Neural Computation

Neural Computation (extended)

Week 1:
Introduction and linear models

[Kashif Rajpoot]

Outline

Module introduction

Outline: module introduction

1. Organisation
2. Assessment and feedback
3. Learning aims and outcomes

Module team

Jinming Duan
j.duan@bham.ac.uk
(Module lead)

Alex Krull
a.f.f.krull@bham.ac.uk

Kashif Rajpoot
k.m.rajpoot@bham.ac.uk
(for Dubai)

Office hours

- Updated office hours schedule and mode
  - https://canvas.bham.ac.uk/courses/70443/pages/office-hours?module_item_id=3245656

Assessment and feedback

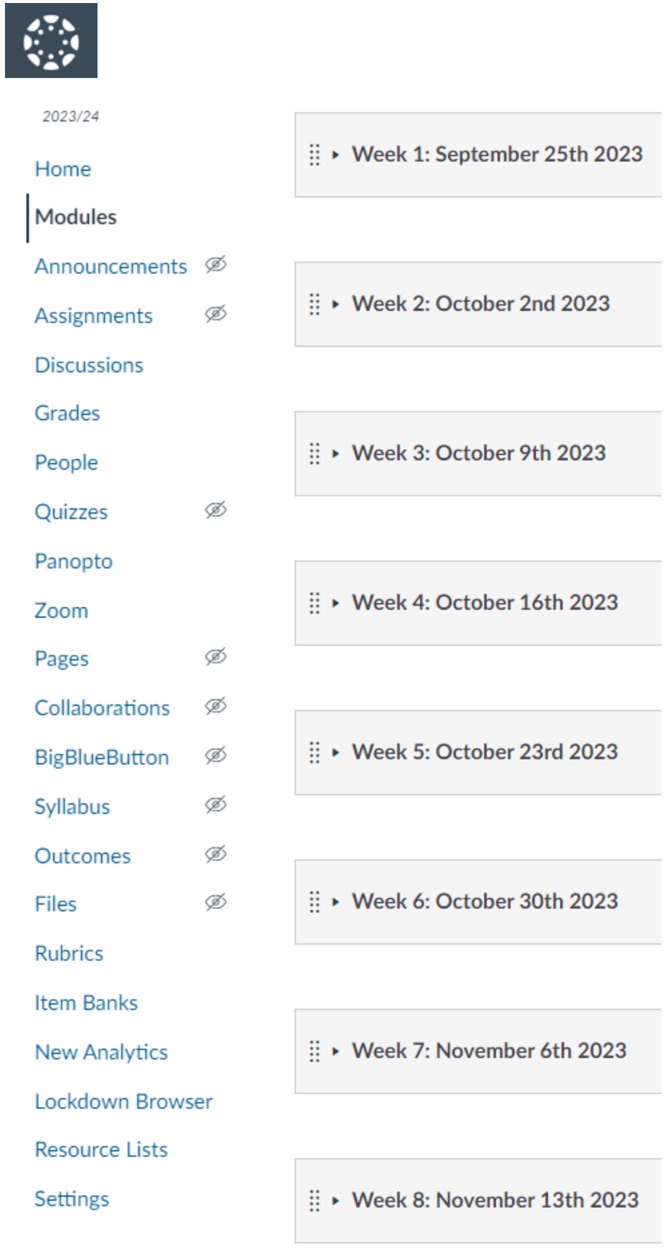
- Written exam: 80% (in May/June period)
- Continuous assessment (CA): 20%

- CA plan (tentative)
  - CA1 (10%), week 6 (via Canvas)
  - CA2 (10%), week 11 (via Canvas)
  - CA answers will be released on Canvas
  - CA feedback will be provided on Canvas

- Non-graded formative assessments
  - Labs (Jupyter Notebook)
  - Exercises

Canvas layout

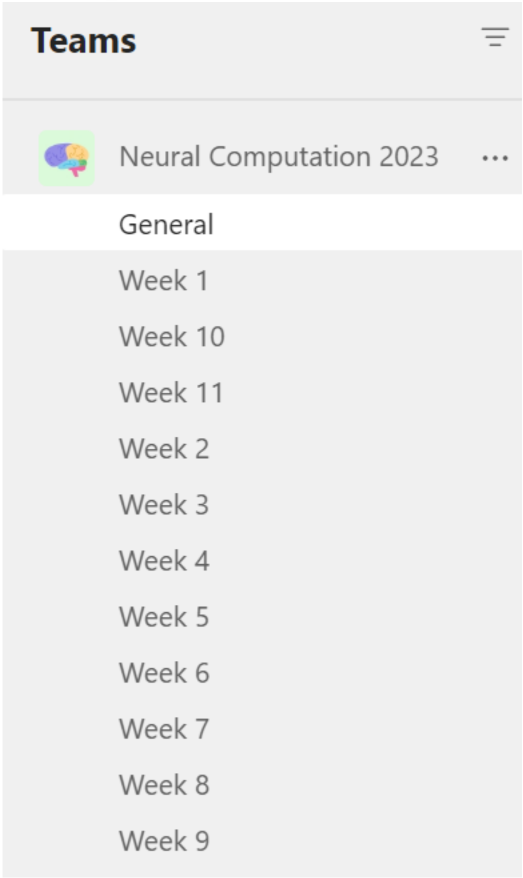
- All learning materials are placed on the Modules page on Canvas
  - https://canvas.bham.ac.uk/courses/70443/modules
- Each section of the Modules page on Canvas typically covers one week
- Introduction is provided for each week, summarising what we talk about, and the order in which we recommend that you study the material
- Each week covers one or more topics.
- Each week will normally have slides, recommended reading material, and a practical lab or tutorial



Microsoft Teams

- We have set up a Microsoft Teams group to enable discussion of any questions about the module.
  - Teaching, learning, coursework, etc.

- To enrol to the Neural Computation 2023 group for participation in discussion
  - Follow instruction at this link to join



Module plan

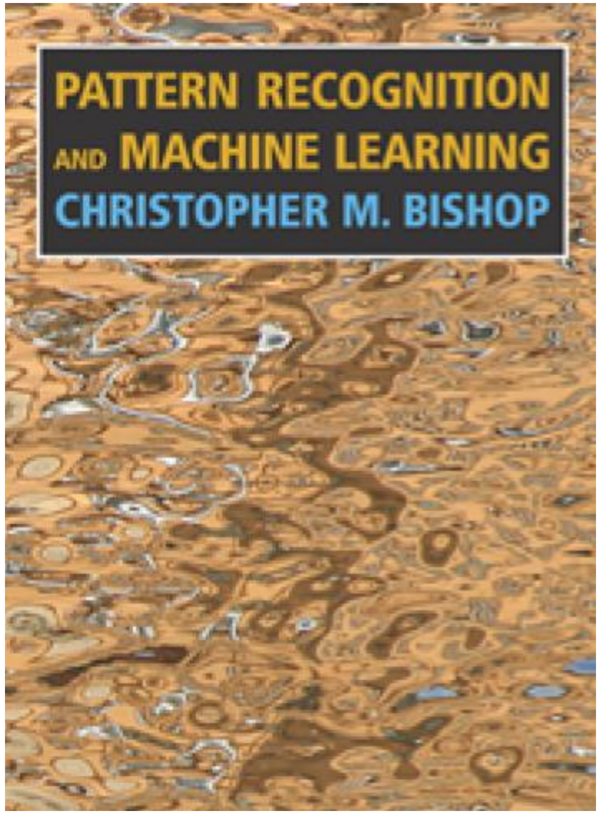
Week	Date	Topic	Edgbaston lecturer	Dubai lecturer	CA	Exam
1	25th Sep	Introduction and Linear Models	Alex	Kashif		
2	2nd Oct	Gradient Descent Methods and Linear Classification	Jinming	Kashif		
3	9th Oct	MLP and Backpropagation	Alex	Kashif		
4	16th Oct	Convolutional Neural Networks	Jinming	Kashif		
5	23rd Oct	Auto-encoders (AEs)	Jinming	Kashif		
6	30th Oct	Consolidation week, assessment and Q/A	Alex	Kashif	CA1 (10%)	
7	6th Nov	Variational AEs	Jinming	Kashif		
8	13th Nov	Generative Adversarial Networks	Jinming	Kashif		
9	20th Nov	Recurrent Neural Networks	Jinming	Kashif		
10	27th Nov	Transformers	Alex	Kashif		
11	4th Dec	Diffusion Models	Alex	Kashif	CA2 (10%)	
May/June 2024						Final exam (80%)

Programming environment

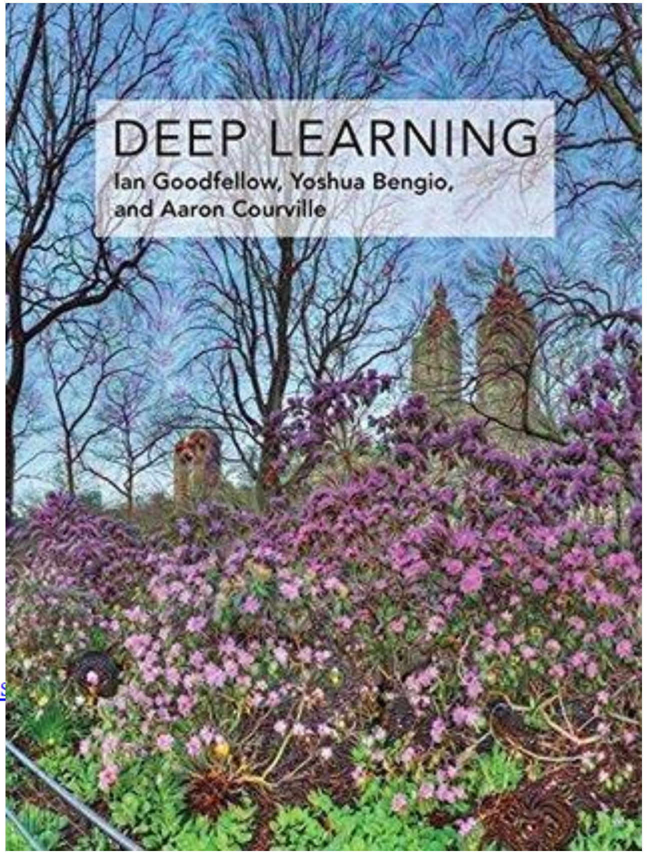
- Online GPU access (details to be shared via Canvas and in class)
- School's machines
  (Edgbaston campus)
- Campus machines
  (Dubai campus)
- Google Colab
  - https://colab.research.google.com/

Recommended textbooks

1. Deep learning, Goodfellow et al., MIT Press, 2016 (online and in Library)
2. Pattern Recognition and Machine Learning, Bishop, Springer, 2007 (online and in Library)



Deep Learning



Free online courses

- Summer School
  - Deep Learning & Reinforcement Learning Summer School
    (http://videolectures.net/DLRLsummerschool2018_toronto/)
- Modules
  - DeepLearning.ai (Andrew Ng)
    (https://www.youtube.com/channel/UCcIXc5mJsHVYTZR1maL5l9w/playlists)
  - CS231n – Convolutional Neural Networks (Stanford)
    (https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv)
  - CS224d – Natural Language Processing (Stanford)
    (https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6)
  - Fast.ai
    (https://www.youtube.com/playlist?list=PLCdvEQLhYkYmKTKWTrH7bHtQ1CsKZaQBl)

Pre-requisite (knowledge)

- This module requires a solid **Maths background**
  - Canvas self-learning module to refresh on Maths
    - https://canvas.bham.ac.uk/courses/60674
    - Email your lecturer to enrol, if you can't access it and need a refresher on Maths
  - Linear Algebra: vector/matrix manipulations
    - MIT Open Course Linear Algebra
  - Calculus: partial derivative, chain rule
    - MIT Open Course Multivariable Calculus
  - Probability
    - MIT Open Course Introduction to Probability
- This module expects familiarity in **Python programming**
  - and NumPy, SciPy, Matplotlib, scikit-learn, Pandas

Module learning aims

- The aims of this module are to:
  1. Introduce some of the fundamental techniques and principles of neural networks
  2. Investigate some common neural-network architectures and their applications
  3. Present neural networks in the larger context of state-of-the-art techniques of automated learning

Module learning outcomes

- On successful completion of this module, the student should be able to:
    1. Describe and explain some of the principal architectures and learning algorithms of neural computation
    2. Explain the learning and generalisation aspects of neural computation networks
    3. Demonstrate an understanding of the benefits and limitations of neural networks in comparison to other machine learning methods
    4. Develop and apply neural network models to specific technical and scientific problems

Summary: module introduction

- Module organization
- Module aims and outcomes
- Module resources
- Programming environment

Machine learning and neural computation

What is machine learning?

