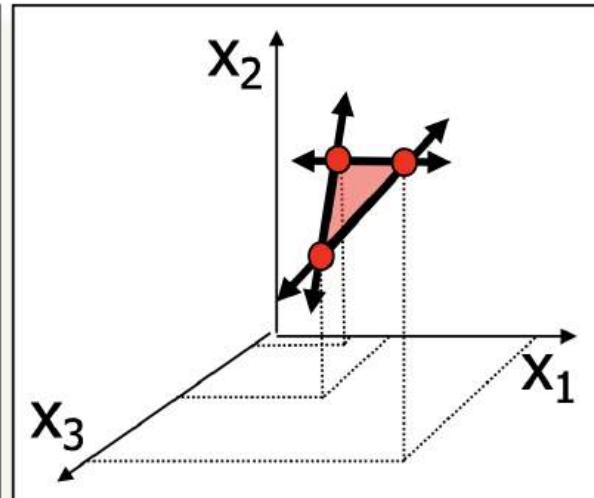
## *Intrinsic dimensionality*



2 objects, 2 dimensions  
→ 1 dimension



2 objects, 3 dimensions  
→ 1 dimension

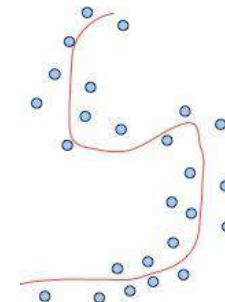
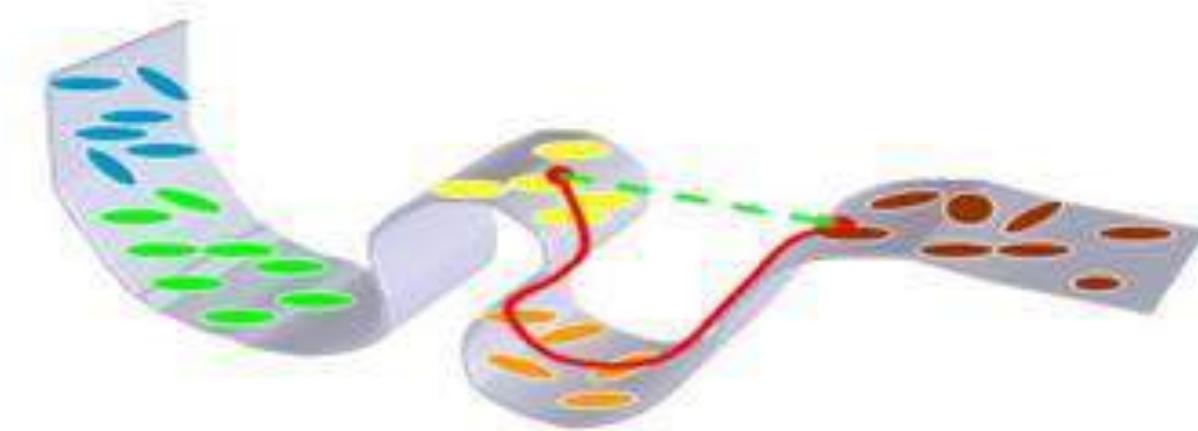


3 objects, 3 dimensions  
→ 2 dimension

# Dimensionality reduction

## *Intrinsic dimensionality*

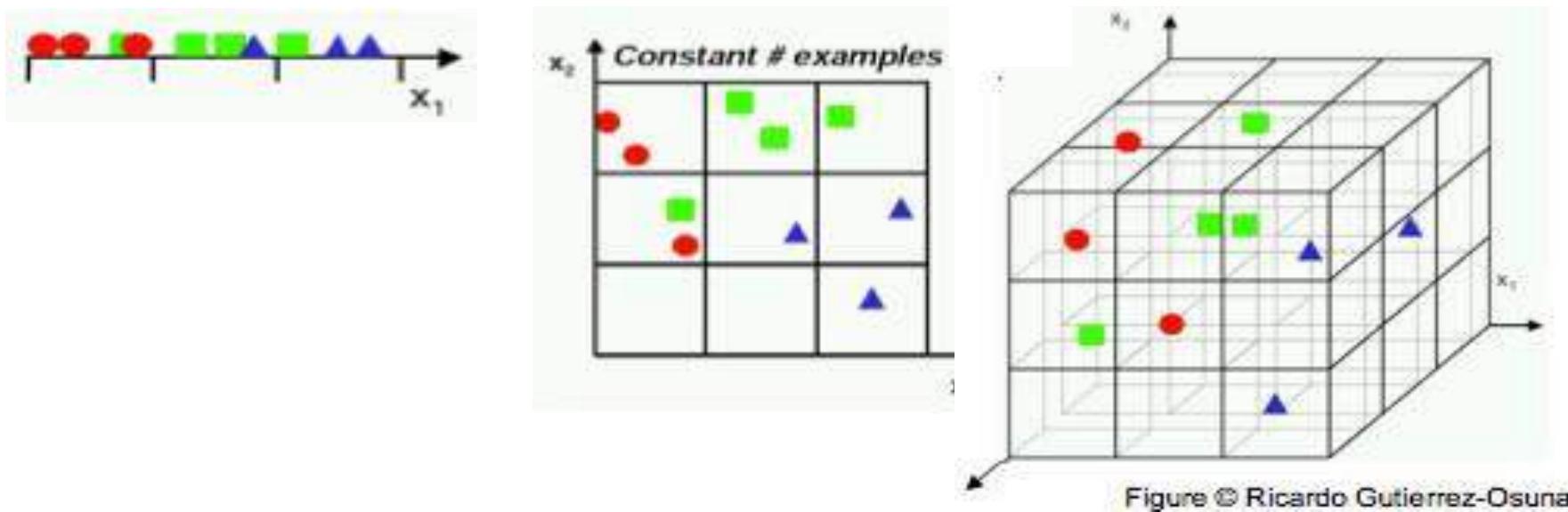
- Data may also be spread across a non-linear lower dimensional space (manifold)



# Dimensionality reduction

## *Curse of dimensionality*

- As dimensionality grows: fewer observations per region
  - 1d: 3 regions, 2d:  $3^2$  regions, 1000d – hopeless
  - statistics need repetition

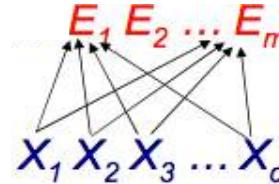


# Dimensionality reduction

- Typically, data sets are *high-dimensional*: each instance is described by many features.
- Why do we want to reduce data dimensionality?
- **What does it mean to reduce dimensionality?**
- How Principal Component Analysis reduces dimensionality?

# Reducing dimensionality: methods

- Feature selection
  - pick a subset of the original dimensions  $X_1 X_2 X_3 \dots X_{d-1} X_d$
  - Use domain knowledge
  - Use statistics-based selection methods
- Feature extraction
  - construct a new set of dimensions  $E_i = f(X_1 \dots X_d)$

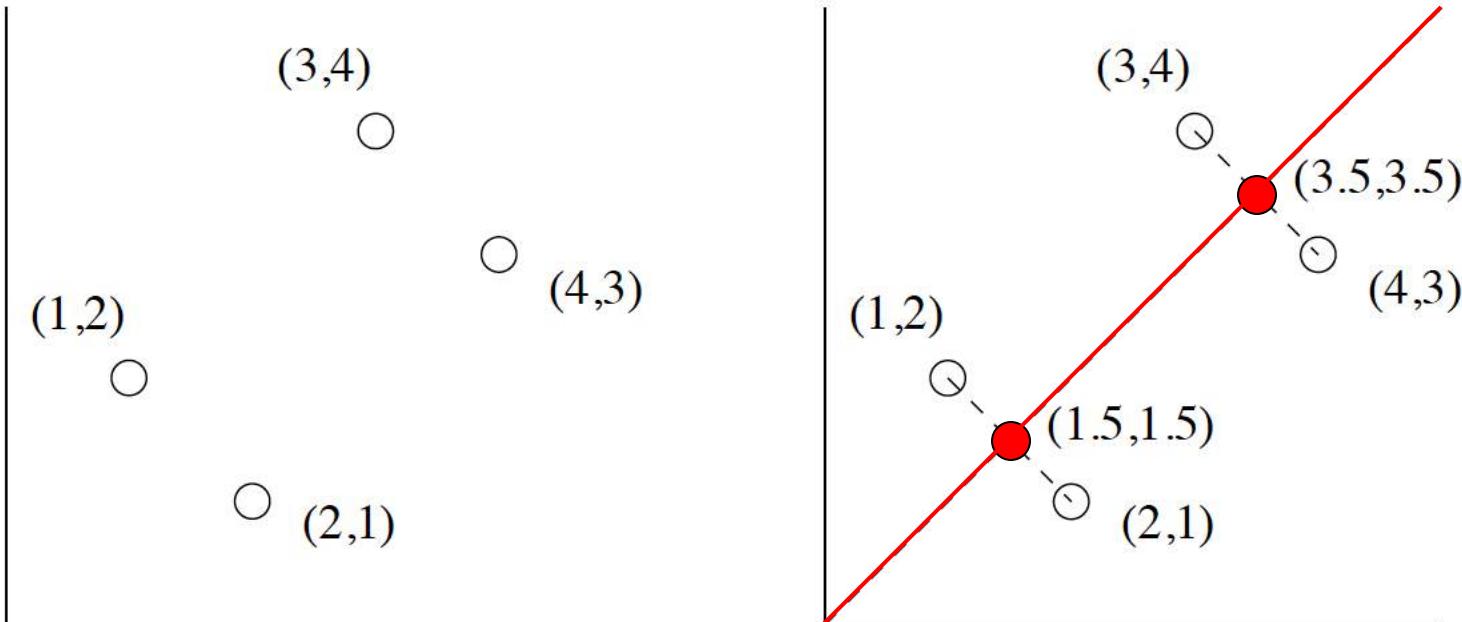


- (linear) combinations of original
- whilst *preserving the structure* in the original data

# Reducing dimensionality

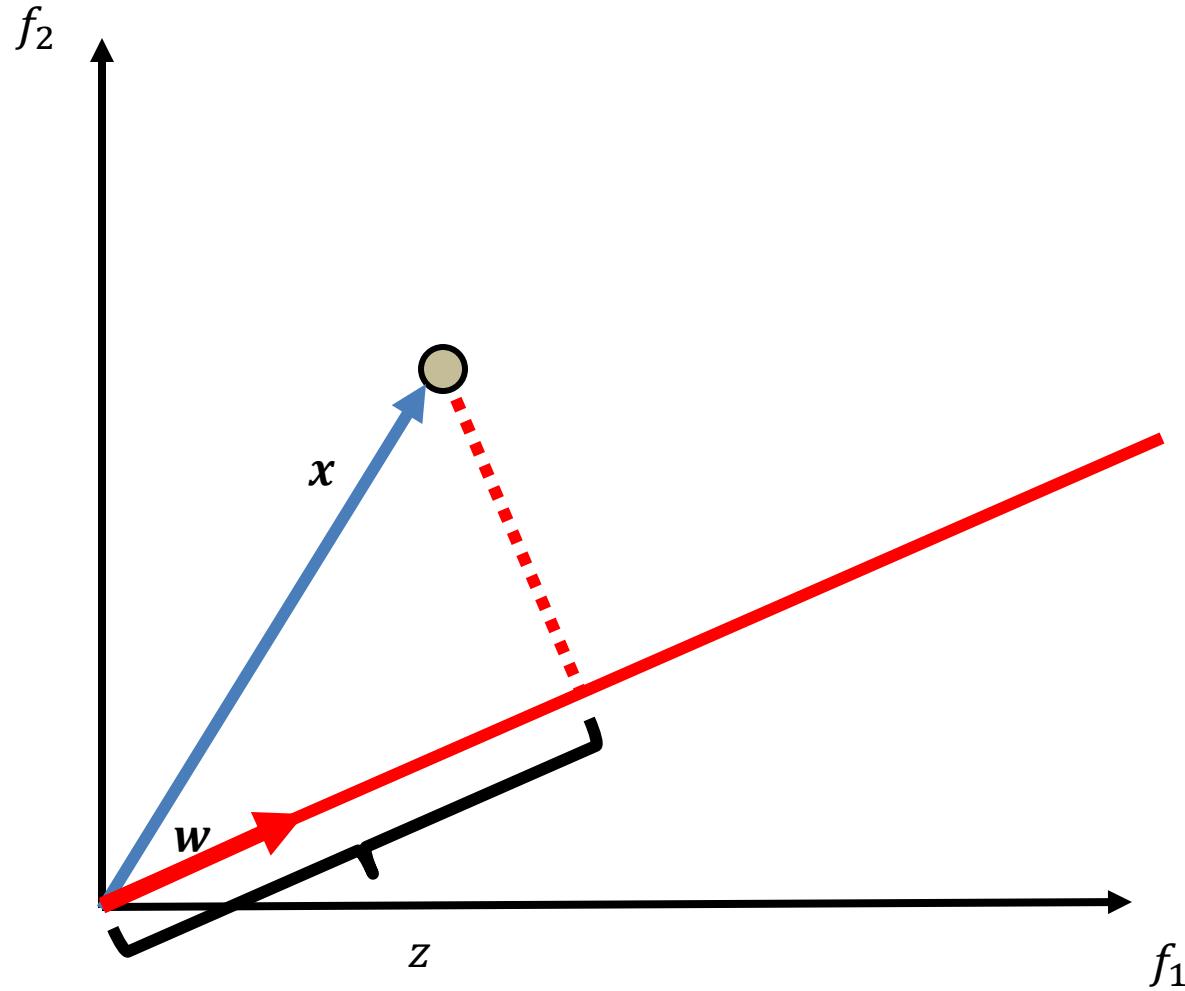
## Feature extraction

- Many important dimensionality reduction techniques are *linear* techniques
- These project the data onto a *linear subspace of lower dimensionality* (e.g. *Principal Components Analysis*)



# Reminder how to project a vector

- $z = w^T x$



# Dimensionality reduction

- Typically, data sets are *high-dimensional*: each instance is described by many features.
- Why do we want to reduce data dimensionality?
- What does it mean to reduce dimensionality?
- **How Principal Component Analysis reduces dimensionality?**

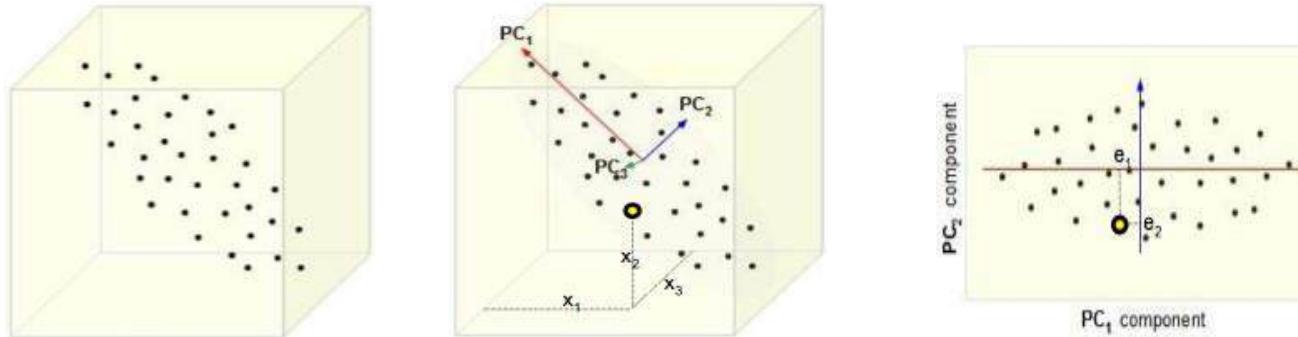
# Principal components analysis

# Principal Components Analysis

- Principal Components Analysis (PCA) maps the data onto a *linear subspace*, such that the ***variance*** of the projected data is ***maximized***.

