Deep Learning

Machine Learning Fundamentals

Tuesday 30th April

Dr. Nicholas Cummins

Emotional Car Reviews

Annotation

Lukas Stappen: lukas.stappen@informatik.uni-augsburg.de, stappen@ieee.org
Alice Barid: alice.baird@informatik.uni-augsburg.de

EmCaR: Emotional Car Reviews

Call-for-Students:
We want to study the interaction between objects and emotions
- And **we want you!**

Easy work! Watching videos and estimate speaker emotions using a Joystick.

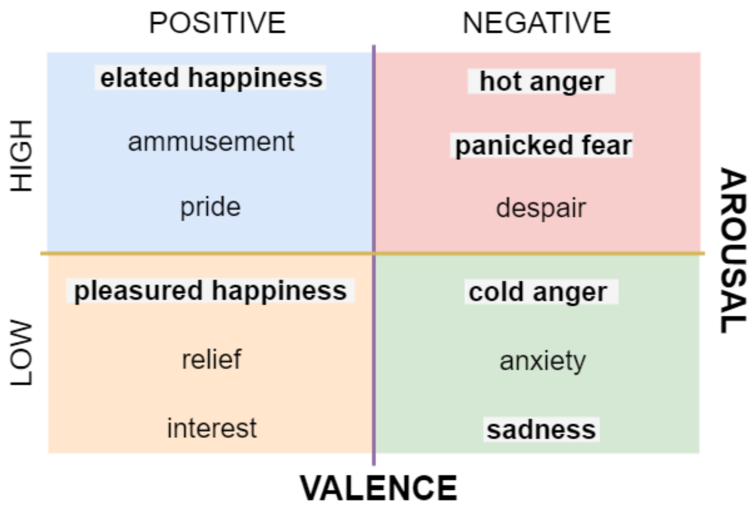
Flexible work!
1. Annotator training for ~2 h at university
2. You can watch and annotate the videos everywhere (at home, during breaks, …)

Payment!
~ 9.5 - 11.5 € per hour (netto, rates depending on degree)
~ 600 - 1.500 € in total (depends on total no of annotators)

Sign-up/ Contact
lukas.stappen@informatik.uni-augsburg.de, stappen@ieee.org
alice.baird@informatik.uni-augsburg.de
or quick form:
https://docs.google.com/forms/d/e/1FAIpQLScSTX-jiLIJyFjU3jogAHv42NAGkk6_DCQDghhKDWtA964_qA/viewform

Requirements
- English speaking (native, stays abroad preferred)
- a few free hours over the next 2-3 months

Further: Students familiar with the data set will be able to write their project, bachelor or master thesis on highly interesting **deep learning topics**.



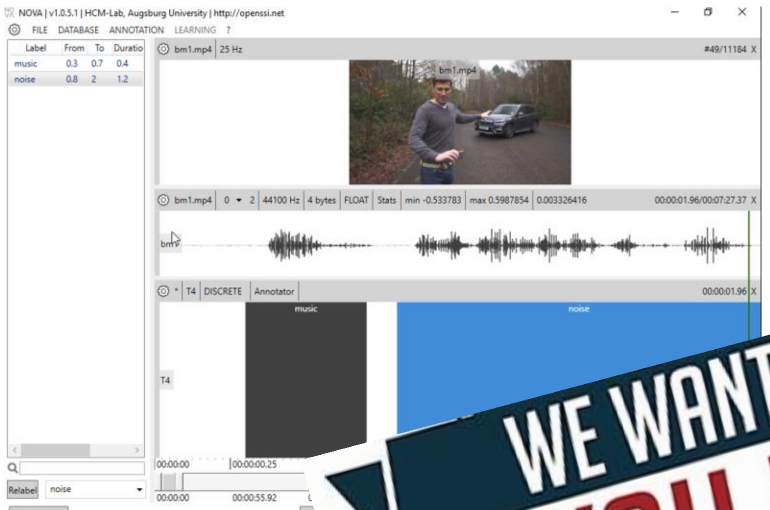


Image Source
https://pixabay.com/

- **Deep Learning**
  - A subfield of machine learning
  - Concerned with artificial neural networks
    - Algorithms inspired by the structure and function of the brain
  - Performs clustering, classification & predictive analysis
    - Clustering or grouping is the detection of similarities in data
    - Classification is the assignment of data instances into two or more discrete output values
    - Predictive analysis or regression is the assignment of data instances onto a continuous scale

Image Source
https://pixabay.com/

- **Empirical learning**
  - When you base a decision on existing data
  - Example:
    - It is Friday night, you have ordered a pizza and will be delivered in approximately 30 minutes, but it is often late
    - Your friend has messaged, can you pick him up?
      - The round trip is 35 minutes
    - Can you make it back in time for your pizza?