- Definition by Tom Mitchell (1997)
  - An algorithm is said to learn from Experience E with respect to some class of Tasks T and Performance Measure P, if its Performance P at Task in T improves with Experience E
- Toy block building
  - E: knowledge of physical world
  - T: building a tower with toy block
  - P: how tall the tower is

Tasks (T)

- Classification
- Regression
- Machine translation

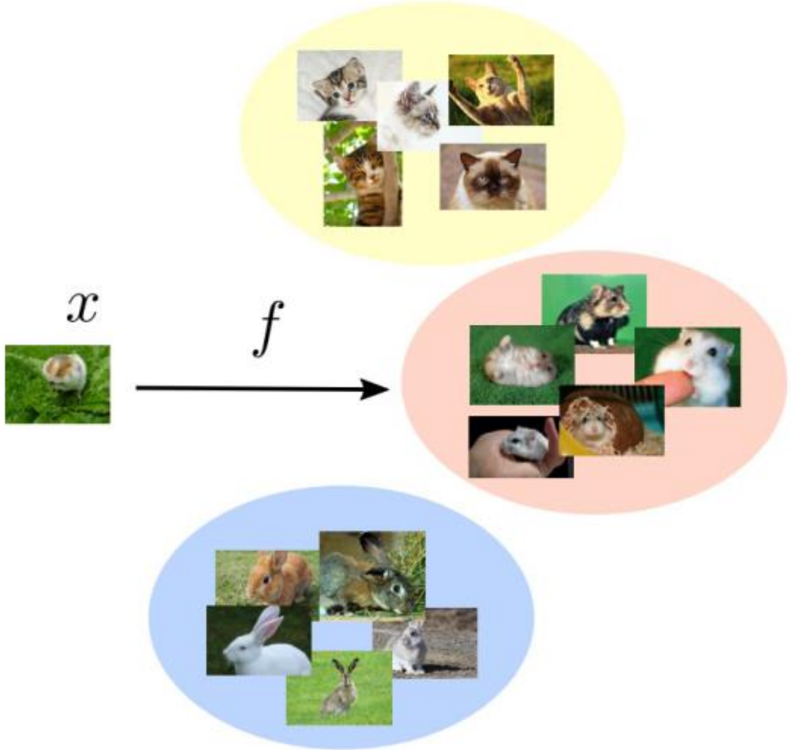
Classification

- Construct a function

$$f : \mathbb{R}^d \mapsto \{1, \dots, k\}$$

such that if an object with features $\boldsymbol{x} \in \mathbb{R}^d$ belongs to a class $y \in \{1, \dots, k\}$ then

$$f(\boldsymbol{x}) = y$$



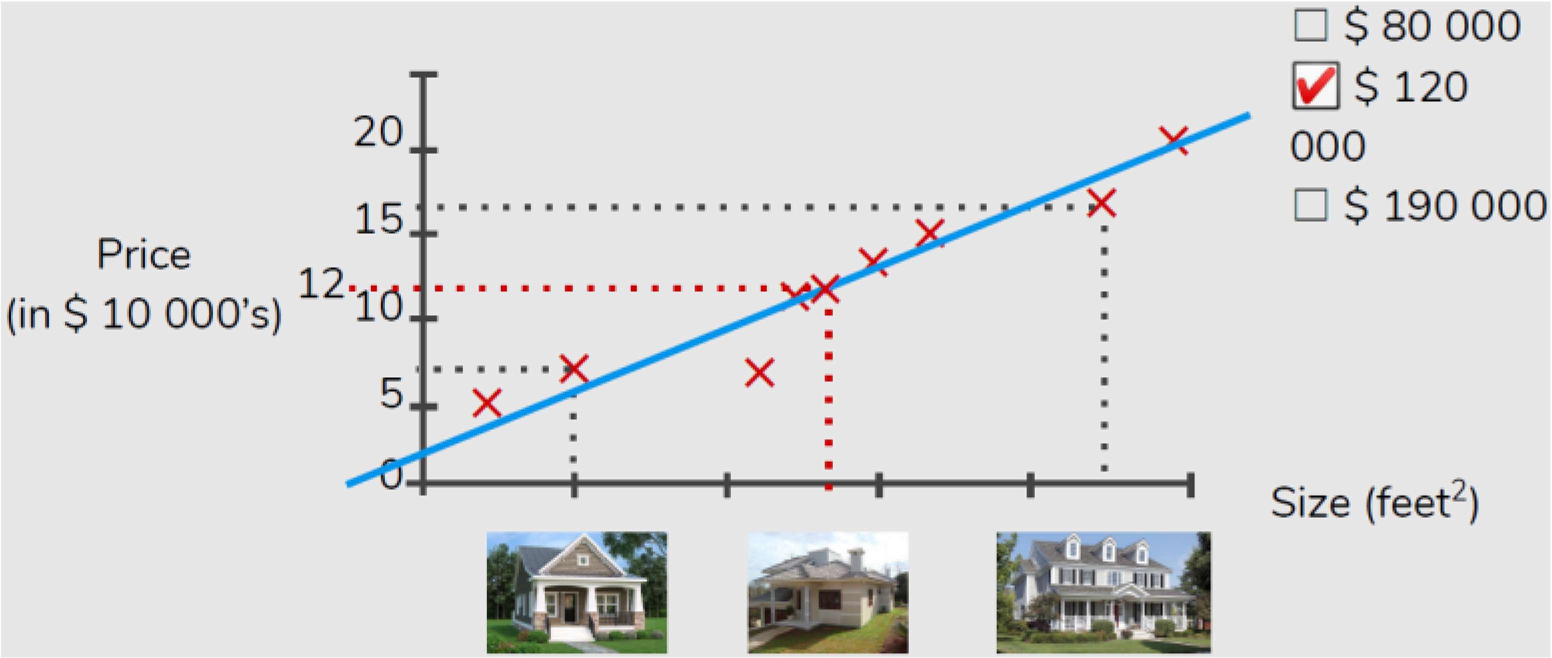
- Alternatively, construct a function which given features returns the probability of each class

Regression

- Predict a numerical output given some input, i.e., a function

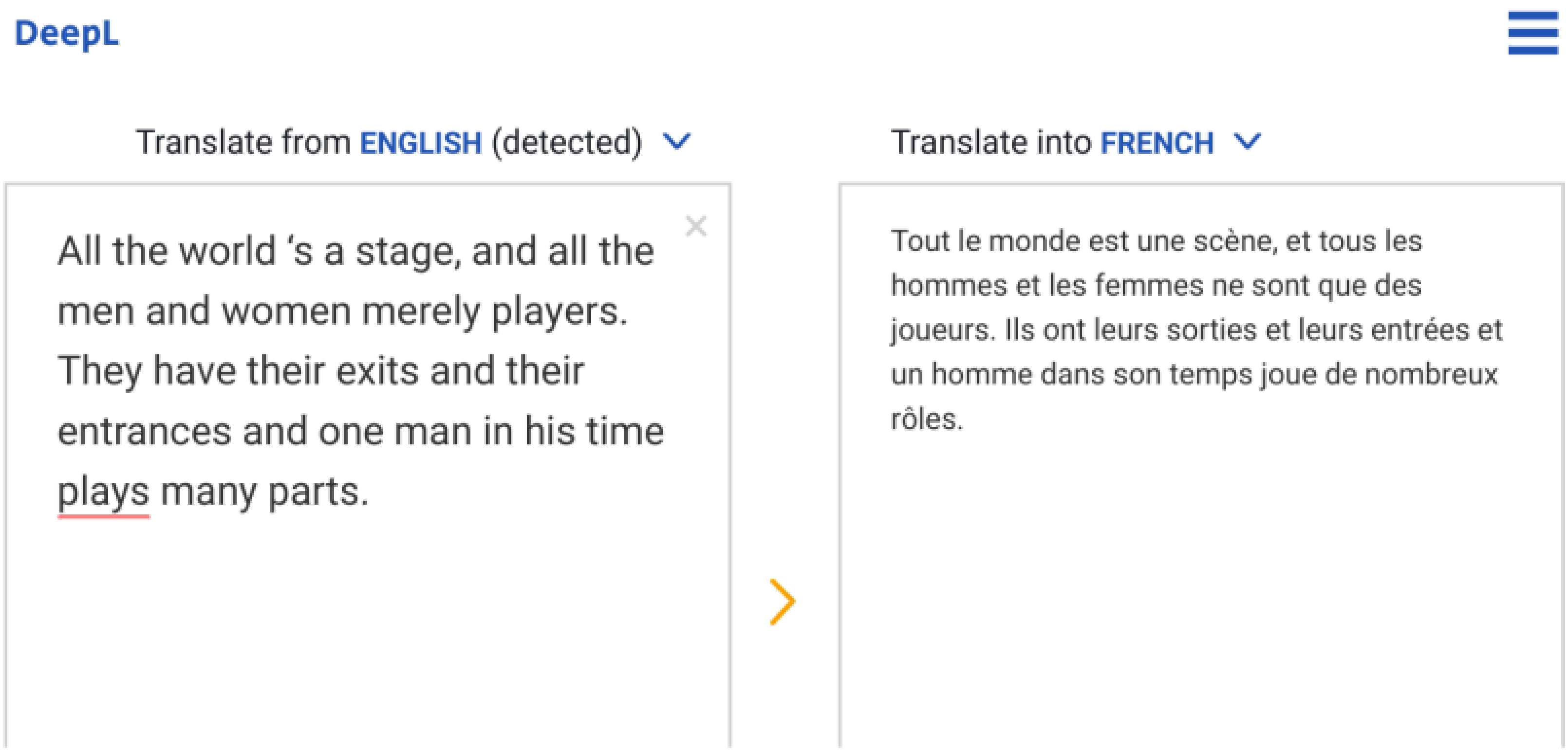
$$f: \mathbb{R}^d \mapsto \mathbb{R}$$

- Example: house price prediction
  - Input: House information (living size, lot size, location, # floors)
  - Output: Price



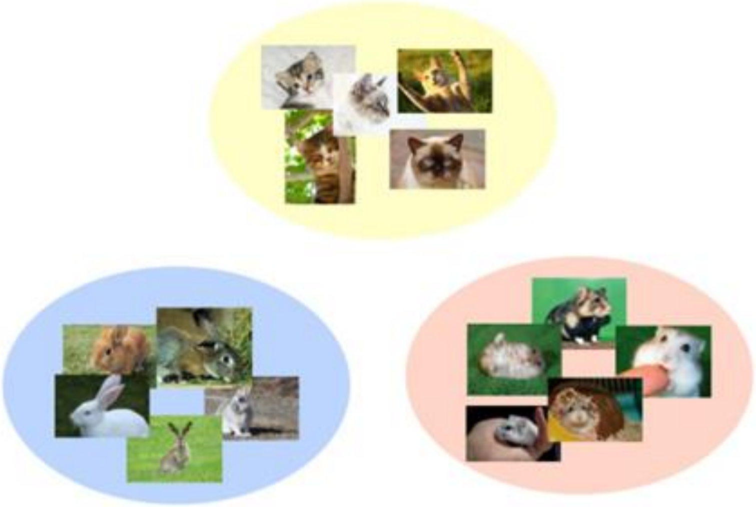
Machine translation

- Translation from a source language to a target language
  - Input: sequence of characters (e.g., English text)
  - Output: sequence of characters (e.g., French text)



Experience (E)

- We obtain experience by observing a dataset from nature
- For classification
  - $D = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \dots, (\boldsymbol{x}^n, y^n)\}$
    where $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in \{1, \dots, k\}$

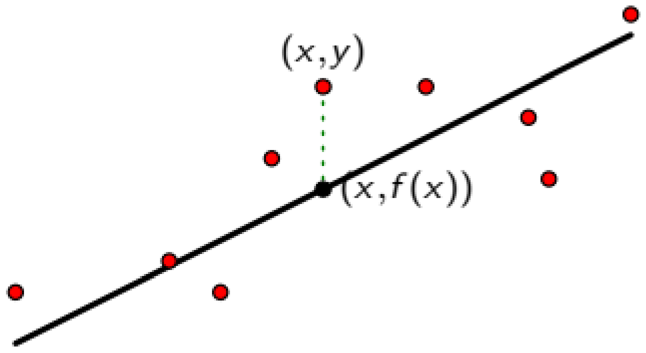


- For regression
  - $D = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \dots, (\boldsymbol{x}^n, y^n)\}$
    where $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$

Performance (P)

- For classification, accuracy is a common performance measure
  - Proportion of correctly classified examples (typically reported as a percentage)

- For regression, residual is a common performance measure
  - e.g., mean of sum of square of differences



What is machine learning?

- Some alternate (and relatively modern) definitions
  - A type of AI in which computers use huge amounts of data to learn how to do tasks rather than being programmed to do them
    - https://www.oxfordlearnersdictionaries.com/definition/english/machine-learning
  - Machine learning is a branch of AI which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy
    - https://www.ibm.com/topics/machine-learning

What is neural computation?

- Neural computation deals with the machine learning problem by neural networks (NNs).
  - At the intersection of neuroscience, computer science, and mathematics.

- This module focuses on:
  1. Fundamental theory
  2. Methodologies for constructing modern deep neural networks
  3. Practical experience of designing and implementing a neural network for a real-world application.

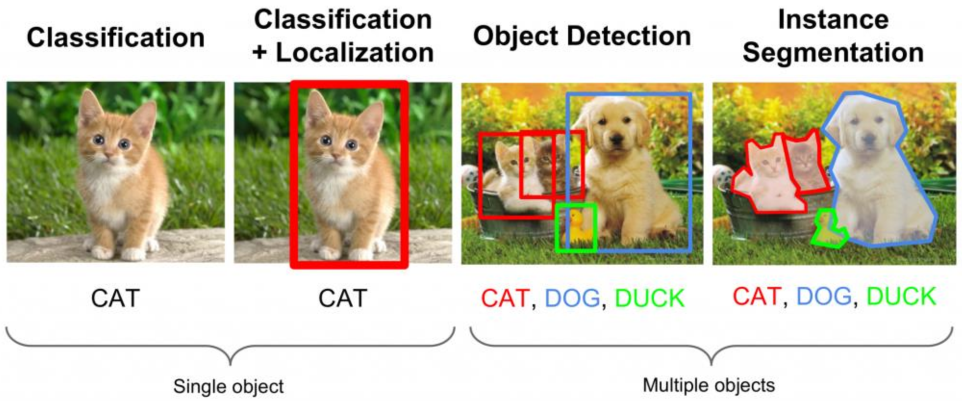
What can we do with deep NNs?