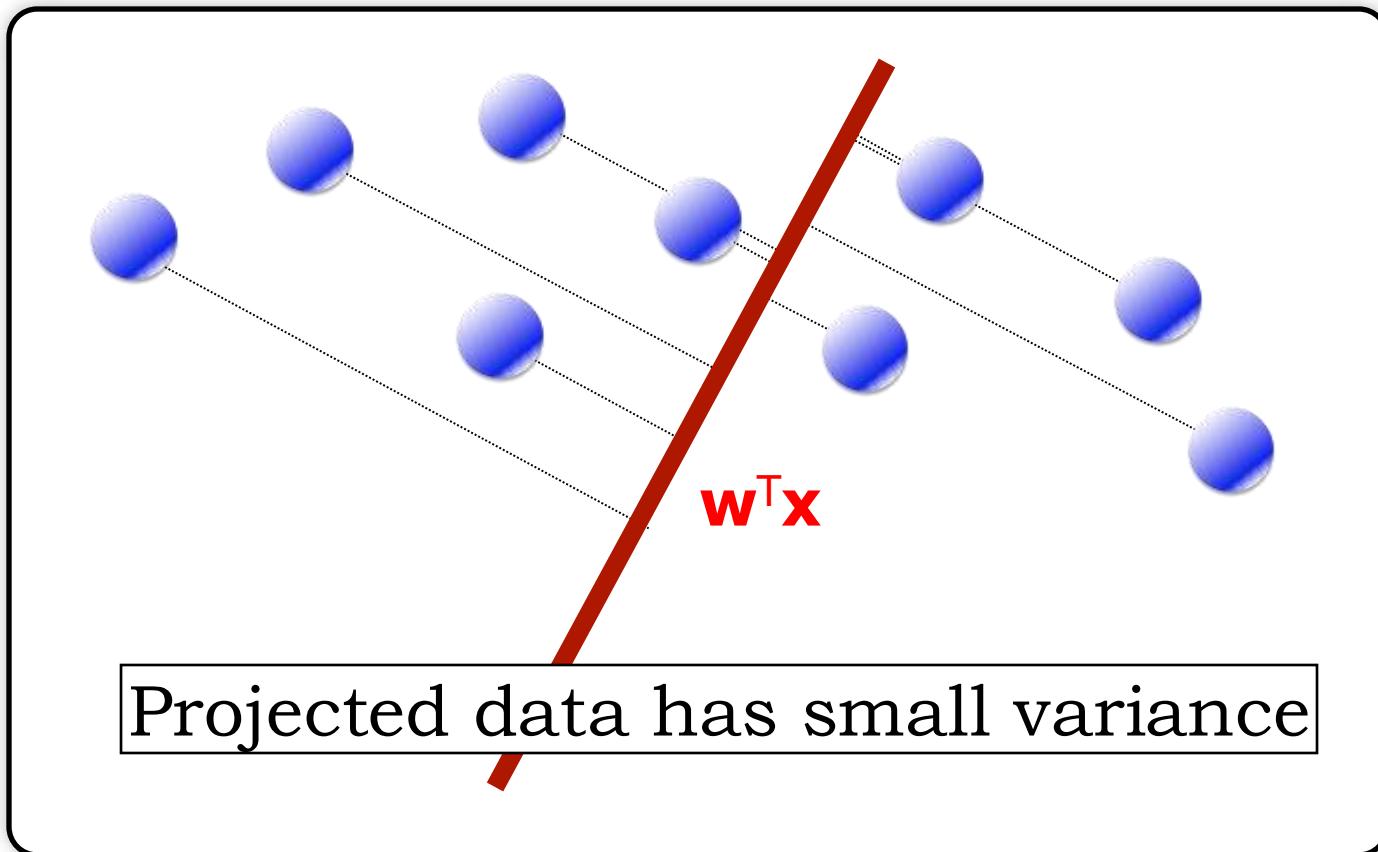
# PCA overview

- Defines a set of principal components
  - 1<sup>st</sup>: direction of the greatest variability in the data
  - 2<sup>nd</sup>: perpendicular to 1<sup>st</sup>, greatest variability of what's left
  - ... and so on until d (original dimensionality)
- First  $m$  components become  $m$  new dimensions
  - change coordinates of every data point to these dimensions



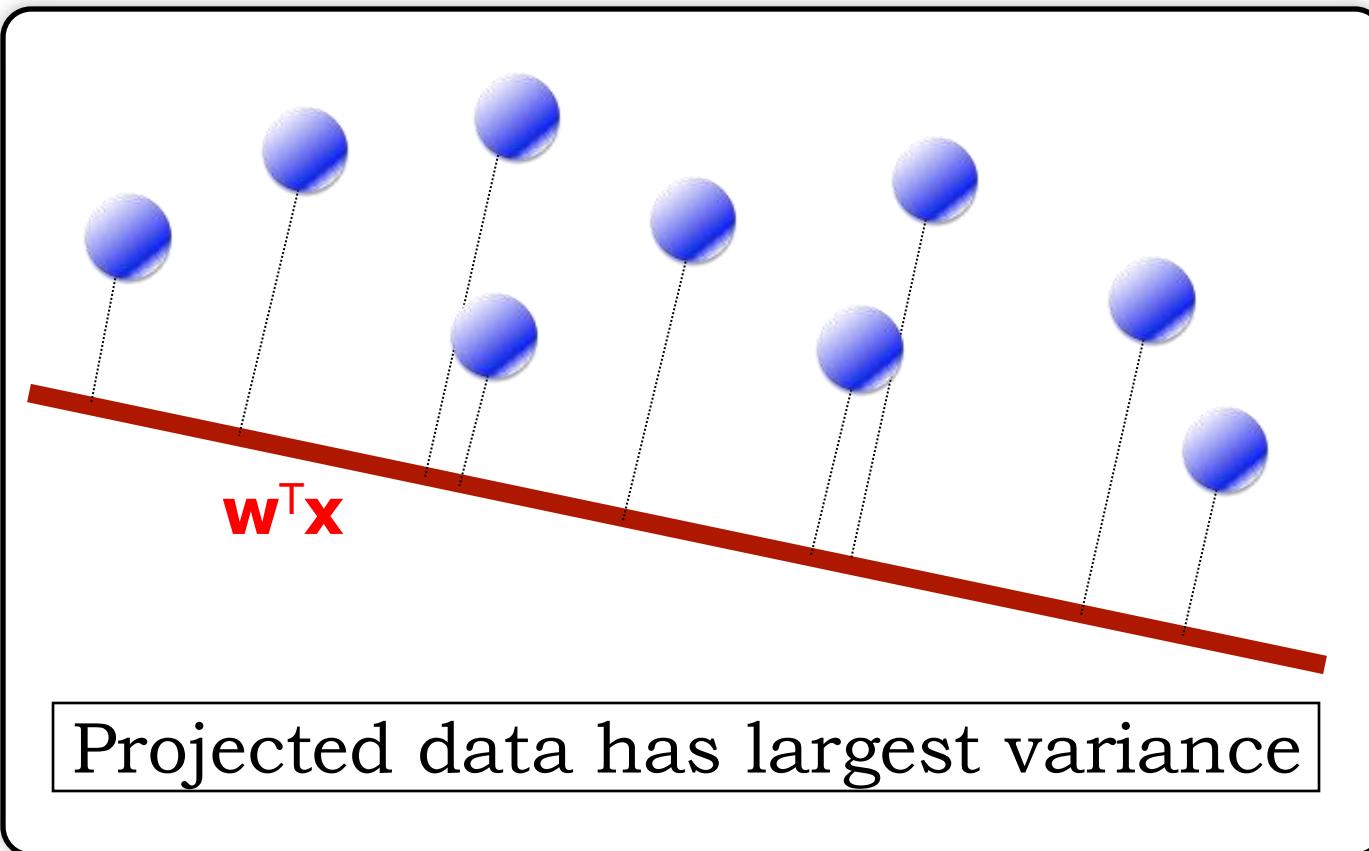
# Principal components analysis

- Principal Components Analysis maps the data onto a *linear subspace*, such that the *variance* of the projected data is *maximized*:



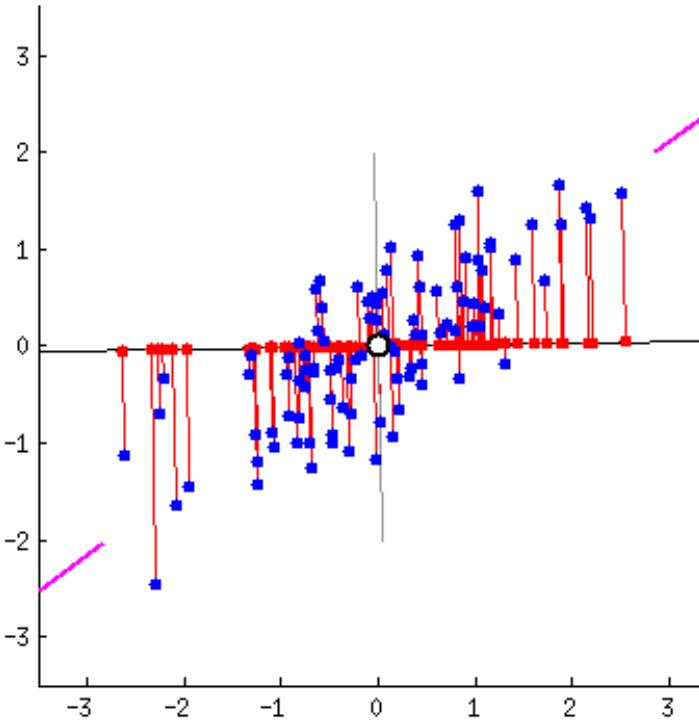
# Principal components analysis

- Principal Components Analysis maps the data onto a *linear subspace*, such that the *variance* of the projected data is *maximized*:



# Principal components analysis

- Principal Components Analysis maps the data onto a *linear subspace*, such that the *variance* of the projected data is *maximized* (equivalent to *smallest reconstruction error*):



# PCA Optimization Problem

- Principal Components Analysis (PCA) maps the data onto a *linear subspace*, such that the *variance* of the projected data is maximized.
- So PCA performs maximization:
$$\max_{\|w\|^2=1} \text{var}(w^T x)$$
- Constrain:  $w^T w = 1 \Rightarrow \|w\|^2 = 1$ 
  - requires  $w$  (direction of projection) to be unit vector; solves scaling problem and guarantees unique solution

# Zero-mean data

- Recall the definition of variance:  $var(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \left( x_n - \frac{1}{N} \sum_{n=1}^N x_n \right)^2$
- For PCA shift your data to zero-mean
  - For each feature, calculate the mean, subtract the mean from each data point in that feature.
- Helps with:
  - Mean does not affect the calculation of variance
  - Simplifies covariance matrix:  $M = \frac{1}{n} XX^T$

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

# Covariance matrix

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$$

- The *covariance matrix* is the matrix with all pairwise covariances:

$$M = \begin{bmatrix} \mathbb{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathbb{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathbb{E}[(X_1 - \mu_1)(X_D - \mu_D)] \\ \mathbb{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathbb{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathbb{E}[(X_2 - \mu_2)(X_D - \mu_D)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}[(X_D - \mu_D)(X_1 - \mu_1)] & \mathbb{E}[(X_D - \mu_D)(X_2 - \mu_2)] & \cdots & \mathbb{E}[(X_D - \mu_D)(X_D - \mu_D)] \end{bmatrix}$$

- If data is *zero-mean*, the covariance matrix is simply:  $M = \frac{1}{n} XX^T$

# PCA Optimization Problem

$$var(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \left( x_n - \frac{1}{N} \sum_{n=1}^N x_n \right)^2$$

- Our objective is to maximize variance:

$$\max_{\|w\|^2=1} var(w^T x)$$

- Assuming zero-mean data:

$$var(w^T x) = \frac{1}{n} (w^T x)(w^T x) = \frac{1}{n} (w^T x)(w^T x)^T = \frac{1}{n} w^T x x^T w = w^T M w$$

# Lagrange multiplier

- Introduce a Lagrange multiplier  $\lambda$  to incorporate the constraint into our optimization

$$L(w, \lambda) = w^T M w - \lambda(w^T w - 1)$$

- $w^T M w$  is the quantity we want to maximize (variance)
- $w^T w - 1 = 0$  is the constraint that  $w$  must be a unit vector.
- It penalizes any deviation from the constraint
  - if the constraint is violated (i.e.,  $w^T w \neq 1$ ), it will either increase or decrease the value of the Lagrangian

# PCA Optimization Problem

- Enforce constraint using *Lagrange multiplier*:

$$\max_{\|w\|^2=1} \text{var}(w^T x) = \max_{w, \lambda} w^T M w - \lambda(w^T w - 1)$$

- Set gradient with respect to  $w$  to zero:  $\frac{\partial_{w^T M w - \lambda(w^T w - 1)}}{\partial w} = 0$   
 $2Mw - 2\lambda w = 0$

$$Mw = \lambda w$$

# Eigenvalues & eigenvectors: Definition

- $M$  square matrix,  $\lambda$  constant,  $\mathbf{e}$  a non-zero column vector
- $\lambda$  is an eigenvalue of  $M$  and  $\mathbf{e}$  is the corresponding eigenvector of  $M$  if

$$M\mathbf{e} = \lambda\mathbf{e}$$

- Avoiding ambiguity regarding length: eigenvector to be *unit vector*
- $\lambda$  and  $\mathbf{e}$  form eigenpairs
- Watch: 3blue1brown: **Eigenvectors and eigenvalues | Essence of linear algebra, chapter 14**

# Principal components analysis

- *Principal components* are given by the eigenvectors of the covariance matrix
- First principal component is given by the eigenvector with the corresponding highest eigenvalue, etc.

