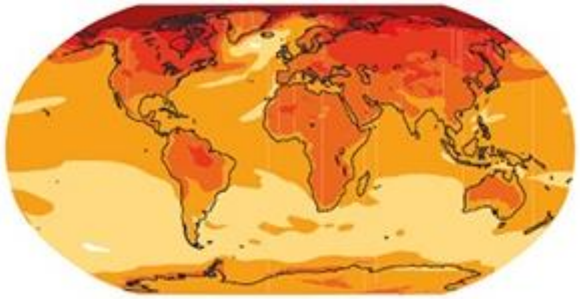
Machine Learning – COMS3007

Introduction to Machine Learning

Benjamin Rosman

Benjamin.Rosman1@wits.ac.za / benjros@gmail.com



https://thispersondoesnotexist.com/

Course Details

- Lecturer:
  - Prof. Benjamin Rosman
- Contact details:
  - The course will be run through the Moodle page
  - Email: Benjamin.Rosman1@wits.ac.za
  - Discord group: https://discord.gg/7437JPX4zp
- Lecture Venues and Times:
  - Lectures will be every Thursday from 08h00-09h45 in WSS2.
- Tutorials and Labs:
  - Tuts/labs are on Tuesday at 14h15-16h00 in the ground floor MSL labs. There will be weekly handins, and occasional quizzes.

Course Outline

1. Introduction to Machine Learning
2. Naïve Bayes and Probability
3. Decision Trees
4. Linear Regression
5. Logistic Regression
6. Neural Networks
7. K-means Clustering
8. Practical Application of ML Methods
9. Principal Components Analysis
10. Reinforcement Learning

Assessments

- Labs/assignments: 40%
  - Most labs will be marked
  - One assignment due at the end of the course (date to be announced)
- Tests/quizzes: 10%
  - There will be a series of small quizzes that will be taken during some of the lab sessions (details to be announced)
- Exam: 50%

Textbooks

There is no prescribed textbook. We will loosely be following these textbooks:

- Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997
- Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer, 2006.
- Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto, The MIT press, 1998.

There are many texts on Machine Learning in the library and on the web that have information on the above topics.

You can also look up the Coursera course “Machine Learning” by Andrew Ng (Stanford University) which follows similar content.

What is expected of you?

- You can collaborate unless otherwise stated, but make sure you acknowledge!
    - Labs and the assignment will be done in groups
- Programming in python will be expected but not taught
    - All labs and assignments require programming
- I advise working in (online) groups a lot: you will learn from each other

Mathematics

This is a maths-heavy course!

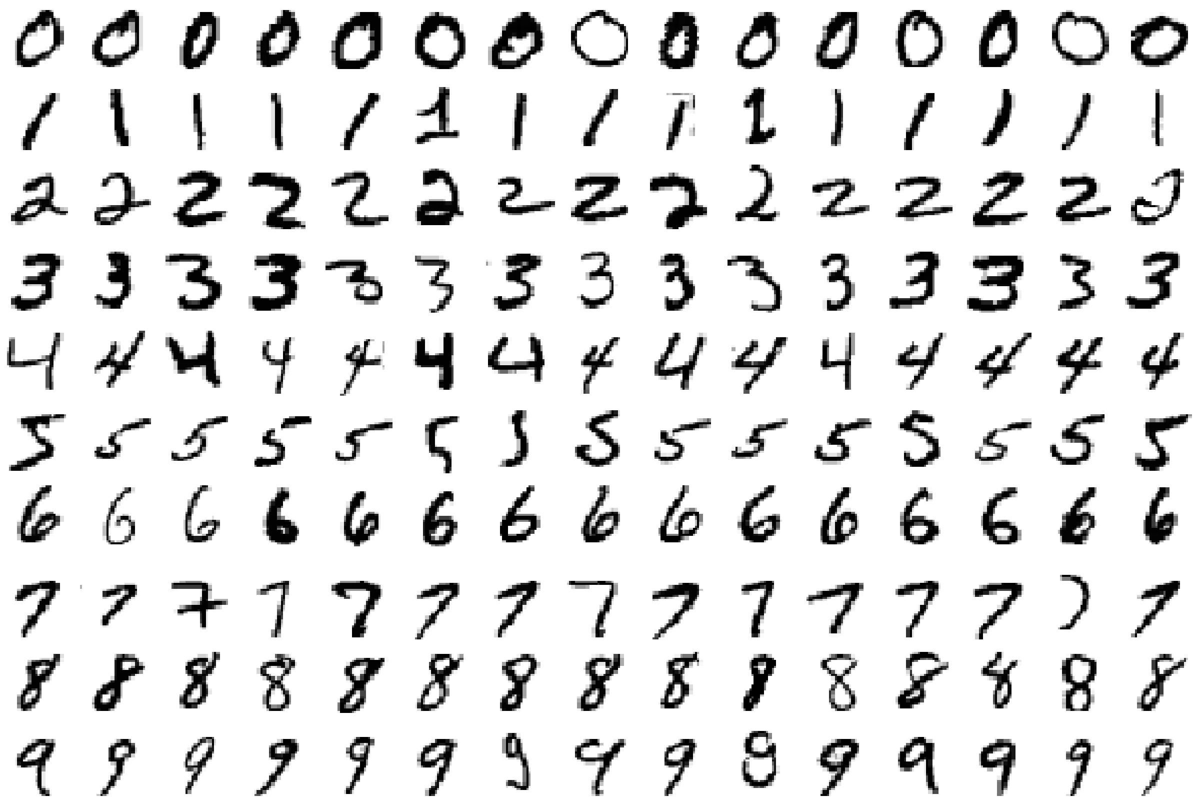
You should have familiarity with linear algebra, calculus, and statistics.

I will explain what is needed as we go along, but you may need to brush up on some of your old notes, or look it up online.

Now, onto the fun stuff…

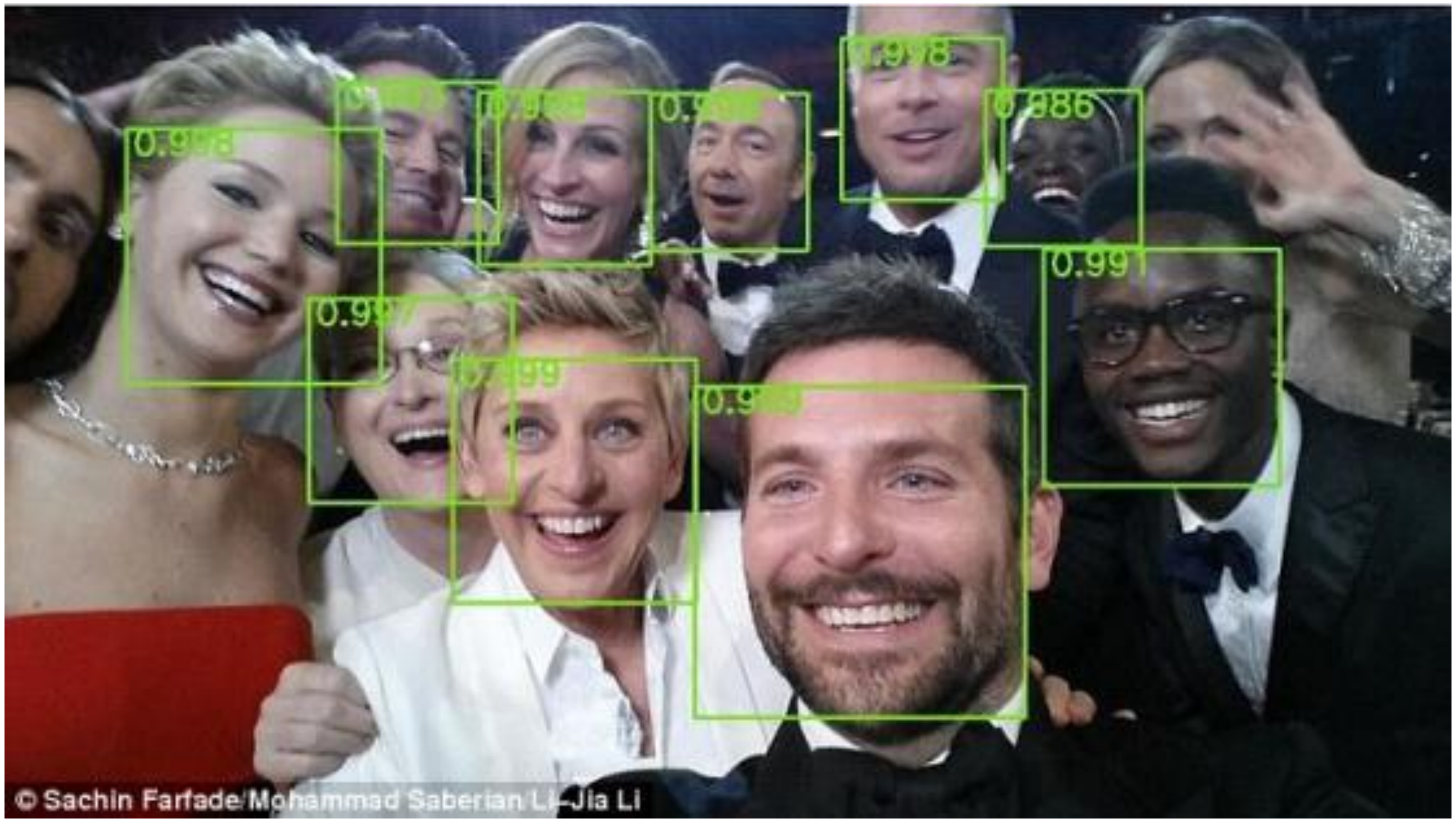
Example: Hand-Written Digits

- Write a program to automatically classify these



Example: Faces

- Write a program to automatically find faces



What is machine learning?