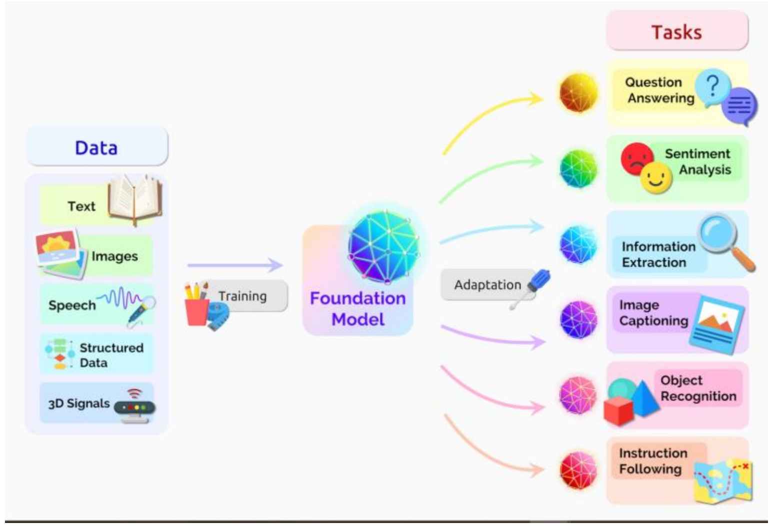
ChatGPT

Bard

FalconLLM

جيس
JAIS

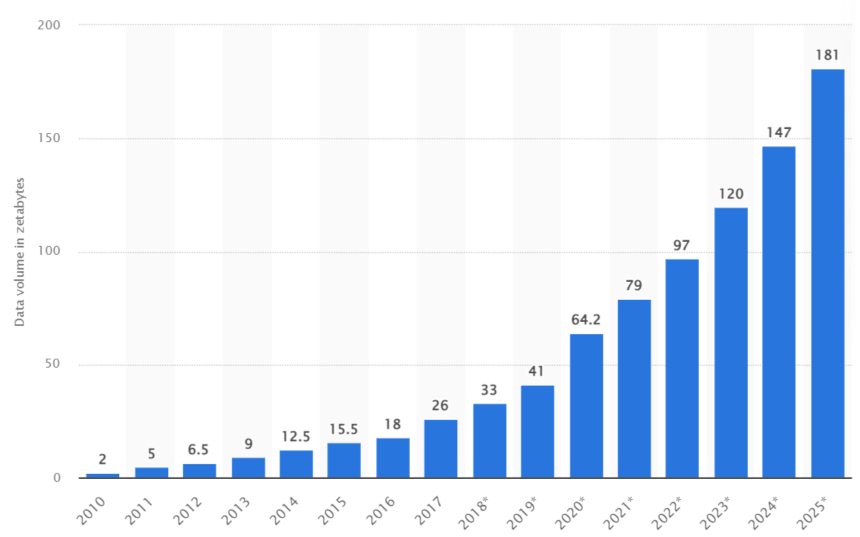




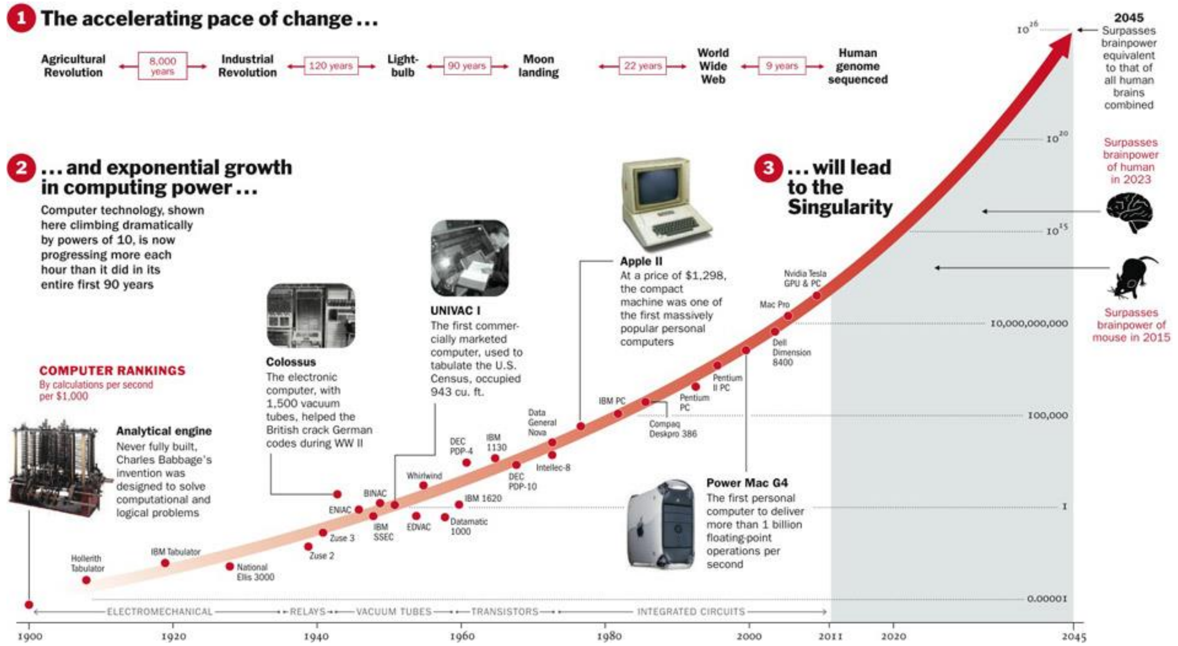
What has led to the surge in NNs (and AI)?

1. Lots of digital data
   - e.g., web, pictures, videos

2. Lots of compute power
   - e.g., GPUs, TPUs



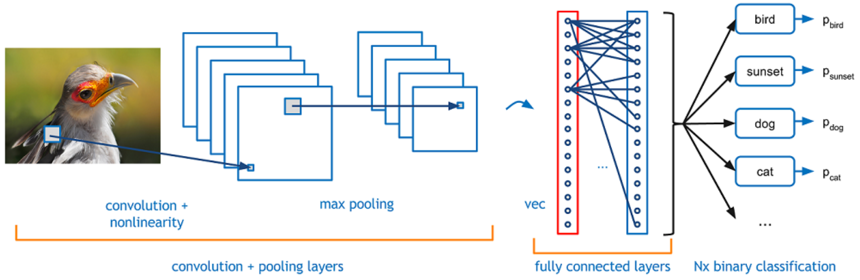
https://www.statista.com/statistics/871513/worldwide-data-created/

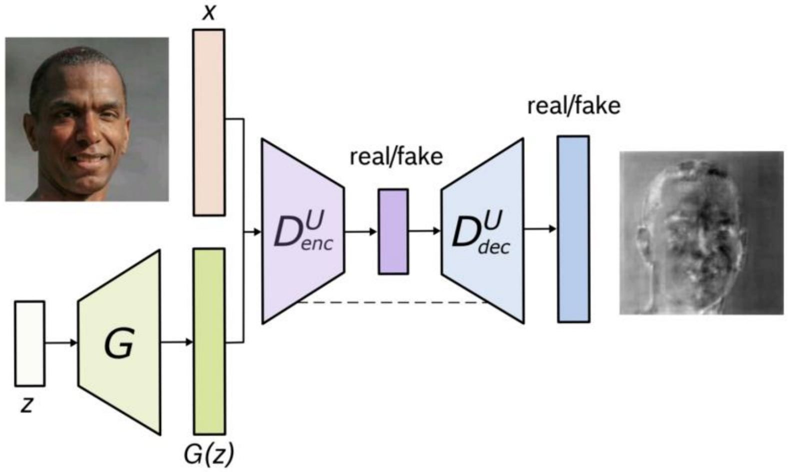


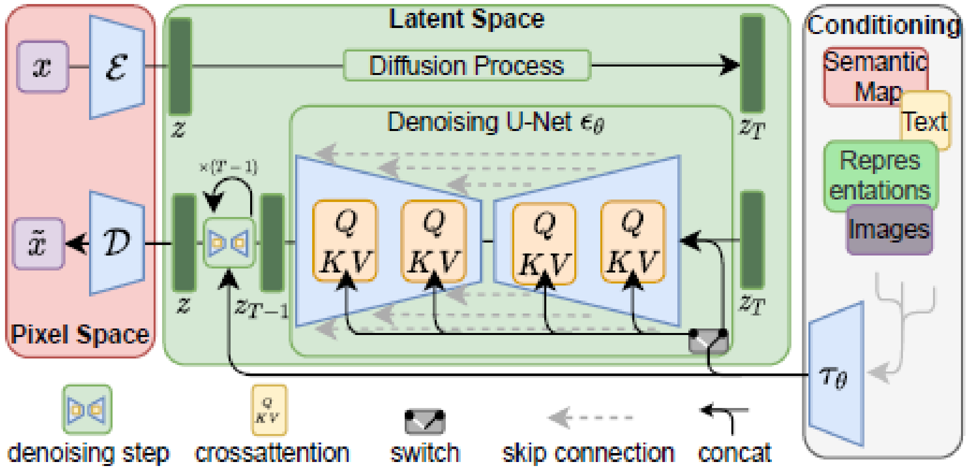
https://content.time.com/time/interactive/0%2C31813%2C2048601%2C00.html

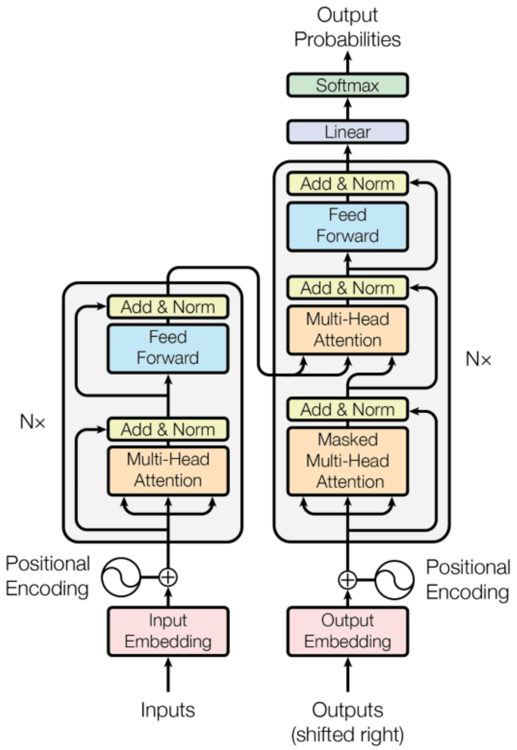
What has led to the surge in NNs (and AI)?

3. Lots of NN innovations and architectures
   - e.g., CNNs, GANs, transformers, diffusion models









NNs and ML

- NNs are just one way to deal with machine learning
- There are several other approaches that have been developed over the decades, for example:
  - Natural computation (e.g., genetic algorithms)
  - Decision trees (e.g., random forests)
  - Statistical approaches (e.g., Bayesian)
  - Decision boundaries (e.g., support vector machines)

- This module primarily focuses on NN based approaches.

Summary: ML and NN

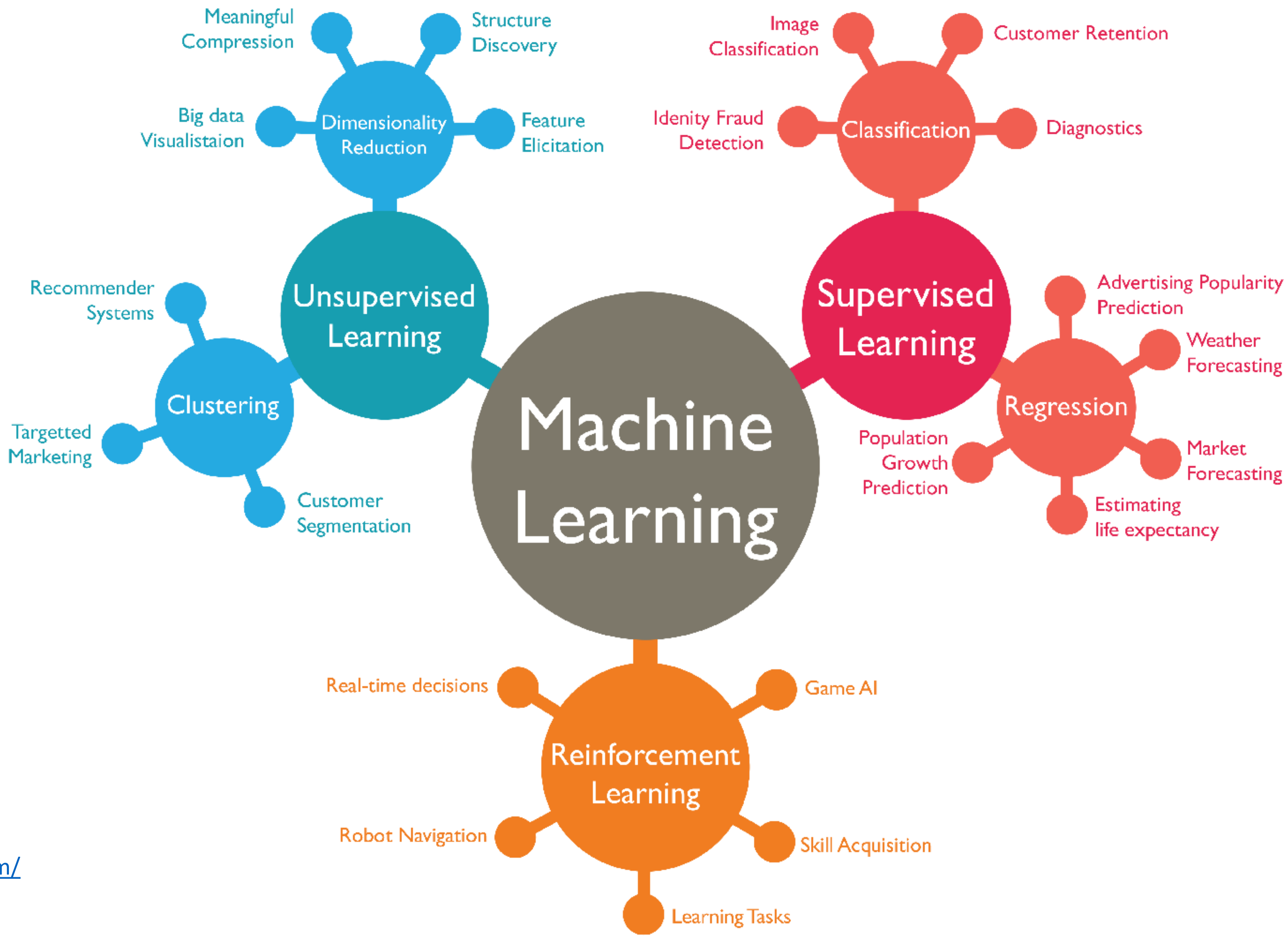
- Machine learning occurs when:
  - performance P of algorithm at task T improves with experience E
- Machine learning introduction
- Neural computation introduction