Image Source
https://pixabay.com/

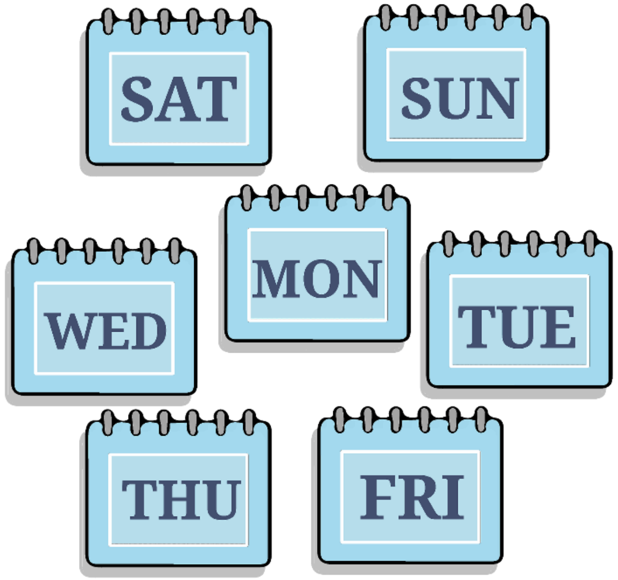
- **Empirical learning**

Can you make it back in time for your pizza?

  – Solution
    - Build a statistical model using previous delivery experiences
    - You have ordered a pizza at your current address 8 times
      – It was late four times (with a 40-minute delivery time)
      – There is a 50% chance the pizza will be late
    - However, we have not taken all variables in account

Universität Augsburg University

Image Source https://pixabay.com/



- **Empirical learning**
  - Need to consider all relevant information:
    - Dependent variable y
      - What we wish to predict, i.e., pizza delivery time
    - Independent variable X
      - Factors which affect the dependent variable
        - » In our example: days of the week
    - We want to model the relationship between independent variable(s) and the dependent variable
      - 3 out of 4 late deliveries have been on Mondays
      - Other information could include traffic conditions

- **Empirical learning**
  - Decision Tree Model

There is a 25% chance your pizza will be late

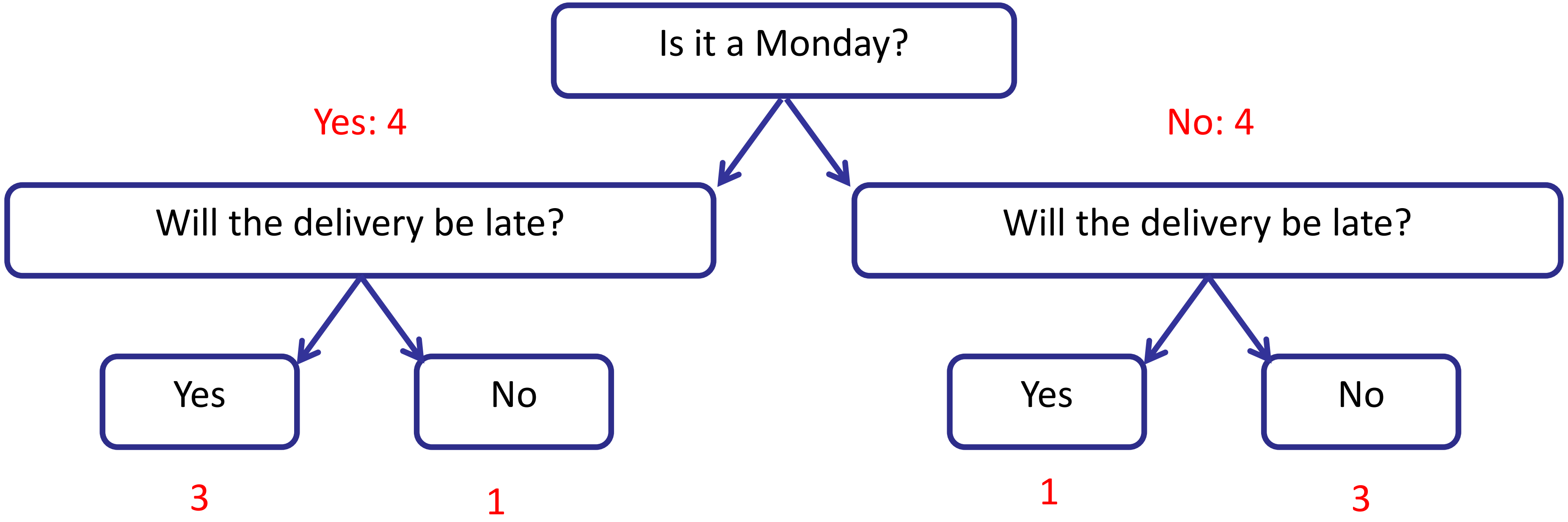


Image Source: iStock/dane_mar

- **Empirical learning**
  - How do we do this in computational analysis?
    - Collect ***labelled*** data
      - E.g. collect data from individuals with a particular health condition
        - » Depression, Bipolar, Parkinson’s, Austism, ….
    - ***Clean*** data to remove unwanted erroneous factors
      - E.g., speech samples recorded with a bad microphone
    - Extracted ***relevant information*** from the signal
      - Raw speech signals are highly complex
      - Feature extraction, information reduction
    - Choose a ***machine learning algorithm***, train and validate it
      - Identify suitable model settings and operating parameters