(patterns)

- *"the **automatic discovery of regularities in data** through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories"*

  [Bishop, 2007]

- *"Machine learning is the study of computer algorithms that **improve automatically through experience**"*
- *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."*

  [Mitchell, 1997]

What is machine learning?

- Automating the process that has driven science for centuries
- Collect data, hypothesise relationships (models), experiment to test hypotheses
- E.g.
  - Kepler’s laws of planetary motion
  - Newton’s laws of motion
  - …

What is machine learning?

- In many of today's problems it is
  - Very hard to write a correct program
    - Heuristics, rules, special cases
    - E.g. based on character strokes, topology, etc
  - But very easy to collect examples
    - Text, emails, images, sales, …
- Idea behind machine learning:
  - From the examples, generate the program (function)
  - This is easier with many examples
    - Define **many** relative to the size of the data, and complexity of the model

Terms

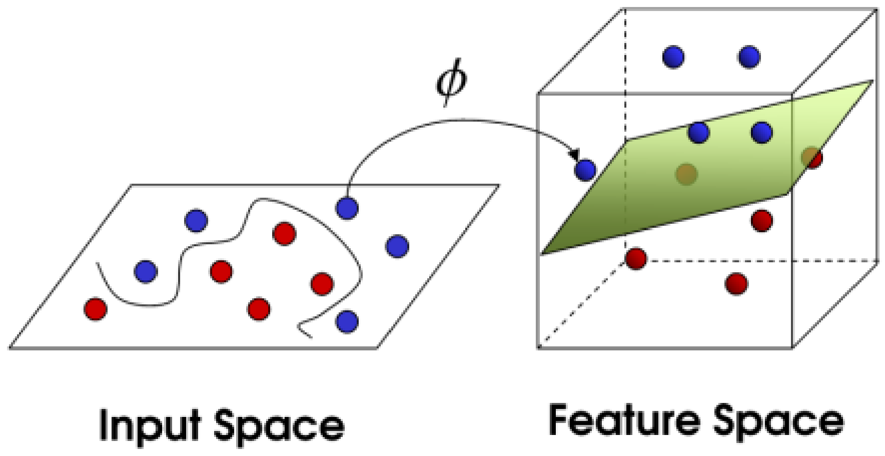
- **Artificial Intelligence (AI)**
- **Machine Learning**
- **Data Science**
- **Big Data**

Terms

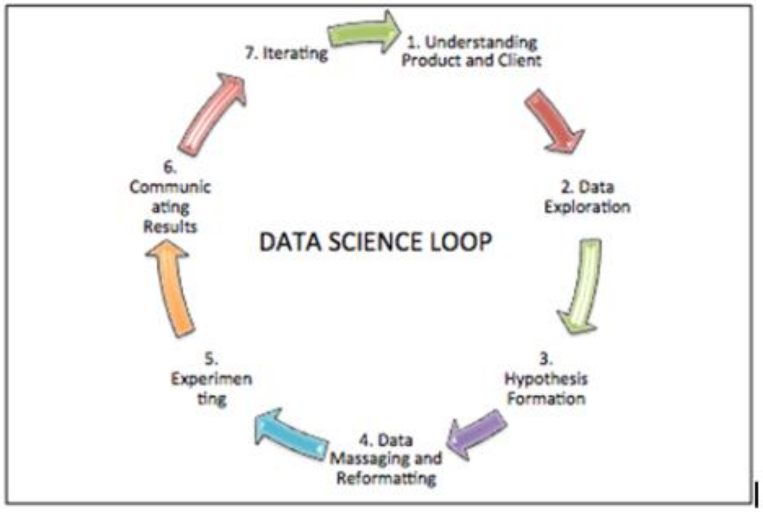
- **Artificial Intelligence** (**AI**) is the field which studies how to create computers and computer software that are capable of intelligent behaviour.
- **Machine Learning**
- **Data Science**
- **Big Data**

Terms

- **Artificial Intelligence (AI)**
- **Machine Learning** explores the study and construction of algorithms that can learn from and make predictions on data.
- **Data Science**
- **Big Data**

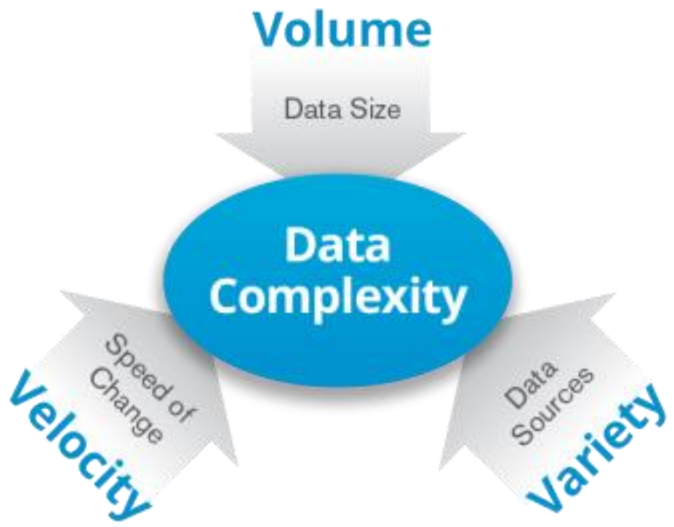


Terms



- **Artificial Intelligence (AI)**
- **Machine Learning**
- **Data Science** is an interdisciplinary field about processes and systems to extract knowledge or insights from large volumes of data.
- **Big Data**

Terms



- **Artificial Intelligence (AI)**
- **Machine Learning**
- **Data Science**
- **Big Data** is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, and information privacy.

ML, Stats and Data Science

Many parts of machine learning are similar to ideas from statistics

Emphasis is typically different

- Focus on **prediction** in machine learning vs **interpretation** of the model in statistics

ML often refers to tasks associated with artificial intelligence (AI)

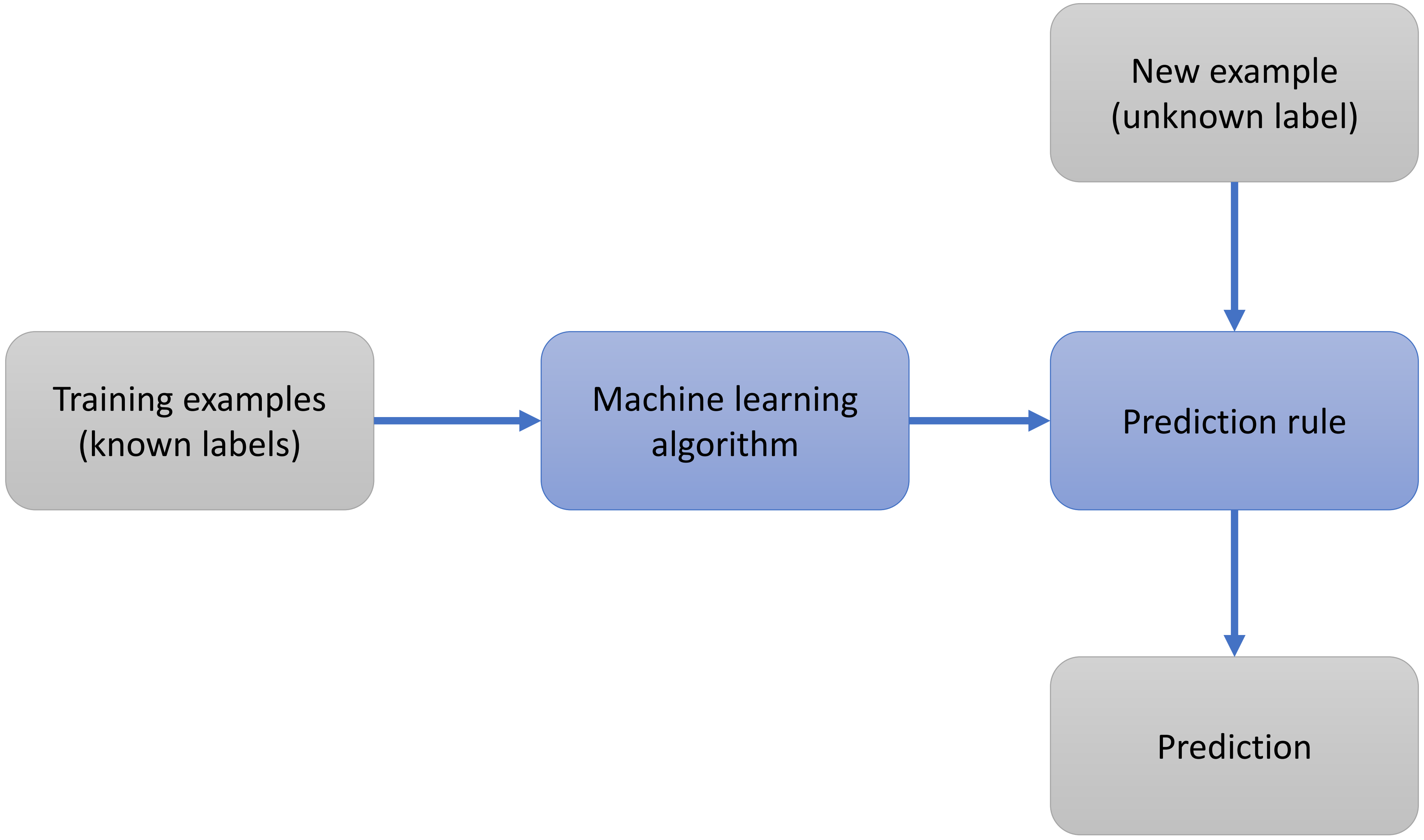
- e.g. recognition, diagnosis, planning, robot control, prediction, etc.