Machine learning fundamentals

Outline: ML fundamentals

1. Types of machine learning
2. Training and testing
3. Overfitting and underfitting
4. ML workflow

Types of machine learning

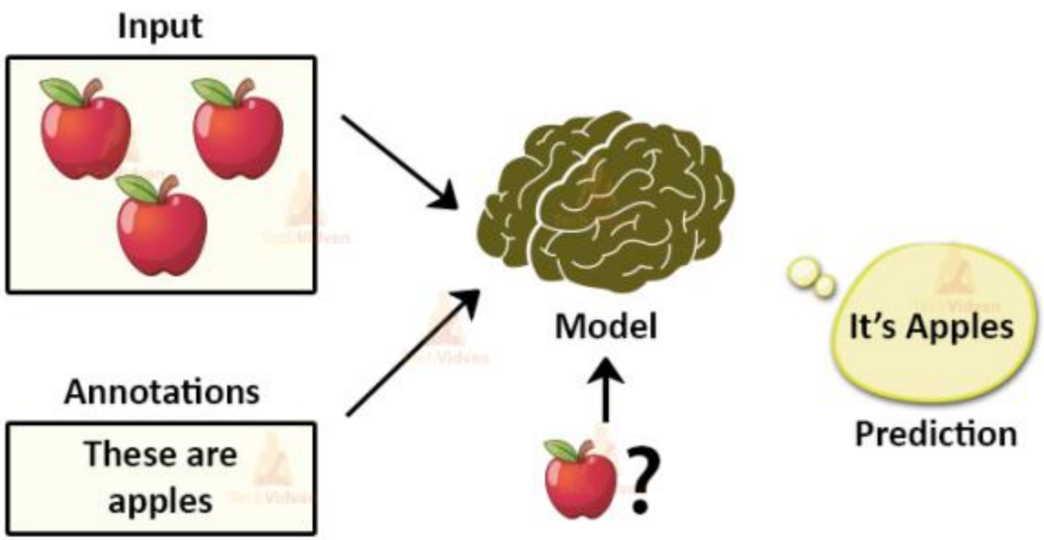


https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1

Supervised learning

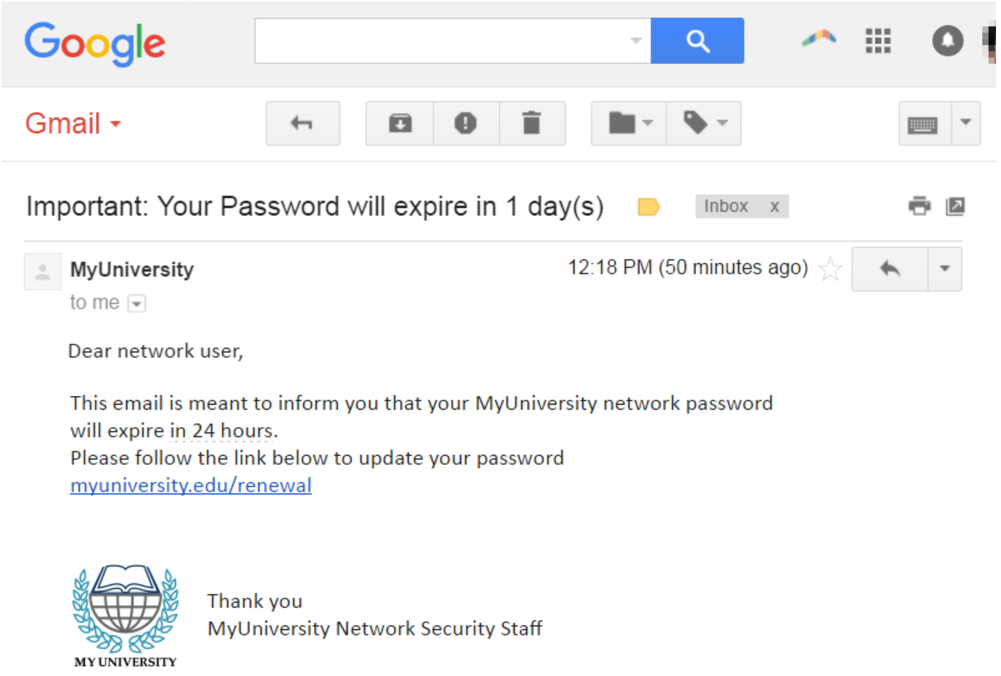
- Correct output known for each training example
- Learn a function that maps an input to an output based on examples of input-output pairs

- Classification learns to predict discrete values (class labels)
- Regression learns to predict continuous values



Supervised learning: binary classification

- Spam email classification
- Input: Incoming email
- Output: “SPAM” or “NOT SPAM”
- A binary classification problem, because only 2 possible outputs
- Performance: the number of messages correctly classified



Supervised learning: multiclass classification

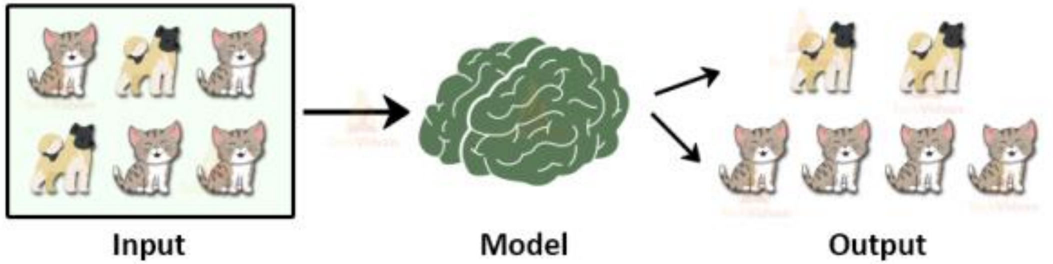
- Medical diagnosis
- Input: Symptoms (fever, cough, fast breathing, shaking, no smell,...)
- Output: Diagnosis (coronavirus, flu, common cold, pneumonia, ...)
- A multiclass classification problem: choosing one of several [discrete] outputs
- Performance: the number of correct diagnosis

- How to express uncertainty?
  - Probabilistic classification

P(coronavirus) = 0.7; P(flu) = 0.2; P(common_cold) = 0.1

Unsupervised learning

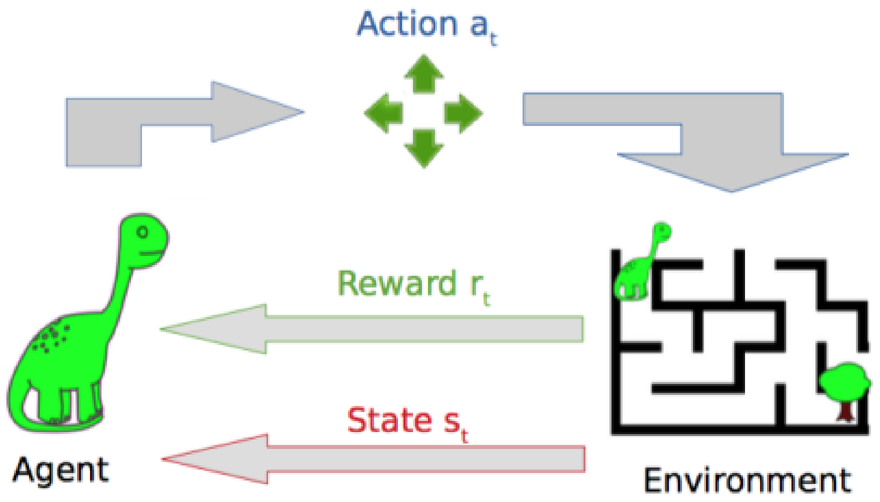
- Create an internal representation of the input, capturing regularities/structure in data
- Clustering algorithm tries to detect similar groups



- Learn interesting patterns from dataset with <u>no labels</u>: $D = \{(\boldsymbol{x}^1), (\boldsymbol{x}^2), \dots, (\boldsymbol{x}^n)\}$
- Dimensionality reduction tries to simplify the data without losing too much information

Reinforcement learning

- An agent learns how to interact with an environment (e.g., a game of maze)
- In each time step
  - the agent receives observations (e.g., (x; y)) which give it information about the state (e.g., positions of the dinosaur)
  - the agent picks an action (e.g., moving direction) which affects the state
- The agent periodically receives a reward (e.g., +1 or −1 per step)
- The agent wants to learn a policy, or mapping from observations to actions, which maximises its average reward over time



Training and testing

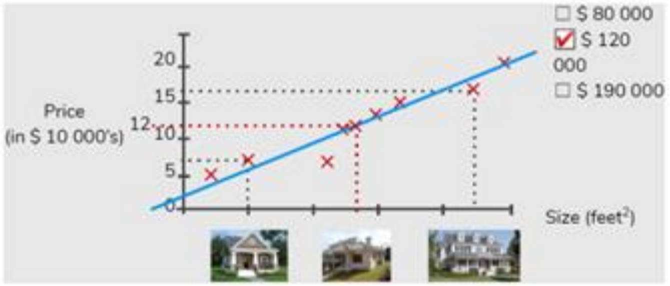
- Input: House information (living size, lot size, location, # floors)
- Output: Price
- Aim: find a model to predict price based on feature information of house
- Training dataset: a sequence of (features, price) pairs

$ 70,000

$ 160,000

- Prediction/testing: given information of new house, predict its price?

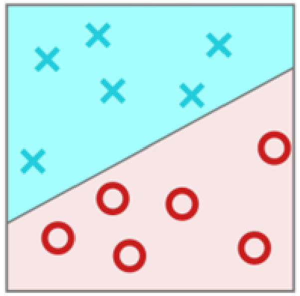
???



- Performance: difference between predicted price and true price

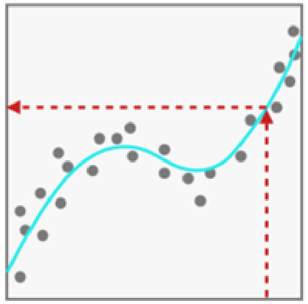
Training and testing

- Training dataset: a dataset that contains n samples
  - $D = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \dots, (\boldsymbol{x}^n, y^n)\}$
  - $\boldsymbol{x}$ represents input, y represents output
- Classification: $y \in \{-1, +1\}$
  - +1 means positive examples
  - −1 means negative examples



- Regression: $y \in \mathbb{R}$
  - Find a function $f: \boldsymbol{x} \mapsto y$ such that

$$y \approx f(\boldsymbol{x})$$



- Prediction: given a new input $\boldsymbol{x}$, use f to do prediction
  - e.g., if a house has *x* square feet, predicting its price?

Training and testing: loss function

- A loss function scores how far off a prediction is from the desired "target" output
  - Loss(prediction, target) returns a number called "the loss"
  - Big Loss = Bad Error
  - Small Loss = Minor Error
  - Zero Loss = No Error
- **0-1 loss** for classification
  - Loss is 1 if prediction is wrong
  - Loss is 0 if prediction is correct
- **Square loss** for regression
  - loss = (predicted - target)^2

Training and testing: evaluation

- Evaluating a prediction function f
  - Data science intern gives you a prediction function $f(\boldsymbol{x})$
    - "Average error on training data was 1%"
  - Product manager says "we can deploy if ≤ 2% error"
  - Shall we deploy this prediction function?
    - No!
- Prediction function needs to do well on new inputs!!!
- The only way to know how well a model will generalise to new cases is to actually try it out on new cases
  - Training set: only for training prediction functions
  - Test set: only for assessing performance (independent of training)
- Key message: Performance on training set has a bias and cannot serve as an estimate of its performance in real world!