# Eigenvalues & eigenvectors: Example

- Let  $M$  be a covariance matrix

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$$

- One of eigenvectors of  $M$  is

$$\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$$

- Corresponding eigenvalue is 7, since

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix} = 7 \begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$$

- Eigenvector is indeed unit vector

$$(1/\sqrt{5})^2 + (2/\sqrt{5})^2 = 1/5 + 4/5 = 1$$

# How to find eigenpairs?

## Pivotal condensation

- Restate definition eigenpair  $M\mathbf{e} = \lambda\mathbf{e}$  as

$$(M - \lambda I)\mathbf{e} = \mathbf{0}$$

- For this to hold the determinant of  $(M - \lambda I)$  must be 0
- Determinant of  $(M - \lambda I)$  is an  $n$ -th degree polynomial from which we can get the  $n$  values for  $\lambda$  that are eigenvalues of  $M$

# Eigenpairs: Pivotal condensation (example)

- Set  $M$  to  $\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$
- Then  $M - \lambda I$  is  $\begin{bmatrix} 3 - \lambda & 2 \\ 2 & 6 - \lambda \end{bmatrix}$
- Determinant is  $(3 - \lambda)(6 - \lambda) - 4$
- Setting to zero, solving equation  $\lambda^2 - 9\lambda + 14 = 0$
- Gives solutions  $\lambda = 7$  and  $\lambda = 2$  being principal eigenvalues
- Let  $\mathbf{e}$  be vector of unknowns  $\begin{bmatrix} x \\ y \end{bmatrix}$
- Solve  $\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 7 \begin{bmatrix} x \\ y \end{bmatrix}$

# Eigenpairs: Pivotal condensation (example)

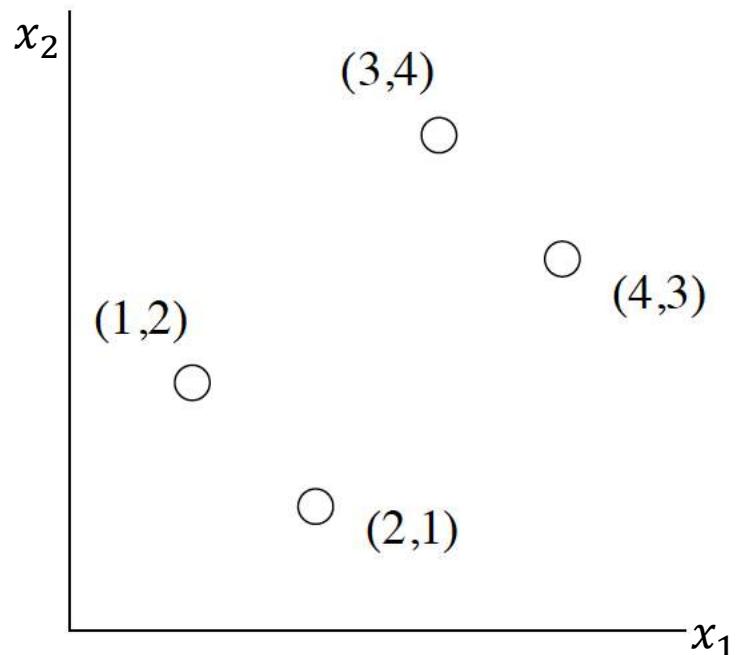
- Two equations: 
$$\begin{bmatrix} 3x + 2y & = & 7x \\ 2x + 6y & = & 7y \end{bmatrix}$$
- Both saying the same thing  $y = 2x$
- Possible eigenvector: 
$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$
- Make unit vector (divide by length): 
$$\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$$
- Second eigenvalue: repeat with  $\lambda = 2$
- Equation becomes:  $x = -2y$
- Second eigenvector: 
$$\begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$$

# Principal components: PCA example

- *Principal components* are given by the eigenvectors of the covariance matrix

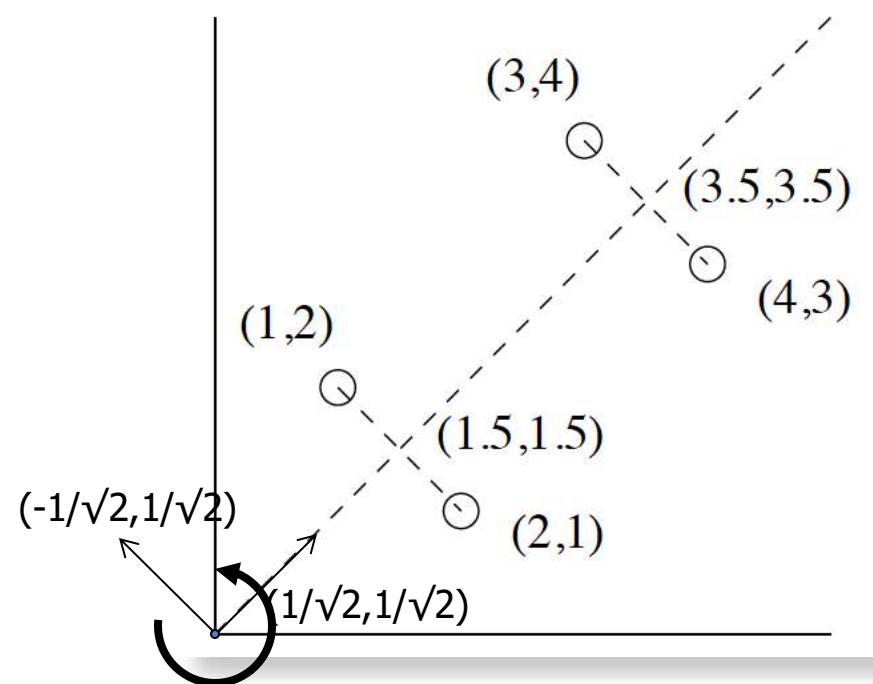
$$Me = \lambda e \quad M = \frac{1}{n} XX^T$$

- Perform PCA for:

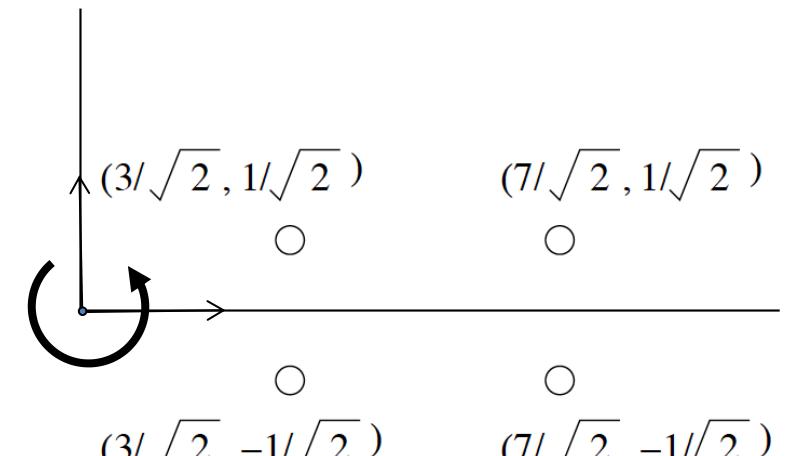


# Principal component analysis (example)

- First point  $[1,2]$  transformed into  $[3/\sqrt{2}, 1/\sqrt{2}]$



Original points, eigenvectors,  
projections

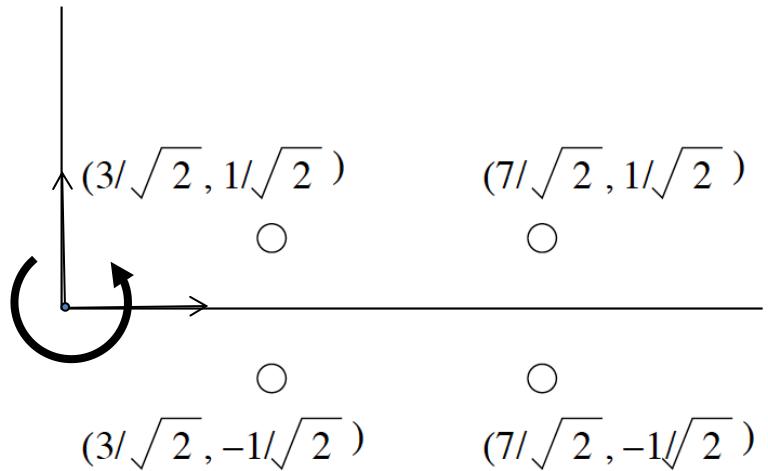


New coordinate system

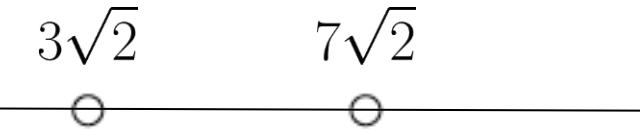
# Principal component analysis (example)

- From 2d to 1d

New coordinate system



$XE_1$   
New coordinate system  
1 dimensional !

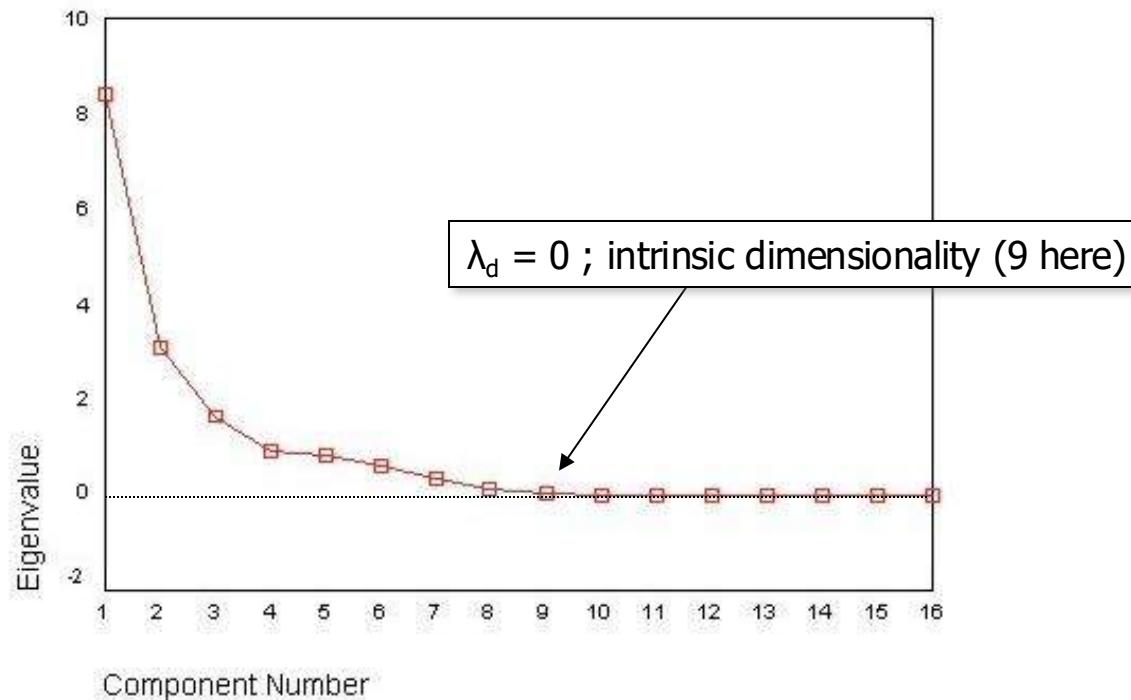


# PCA in a nutshell

- $X$  matrix whose rows represent (zero-mean) points in Euclidean space
- Compute covariance  $XX^T$  and its eigenpairs
- $E$  matrix whose columns are the eigenvectors, ordered as largest eigenvalues first
- $X^TE$ : points of  $X$  transformed into new coordinate space
  - First axis (largest eigenvalue) most significant
  - Second axis (second eigenpair), next most significant
- Let  $E_k$  be first  $k$  columns of  $E$
- Then  $X^TE_k$  is  $k$ -dimensional representation of  $X$

# PCA scree plot

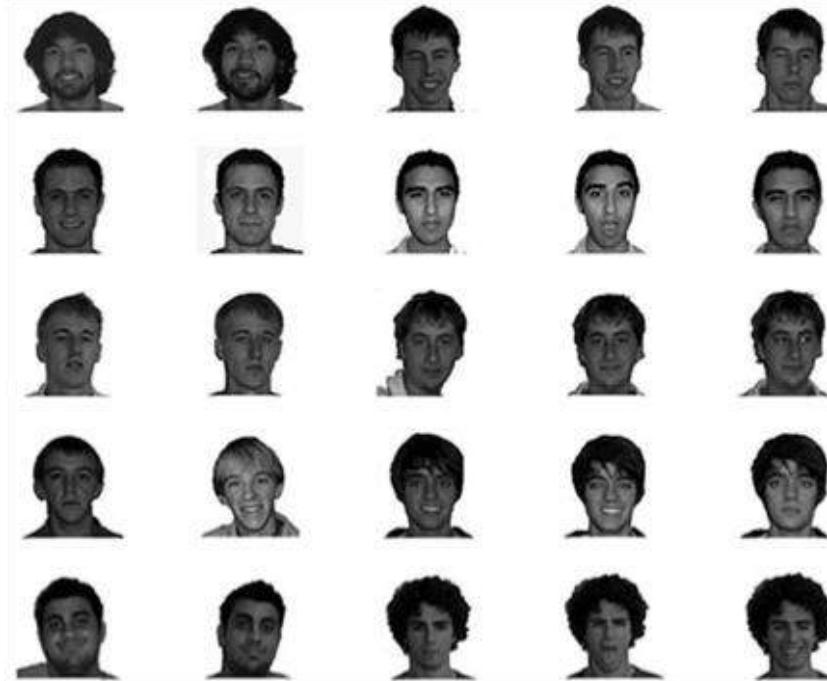
- Scree plot of eigenvalues shows amount of variance retained by the eigenvectors (*principal components, PCs*):



- First  $K$  PCs explain  $\frac{\sum_{d=1}^K \lambda_d}{\sum_{d'=1}^D \lambda_{d'}} \times 100\%$  of variance

# Eigenfaces

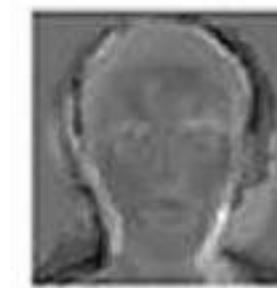
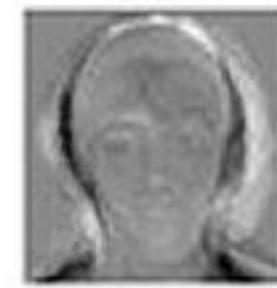
- Suppose we are applying PCA on the following set of face images:



- Image is matrix; *but* represented as a row vector !
- Eigenvectors also row vector, so eigenvector is also an image !