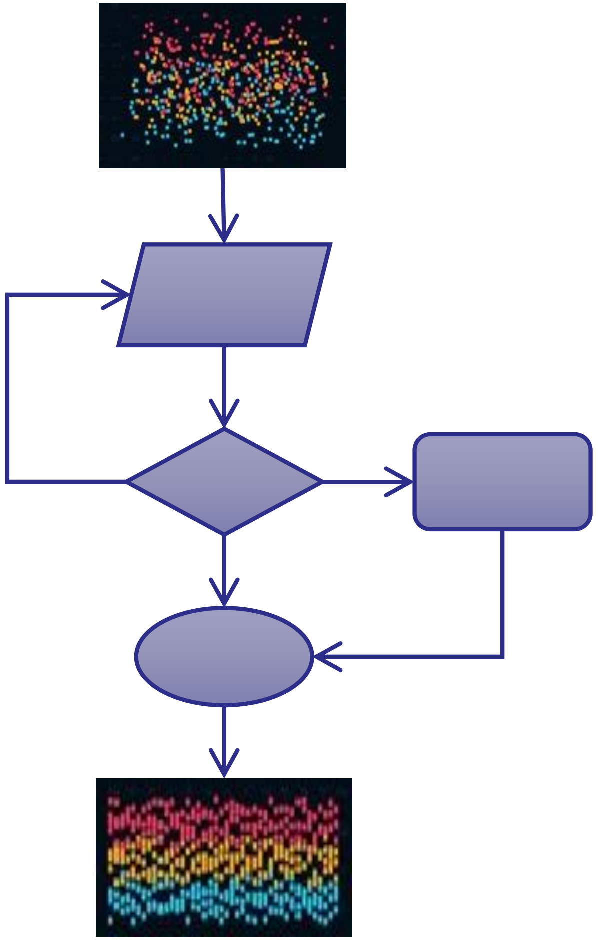
- **What are features?**
  - The representation of the data presented to the machine learning algorithm
  - Each feature can be thought of as a single piece of information the algorithm can use when making a decision
  - Typically hundreds or thousands of such pieces of information are concatenated together to form a feature vector
  - The role of the machine learning algorithm is to identify patterns from a collection of feature vectors

Universität Augsburg University

- **What is machine learning?**
  - Creation of (robust) models to cluster/predict/classify a particular output (y) from a selected independent variables (X – features) from a dataset
    - Primarily concerned with the identification of patterns within (large amounts of) data
    - Machine learning algorithms are used to perform the process of pattern identification via an iterative process
    - Learning phase: the algorithm optimises its parameters with the goal of improving (recognition) performance on a particular task

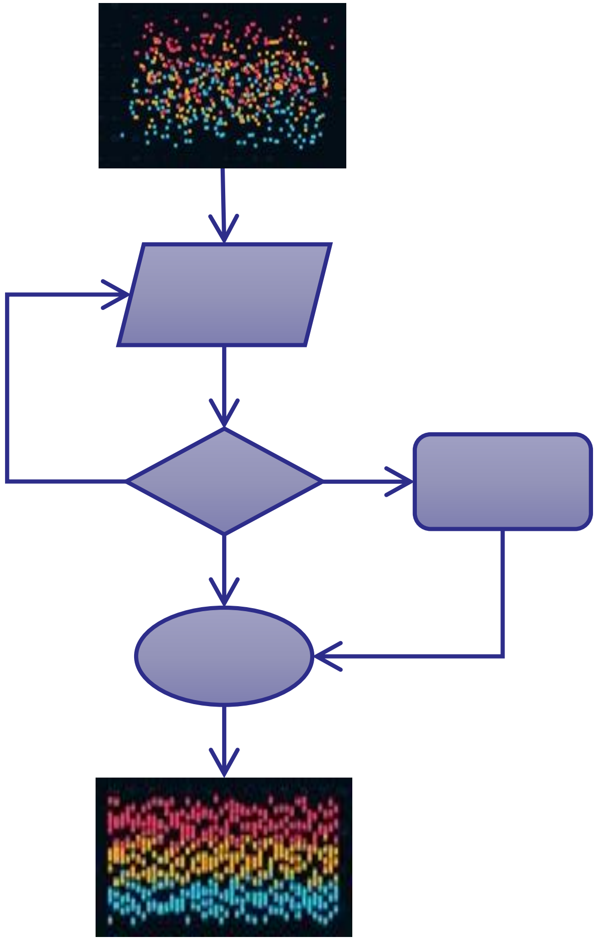
Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- Machine Learning Fundamentals
- Generalisation
- Gradient Descent
- Summary



Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- Machine Learning Fundamentals
- Generalisation
- Gradient Descent
- Summary



Why Linear Algebra?