Training and testing: error

- Training error: the error for using a model f to do prediction on training data

$$Err_{train}(f) = \frac{1}{n}\sum_{i=1}^{n} Loss\big(f(\boldsymbol{x}^i), y^i\big)$$

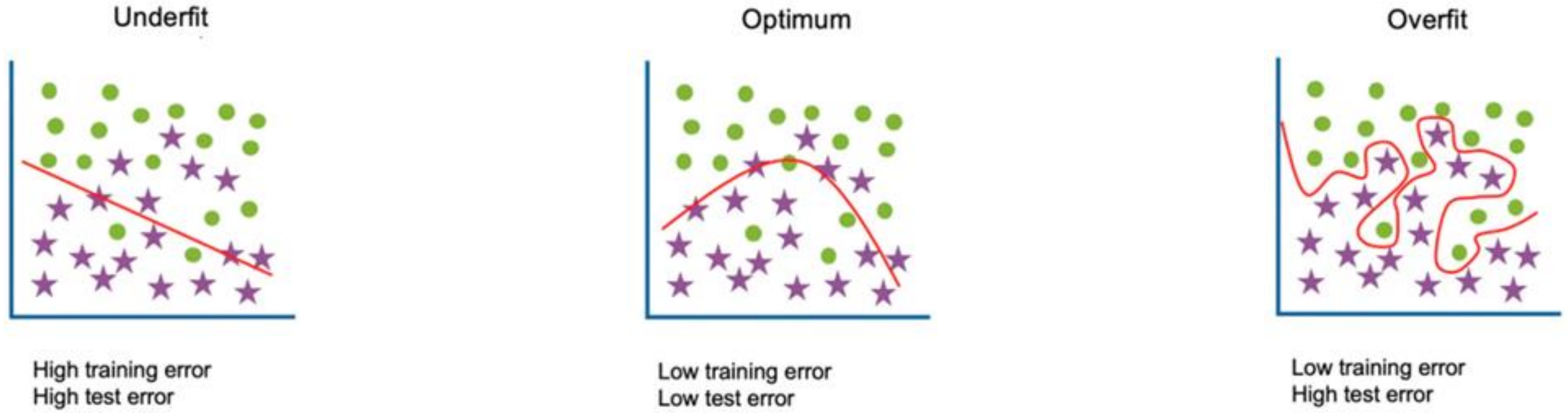
  - For house price prediction, this can be $Loss\big(f(\boldsymbol{x}^i), y^i\big) = \big(f(\boldsymbol{x}^i) - y^i\big)^2$
- Testing error: the error $Err_{test}(f)$ for using a model f to do prediction on test data
- Error decomposition: we decompose the test error by

$$Err_{test}(f) = Err_{train}(f) + \underbrace{Err_{test}(f) - Err_{train}(f)}_{\mathrm{Gen_{gap}}(f)}$$

  - Typically, the generalisation gap $\mathrm{Gen_{gap}}(f)$ is greater than 0
  - A good model has a small $\mathrm{Gen_{gap}}(f)$

Underfitting and overfitting

- Loosely speaking, we say a model underfits when
  - training performance is poor
- We say a model overfits when
  - training performance is good but
  - test performance is poor



Underfitting and overfitting

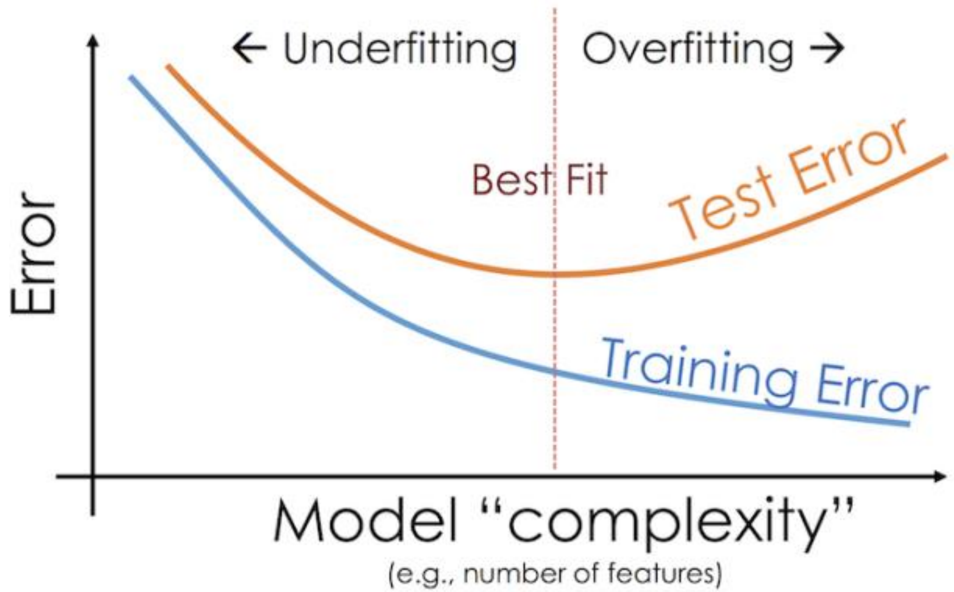
- <u>Overfitting</u> the training data
  - Overfitting means that the model performs well on the training data, but it does not generalise to testing data
  - It happens when the model is too complex relative to the amount and noisiness of the training data

Underfitting and overfitting

- Underfitting the training data
  - Underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data

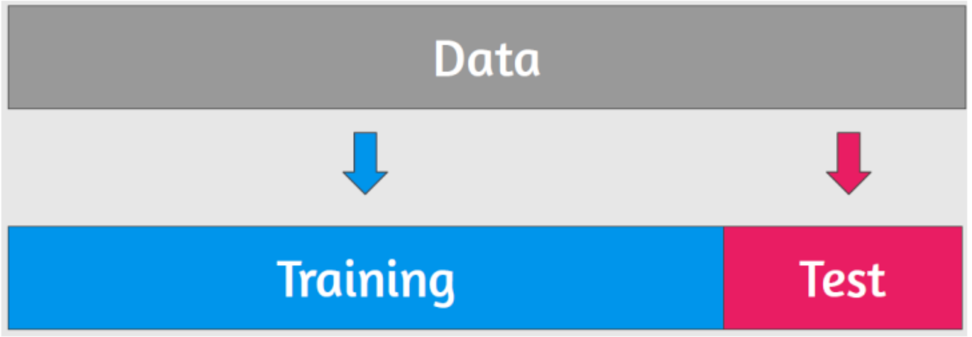
Underfitting and overfitting

- In general, the training error decreases as we add complexity to our model with additional features or more complex prediction mechanisms
- The test error, on the other hand, decreases up to a certain amount of complexity then increases again as the model overfits the training set



ML workflow: data split

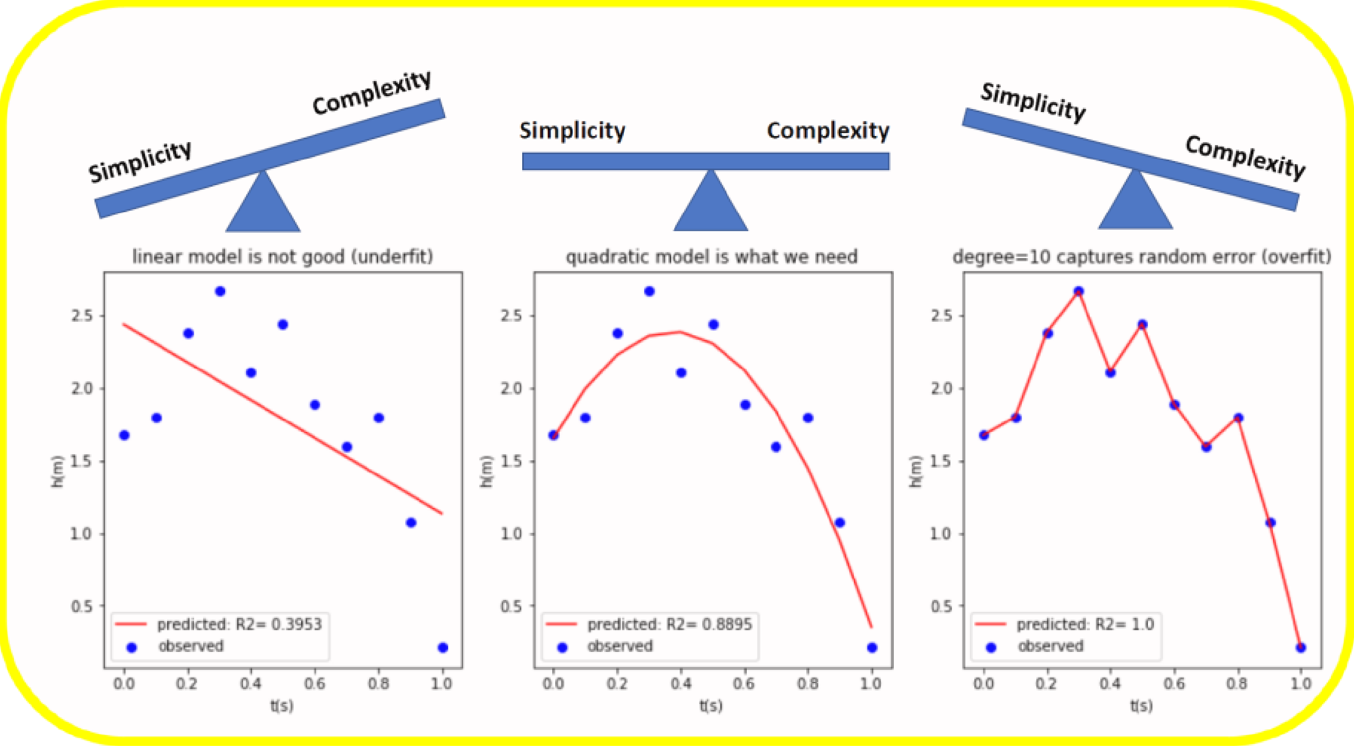
- It is common to use 80% of the data for training and hold out 20% for testing
  - Training set: only for training prediction functions
  - Test set: only for assessing performance (independent of training)



- Give training set to data science intern, you keep the test set
- Intern gives you a prediction function
- You evaluate prediction function on test set
- No matter what intern did with training set
  - "test performance should give you good estimate of deployment performance"

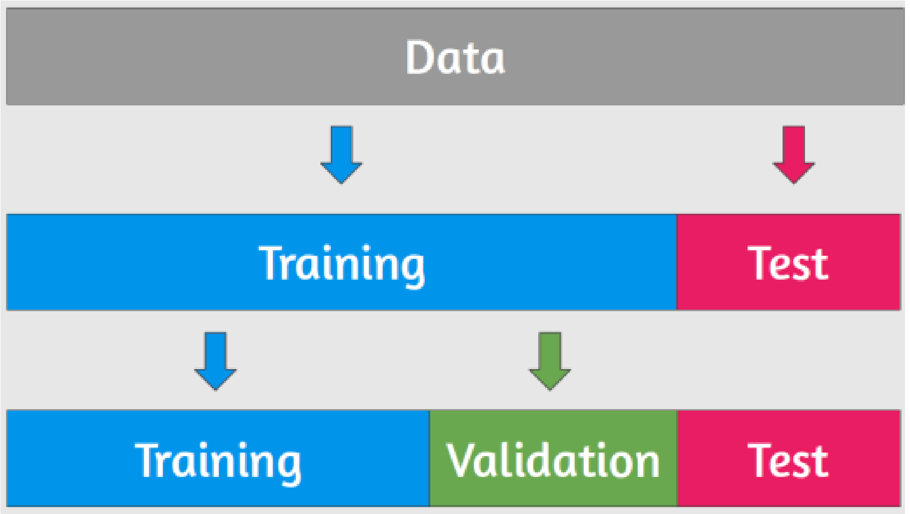
ML workflow: how to find a good model?

- Model selection problem
  - Intern wants to try many fancy ML models: each gives a different prediction function.
  - How should they choose one model?



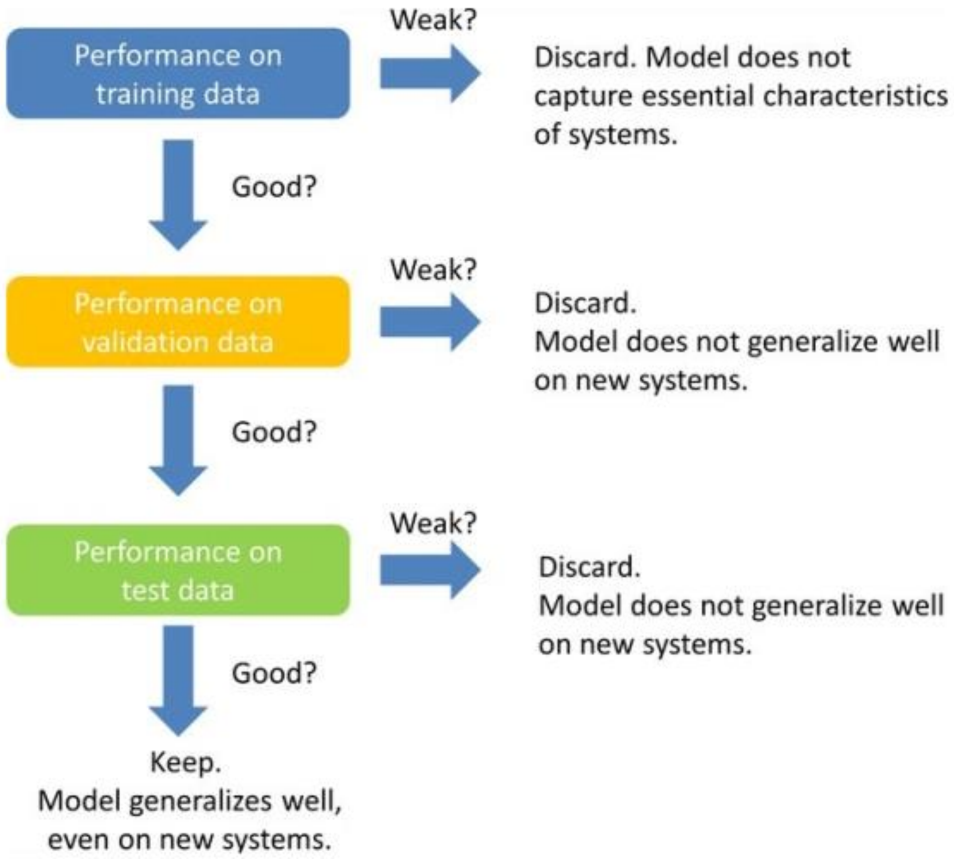
ML workflow: how to find a good model?