# Eigenfaces

- Example of first eigenvectors of set of face images (faces were aligned):



# Principal components analysis

- Since we have projected onto a subspace, we can reconstruct the data in the original data space by performing the inverse of the projection:

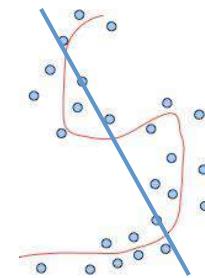
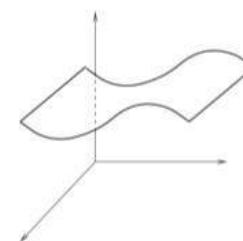
$$\hat{\mathbf{x}} = \mathbf{w}\mathbf{w}^T\mathbf{x}$$

- Example reconstructions of face images and digits (using 30D PCA subspace):



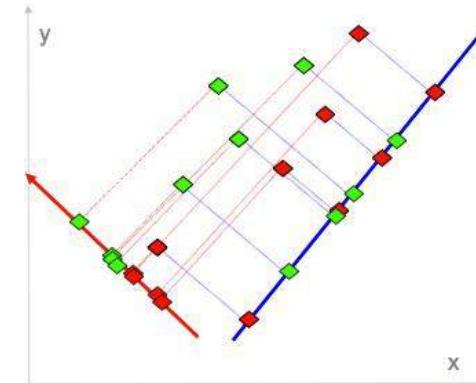
# PCA: practical issues

- Covariance extremely sensitive to large values
  - Multiply some dimensions by 1000
    - Dominates covariance
    - Becomes a principal component
  - Normalize each dimension to zero mean and unit variance:  $x' = \frac{x - \mu}{\sigma}$
- PCA assumes underlying subspace is linear
  - 1d: straight line, 2d: plane



# PCA and classification

- PCA is unsupervised
  - maximizes overall variance of the data along a small set of directions
  - does not know anything about class labels
  - can pick direction that makes it hard to separate classes
- Discriminative approach
  - look for a dimension that makes it easy to separate classes

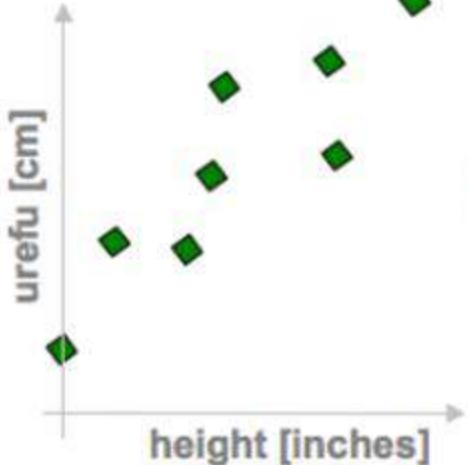


# PCA pros and cons

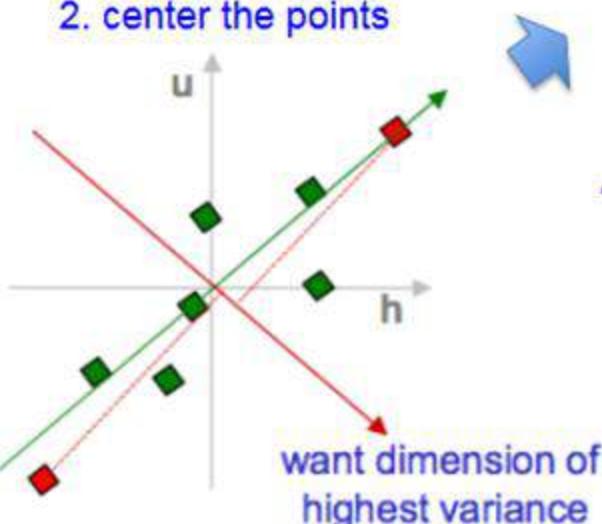
- Pros
  - reflects our intuitions about the data
  - dramatic reduction in size of data
    - faster processing (as long as reduction is fast), smaller storage
- Cons
  - too expensive for many applications (Twitter, web)
  - understand assumptions behind the methods (linearity etc.)

# PCA in a nutshell

1. correlated hi-d data  
("urefu" means "height" in Swahili)



2. center the points



3. compute covariance matrix

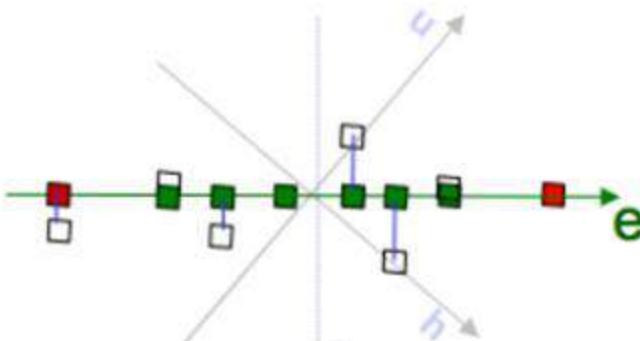
$$\begin{matrix} h & u \\ \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} & \rightarrow \text{cov}(h,u) \end{matrix}$$

4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

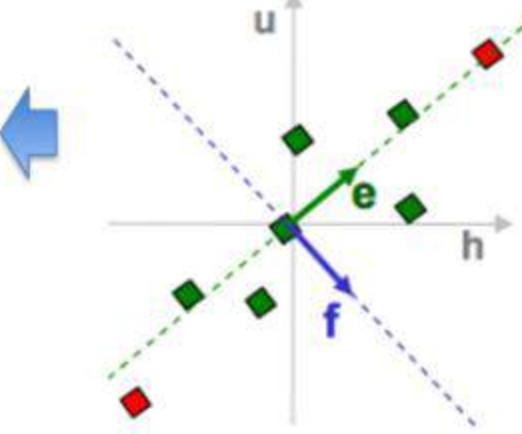
7. uncorrelated low-d data



6. project data points to those eigenvectors

$$x_e = x^T e = \sum_{j=1}^d x_{ij} e_j$$

5. pick  $m < d$  eigenvectors w. highest eigenvalues



# Recap

- Dimensionality reduction builds a condensed data representation
- This removes redundant or noisy features, and identifies correlations
- Principal components analysis projects data onto the principal eigenvectors of the covariance matrix: maximizes variance of the projection

# PCA resources

- <http://setosa.io/ev/principal-component-analysis/>
- <http://peterbloem.nl/blog/pca>

# Shallow Introduction to Deep Learning



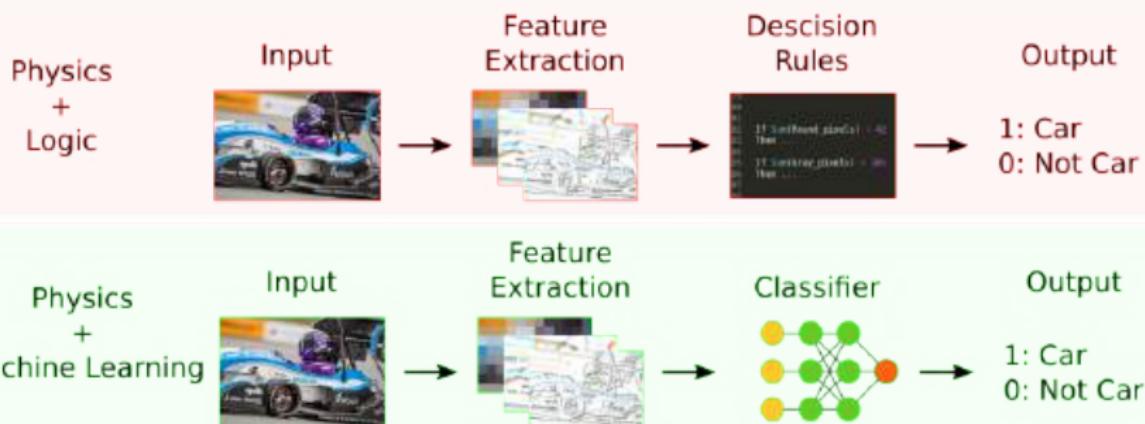
Delft University of Technology

Lecturer: Jan van Gemert

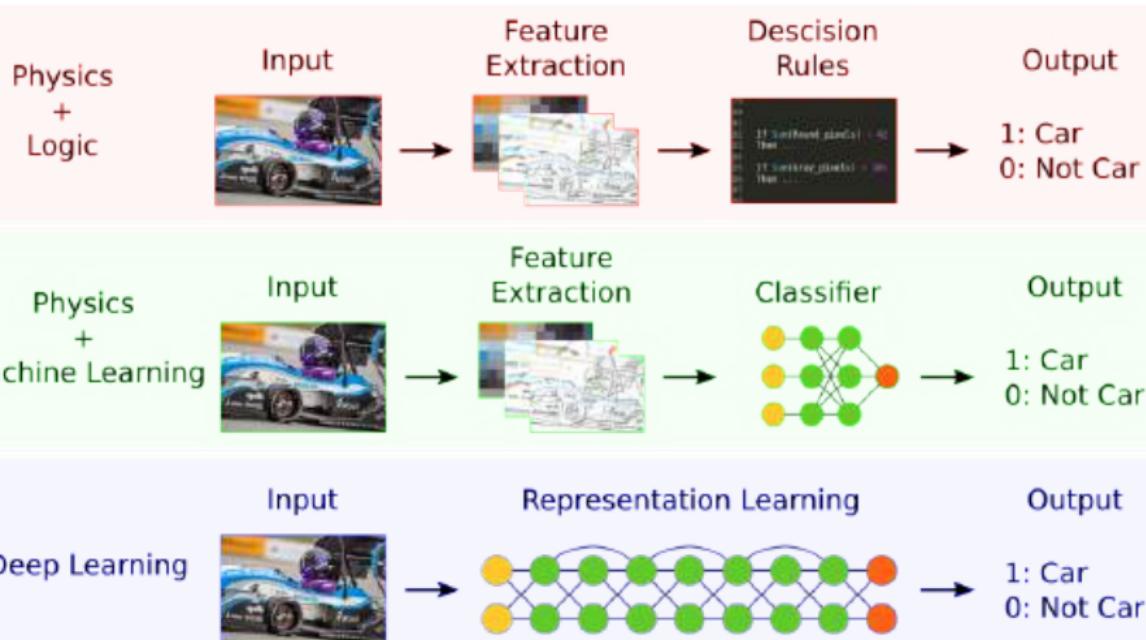
# Feature extraction vs deep learning



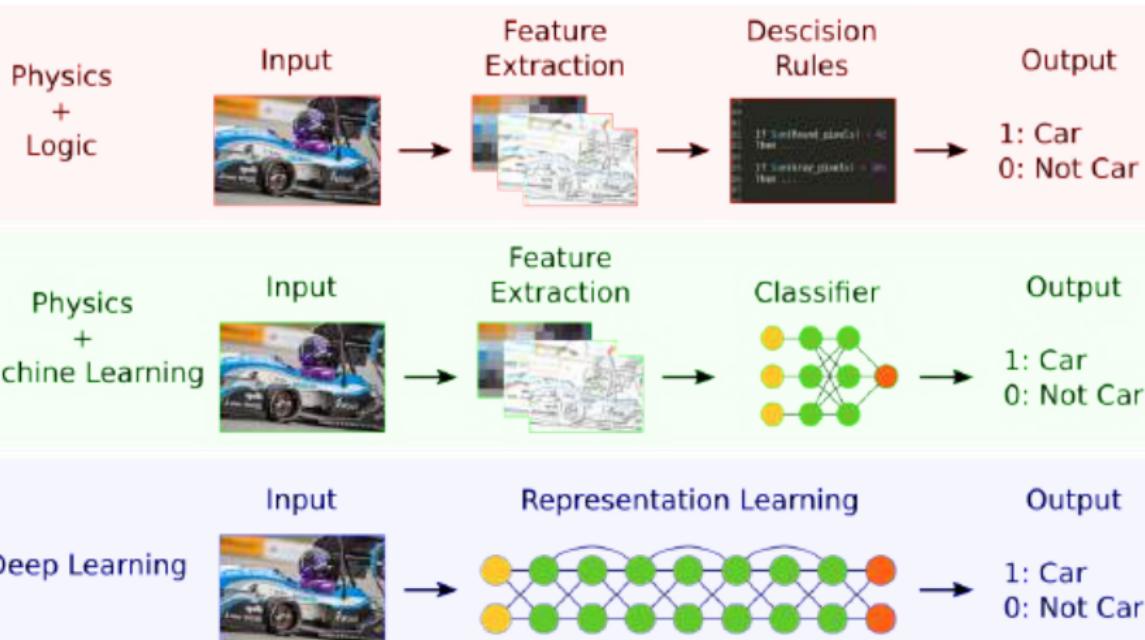
# Feature extraction vs deep learning



# Feature extraction vs deep learning



# Feature extraction vs deep learning



End-to-End learning: End goal (output) used to learn feature extraction (input)

# A bit of image processing

Q: How to get rid of noisy pixels?



# A bit of image processing

Q: How to get rid of noisy pixels?



A: Simple solution: replace pixel by neighborhood average

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$


# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(x, y)$


# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(x, y)$

			0						

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$

			0							

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$

			0	10						

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$

			0	10						

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	90	0	0
0	0	0	90	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$

	0	10	20							

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(x, y)$

	0	10	20						

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(x, y)$

	0	10	20	30					

# Moving Neighborhood Average

$F(x, y)$

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	0	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$

	0	10	20	30	30	30	20	10			
	0	20	40	60	60	60	40	20			
	0	30	60	90	90	90	60	30			
	0	30	50	80	80	90	60	30			
	0	30	50	80	80	90	60	30			
	0	20	30	50	50	60	40	20			
	10	20	30	30	30	30	20	10			
	10	10	10	0	0	0	0	0			

# Moving Neighborhood Average

Q: What do you notice?

$F(x, y)$

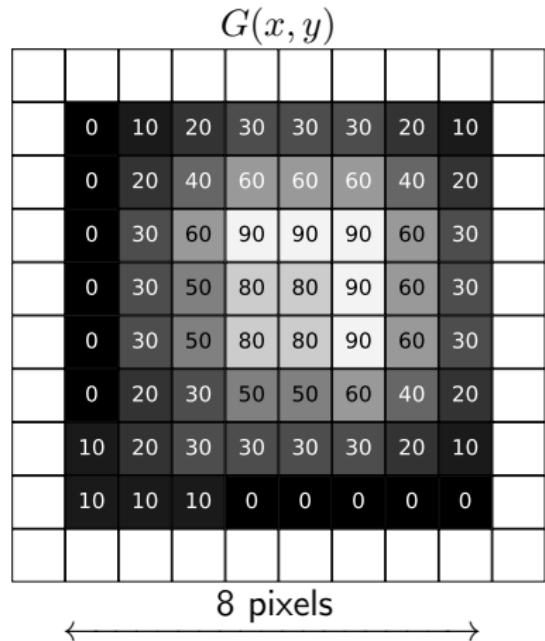
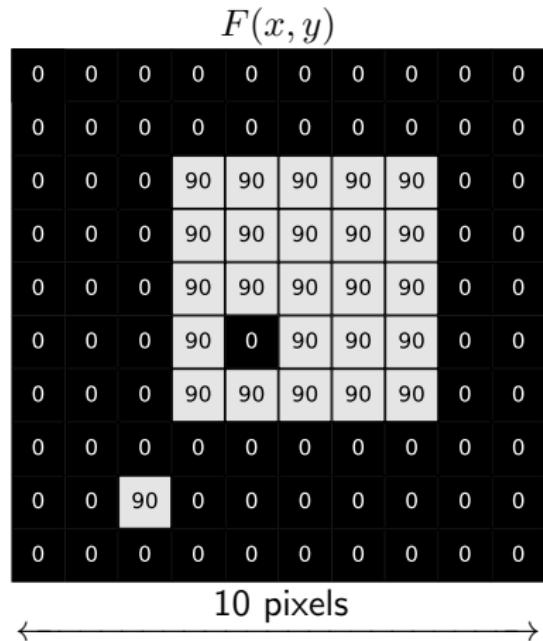
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	0	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

$G(x, y)$

	0	10	20	30	30	30	20	10			
	0	20	40	60	60	60	40	20			
	0	30	60	90	90	90	60	30			
	0	30	50	80	80	90	60	30			
	0	30	50	80	80	90	60	30			
	0	20	30	50	50	60	40	20			
	10	20	30	30	30	30	20	10			
	10	10	10	0	0	0	0	0			

# Moving Neighborhood Average

Q: What do you notice?