Goals can be autonomous machine performance, or enabling humans to learn from data (data mining)

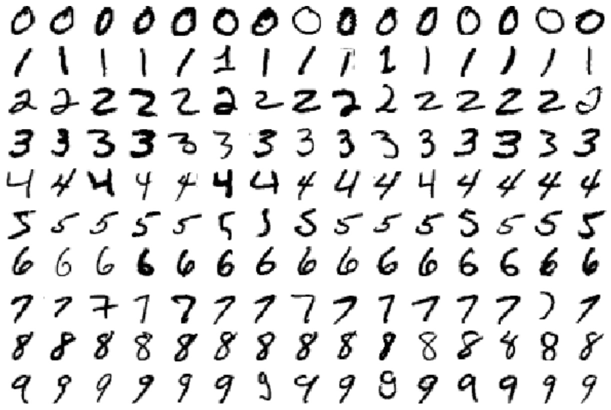
Data science: typically refers to entire pipeline

- From data acquisition and storage to presenting results
- ML is typically the modelling and analysis phase

Typical $\overset{\text{supervised}}{\text{learning}}$ problem


New example (unknown label)
Training examples (known labels)
Machine learning algorithm
Prediction rule
Prediction

Hand-Written Digits



What is the ML problem here?

Learn a **function** y = f(x):

- Mapping from **data** (x=image) to a **class** (y=number label)
- Think of this as a program
- E.g. f( ) = “8”

How is this done?

- The function f is some **model** of the digits
  - We provide the general form
- We present the model with a set of N images $\{x_1, x_2, \ldots, x_N\}$ where the true solutions $\{y_1, y_2, \ldots, y_N\}$ are known
- This **training** tweaks the parameters of the model
- During **testing** provide a new x’ to determine y’ = f(x’)

Formally

Given some data (input-output pairs):

- x = training input, y = training output
- $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

We need a model:

- *Modelling assumptions (or knowledge about the problem) go here!*
- $y = f(x; \theta)$
- θ = parameters of model

Now, learning task is to find "best" model parameters θ

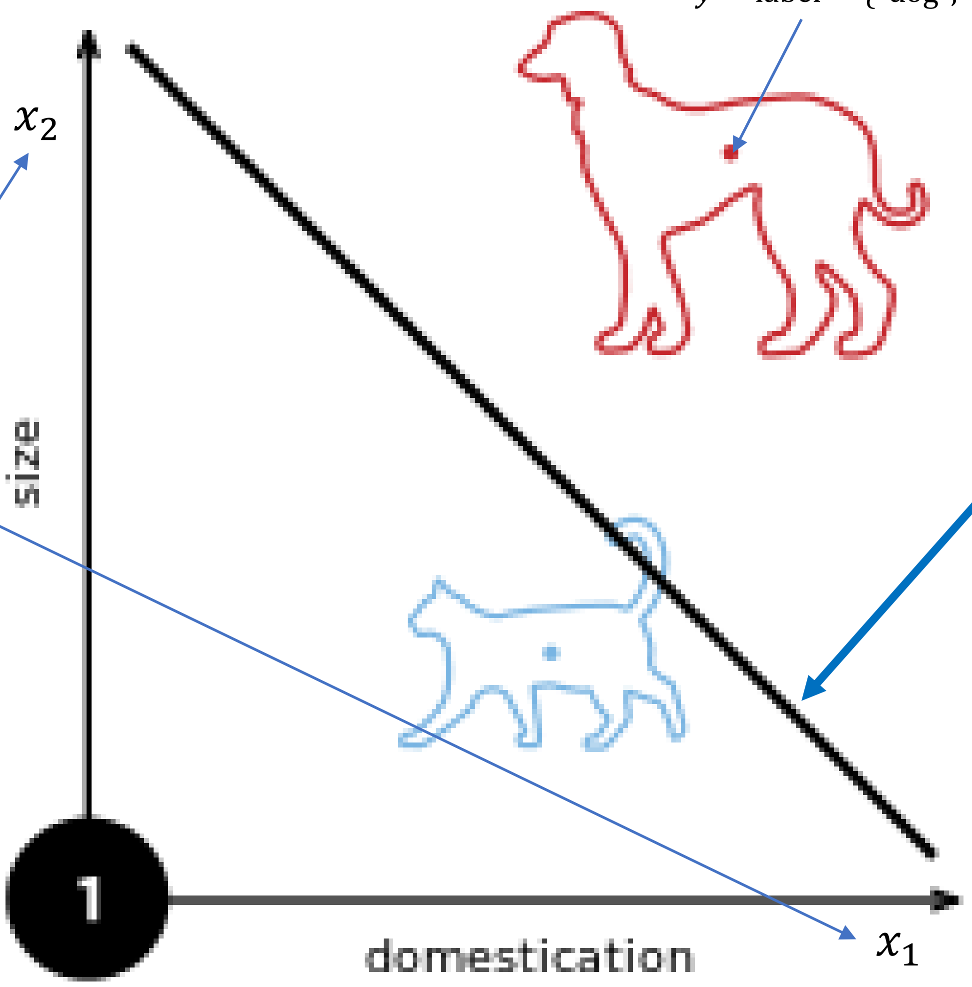
- Usually measure some error between y and $f(x; \theta)$
- Treated as an optimisation problem

An example

- Given (features):
  - Animal size
  - Level of domestication
- Predict (label):
  - Cat or dog?

An example

Training



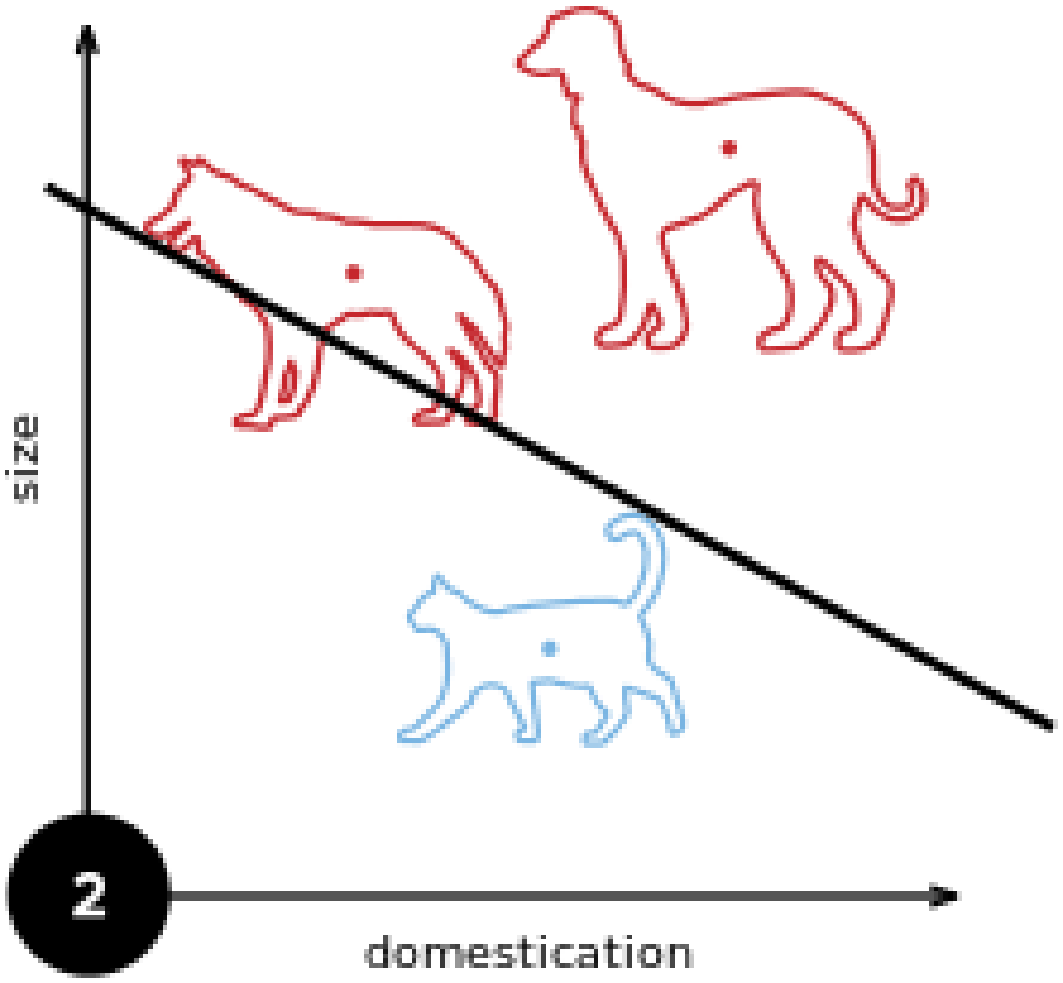
Features of the data (2D)