- Model selection problem
  - Intern needs her own test set to evaluate prediction functions
  - Intern should randomly split data again into (80/20 split)
    - training set
    - validation set
  - Validation set is like test set, but used to choose best among many prediction functions
  - Test set is just used to evaluate the final chosen prediction function



ML workflow: data split

- Split labelled data into training, validation, and test sets
- Repeat until happy with performance on validation set
  - Choose some ML algorithm
  - Train ML models with various complexities
  - Evaluate prediction functions on validation set
- Evaluate performance on test set



Summary: ML fundamentals

- Types of ML
  - supervised learning: predict input-output mapping
  - unsupervised learning: find pattern among input data
  - reinforcement learning: interaction with environment
- Key concepts in ML
  - training and testing
  - overfitting and underfitting
  - ML workflow

Linear regression

Linear regression

- Why study linear regression?
  - A fundamental and easy to understand method
  - Theoretically sound
  - More complex models require understanding linear regression
  - Familiarity with key concepts of machine learning (e.g., function modelling, regularisation)
  - Familiarity with key notations of machine learning (e.g., features, target, loss)

Linear regression: toy example

- Commute time on bus
  - Want to predict commute time to University
  - Input variables (features)?
    - Distance to University
    - Day of the week
  - Output / target?
    - Commute time
  - Data
    - day = 1 if weekday, day = 0 otherwise

Dist (km)	Day	Commute time (min)
2.7	1	25
4.1	1	33
1.0	0	15
5.2	1	45
2.8	0	22

Linear regression: problem setup

- Dataset D: n input/output pairs (Experience (E))

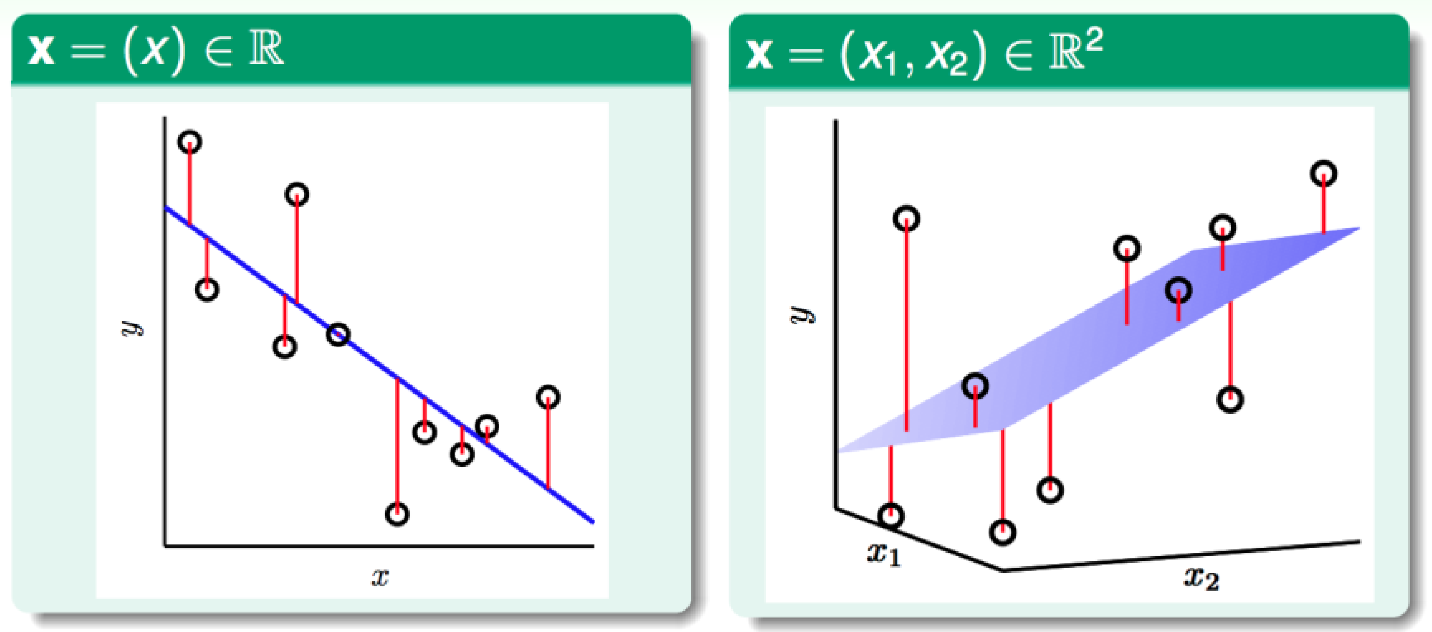
$$D = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^n, y^n)\}$$

  - $\boldsymbol{x}^i \in \mathbb{R}^d$ is the "input" for the i^{th} data point as a feature vector with d elements (e.g., $d = 2$ for day and dist features)
  - $y^i \in \mathbb{R}$ is the "output" for the i^{th} data point (e.g., commute time)
- Regression task (T): find a model, i.e., a function $f: \mathbb{R}^d \mapsto \mathbb{R}$ such that the predicted output $f(\boldsymbol{x})$ is close to the true output y
- Linear Model: a linear regression model has the form

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + \cdots + w_d x_d$$

  - bias (intercept): w_0
  - weight parameters: $w_{1,} w_2, \ldots, w_d$
  - feature: x_i is the i^{th} component of $\boldsymbol{x} \in \mathbb{R}^d$

Linear regression: illustration



- Linear regression: find linear function with small discrepancy!

Linear regression

- Adding a feature for bias term
  - We can add 1 to get an expanded feature vector

Dist (km)	Day	Commute time (min)
x_1	x_2	y
2.7	1	25
4.1	1	33
1.0	0	15
5.2	1	45
2.8	0	22

$\Longrightarrow$

One	Dist (km)	Day	Commute time (min)
x_0	x_1	x_2	y
1	2.7	1	25
1	4.1	1	33
1	1.0	0	15
1	5.2	1	45
1	2.8	0	22

- This allows us to consider the bias in the linear model:

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + \cdots + w_d x_d = (w_0 + w_1 + \cdots + w_d) \binom{1}{\boldsymbol{x}} = \boldsymbol{w}^T \overline{\boldsymbol{x}}$$

- For brevity, we use notation $\boldsymbol{x}$ to represent the extended feature $\overline{\boldsymbol{x}}$:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

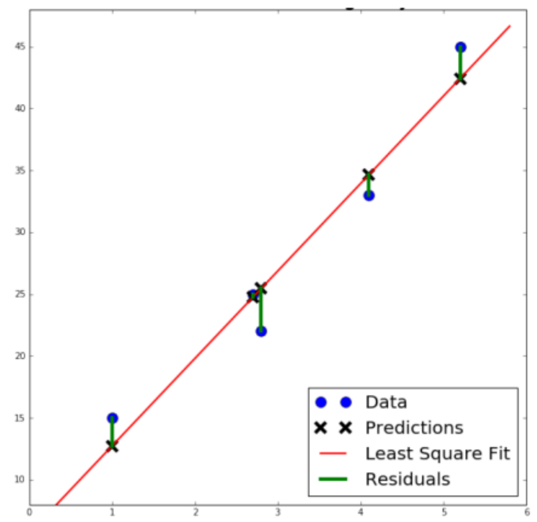
Linear regression: performance measure (P)

- We want a function $C(\boldsymbol{w})$ which quantifies the error in the predictions for a given parameter $\boldsymbol{w}$
- The "residual" on the i^{th} data point can be defined as:

$$e^i = y^i - \boldsymbol{w}^T \boldsymbol{x}^i = y^i - \boldsymbol{x}^{i^T} \boldsymbol{w}$$

- The following mean square error (MSE) $C(\boldsymbol{w})$ takes into account the errors e for all n data points:

$$C(\boldsymbol{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^i - \boldsymbol{x}^{i^T} \boldsymbol{w} \right)^2$$



- By squaring the residual e, we
  - ignore the sign of the residuals
  - penalise large residuals more (if $e^i > 1$)
- Find the parameter $\boldsymbol{w}$ which minimises the loss $C(\boldsymbol{w})$

Linear regression: least squares (1d)

- Let $d = 1$, then:

$$C(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^i - x^i w\right)^2 = \frac{1}{2n}\sum_{i=1}^{n}\left(\underbrace{{x^i}^2 w^2}_{\text{quadratic}} - \underbrace{2y^i x^i w}_{\text{linear}} + \underbrace{{y^i}^2}_{\text{constant}}\right)$$

- It then follows that:

$$C'(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(2{x^i}^2 w - 2y^i x^i\right) = \frac{1}{n}\sum_{i=1}^{n}{x^i}^2 w - \frac{1}{n}\sum_{i=1}^{n} y^i x^i$$

- According to the first-order optimality condition, we know the optimal w^* satisfies

$$C'(w^*) = 0 \Longrightarrow \frac{1}{n}\sum_{i=1}^{n}{x^i}^2 w^* = \frac{1}{n}\sum_{i=1}^{n} y^i x^i$$

- It then follows that:

$$w^* = \frac{\sum_{i=1}^{n} y^i x^i}{\sum_{i=1}^{n} {x^i}^2}$$

- How about the general case (i.e., >1d)?

Linear regression: matrix form

- Let's recall that $C(\boldsymbol{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^i - \boldsymbol{x}^{i^T}\boldsymbol{w}\right)^2$
- Let's consider $\boldsymbol{x}^{i^T} = \left(x_1^i, x_2^i, \dots, x_d^i\right)$

$$X = \begin{pmatrix} \boldsymbol{x}^{1^T} \\ \vdots \\ \boldsymbol{x}^{n^T} \end{pmatrix} \in \mathbb{R}^{n\times d}, \boldsymbol{y} = \begin{pmatrix} y^1 \\ \vdots \\ y^n \end{pmatrix} \in \mathbb{R}^n \Longrightarrow X\boldsymbol{w} - \boldsymbol{y} = \begin{pmatrix} \boldsymbol{x}^{1^T}\boldsymbol{w} - y^1 \\ \vdots \\ \boldsymbol{x}^{n^T}\boldsymbol{w} - y^n \end{pmatrix}$$

- It then follows that:

$$(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y}) = \begin{pmatrix} \boldsymbol{x}^{1^T}\boldsymbol{w} - y^1 & \cdots & \boldsymbol{x}^{n^T}\boldsymbol{w} - y^n \end{pmatrix} \begin{pmatrix} \boldsymbol{x}^{1^T}\boldsymbol{w} - y^1 \\ \vdots \\ \boldsymbol{x}^{n^T}\boldsymbol{w} - y^n \end{pmatrix}$$

$$= \sum_{i=1}^{n}\left(\boldsymbol{x}^{i^T}\boldsymbol{w} - y^i\right)^2 = 2nC(\boldsymbol{w})$$

and

$$C(\boldsymbol{w}) = \frac{1}{2n}(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y}) = \frac{1}{2n}(\boldsymbol{w}^T X^T - \boldsymbol{y}^T)(X\boldsymbol{w} - \boldsymbol{y})$$

$$= \frac{1}{2n}(\boldsymbol{w}^T X^T X\boldsymbol{w} - 2\boldsymbol{w}^T X^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y})$$

note $(X\boldsymbol{w})^T = \boldsymbol{w}^T X^T$

Linear regression: closed-form solution

- Let's recall the objective function:

$$C(\boldsymbol{w}) = \frac{1}{2n}\left(\underbrace{\boldsymbol{w}^T X^T X \boldsymbol{w}}_{\text{quadratic}} - \underbrace{2\boldsymbol{w}^T X^T \boldsymbol{y}}_{\text{linear}} + \underbrace{\boldsymbol{y}^T \boldsymbol{y}}_{\text{constant}}\right)$$

- The gradient of $C(\boldsymbol{w})$ is:

$$\nabla C(\boldsymbol{w}) = \frac{1}{2n}(2X^T X \boldsymbol{w} - 2X^T \boldsymbol{y})$$

- By the first-order optimality condition, we know that optimal $\boldsymbol{w}^*$ satisfies:

$$\nabla C(\boldsymbol{w}^*) = \frac{1}{n}(X^T X \boldsymbol{w}^* - X^T \boldsymbol{y}) = 0 \Rightarrow X^T X \boldsymbol{w}^* = X^T \boldsymbol{y}$$

(normal equation)

- If $X^T X$ is invertible, we get $\boldsymbol{w}^* = (X^T X)^{-1} X^T \boldsymbol{y}$
- Consequently, we can get the prediction:

$$\hat{\boldsymbol{y}} = X\boldsymbol{w}^* \quad \Longrightarrow \quad \hat{\boldsymbol{y}} = X(X^T X)^{-1} X^T \boldsymbol{y}$$

Linear regression: back to toy example

- In this example, we have $n = 5$, $d = 3$, so we get:

$$\boldsymbol{y} = \begin{pmatrix} 25 \\ 33 \\ 15 \\ 45 \\ 22 \end{pmatrix}, X = \begin{pmatrix} 1 & 2.7 & 1 \\ 1 & 4.1 & 1 \\ 1 & 1.0 & 0 \\ 1 & 5.2 & 1 \\ 1 & 2.8 & 0 \end{pmatrix}, \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

One	Dist (km)	Day	Commute time (min)
x_0	x_1	x_2	y
1	2.7	1	25
1	4.1	1	33
1	1.0	0	15
1	5.2	1	45
1	2.8	0	22

- By the closed-form solution, we get:

$$\boldsymbol{w}^* = \begin{pmatrix} 6.09 \\ 6.53 \\ 2.11 \end{pmatrix}, \hat{\boldsymbol{y}} = \begin{pmatrix} 25.84 \\ 34.99 \\ 12.62 \\ 42.17 \\ 24.38 \end{pmatrix} \Longrightarrow \hat{\boldsymbol{e}} = \begin{pmatrix} -0.84 \\ -1.99 \\ 2.38 \\ 2.82 \\ -2.38 \end{pmatrix}$$

Solve this toy problem in Python!