- It is a key foundation to the field of machine learning
  - Present from notations used to describe the operation of algorithms to the implementation of algorithms in code
- Also needed to understanding the calculus and statistics used in machine learning
- Enables a deeper intuition in algorithms
  - Implement algorithms from scratch
  - Devise new algorithms

- **Scalar**
  - A one-dimensional vector, i.e. 1 x 1
- **Vector**
  - A single-dimensional array of numbers
  - i.e. 1 x m
- **Matrix**
  - A two-dimensional array of numbers
  - An $m \times n$ matrix has m rows and n columns
- **Tensor**
  - A multidimensional array of numbers

Vector:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

Matrix:

$$\begin{array}{cc} & n \text{ columns} \xrightarrow{j} \\ m \text{ rows} \; i \downarrow & \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{m1} & \dots & \dots & a_{mn} \end{bmatrix} \end{array}$$

- **Norm of a vector**
  - The norm is a measure of magnitude
    - The l^p norm is given by:

$$l^p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/P}$$

  - Machine learning generally uses the l^1 and l^2 norms
    - The least squares cost function is the l^2 norm of an error vector
    - Norm of a parameter vector can be used in regularization

- **Norm of a vector**
  - l^1 norm example

$$v = \begin{bmatrix} 1 \\ -4 \\ 5 \end{bmatrix}, \|v\|_1 = |1| + |-4| + |5| = 10$$

  - l^2 norm example

$$v = \begin{bmatrix} 1 \\ -4 \\ 5 \end{bmatrix}, \|v\|_2 = \sqrt{|1|^2 + |-4|^2 + |5|^2} = \sqrt{42}$$

- **Dot product**
  - The dot product of two vectors, $v_1 \in \mathbb{R}^{n\times 1}$ and $v_2 \in \mathbb{R}^{n\times 1}$, is the sum of the product of the corresponding elements:

$$v_1 \cdot v_2 = v_1^T v_2 = v_2^T v_1 = v_{1_1} v_{2_1} + v_{1_2} v_{2_2} + \cdots + v_{1_n} v_{2_n} = \sum_{k=1}^{n} v_{1_k} v_{2_k}$$

- **Dot product**
  - Example:

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, v_2 = \begin{bmatrix} 3 \\ 5 \\ -1 \end{bmatrix}$$

$$v_1 \cdot v_2 = \; v_1^T v_2 == 1 \times 3 + 2 \times 5 - 3 \times 1 = 10$$

- **Why is the dot product important?**
  - The dot product encodes information about the angle between two vectors

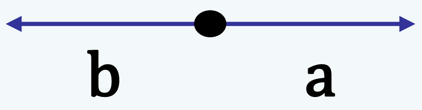
$$\boldsymbol{v}_1 \cdot \boldsymbol{v}_1 = \|\boldsymbol{v}_1\| \|\boldsymbol{v}_1\| \cos\theta$$

$$\theta = arccos\left(\frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_1}{\|\boldsymbol{v}_1\| \|\boldsymbol{v}_1\|}\right)$$

Linear Algebra (Refresher)

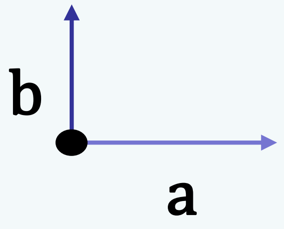
- **Why is the dot product important?**

$$\cos \pi = -1$$



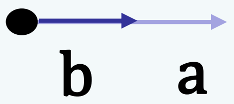
Dot product is negative

$$\cos \frac{\pi}{2} = 0$$



Dot product is zero

$$\cos 0 = 1$$



Dot product is positive

  – The dot product measures how similar two vectors are

- **Matrix multiplication**
  - For $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$ to be multipliable n must equal p and the resulting matrix is $\mathrm{C} \in \mathbb{R}^{m \times q}$

$$C_{i,j} = \sum_{k=1}^{n} a_{i,k}\, b_{k,j}$$

$$\forall\, i \in \{1,2, \dots, m\}$$

$$\forall\, j\ \{1,2, \dots, q\}$$

Universität Augsburg University

- **Matrix multiplication**
  - Example:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}, C = A \times B$$

$$C_{11} = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix}, 1 \times 5 + 2 \times 7 = 19\,, C_{12} = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 6 \\ 8 \end{bmatrix}, 1 \times 6 + 2 \times 8 = 22$$

$$C_{21} = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix}, 3 \times 5 + 4 \times 7 = 43\,, C_{22} = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 6 \\ 8 \end{bmatrix}, 3 \times 6 + 4 \times 8 = 50$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