A: 2 pixels lost to boundary (1 on each side)

## Convolution

Chapter 9.1

Generalize to a kernel: **Multiply weights variables per pixel and sum**

Q: What values to fill in kernel  $H$  to obtain a moving neighborhood average?

# Convolution

## Chapter 9.1

Generalize to a kernel: **Multiply weights variables per pixel and sum**

$$F(x, y) \star H(x, y) = G(x, y)$$

Diagram illustrating the convolution operation:

Input image  $F(x, y)$  (3x9 grid):

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	0	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Kernel  $H(x, y)$  (3x3 grid):

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

Output image  $G(x, y)$  (3x9 grid):

			0	10	20	30		

Q: What values to fill in kernel  $H$  to obtain a moving neighborhood average?

# Practice with kernels

 $F(x, y)$  $\star \quad H(x, y) =$  $G(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

# Practice with kernels

 $F(x, y)$  $\star$   
 $H(x, y)$  $=$   
 $G(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$



# Practice with kernels

 $F(x, y)$  $\star$  $H(x, y)$  $=$  $G(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$

# Practice with kernels

 $F(x, y)$  $\star$   
 $H(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$

 $=$   
 $G(x, y)$ 

# Practice with kernels

 $F(x, y)$  $\star \quad H(x, y) =$  $G(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$

# Practice with kernels

 $F(x, y)$  $\star$   
 $H(x, y)$  $=$   
 $G(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$



# Practice with kernels

 $F(x, y)$  $\star \quad H(x, y) \quad =$  $G(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{matrix}$$

# Practice with kernels

 $F(x, y)$  $\star$   
 $H(x, y)$ 

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{matrix}$$

 $=$   
 $G(x, y)$ 

# Practice with kernels

(Red = positive; blue is negative)

$$F(x, y)$$



\*

$$H(x, y)$$



=

$$G(x, y)$$

# Practice with kernels

(Red = positive; blue is negative)

$$F(x, y)$$



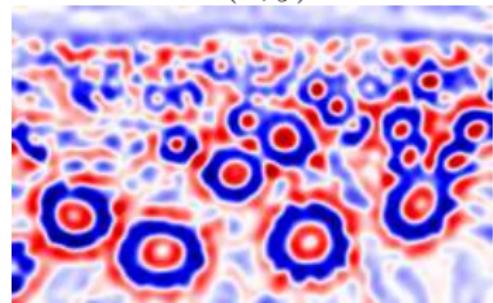
\*

$$H(x, y)$$



=

$$G(x, y)$$



# Practice with kernels

(Red = positive; blue is negative)

$$F(x, y)$$



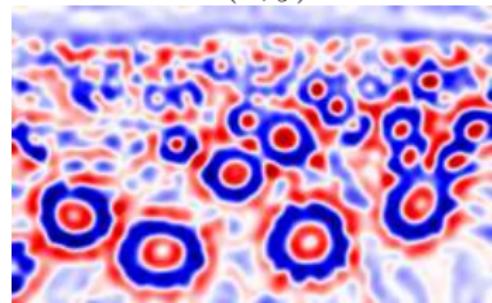
\*

$$H(x, y)$$



=

$$G(x, y)$$



Q: How about a smaller kernel?



# Practice with kernels

(Red = positive; blue is negative)

$$F(x, y)$$



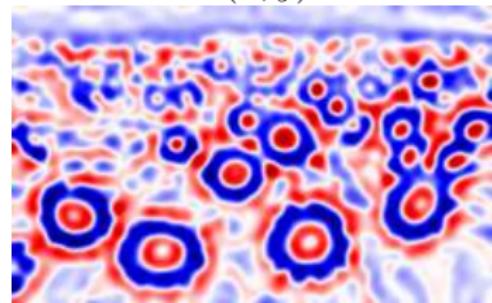
\*

$$H(x, y)$$

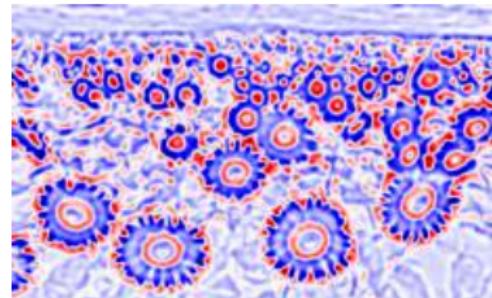


=

$$G(x, y)$$



Q: How about a smaller kernel?



# Practice with kernels

(Red = positive; blue is negative)

$$F(x, y)$$



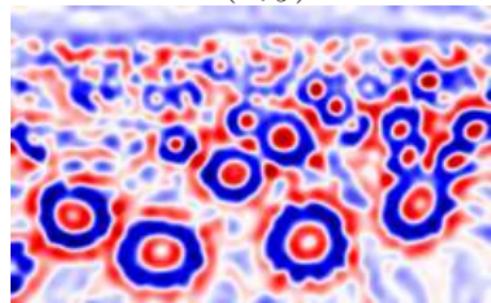
\*

$$H(x, y)$$

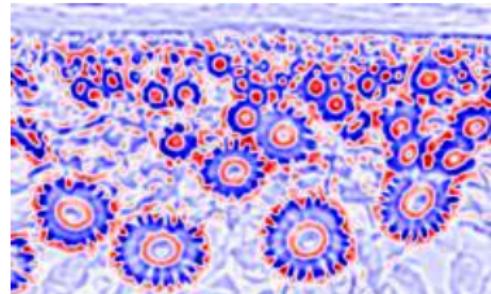


=

$$G(x, y)$$



Q: How about a smaller kernel?



Q: How about a smaller input image?

# Practice with kernels

(Red = positive; blue is negative)

$$F(x, y)$$



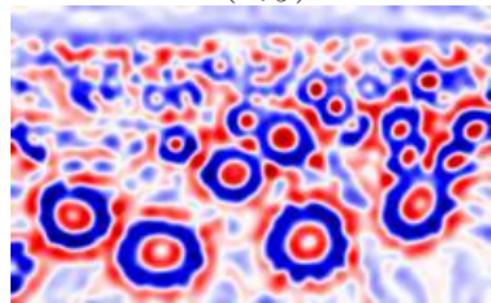
\*

$$H(x, y)$$

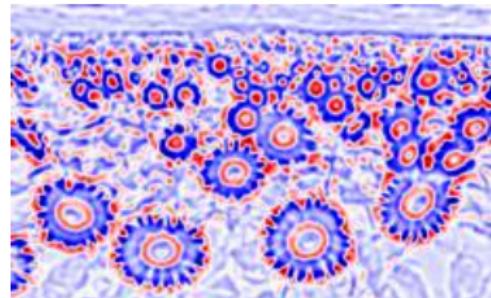


=

$$G(x, y)$$



Q: How about a smaller kernel?



Q: How about a smaller input image?

A: Smaller image = larger kernel.

# Where is Waldo?



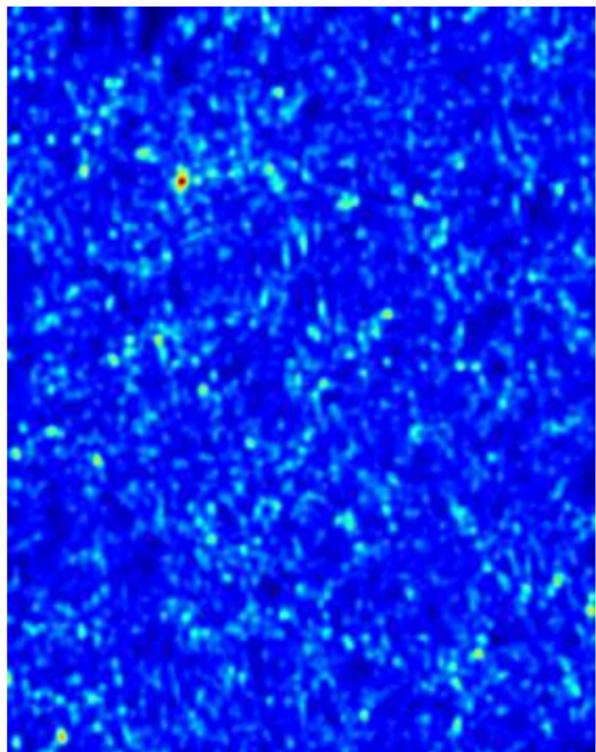
# Where is Waldo?



Use this normalized kernel:



# Where is Waldo?



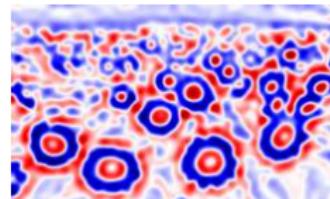
# Representation learning



\*



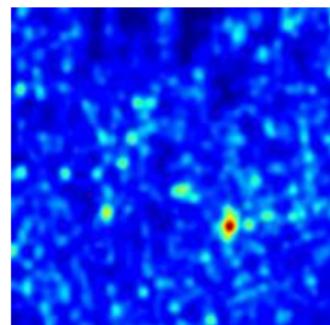
=



\*



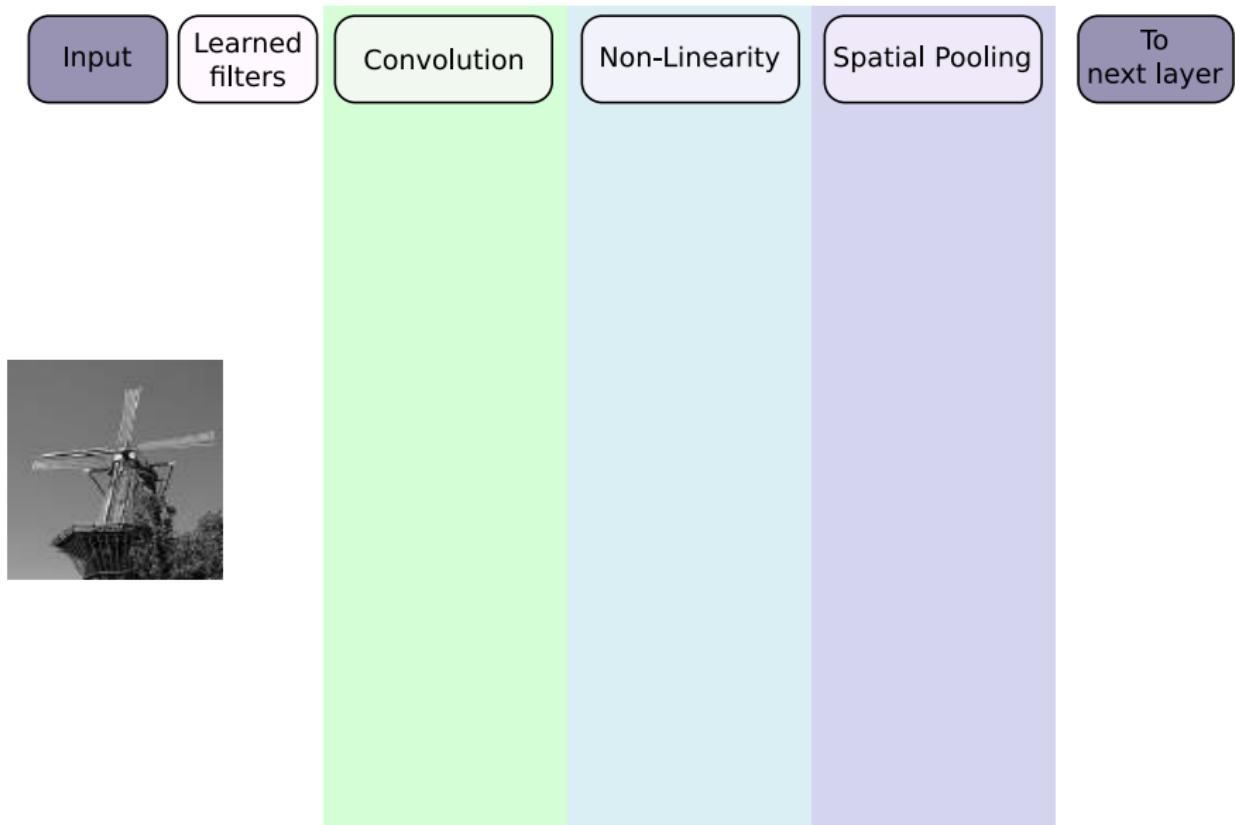
=



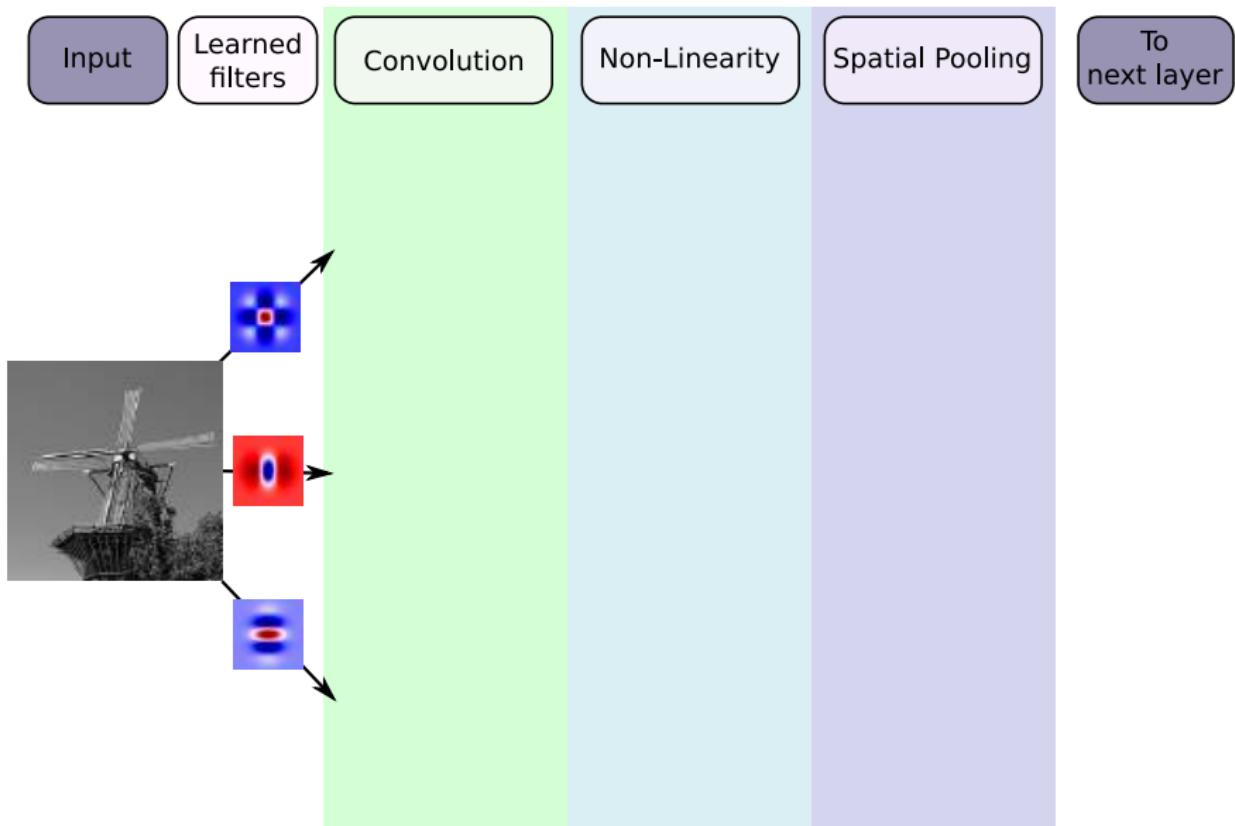
- Kernel weights are feature detectors
- Learning weights = Learning features
- Convnet learns the feature representation