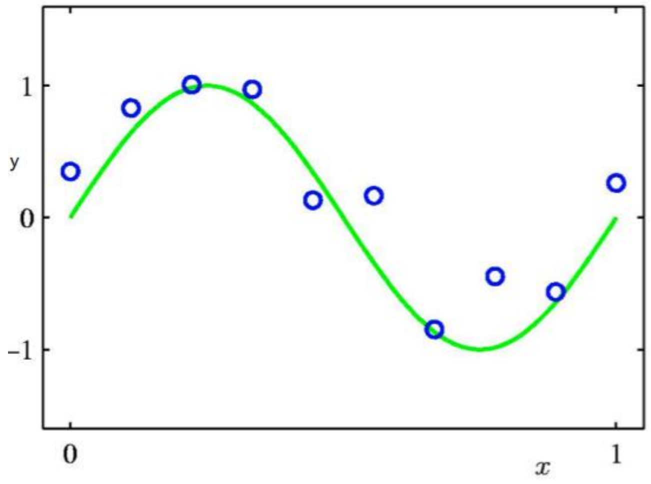
Summary: linear regression

- Linear regression (or least square regression)
  - model linear relationship between input and output (task T)
  - Example points (experience E)
  - mean square error as loss function (performance P)
  - closed-form solution (or exact solution)

Polynomial regression

Polynomial regression

- Suppose we want to model the following data



- The input-output relationship is nonlinear!
- How about we try to fit a polynomial?
  - This is known as polynomial regression

$$f(x) = w_0 + w_1 x + w_1 (x)^2 \ldots, w_M (x)^M$$

where $(x)^i$ denotes i^{th} power of x.

- Do we need to derive a whole new regression algorithm?

Polynomial regression: feature mappings

- Define the feature map:

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ (x)^2 \\ (x)^3 \end{pmatrix}$$

- Polynomial regression model now becomes a linear model w.r.t. the new features

$$f(x) = \boldsymbol{w}^T \phi(x) = w_0 + w_1 x + w_2 (x)^2 + w_3 (x)^3 = \phi(x)^T \boldsymbol{w}$$

We've transformed a univariate nonlinear problem to a multivariate linear problem!

- The derivations and algorithms so far in this lecture remain the same!

$$X = \begin{pmatrix} \boldsymbol{x}^{1^T} \\ \boldsymbol{x}^{2^T} \\ \vdots \\ \boldsymbol{x}^{n^T} \end{pmatrix} \longmapsto \begin{pmatrix} \phi(x^1)^T \\ \phi(x^2)^T \\ \vdots \\ \phi(x^n)^T \end{pmatrix} = \begin{pmatrix} 1 & x^1 & (x^1)^2 & (x^1)^3 \\ 1 & x^2 & (x^2)^2 & (x^2)^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x^n & (x^n)^2 & (x^n)^3 \end{pmatrix} = \bar{X}$$

Polynomial regression: feature mappings

- The linear regression model becomes:

$$\begin{pmatrix} f(x^1) \\ f(x^2) \\ \vdots \\ f(x^n) \end{pmatrix} = \begin{pmatrix} \phi(x^1)^T\boldsymbol{w} \\ \phi(x^2)^T\boldsymbol{w} \\ \vdots \\ \phi(x^n)^T\boldsymbol{w} \end{pmatrix} = \bar{X}\boldsymbol{w}$$

- The optimal weights can be found as:

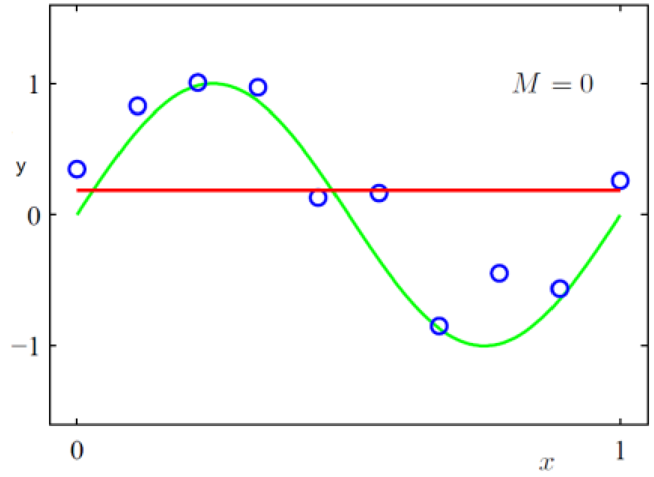
$$\boldsymbol{w}^* = (\bar{X}^T\bar{X})^{-1}\bar{X}^T\boldsymbol{y}$$

- Thus, we can estimate the model prediction:

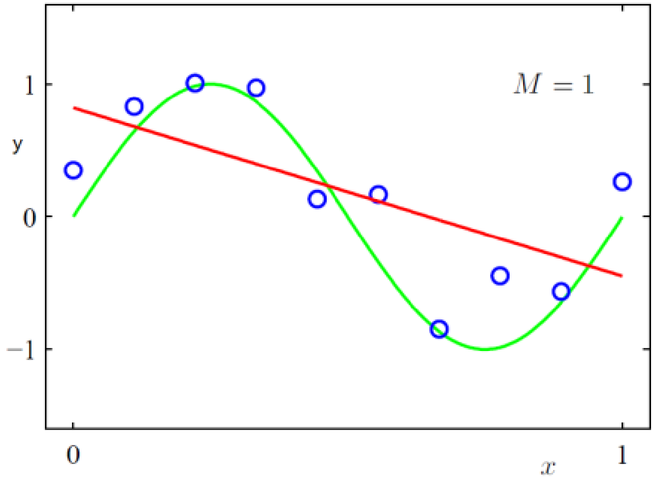
$$\widehat{\boldsymbol{y}} = \bar{X}\boldsymbol{w}^* \quad \Longrightarrow \quad \widehat{\boldsymbol{y}} = \bar{X}(\bar{X}^T\bar{X})^{-1}\bar{X}^T\boldsymbol{y}$$

Polynomial regression: fitting polynomials

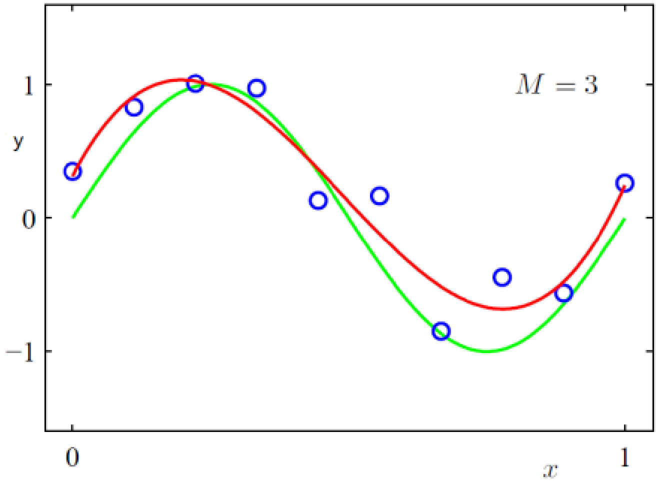
$y = w_0$



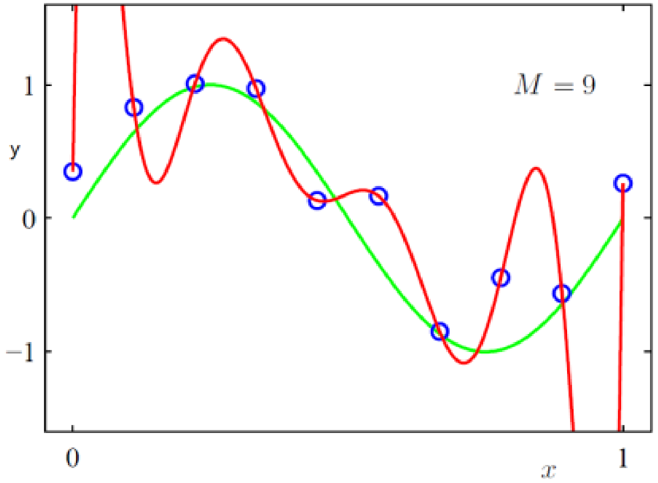
$y = w_0 + w_1 x$



$y = w_0 + w_1 x + w_2 (x)^2 + w_3 (x)^3$



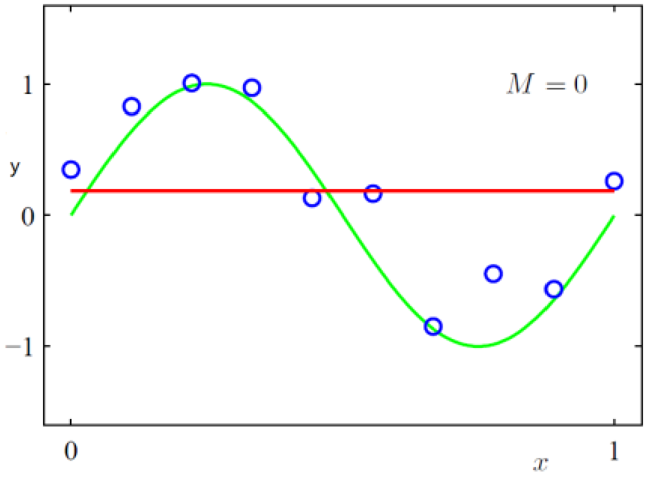
$y = w_0 + w_1 x + w_2 (x)^2 + \cdots + w_9 (x)^9$



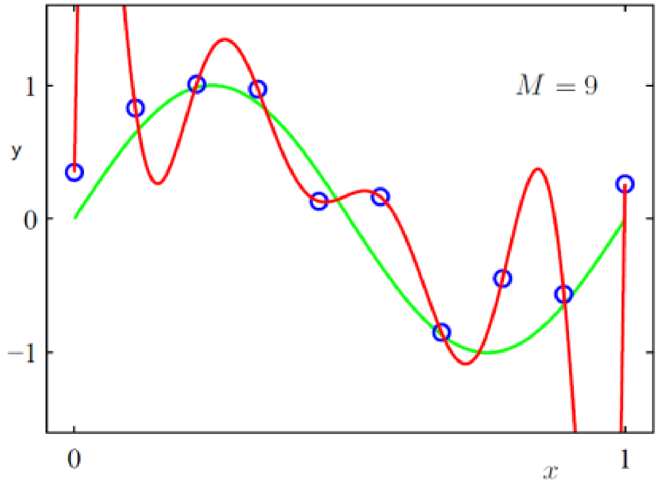
Polynomial regression

- Underfitting and Overfitting
- Underfitting
  - the model is too simple
  - does not fit the data
- Overfitting
  - the model is too complex
  - fits perfectly to training data
  - does not generalise to new data!

$y = w_0$



$y = w_0 + w_1 x + w_2 (x)^2 + \cdots + w_9 (x)^9$



Polynomial regression: generalisation

- Generalisation
  - model's ability to predict the unseen data
- Our model with $M = 9$ overfits the data
- One way to handle this is to encourage the weights to be small
  - This way, no feature will have too much influence on prediction
  - This is called regularisation!

Polynomial regression: regularisation

- Regularised least squares regression
  - Given dataset $D = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^n, y^n)\}$ and a regularisation parameter $\lambda > 0$, find a model to minimise:

$$C(\boldsymbol{w}) = \underbrace{\frac{1}{2n}(\boldsymbol{w}^T X^T X \boldsymbol{w} - 2\boldsymbol{w}^T X^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y})}_{\text{fitting to data}} + \underbrace{\frac{\lambda}{2}||\boldsymbol{w}||_2^2}_{\text{regulariser}}$$

where $||\boldsymbol{w}||_2^2$ is the 2-norm or Euclidean norm

- The regularised least square regression also has a closed-form solution
- One can show that $\nabla||\boldsymbol{w}||_2^2 = 2\boldsymbol{w}$ and therefore

$$\nabla C(\boldsymbol{w}) = \frac{1}{2n}(2X^T X \boldsymbol{w} - 2X^T \boldsymbol{y}) + \lambda \boldsymbol{w}$$

Polynomial regression: regularisation

- By the first-order optimality condition, we know that optimal $\boldsymbol{w}^*$ satisfies:

$$\nabla C(\boldsymbol{w}^*) = \frac{1}{n}(X^T X \boldsymbol{w}^* - X^T \boldsymbol{y}) + \lambda \boldsymbol{w}^* = 0 \Longrightarrow \left(\frac{1}{n}(X^T X) + \lambda \mathbb{I}\right)\boldsymbol{w}^* = \frac{1}{n} X^T \boldsymbol{y}$$

where $\mathbb{I} \in \mathbb{R}^{n \times n}$ is the identity matrix.

- It then follows that:

$$\boldsymbol{w}^* = \left(\frac{1}{n}(X^T X) + \lambda \mathbb{I}\right)^{-1} \left(\frac{1}{n} X^T \boldsymbol{y}\right)$$

$$\hat{\boldsymbol{y}} = X\boldsymbol{w}^* \quad \Longrightarrow \quad \hat{\boldsymbol{y}} = X\left(\frac{1}{n}(X^T X) + \lambda \mathbb{I}\right)^{-1} \left(\frac{1}{n} X^T \boldsymbol{y}\right)$$

- If $\lambda = 0$, then this becomes the solution of the least squares regression problem.
- If $\lambda = \infty$, we get $\boldsymbol{w}^* = 0$, which is a trivial solution. We need to choose an appropriate λ.