- **Matrix transpose**
  - The transpose of a matrix $A \in \mathbb{R}^{m \times n}$ is generally represented as $A^T \in \mathbb{R}^{n \times m}$
  - This is performed by transposing the column vectors as row vectors

$$a'_{ji} = a_{i,j}, \forall\, i \in \{1,2, \dots, m\}, \forall\, j\, \{1,2, \dots, n\}$$

where $a'_{ji} \in A^T$ and $a_{i,j} \in A$

Universität Augsburg University

- **Matrix transpose**
  - Examples:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ then } A^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 4 & 3 \\ 8 & 2 & 6 \\ 7 & 8 & 3 \\ 4 & 9 & 6 \\ 7 & 8 & 1 \end{bmatrix} \text{ then } A^T = \begin{bmatrix} 1 & 8 & 7 & 4 & 7 \\ 4 & 2 & 8 & 9 & 8 \\ 3 & 6 & 3 & 6 & 1 \end{bmatrix}$$

Universität Augsburg University

- **Linear independence**
  - A vector is said to be linearly dependent on other vectors if it can be expressed as the linear combination of other vectors.
    - Given a set of vectors $\boldsymbol{v}_i \in \mathbb{R}^{n\times 1}$ and a set of scalars a_i

$$[\boldsymbol{v}_1, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n]\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = 0$$

    - Then the set of vectors $\boldsymbol{v}_i$ is said to be linear independent

- **Rank of a matrix**
  - The rank of a matrix is the number of linearly independent column vectors or row vectors
  - Example:

$$A = \begin{bmatrix} 1 & 3 & 4 \\ 2 & 5 & 7 \\ 3 & 7 & 10 \end{bmatrix}$$

$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $\begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}$ are linearly independent

$$\begin{bmatrix} 4 \\ 7 \\ 10 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}$$

Therefore:

$$rank(A) = 2$$

- **Rank of a matrix**
  - *Why rank is important in machine learning?*
    - When the rank of a square matrix $A \in \mathbb{R}^{n \times n}$ is n, it is said to be full rank
    - A singular matrix has an undefined matrix inverse and zero determinant.
    - Advanced techniques such as *Singular-Value Decomposition* have to be used to determine a pseudo-inverse of a singular matrix

Universität Augsburg University

- **Determinant of a matrix**
  - The determinant of a square matrix $A \in \mathbb{R}^{n \times n}$ is a number denoted by $|A|$ or $\det(A)$ and is given by:

$$\boldsymbol{det}(\boldsymbol{A}) = \pm \prod a_{1j_i}\, a_{2j_2}, \dots, a_{nj_n}$$

  - where the column indices $j_1, j_2,.., j_n$ are taken from the set $\{1, 2, \dots\ n\}$, with no repetitions allowed. The plus (minus) sign is taken if the permutation $(j_1, j_2,.., j_n)$ is even (odd)

- **Determinant of a matrix**
  - Examples:

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$= a_{11}(a_{22}a_{33} - a_{32}a_{23}) - a_{21}(a_{12}a_{33} - a_{32}a_{13}) + a_{31}(a_{12}a_{23} - a_{22}a_{13})$$

- **Inverse of a matrix**
  - For a square matrix $A \in \mathbb{R}^{n \times n}$, the inverse is denoted as A^{-1} and produces the identity when multiplied by A

$$AA^{-1} = A^{-1}A$$

$$A^{-1} = \frac{(cofactor\ matrix\ of\ A)^T}{\det(A)}$$

  - Cofactor for $a_{\mathrm{i,j}} = (-1)^{i_{-}+j} d_{ij}$
    - Where d_{ij} is the determinate of the matrix formed by deleting row i and column j from A

Universität Augsburg University

- **Inverse of a matrix**
  - Examples

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{a_{11} \times a_{22} - a_{21} \times a_{12}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

$$\begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}^{-1} = \frac{1}{4 \times 6 - 2 \times 7} \begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \frac{1}{10} \begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$$