Parameters of model define this line

The model here is a straight line: linear discriminant

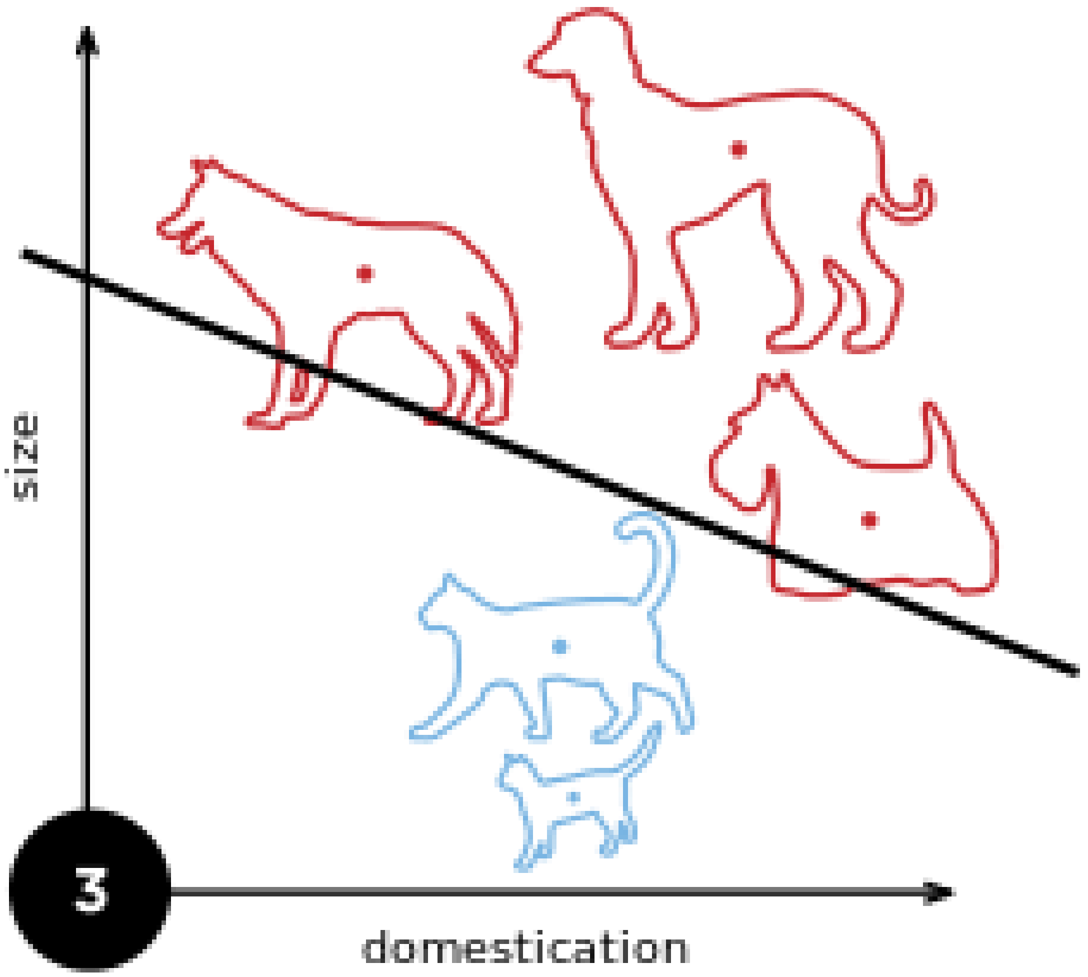
An example

Training



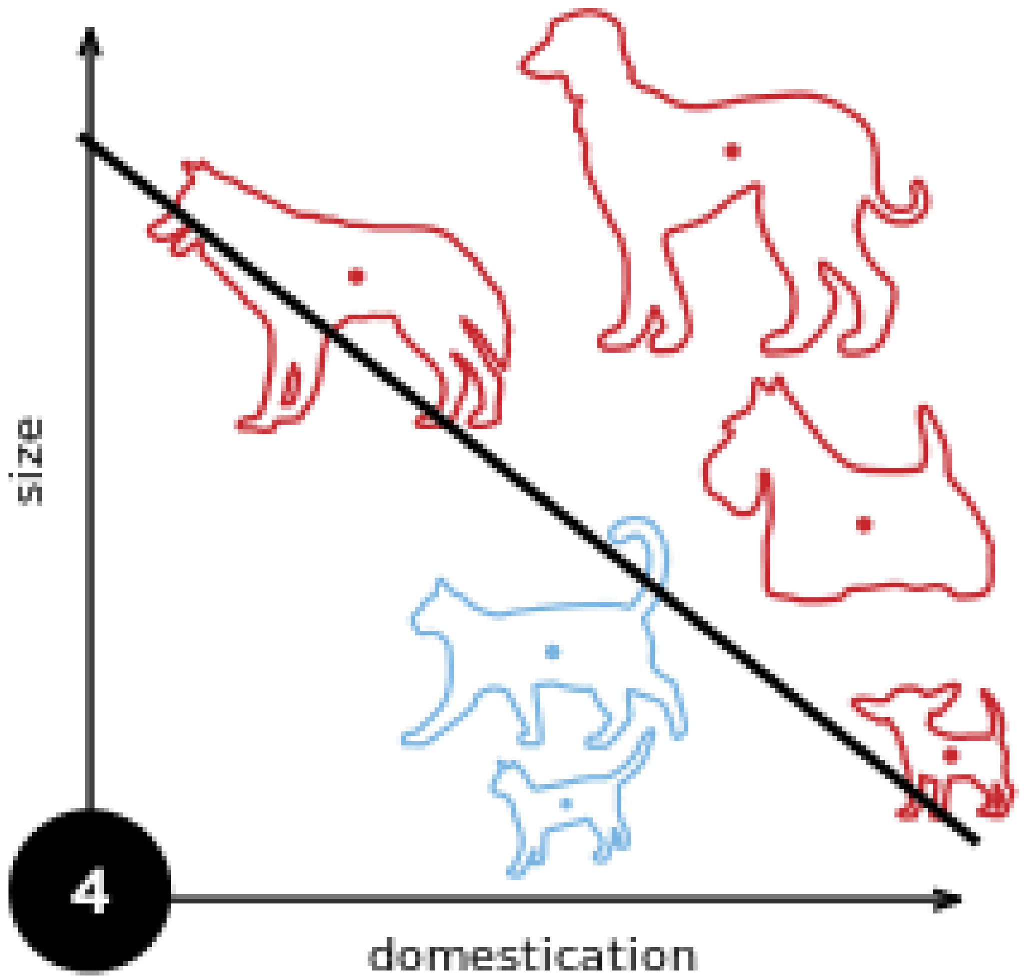
An example

Training



An example

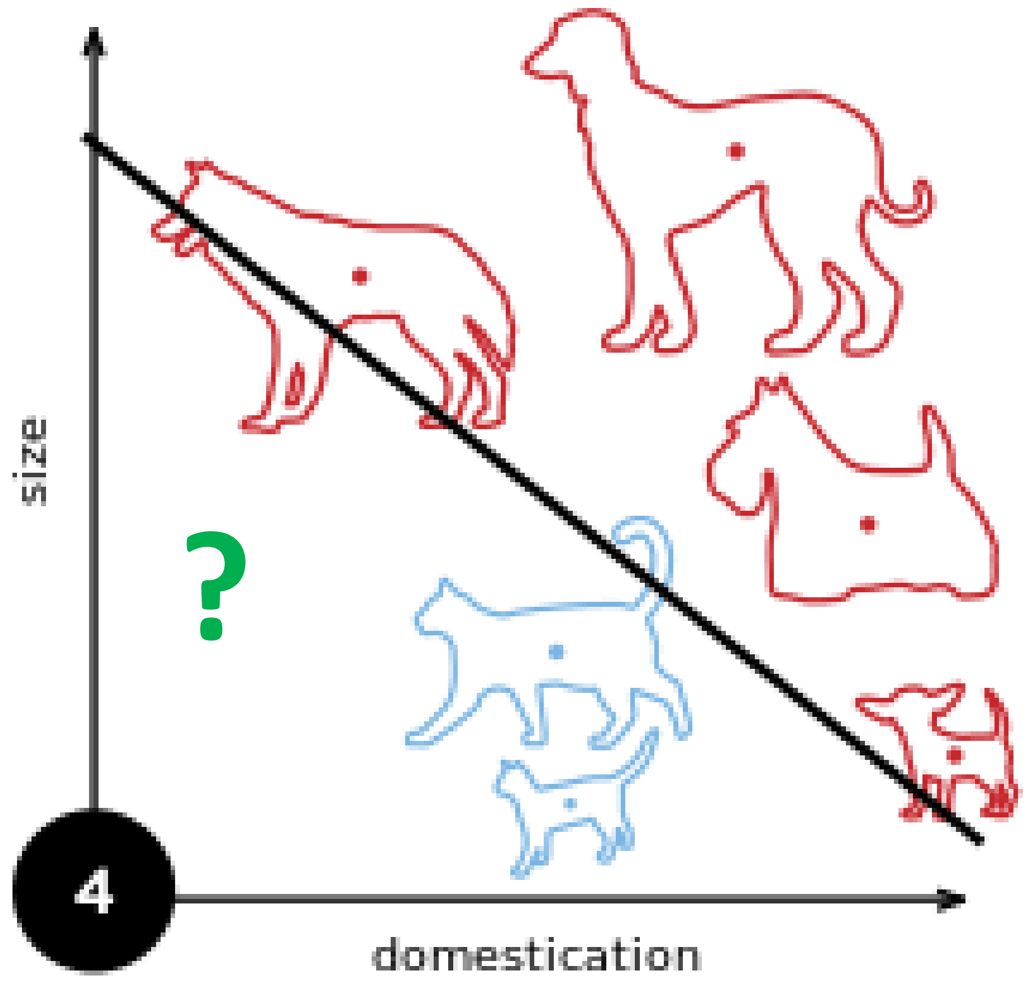
Training



An example

We want the model to **generalise** to any point in this space that we haven't seen yet

Querying



Categories of ML

- Supervised learning
  - Predict output y when given input x
  - Learn from labelled data: $\{(x_i, y_i)\}$
  - Classification: y categorical
  - Regression: y real-valued

(Make predictions!)