Summary: polynomial regression

- Polynomial regression
  - Polynomial fitting
  - Feature mapping
  - Underfitting
  - Overfitting
  - Regularisation

Maths refresher
(optional, self-study)

Maths refresher: matrices and matrix operations

- For a matrix $A \in \mathbb{R}^{m \times n}$, $A_{i,j}$ denotes the element in the i-th row and j-th column.

matrix transpose: $A^\top$ is defined by

$$A^\top_{i,j} = A_{j,i}, \quad A^\top \in \mathbb{R}^{n \times m}$$

Transposing a 2x3 matrix to create a 3x2 matrix

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^\top = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

matrix multiplication: If $B \in \mathbb{R}^{n \times r}$, then

$$(AB)_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}, \quad AB \in \mathbb{R}^{m \times r}$$

$$\overset{A}{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} \times \overset{B}{\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$1 \times 5 + 2 \times 7 = 19$$
$$1 \times 6 + 2 \times 8 = 22$$
$$3 \times 5 + 4 \times 7 = 43$$
$$3 \times 6 + 4 \times 8 = 50$$

- For two vectors $\mathbf{u} = (u_1, \ldots, u_m)^\top, \mathbf{v} = (v_1, \ldots, v_m)^\top \in \mathbb{R}^m$

vector addition

$$\mathbf{u} + \mathbf{v} = \begin{pmatrix} u_1 + v_1 \\ \vdots \\ u_m + v_m \end{pmatrix}$$

dot product

$$\mathbf{u}^\top \mathbf{v} = (u_1, \ldots, u_m) \begin{pmatrix} v_1 \\ \vdots \\ v_m \end{pmatrix} = \sum_{i=1}^{m} u_i v_i$$

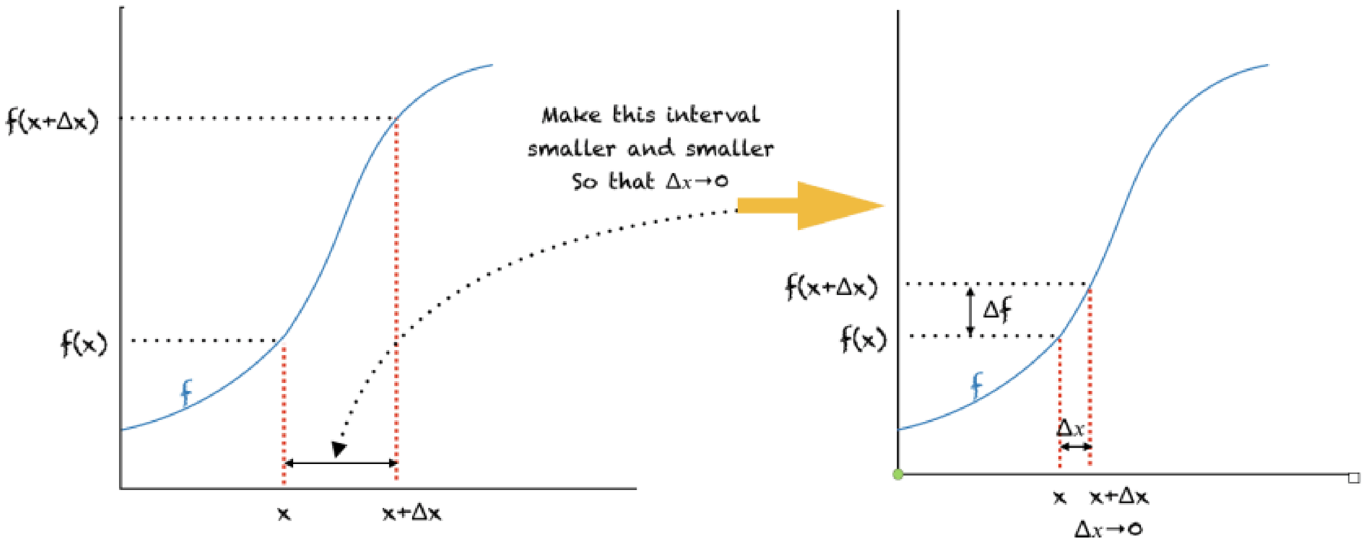
Hadamard product

$$\mathbf{u} \odot \mathbf{v} = \begin{pmatrix} u_1 v_1 \\ \vdots \\ u_m v_m \end{pmatrix}$$

Maths refresher: derivative

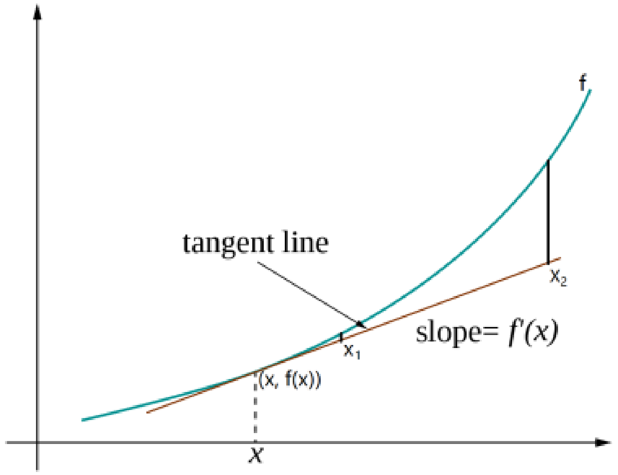
The derivative of a function $f : \mathbb{R} \mapsto \mathbb{R}$ is the rate of change of f

$$f'(x) = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}.$$



- The derivative of f at x is the slope of the tangent line to the graph of f at $(x, f(x))$
- The tangent line is the best linear approximation of the function near that input value

$$f(\bar{x}) \approx \underbrace{f(x) + f'(x)(\bar{x} - x)}_{\text{tangent line}}.$$



Maths refresher: partial derivatives

Partial derivative

The partial derivative of a **multivariate** function $f(x_1, \ldots, x_d)$ in the direction of variable x_i at $\mathbf{x} = (x_1, \ldots, x_d)$ is

$$\frac{\partial f(x_1, \ldots, x_d)}{\partial x_i} = \lim_{h \to 0} \frac{f(\ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots) - f(x_1, \ldots, x_i, \ldots, x_d)}{h}$$

- Intuitively, $\frac{\partial f}{\partial x_i}$ the derivative of a univariate function

$$g(x_i) := f(x_1, \ldots, x_i, \ldots, x_d),$$

  where all variables except x_i are fixed as constants.

- Example: $f(x_1, x_2) = 2x_1^2 + x_2^2 + 3x_1x_2 + 4$. Then

$$\frac{\partial f}{\partial x_1} = \frac{\partial(2x_1^2 + 3x_1x_2)}{\partial x_1} = 4x_1 + 3x_2$$
$$\frac{\partial f}{\partial x_2} = \frac{\partial(x_2^2 + 3x_1x_2)}{\partial x_2} = 2x_2 + 3x_1.$$

Maths refresher: gradient

Let $f : \mathbb{R}^d \mapsto \mathbb{R}$. The gradient of f with respect to $\mathbf{x} \in \mathbb{R}^d$ is defined as

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d}\right)^\top .$$

Example (Linear function). $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$, where $\mathbf{a} = (a_1, \ldots, a_d)^\top$. In this case, the gradient is

$$\nabla(\mathbf{a}^\top \mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_d} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x_1} a_1 x_1 + \frac{\partial}{\partial x_1} \sum_{j \neq 1} a_j x_j \\ \frac{\partial}{\partial x_2} a_2 x_2 + \frac{\partial}{\partial x_2} \sum_{j \neq 2} a_j x_j \\ \vdots \\ \frac{\partial}{\partial x_d} a_d x_d + \frac{\partial}{\partial x_d} \sum_{j \neq d} a_j x_j \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_d \end{pmatrix} = \mathbf{a}. \tag{1}$$

Exercise (Quadratic function). If $f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} = \sum_{i,j=1}^d a_{i,j} x_i x_j$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,d} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,d} \\ \vdots & \vdots & \ldots & \vdots \\ a_{d,1} & a_{d,2} & \cdots & a_{d,d} \end{pmatrix}, \quad \text{prove } \nabla(\mathbf{x}^\top A \mathbf{x}) = A\mathbf{x} + A^\top \mathbf{x}. \tag{2}$$

Gradient is a key concept in optimisation!

Maths refresher: unconstrained minimisation

Given an objective function $C : \mathbb{R}^d \mapsto \mathbb{R}$, we want to solve

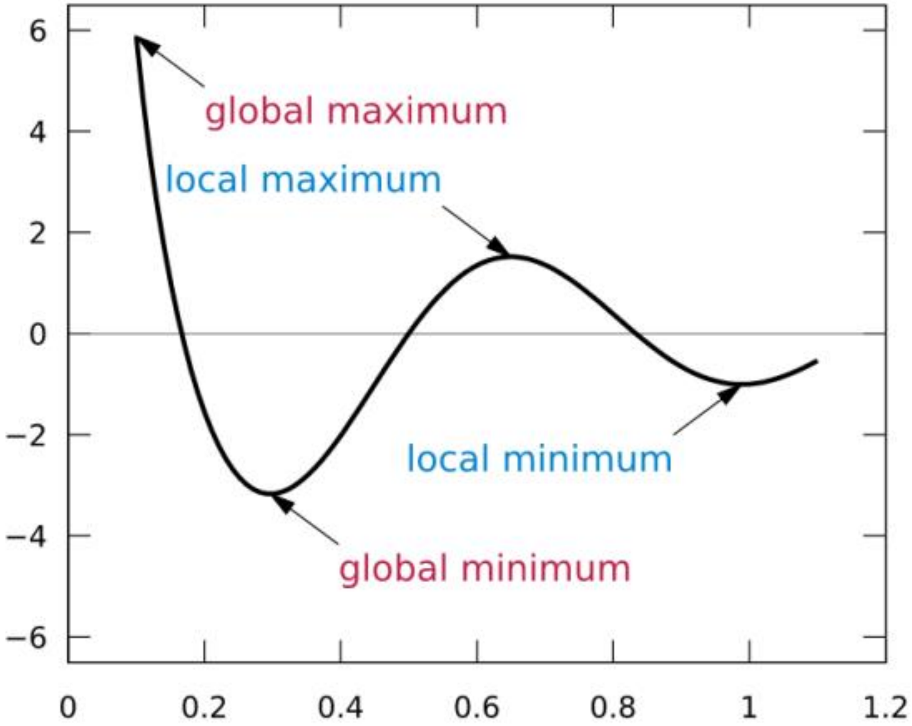
$$\min_{\mathbf{w} \in \mathbb{R}^d} C(\mathbf{w})$$

- $\mathbf{w}^*$ is a Global Minimum Point if

$$C(\mathbf{w}) \geq C(\mathbf{w}^*) \quad \forall \mathbf{w} \in \mathbb{R}^d$$

- $\mathbf{w}^*$ is a Local Minimum Point if there exists $\epsilon > 0$ such that

$$C(\mathbf{w}) \geq C(\mathbf{w}^*) \quad \text{for all } \mathbf{w} \text{ within distance } \epsilon \text{ of } \mathbf{w}^*.$$



Maths refresher: first-order optimality condition

First-order Necessary Optimality Condition

If $\mathbf{w}^*$ is a local minimum of a differentiable function C, then

$$\nabla C(\mathbf{w}^*) = 0. \tag{3}$$

We say $\mathbf{w}^*$ satisfying Eq. (3) a stationary point.

If $\nabla C(\mathbf{w}^*) \neq 0$, we can move along the direction $-\nabla C(\mathbf{w}^*)$ to get a smaller function value!

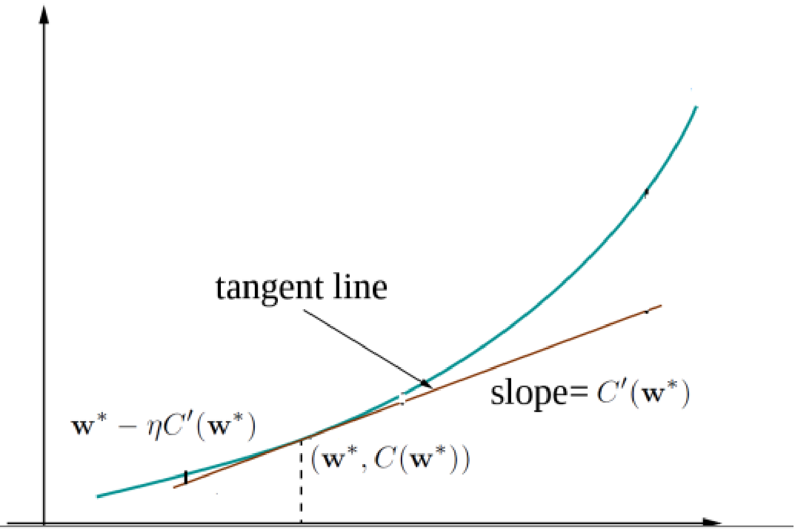
Understanding: (we assume $d = 1$ for brevity)

- If $C'(\mathbf{w}^*) \neq 0$, then $-\nabla C'(\mathbf{w}^*)$ is a descent direction
- We can move from $\mathbf{w}^*$ along the direction $\mathbf{v}^* := -C'(\mathbf{w}^*)$ with a step size η

$$C(\mathbf{w}^* + \eta\mathbf{v}^*) \approx \underbrace{C(\mathbf{w}^*) + C'(\mathbf{w}^*)(\eta\mathbf{v}^*)}_{\text{linear approximation}}$$

$$= C(\mathbf{w}^*) - \eta(C'(\mathbf{w}^*))^2 < C(\mathbf{w}^*)$$

for sufficiently small η, showing that $\mathbf{w}^*$ is not a local minimum!



Summary and further reading

Week 1: Summary

- Introduction
  - Module
  - Machine learning and its fundamentals
  - Neural computation

- Regression
  - Linear regression, exact solution
  - Linear regression, vectorised solution for multi-variate case
  - Polynomial regression, through multi-variate case

- Maths refresher (optional, self-study)

Further reading

- Maths refresher
  - Sections 2.1-2.3 of the Deep Learning book

- ML fundamentals and linear regression
  - Chapter 1 of the Deep Learning book
  - Section 4.5 of the Deep Learning book
  - Sections 5.1-5.3 of the Deep Learning book
  - Section 1.3 of the Pattern Recognition and Machine Learning book
- Polynomial and regularised linear regression
  - Section 1.1 of the Pattern Recognition and Machine Learning book
  - Section 3.1.4 of the Pattern Recognition and Machine Learning book

Practical

- Python refresher (optional, self-study)
  - See wk1_python_refresher.ipynb on Canvas
  - Introduce or refresh yourself with this material

- Week 1 lab on Linear Regression (exact solution)
  - See wk1_lab_linear_regression.ipynb on Canvas
  - Complete the missing bits in this code to complete the solution for linear regression

- Solve the problem presented on slide 65 in Python.

Exercise

- Solve exercise ex1 as released on Canvas
- Solution will be released at the start of Week 2

Credits

- Parts of the material are derived from Yunwen Lei's materials.