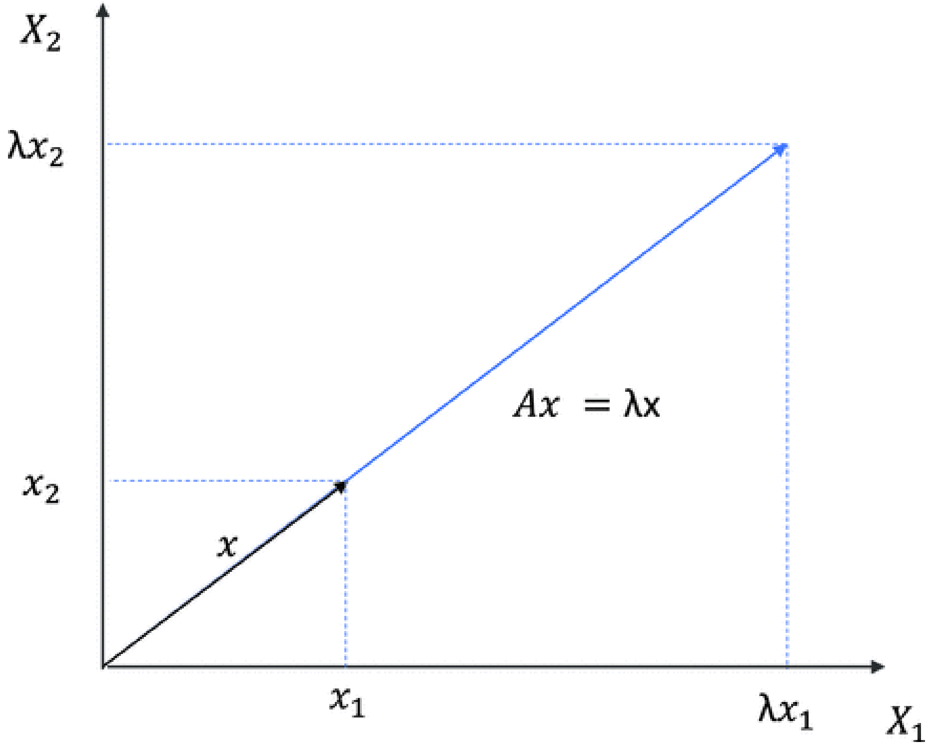
$$\begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}^{-1} = \frac{1}{3 \times 8 - 6 \times 4} \begin{bmatrix} 8 & -4 \\ -6 & 3 \end{bmatrix}$$ → inverse does not exist

- **Eigenvectors and eigenvalues**
  - When a matrix $A \in \mathbb{R}^{n \times n}$ works on a vector $x \in \mathbb{R}^{n \times 1}$ the resulting vector is $Ax \in \mathbb{R}^{n \times 1}$
    - Generally the magnitude and direction of Ax differs from x
  - *However,*
    - When Ax has the same (or directly opposite) direction of x the resulting vectors is known as an *eigenvector*
    - The magnitude by which that vector gets stretched is known as the *eigenvalue*

- **Eigenvectors and eigenvalues**

$$Av = \lambda v$$

- $Av = \lambda v$
- $(A - \lambda I) = 0$
- $A - \lambda I$ is singular
- $det(A - \lambda I) = 0$



Universität Augsburg University

- **Eigenvectors and eigenvalues**
  - Example

Find eigenvalues of: $A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$

$$|\lambda I - A| = \begin{bmatrix} \lambda - 2 & 12 \\ -1 & \lambda + 5 \end{bmatrix} = (\lambda - 2)(\lambda + 5) + 12$$

$$= \lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2)$$

Therefore, eigenvalues of A are $-1, -2$

- **Positive semi-definite and positive definite**
  - A square matrix $A \in \mathbb{R}^{n \times n}$ is *positive semi-definite*
    - If for any non-zero vector the $x \in \mathbb{R}^{n \times 1} \to x^T A x \geq 0$
      - All eigenvalues should be non-negative
  - A square matrix $A \in \mathbb{R}^{n \times n}$ is *positive definite*
    - If for any non-zero vector the $x \in \mathbb{R}^{n \times 1} \to x^T A x > 0$
      - All eigenvalues should be positive

Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- Machine Learning Fundamentals
- Generalisation
- Gradient Descent
- Summary