- Unsupervised learning
  - Learn from unlabelled data: $\{x_i\}$
  - Clustering
  - Learning some structure in the data

(Understand data!)

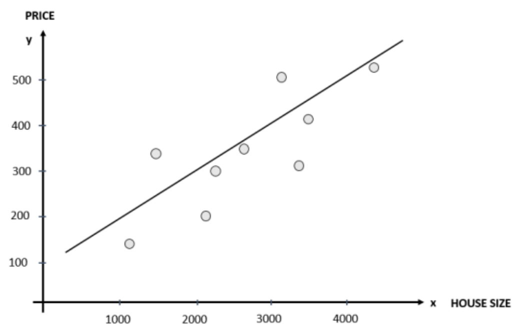
- Semi-supervised learning
  - Only some labels provided

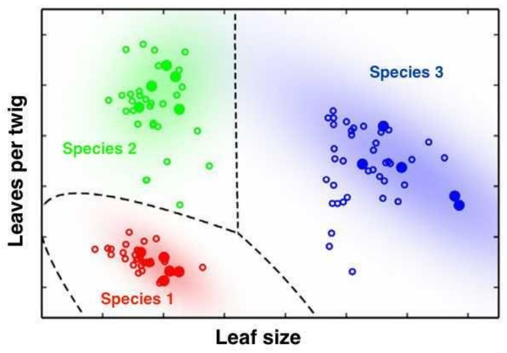
(Combine the above!)

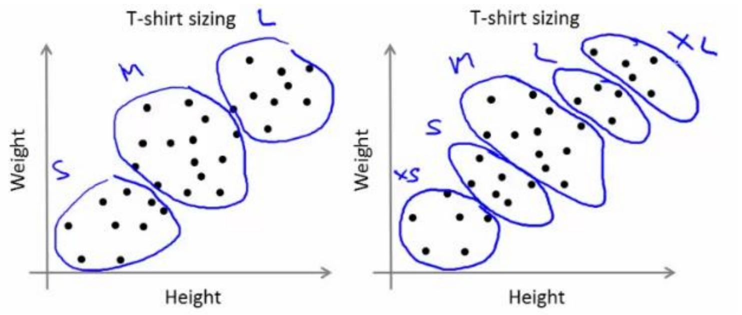
- Reinforcement learning
  - Learn from rewards (typically delayed)
  - Generate own data (experience) through interacting with an environment

(Learn to act/make decisions!)

Common ML Problems







Regression
- Predict a continuous output y, given some input x
- Training: examples of (x,y) pairs
- Supervised learning

Classification
- Predict a discrete class output y, given some input x
- Training: examples of (x,y) pairs
- Supervised learning

Clustering
- Given input data x, return discrete cluster membership
- No training clusters are given, so the returned clusters may not mean anything
- Unsupervised learning

NB: the labels (S, M, L) added to this t-shirt example were the result of someone looking at the clustering results and interpreting them!

Note: inputs are usually multidimensional

Reinforcement Learning (Learn to act!)

Given an unknown dynamic process (environment), and an unknown reward process (goal), learn to take a sequence of actions to maximise reward.

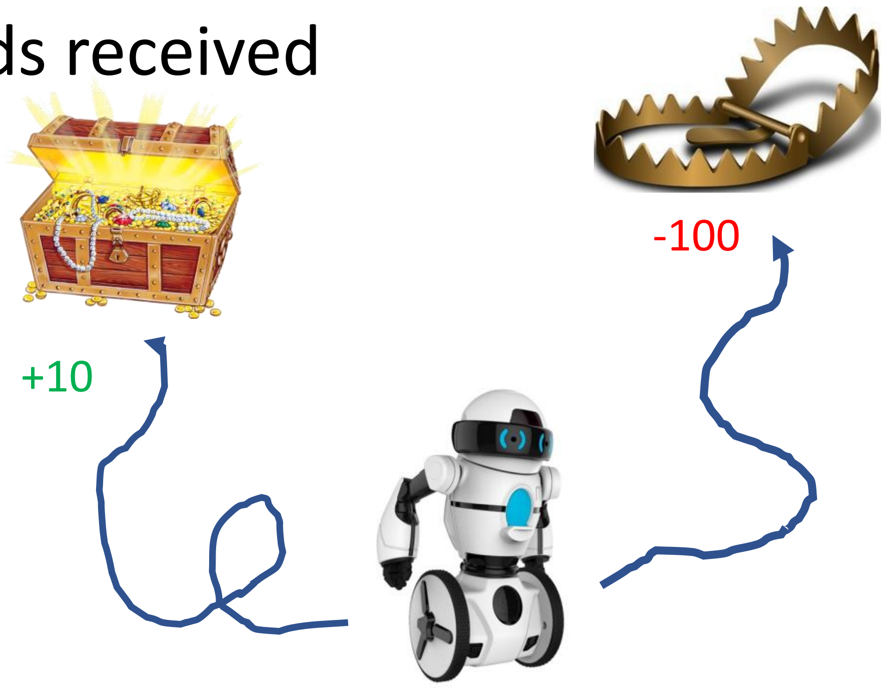
Learn a behaviour by rewards received

Many attempts

- Trial and error

Examples:

- Learn to fly a helicopter
- Learn to make coffee
- Learn to play chess



Generalising

During training, only a small fraction of possible data will be provided

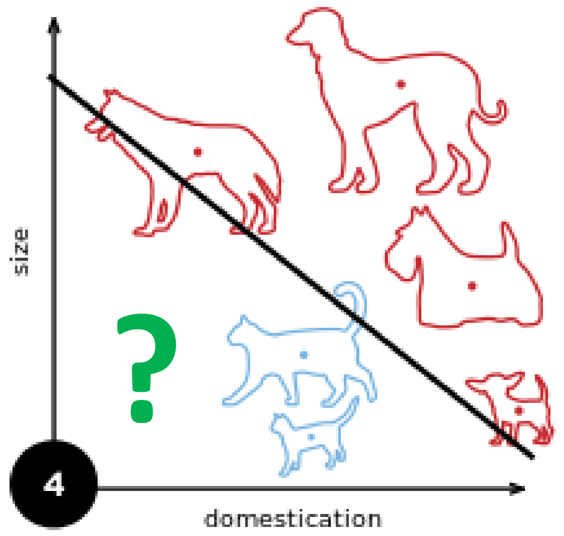
Need the resulting f to **generalise** to (ideally) all cases

- This is what we really care about: we are unlikely to only ever see the training data!

Beware of over-/under- fitting!

- Performs well on training data
- Poor generalisation!

The more data the better!

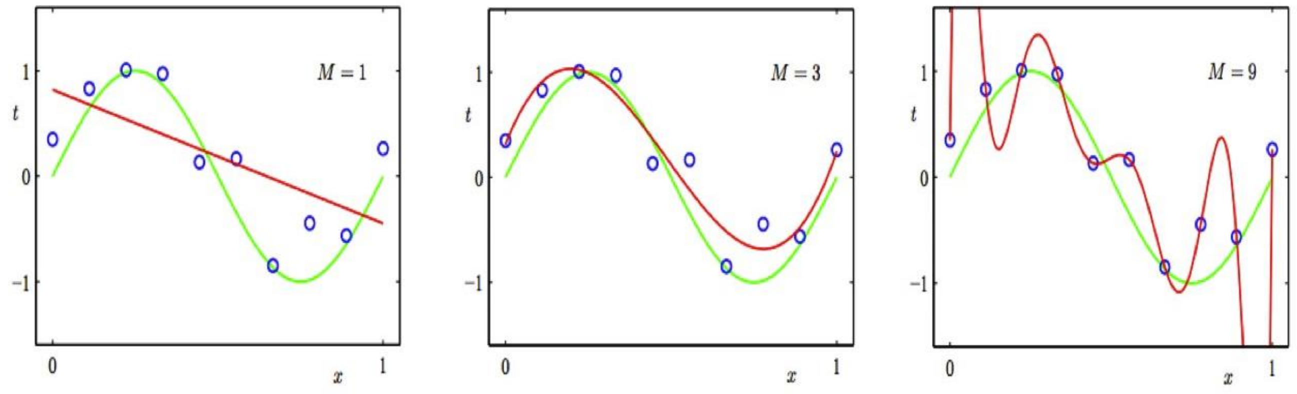


Generalising

Want a model that is expressive, but not too general for the amount of data you have

Regression:

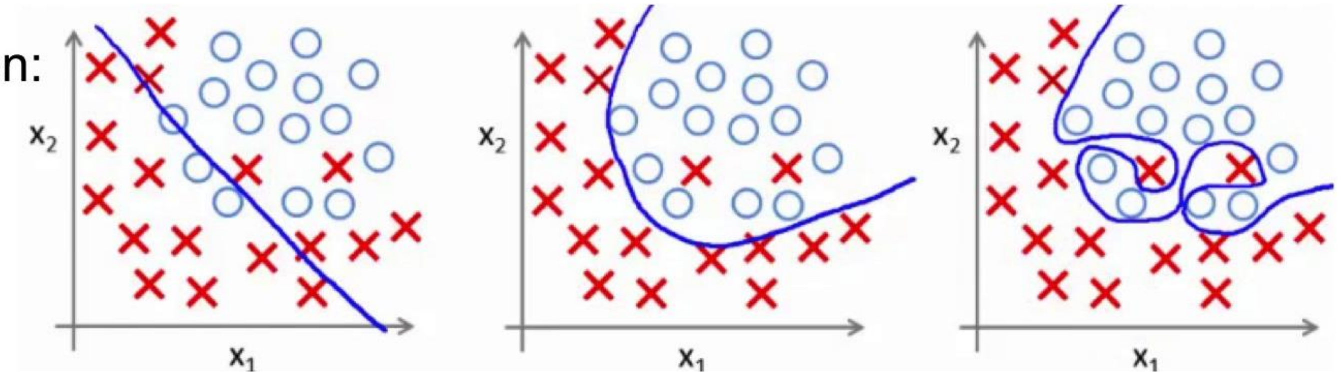
Note: ground truth (green line) is unknown to algorithm



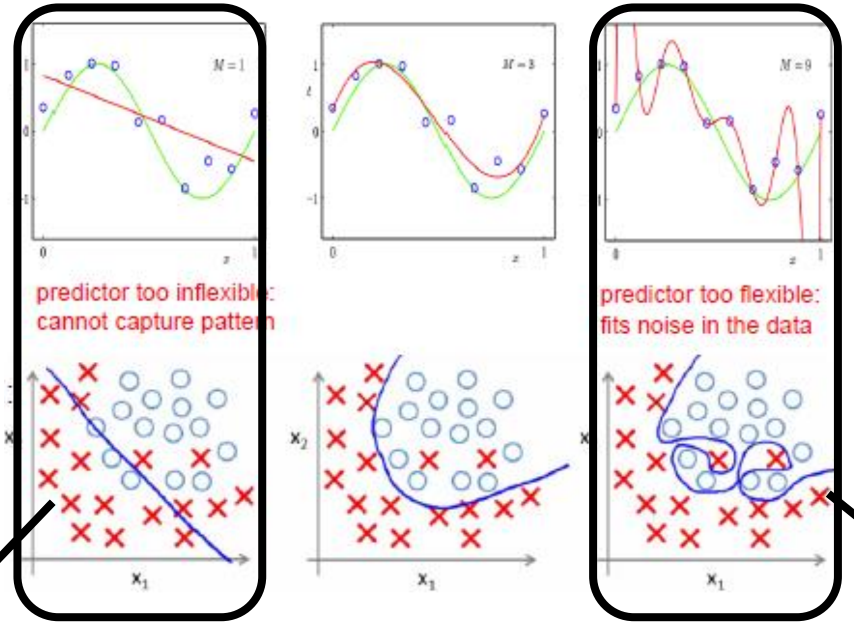
predictor too inflexible: cannot capture pattern

predictor too flexible: fits noise in the data

Classification:



Bias-variance trade-off



These models are said to have a high **bias**: there is error from erroneous assumptions in the model. This causes **underfitting**.

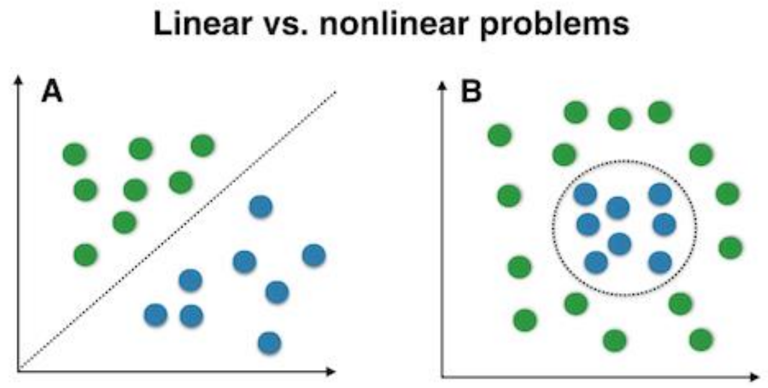
These models are said to have a high **variance**: there is error from small fluctuations in the data. This causes **overfitting**.

The bias-variance trade-off involves balancing these two factors: both are different sources of error that affect generalisation.

Representations

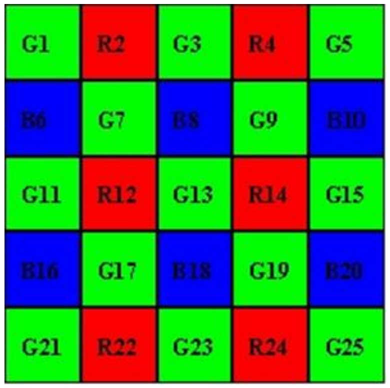
How the data is represented is fundamental!

- Determines if problem can be solved with chosen model



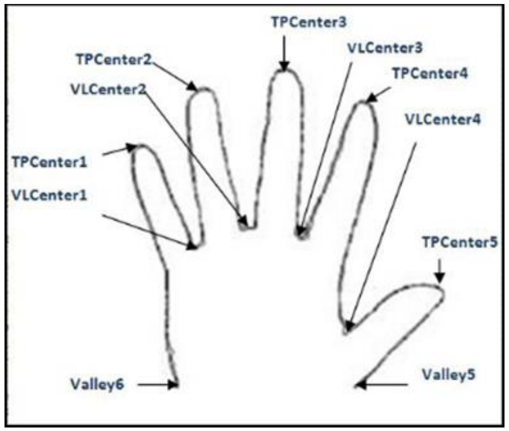
- Or can be solved at all!
  - Identify as elephant vs dog
    1. Features: mass, height
    2. Features: number of legs, number of ears

Representations



Represent data using features

- Low level: pixels, characters, …
- High level: objects, words, regions, …



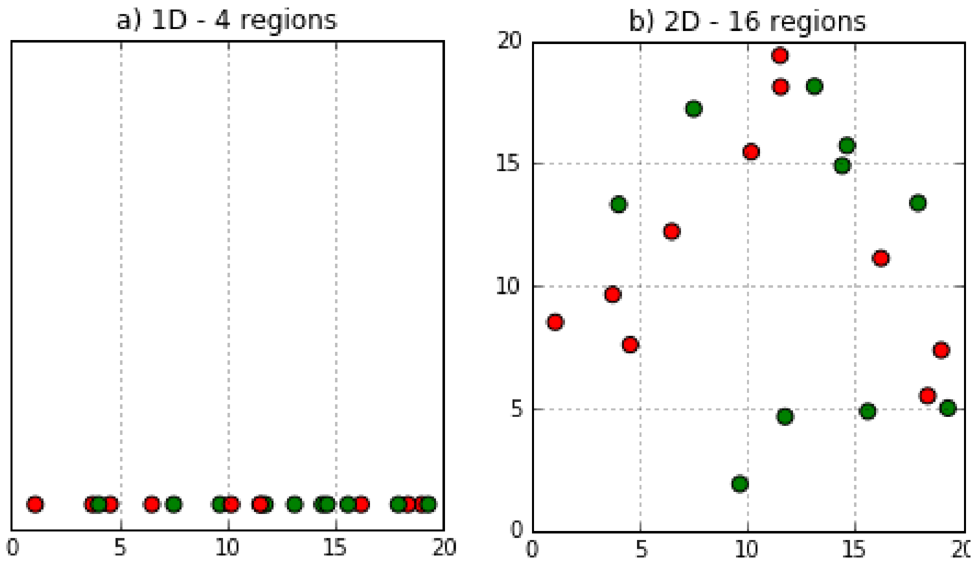
Trade-off between

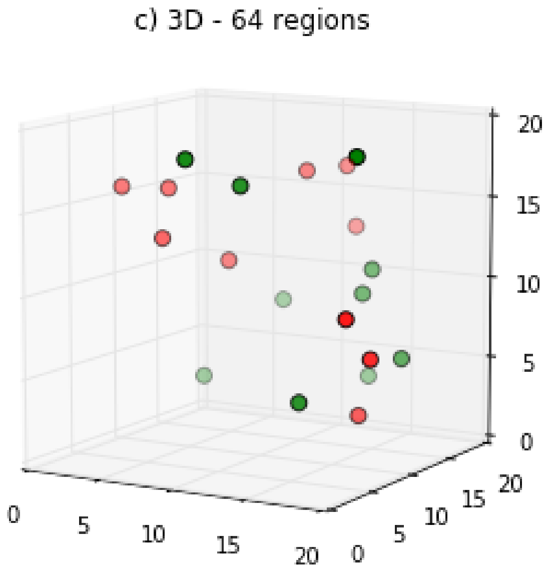
- Expressive: accurately capture distinctions in data
- Sparse: not need prohibitive amounts of data

One of the hardest and most important parts of ML!