# Questions?

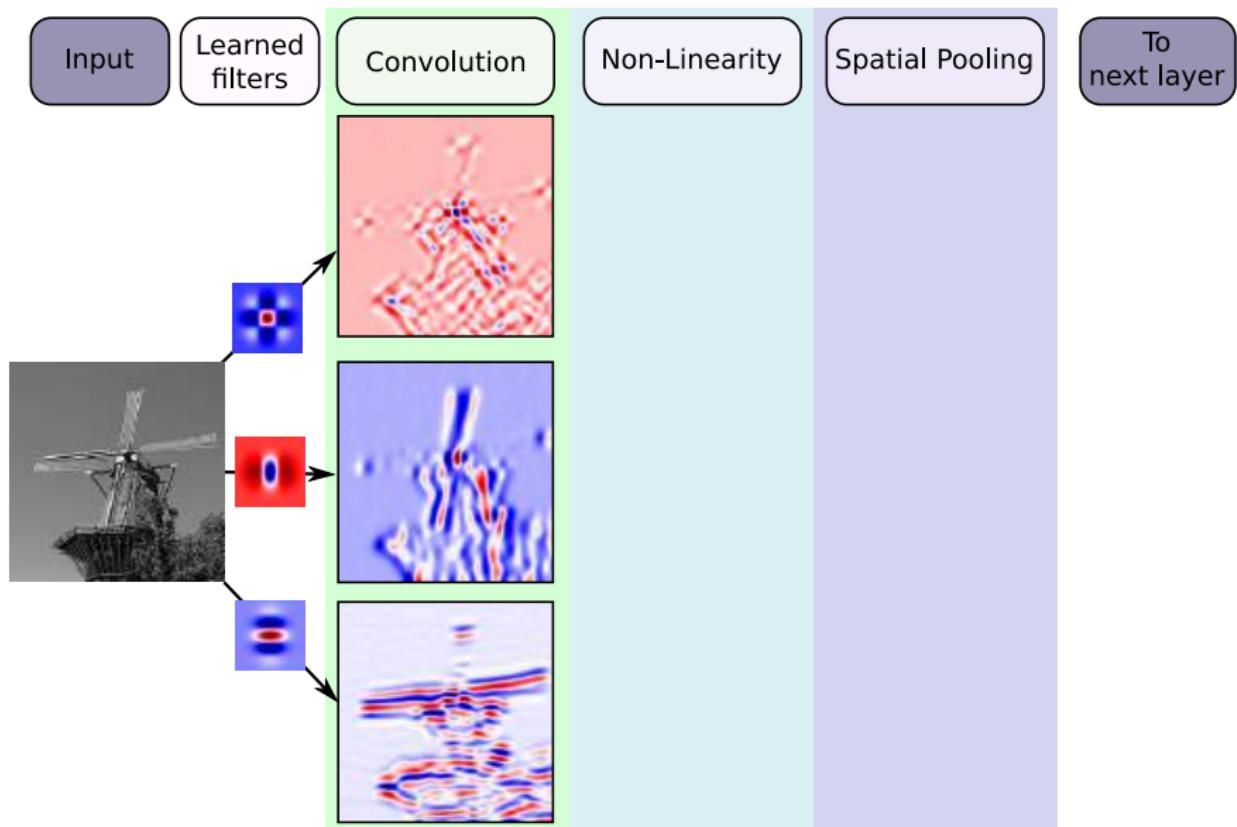
# Convolutional Network



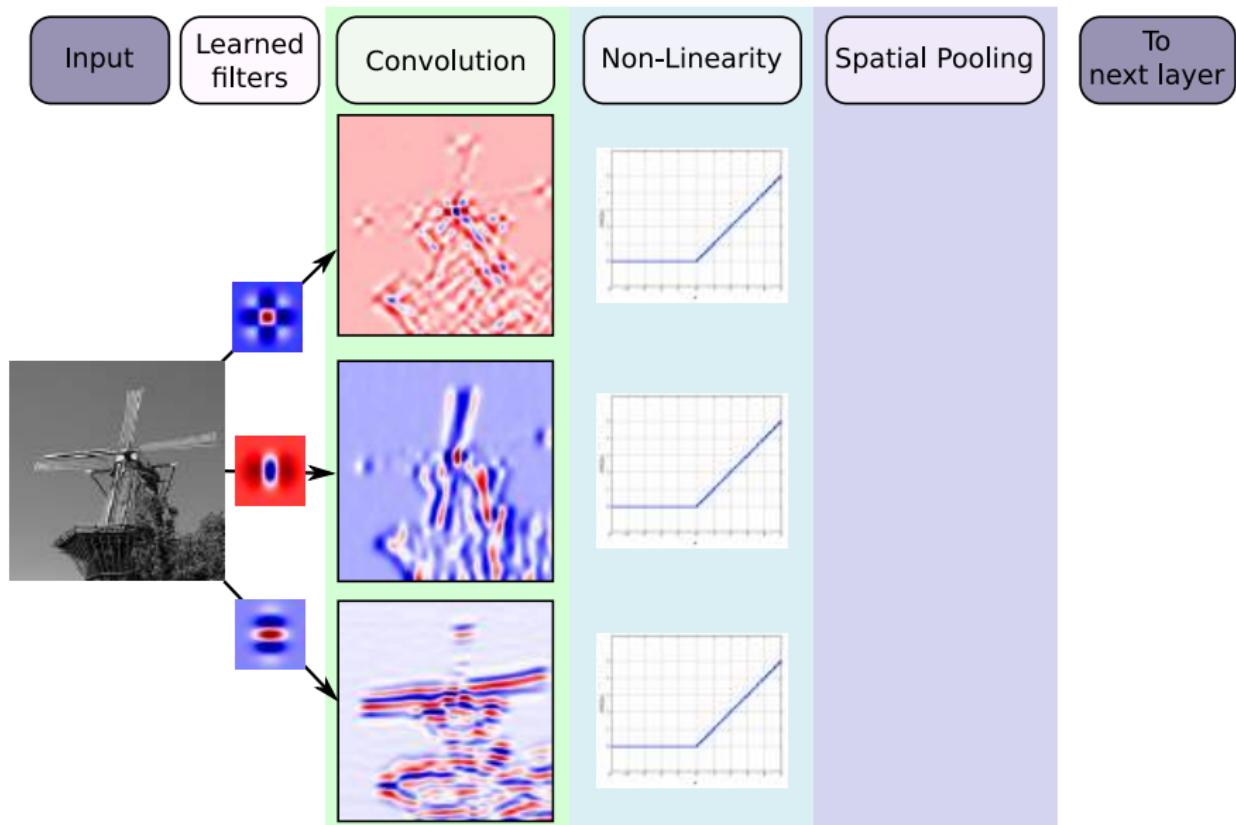
# Convolutional Network: Filter



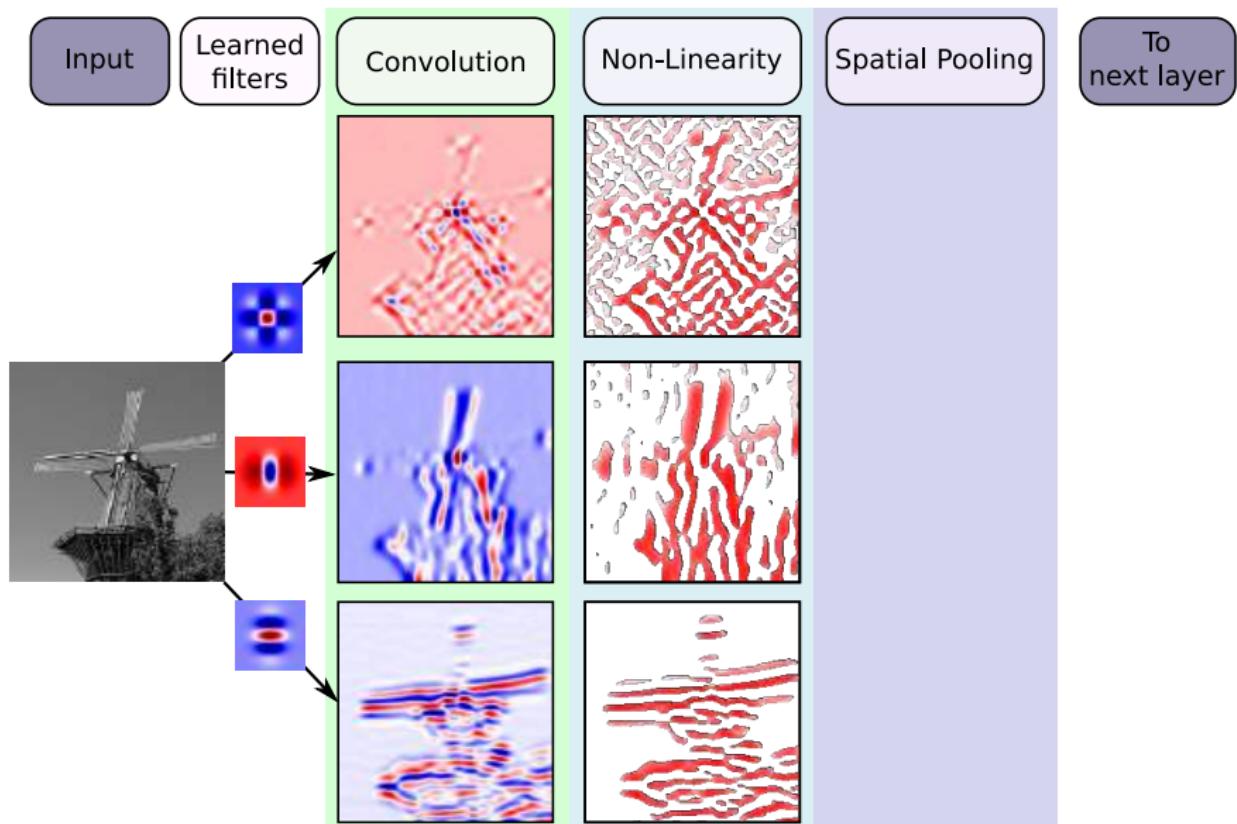
# Convolutional Network: Featuremaps



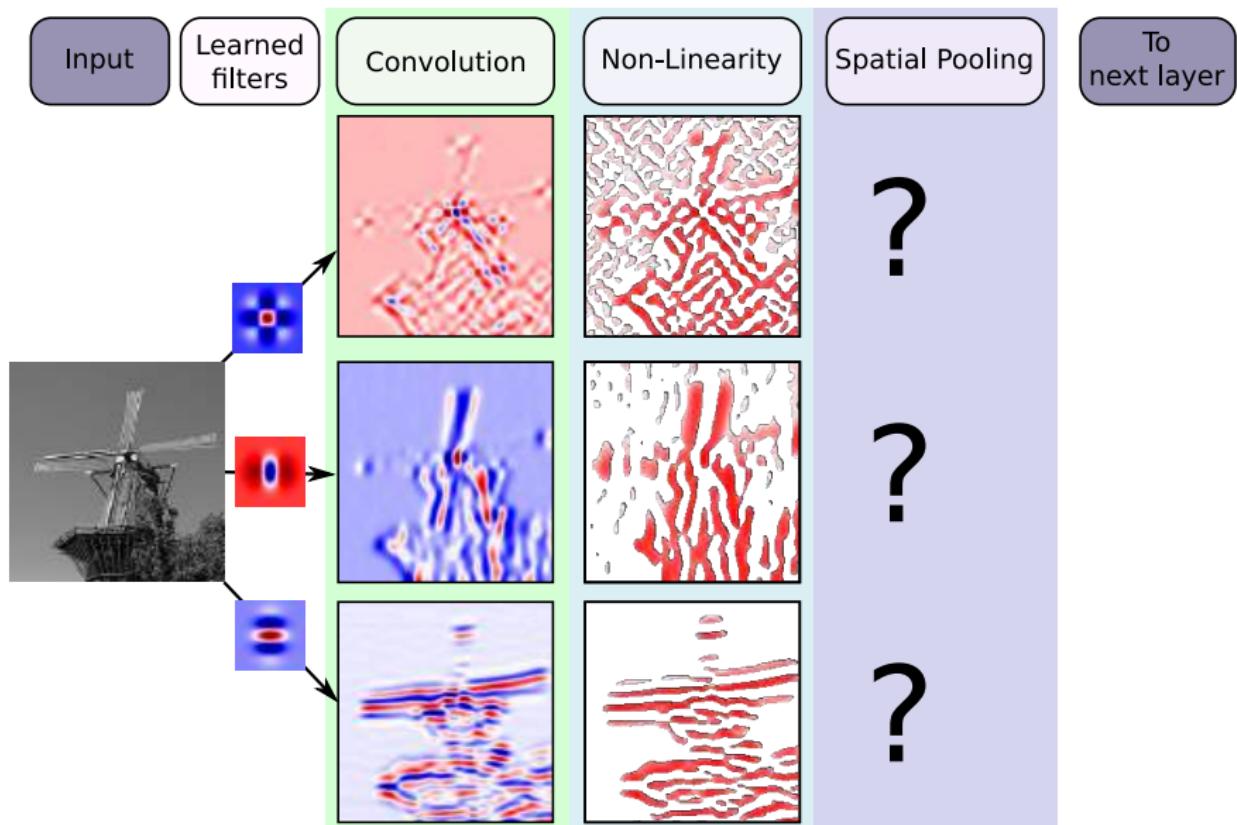
# Convolutional Network: ReLu ( $f(x) = \max(0, x)$ )