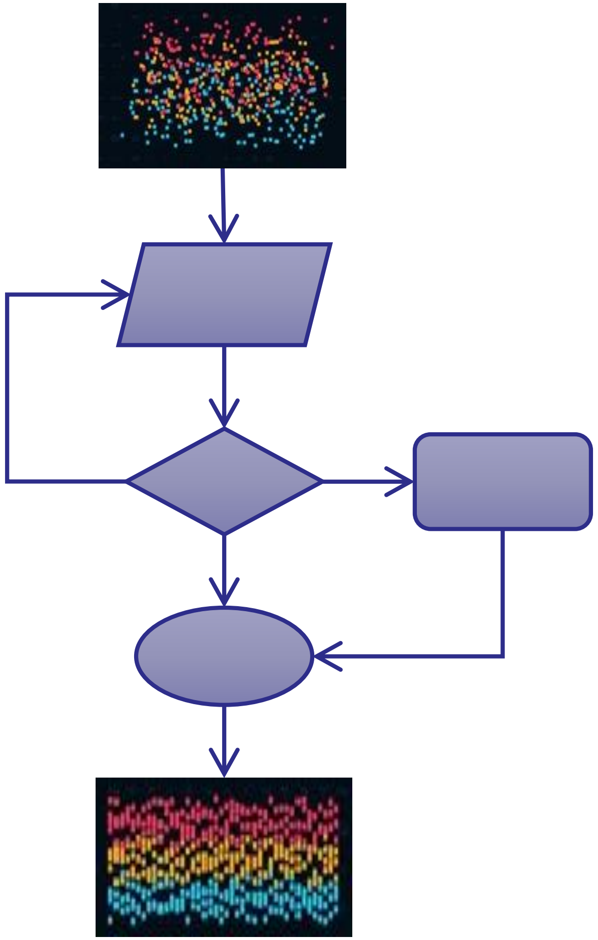


Image Source
https://pixabay.com/

- **Why probability?**
  - Machine learning must always deal with uncertain quantities
  - Laws of probability govern how a machine learning algorithm should reason
    - We design machine learning algorithms to approximate expressions derived from probability theory
  - Analyse the behaviour of a proposed approach

- **Definitions**
  - **Experiment:** any process of observation
  - **Random experiment:** An experiment in which the outcomes cannot be precisely predicted
  - **Sample space:** set of all possible outcomes
  - **Probability measure P:** an assignment of a number between 0 and 1 to a particular event in the sample space

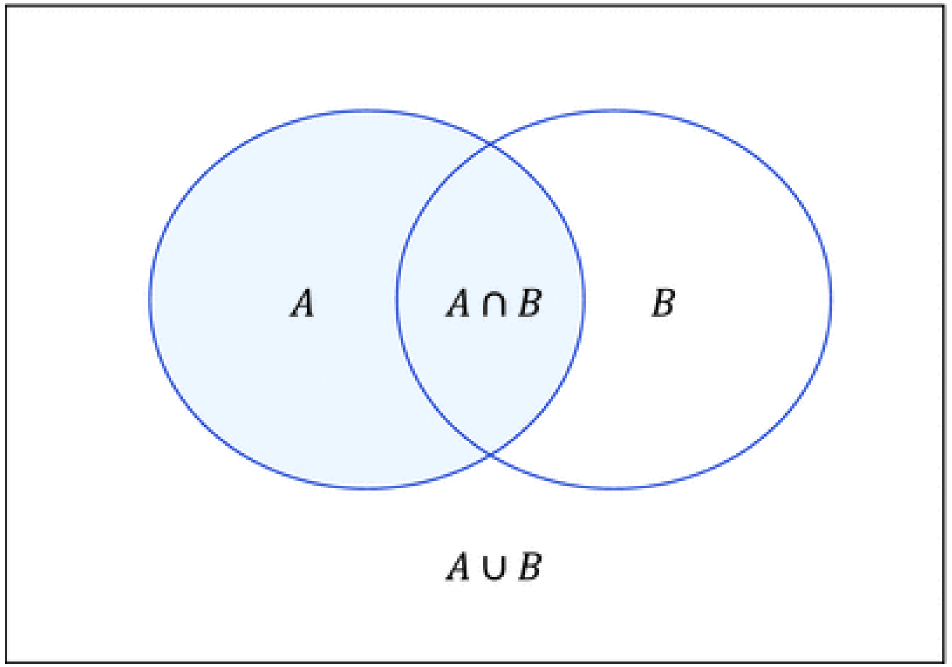
$P(A)$: the probability that an event A will occur

$$0 \leq P(A) \leq 1$$

- **Rules of Probability**
  - **Intersection of events**
    - The probability that Events A *and* B occur, denoted $P(A \cap B)$
  - **Mutually exclusive events**
    - Cannot occur at the same time i.e. $P(A \cap B) = 0$
  - **Union of events**
    - The probability that events A *or* B occur, denoted $P(A \cup B)$
  - **Conditional probability**
    - The probability that Event A occurs, given that Event B has occurred
    - Denoted $P(A|B)$

- **Rules of probability**
  - **Rule of multiplication**
    - $P(A \cap B) \; = \; P(A)\,P(B|A)$

      $\qquad\qquad\; = \; P(B)P(A|B)$
    - Note, $P(A \cap B)$ is denoted as $P(AB)$
  - **Rule of addition**
    - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$



- **Rules of probability**
  - **Chain rule of probability**
    - Extension of the rule of multiplication

$$P(A_1A_2A_3 \dots A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1A_2)P(A_n|A_1A_2A_3 \dots A_{n-1})$$
$$= P(A_1)\prod_{i-2}^{n} P(A_i|A_1A_2A_3 \dots A_{n-1})$$