Curse of Dimensionality

- As dimensionality of model or feature space grows, may need **exponentially** more data





How sparse are these 20 data points in each case?

Handling representations

- Feature extraction
  - Manual pre-processing
- Feature selection
  - Autonomously identify important dimensions
- Feature learning
  - Combine simpler features into more complex ones
  - E.g. deep learning (when we talk about neural networks)

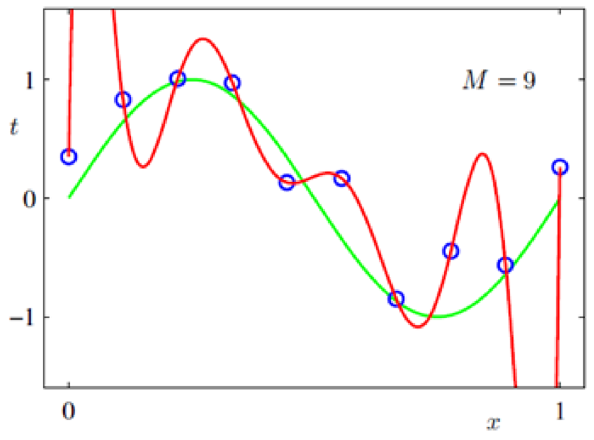
Data

For any ML algorithm to work, we need data, and more is always better. In ML, we “let the data do the talking”.

Much work goes into collecting data sets. For large models (many parameters), we may need many millions of examples to learn a good model.

But, how do we know how well

the model will generalise?



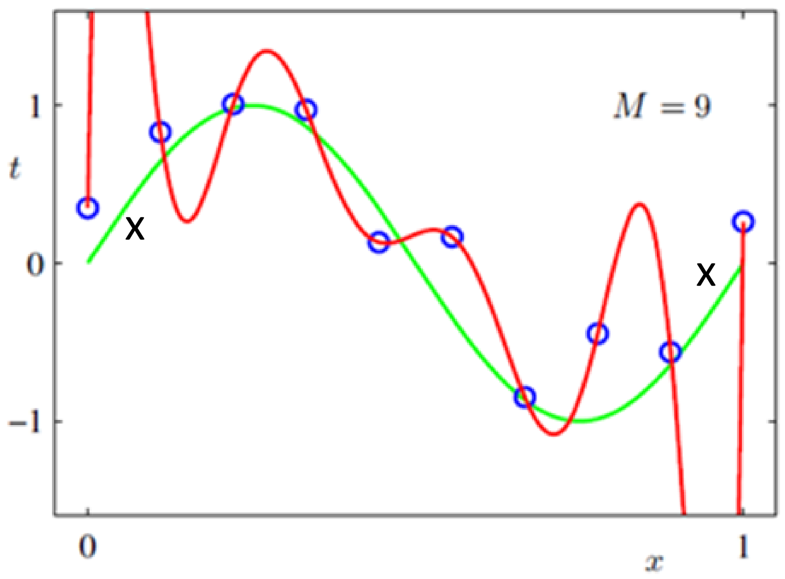
Protip: Never trust people that mess this up!

Splitting the Data

Typically divide the full data set into three:

- **Training data**: learn the model parameters
  - This is the core learning part, and so it needs the most data
  - +/- 60% of the data
- **Validation data**: learn the model hyperparameters
  - Hyperparameters are values set before training begins, e.g. the degree of the polynomial, the complexity of the neural network
  - +/- 20% of the data
- **Testing data**: report quality of model
  - This is used to report an unbiased evaluation of the final model
  - +/- 20% of the data

Why split the data?



This red model has a perfect fit to the blue training points: so they will not give a reliable estimate of how well the model will generalise.

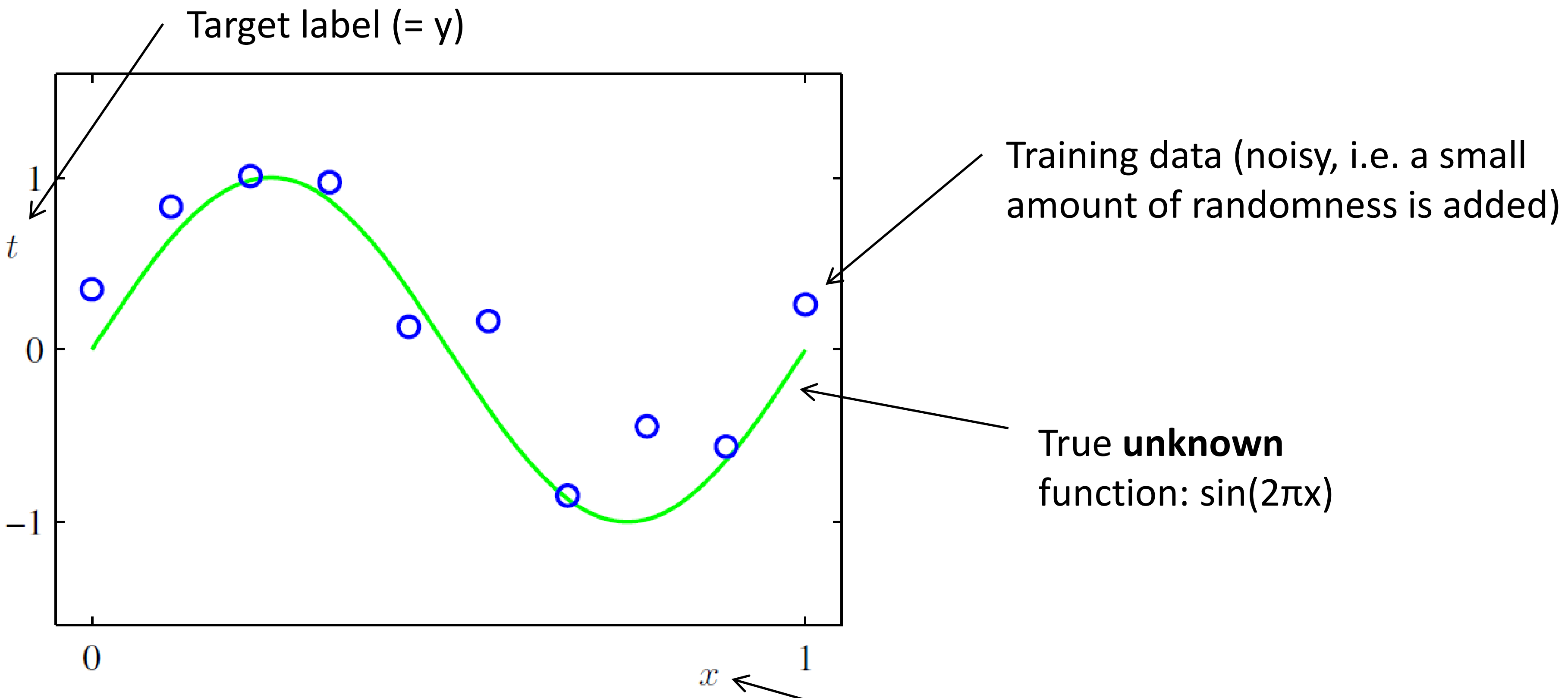
Instead, we want to **test** it on new data points that **it has never seen during training**. This gives a better idea of its performance.

Similarly, we may be learning the hyperparameter of the degree of the model (M), by training a straight line model (M=1), quadratic model (M=2), … up to M=9 and then seeing which is best. We can train them all on the same training data, but we need to use **different validation data** to choose the best one. Again, we can't just report its performance on that data, as it is already biased. So, we then need a **different testing set** to report final scores.

The test data must not be touched until the very end! It is the "blind/surprise test".

Example: Polynomial Curve Fitting

Simple regression (supervised learning) problem



Feature (1D)

Goal: given a new x, predict t (target)

A Polynomial Function

Assume the function is polynomial:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Note: this is a linear model

- Linear function of coefficients w (the parameters)

Evaluate:

- Use an error function

$$E(\mathbf{w}) = \boxed{\frac{1}{2}}\boxed{\sum_{n=1}^{N}}\boxed{\{\boxed{y(x_n, \mathbf{w})} - t_n\}}^{\boxed{2}}$$

Learning:

- Find the weight vector w to minimise error E(w)
- Unique solution w*
  - E(w) is quadratic in w
  - E'(w) is linear in w