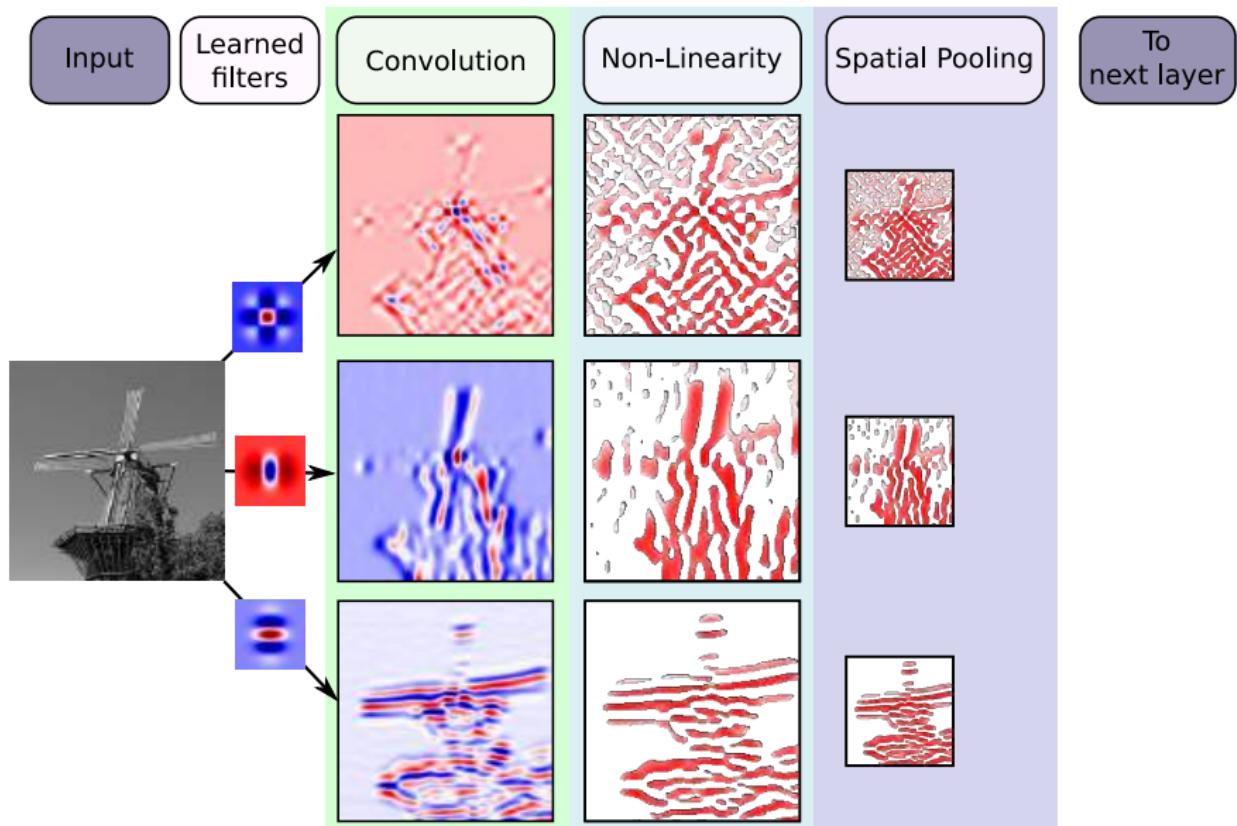
# Convolutional Network: All negative values removed



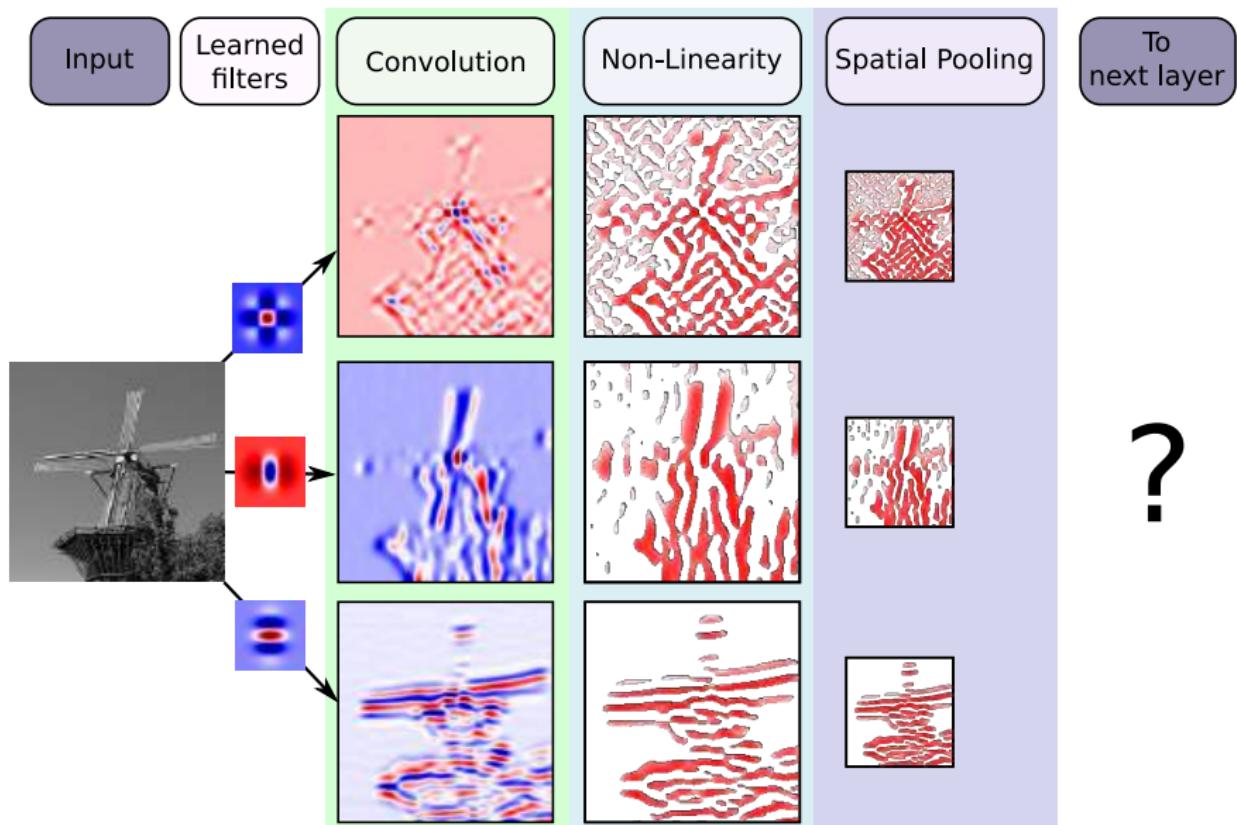
# Convolutional Network: $2 \times 2$ max, $2 \times 2$ sub-sample



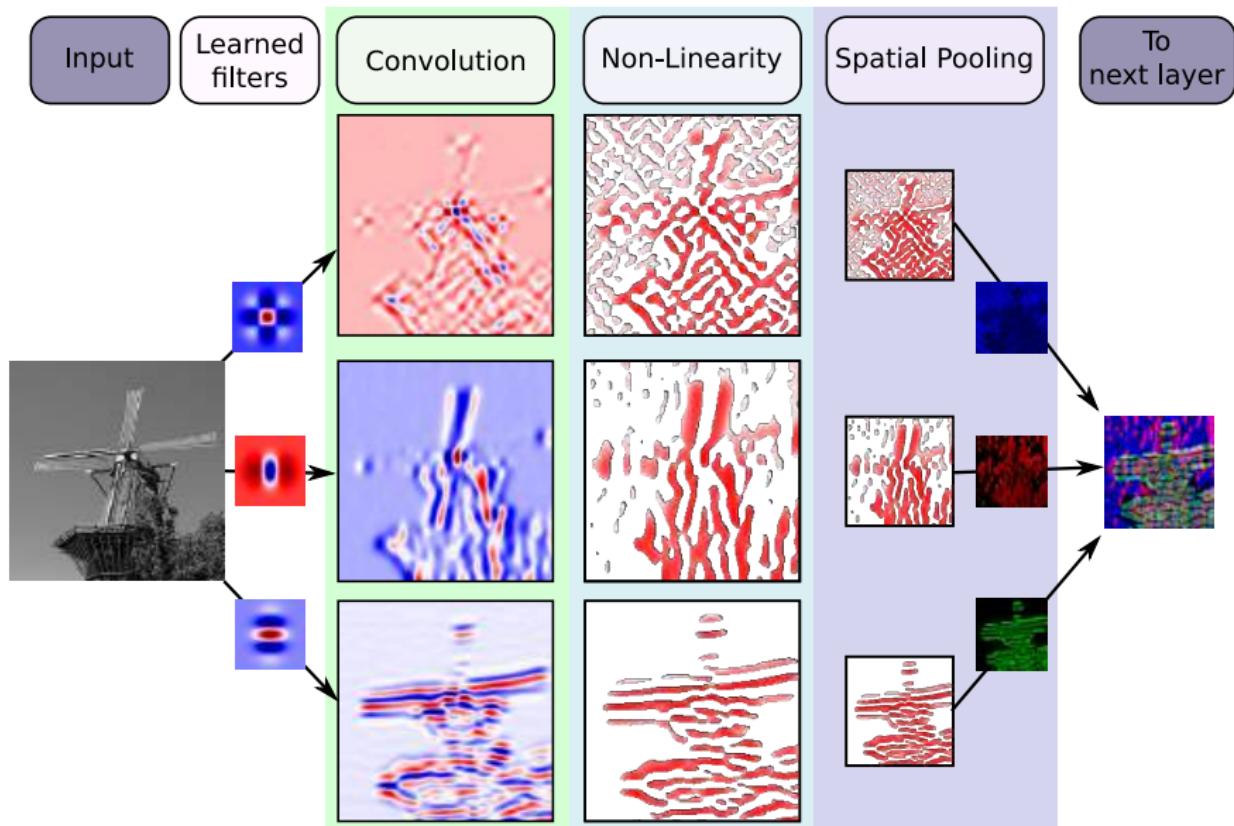
# Convolutional Network: Smaller feature maps



# Convolutional Network: To the next layer

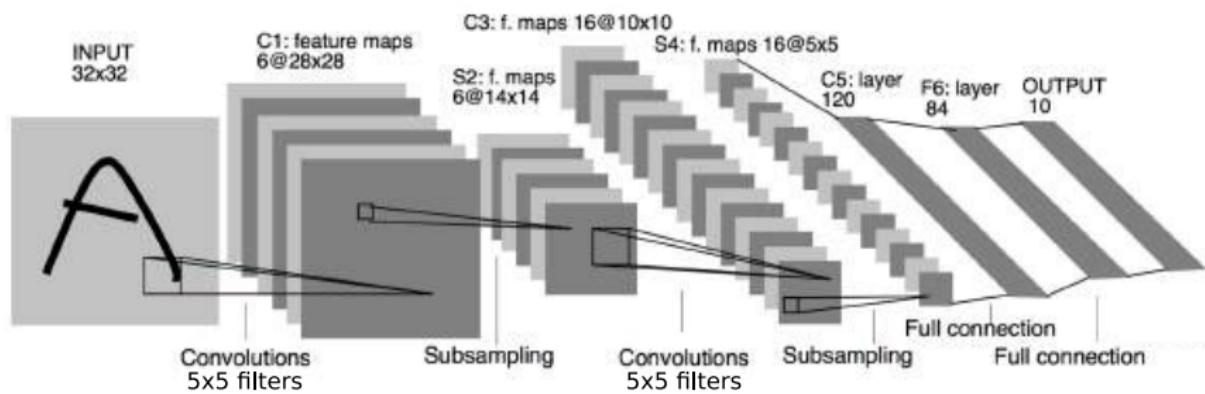


# Convolutional Network: Featuremaps as new channels



# Questions?

# Questions?



Q: Can you explain this CNN?

# *Your opinion counts!*

We hope you are willing to letting us know what you think about the teaching methods, online tools, course organization, assessment, etc. of this course. With your feedback we can further improve our education. Therefore, we kindly ask you to take 5-10 minutes time to fill in the questionnaire.

Please follow the link or scan the QR code below to open the questionnaire.

In order to obtain a reliable evaluation result, we hope that many of you will complete this questionnaire. Your answers will be processed anonymously and handled confidentially.

<https://evasys-survey.tudelft.nl/evasys/online.php?p=G6UCD>



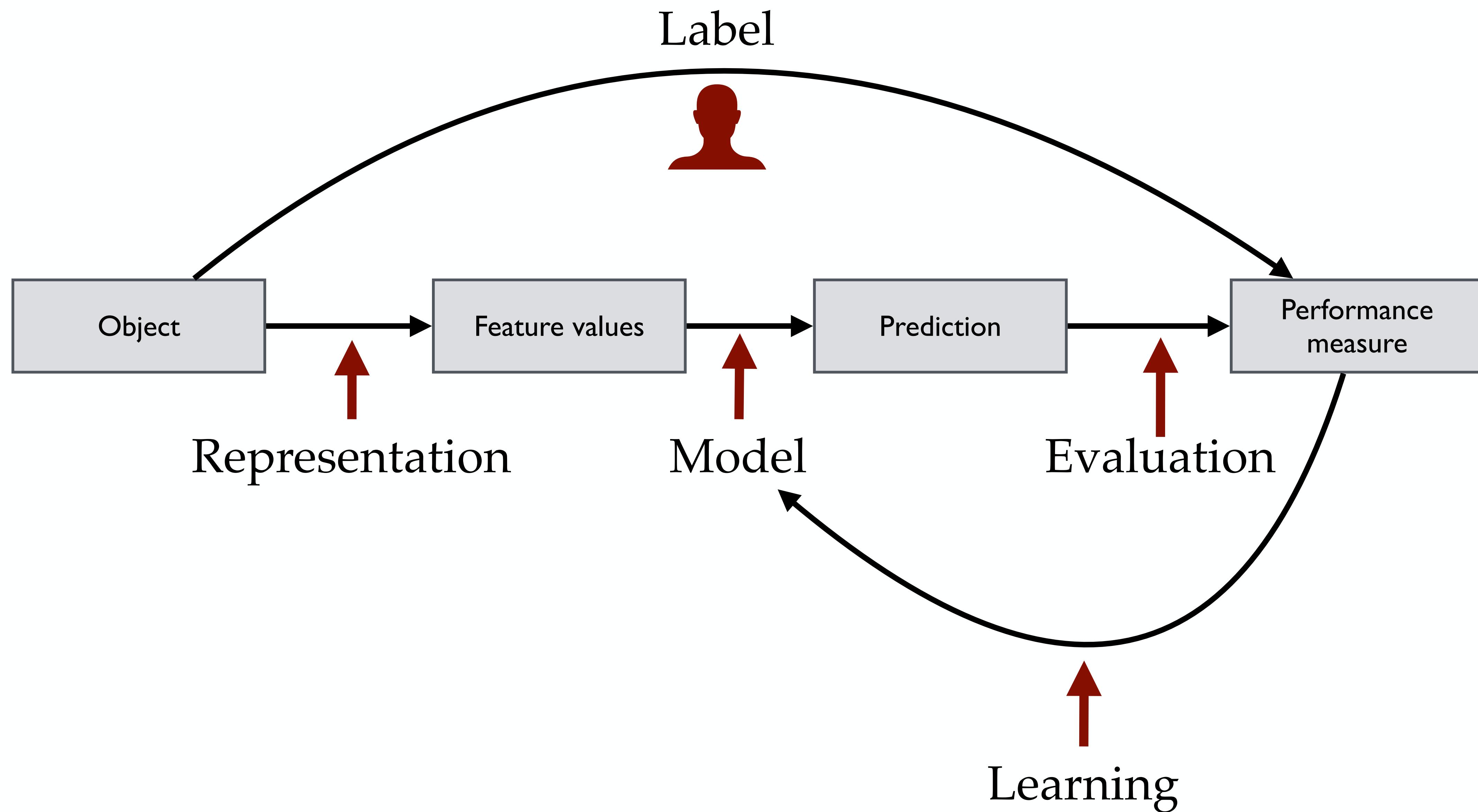
# WRAPPING UP + Q & A

JESSE KRIJTHE  
CSE2510 - MACHINE LEARNING



# MACHINE LEARNING “PIPELINE”

# *Steps involved in Machine Learning*



## Client Question

Deloitte was involved in Tulip Treasures' acquisition process before expanding its role to offer cloud migration support services.