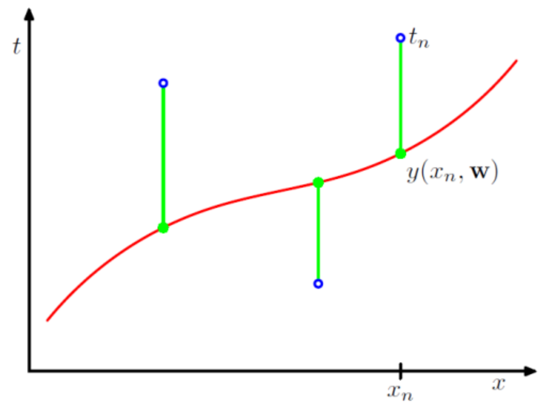
Predicted value at x

Error between predicted and true value t

Squared so it is symmetrical

Sum over every data point

½ to make the maths simpler after differentiating

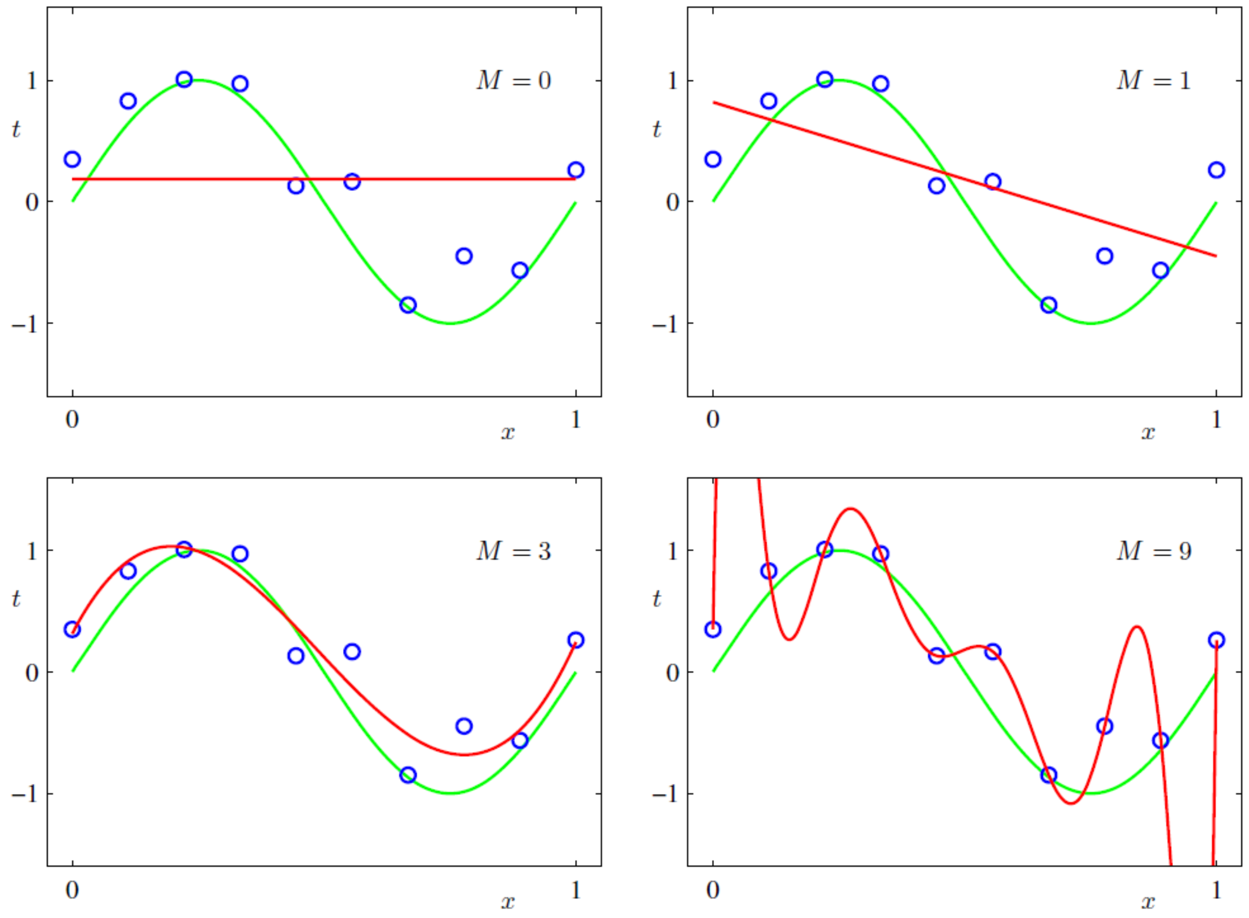


More on this example in the linear regression lecture.

Model Selection

Choosing M (polynomial order)

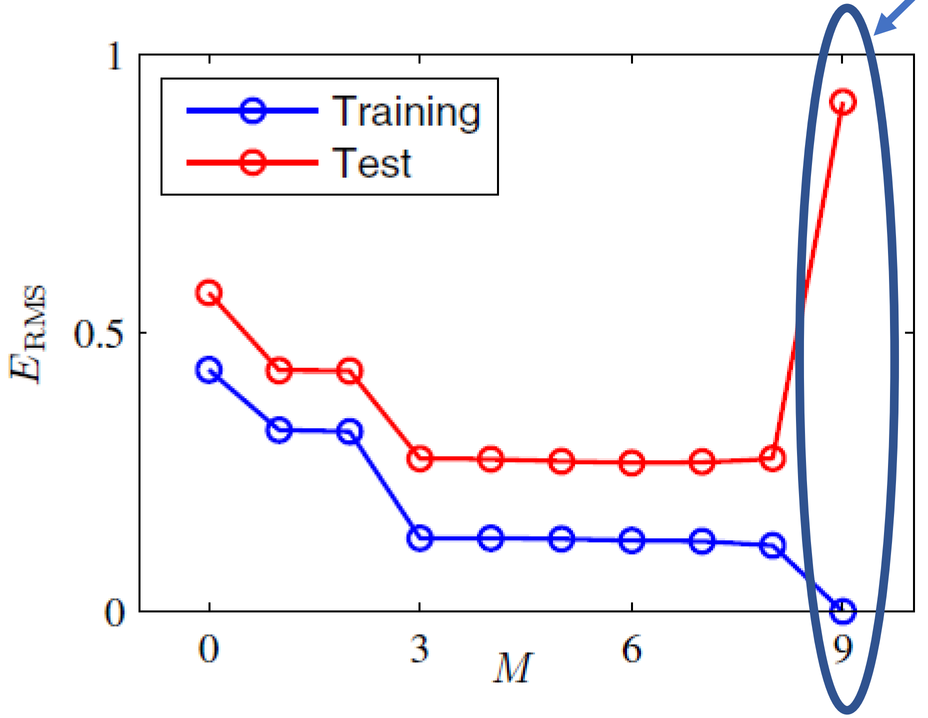
For M = 9, E(w*) = 0! But goal is to **generalise**!



Training vs Testing Error

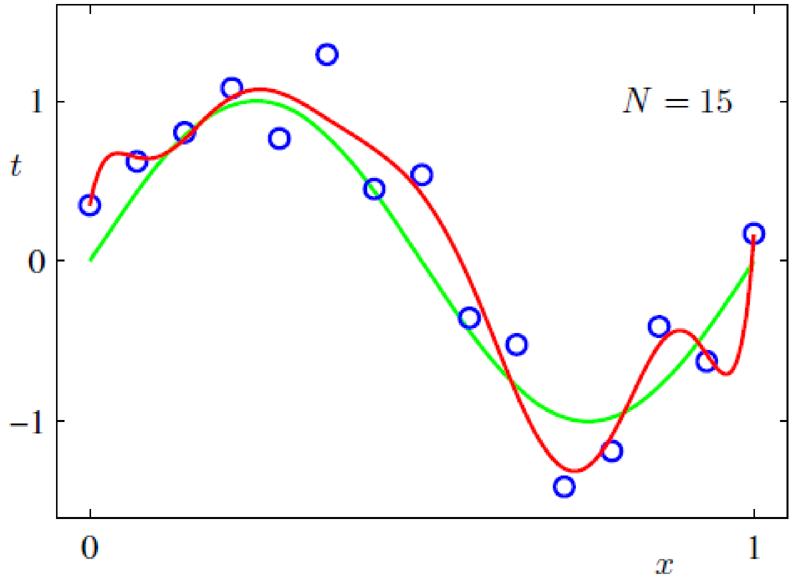
Define error to compare across N:

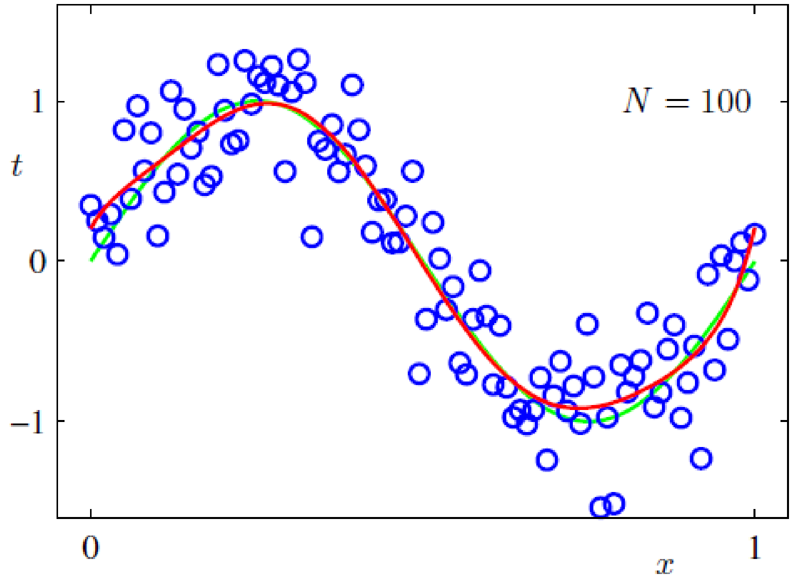
Overfitting: high error on test data, low error on training data



Training error is always better than test error. Why?

Adding More Training Data





More data
- Less severe over-fitting
- More complex model we can fit

There are other strategies to solve this problem (see regularisation later).

Recap

- What is ML and why do we need it
- Example problems
- Supervised, unsupervised, reinforcement learning
- Generalisation, bias-variance trade-off
- Representations
- Curse of dimensionality
- Training, validation, testing data
- Curve-fitting example

Make sure you are comfortable with this all by next week!