Tulip Treasures' Data Science Team approached Deloitte asking for ML engineering support.



Can Deloitte's ML engineers work on our list of technical debt?



True. We're really interesting in finding out how to improve our overall ML systems?

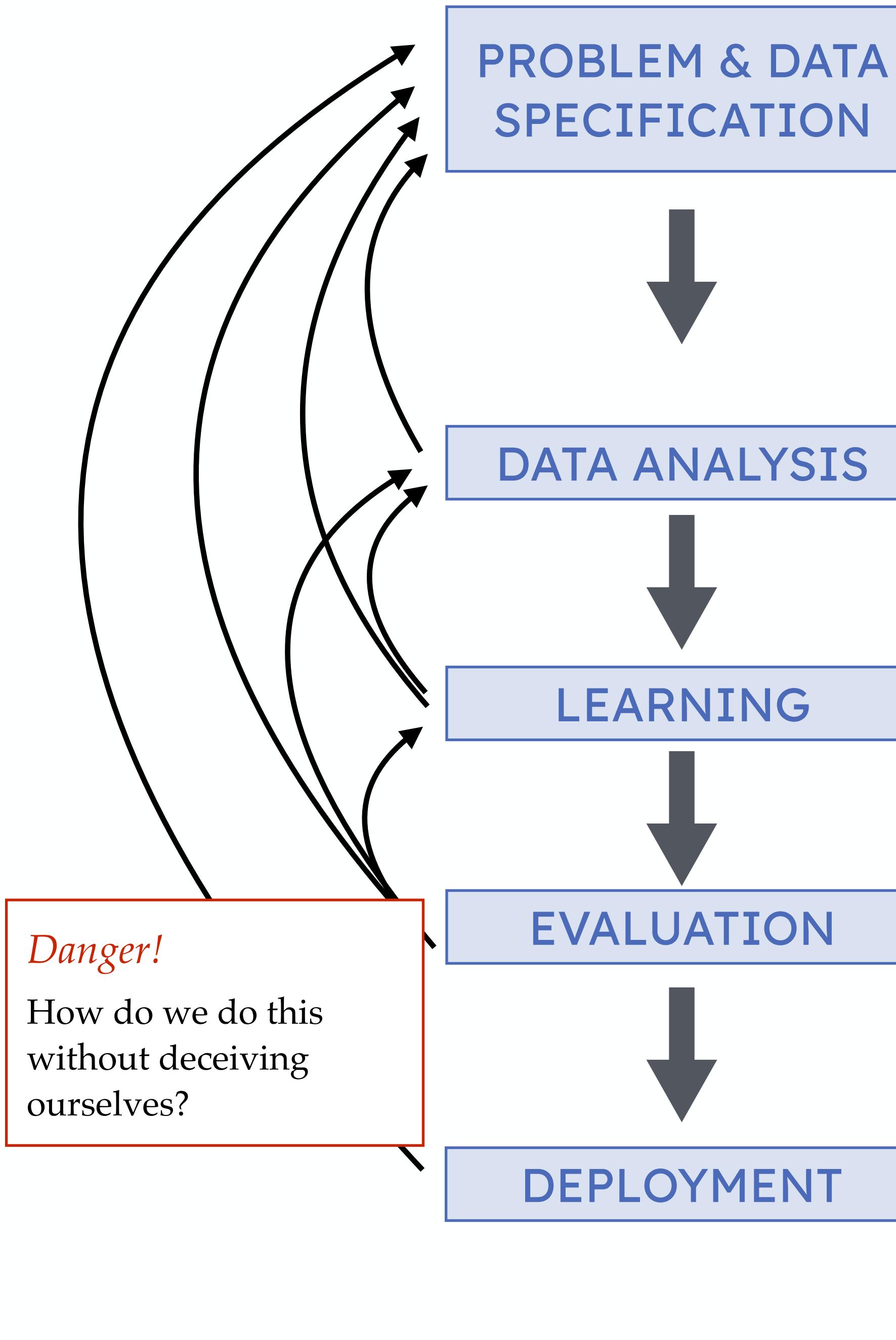


Sure! But are you sure that you are asking us the right question?



Let's take a systematic approach to figuring this out together.





- Defining the problem you want to solve (+performance measure)
- Identifying what objects to gather
- Eliciting prior knowledge that can be helpful for the problem
- Choosing the measurements on these objects
- Gathering and Labeling objects

- Inspecting the data
- Visualising, understanding relationships in the data
- Preprocessing (e.g. normalising, feature extraction)

- Fitting models
- Selecting/combining models (hyperparameter tuning, ensembles)

- Evaluating performance
- Is the performance estimate trustworthy?

- Using the model in practice? Communicating what we learned by building the model?
- How do we make sure the model keeps working?

# QUESTIONS & (HOPEFULLY) ANSWERS

# *Exam Material*

- What about Week 4: Multi-class?
- What about Week 8?

# *Exam Format*

- On campus, but in Weblab (so make sure you are enrolled in Weblab!)
- When you enter the exam, you get a personal exam key
- You have to login to Weblab using your netid
- Structure is very similar to the practice exams:
  - 2-3 programming questions
  - 2-4 open questions
  - ~20 multiple choice questions
- **Bring a calculator**, but you can not bring books or notes.

## *Exam preparation*

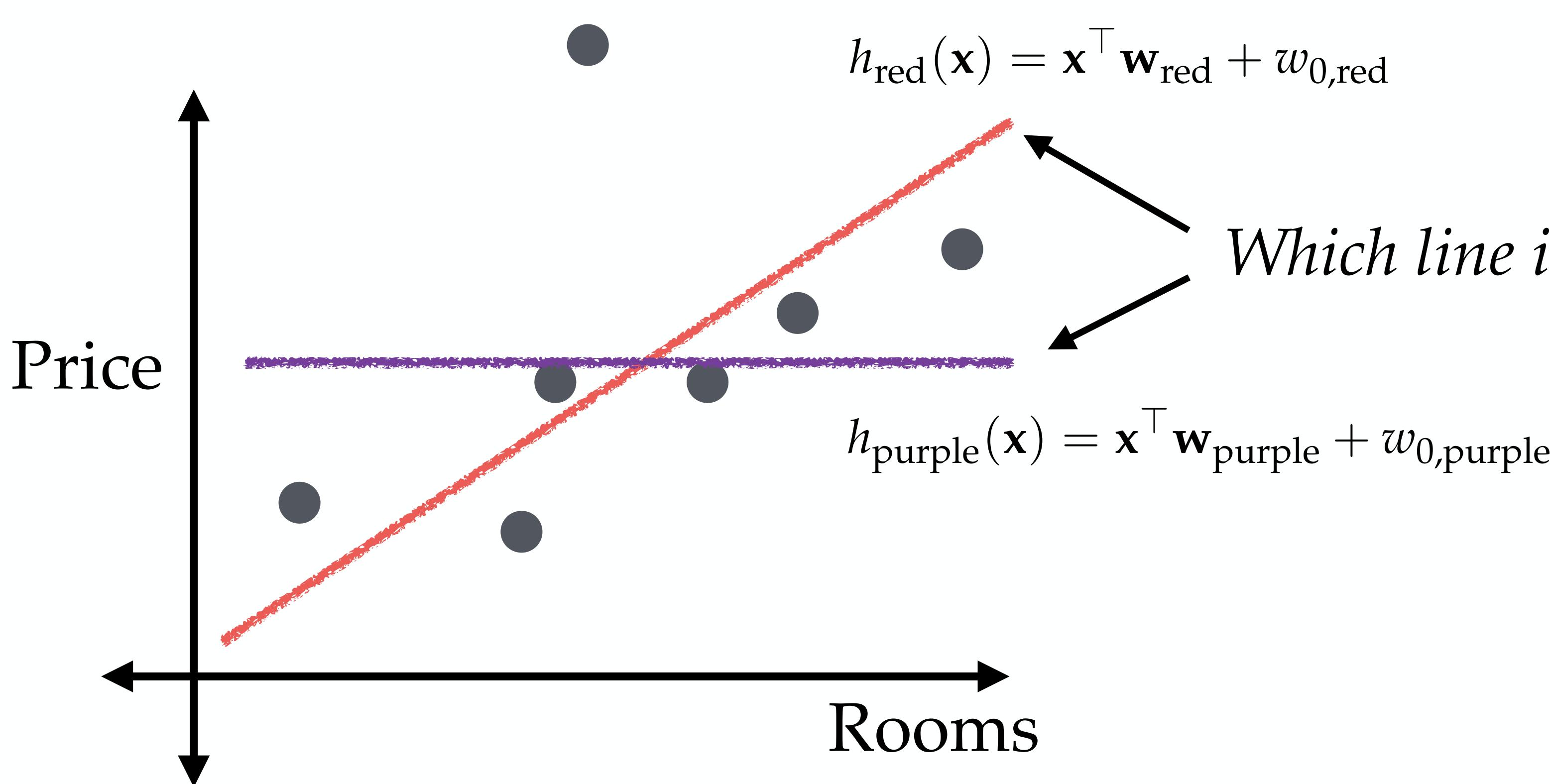
- Final lab session today: ask your questions there or use the [answers.ewi.tudelft.nl](http://answers.ewi.tudelft.nl) site
- **Don't wait till the last days before the exam to ask questions:** they might not get answered

# Regression Problem

We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Task: predict the price of a house, given the number of rooms (note: continuous outcome, not discrete classes)



*Which line is better?*

$$\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w}_{\text{red}} + w_{0,\text{red}} - y_i)^2$$

$$\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{w}_{\text{purple}} + w_{0,\text{purple}} - y_i)^2$$

# *k*-NN: Discriminative or Generative?

---

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}. \quad (2.253)$$

Similarly, the unconditional density is given by

$$p(\mathbf{x}) = \frac{K}{NV} \quad (2.254)$$

while the class priors are given by

$$p(\mathcal{C}_k) = \frac{N_k}{N}. \quad (2.255)$$

We can now combine (2.253), (2.254), and (2.255) using Bayes' theorem to obtain the posterior probability of class membership

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}. \quad (2.256)$$

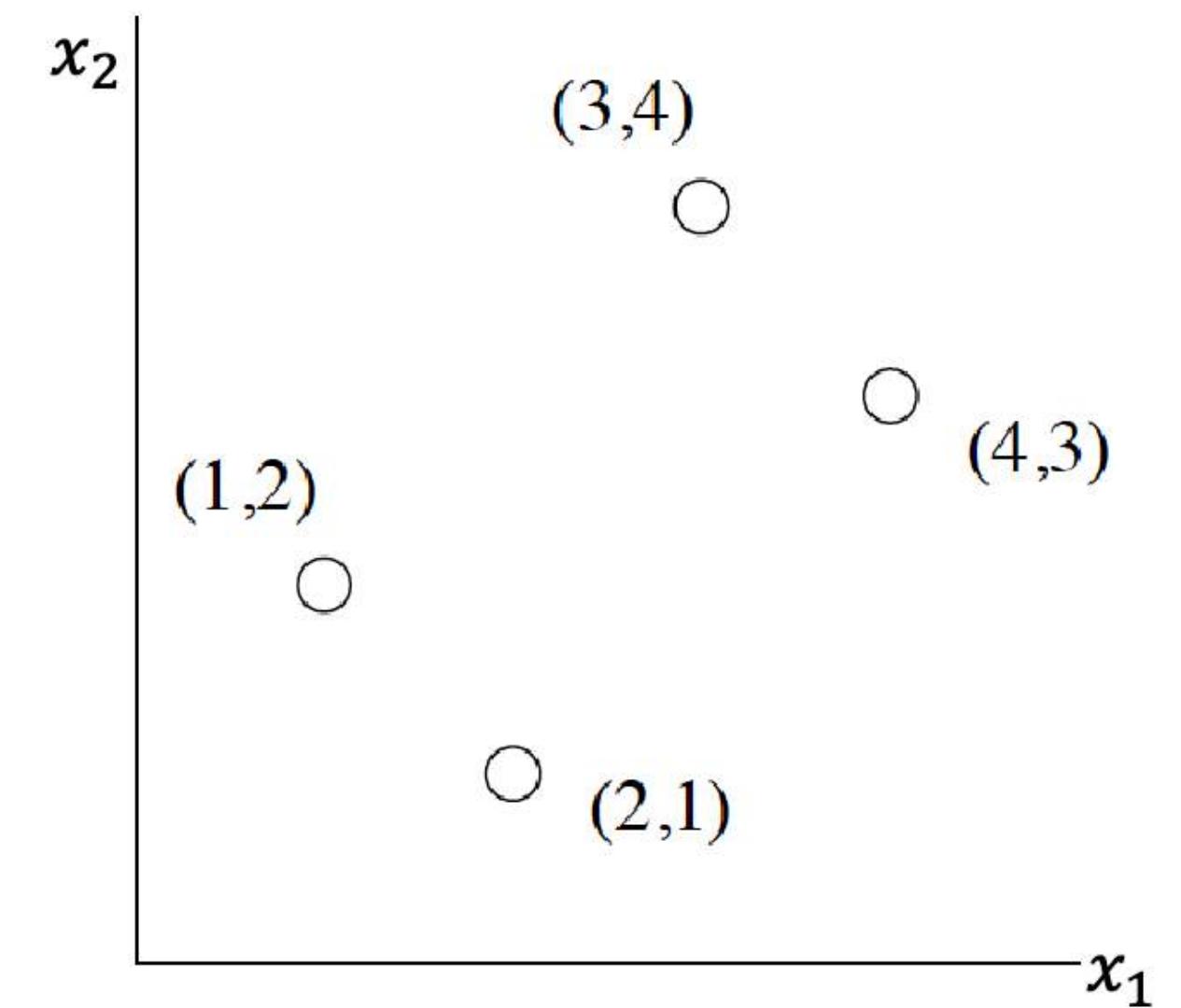
# PCA Notation

## Principal components: PCA example

- *Principal components* are given by the eigenvectors of the covariance matrix

$$Me = \lambda e \quad M = \frac{1}{n} XX^T$$

- Perform PCA for:



# *Number of Parameters: LDA & QDA*

## **The parameters of an LDA**

Assume we have a 10-dimensional dataset with 2 equally likely classes, and assume we want to fit a Linear Discriminant Analysis on this dataset. How many free parameters do we have to estimate?

(Note: the number of free parameters of a 10-dimensional mean is 10.)

# *Your opinion counts!*

We hope you are willing to letting us know what you think about the teaching methods, online tools, course organization, assessment, etc. of this course. With your feedback we can further improve our education. Therefore, we kindly ask you to take 5-10 minutes time to fill in the questionnaire.

Please follow the link or scan the QR code below to open the questionnaire.

In order to obtain a reliable evaluation result, we hope that many of you will complete this questionnaire. Your answers will be processed anonymously and handled confidentially.

<https://evasys-survey.tudelft.nl/evasys/online.php?p=G6UCD>