  - **Mutually exclusive events $\mathbf{P(AB) = 0}$**

$$P(A_1 \cup A_2 \cup A_3 \cup \cdots \cup A_n) = P(A_1) + P(A_2) + P(A_3) + \cdots + P(A_n)$$
$$= \sum_{i=1}^{n} P(A_n)$$

- **Rules of probability**
  - **Independence of events**

$$P(AB) = P(A)P(B)$$

  - **Bayes' rule**
    - From multiplication rule

$$P(AB) = P(A)P(B|A) = P(B)P(A|B)$$

    - Therefore

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Probability (Refresher)

- **Random experiment**
  - An experiment or a process for which the outcome cannot be predicted with absolute certainty
    - However, we have knowledge of the *sample space*, the set of all possible outcomes
- **Random variable**
  - Individual outcomes of a random experiment
    - A function that maps the outcomes of random experiment (the samples space) to a subset of real numers (i.e. $\mathbb{R}$).
      - E.g. A random variable can be used to describe the process of rolling a fair die and the possible outcomes $\{1, 2, 3, 4, 5, 6\}$

- **Probability Mass Function (PMF)**
  - Let X be a random variable with domain D
  - The probability mass function is then defined as the probability that X is equal to some value x

$$\sum_{x \in D} P(X = x) = 1$$

  - To be a PMF, P must satisfy
    - P must be the sets of all possible states of X
    - $0 \leq P(X) \leq 1$
    - $\sum_{x \in D} P(X) = 1$

Probability (Refresher)

- **Expectation**
  - The average value that some function takes when x is drawn from P

$$\mathbb{E}_{x \sim P}[f(x)] = \mu = \sum_x P(x) f(x)$$

- **Variance**
  - Variation in different sample values of x when drawn from its probability distribution

$$Var[f(x)] = \sigma^2 = \mathbb{E}[f(x) - \mathbb{E}[f(x)]^2]$$

- **Covariance**
  - Measure joint variability between two random variables

$$Cov(f(x), g(y)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])(g(x) - \mathbb{E}[g(x)])]$$

Random processes

- A collection of random variables defined over a common PMF
  - Consider a random process η with N observed values

  - **Mean**

$$\mu_{\boldsymbol{\eta}} = \frac{1}{N}\sum_{n=0}^{N-1} \eta(n)$$

  - **Mean-Square**
    - The ‘power’ of the process

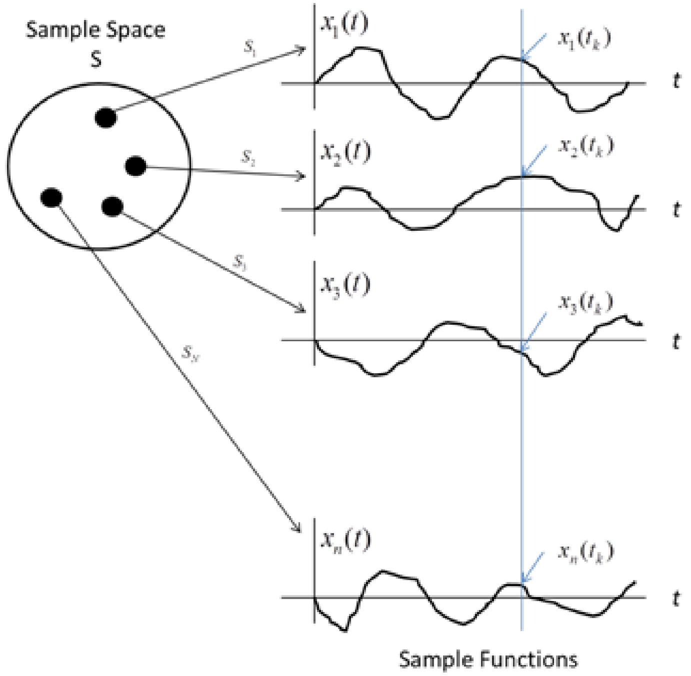
$$MS_{\eta} = \frac{1}{N}\sum_{n=0}^{N-1} (\eta(n))^2$$

  - **Variance**

$$\sigma_{\eta}^2 = \frac{1}{N}\sum_{n=0}^{N-1} \left(\eta(n) - \mu_{\eta}\right)^2$$

Image Source:
https://www.vocal.com

- **Random signal**
  - When the values of a random process η form a time series
    - Also known as a stochastic process
  - Denoted $\eta(t)$
- Key properties
  - μ_η represents the DC component
    - DC component is the amplitude signal fluctuates around
      - Assumed to be zero for random noise
  - MS_η represents the average power
    - If μ_η is zero $\sigma_\eta^2 = MS_\eta$

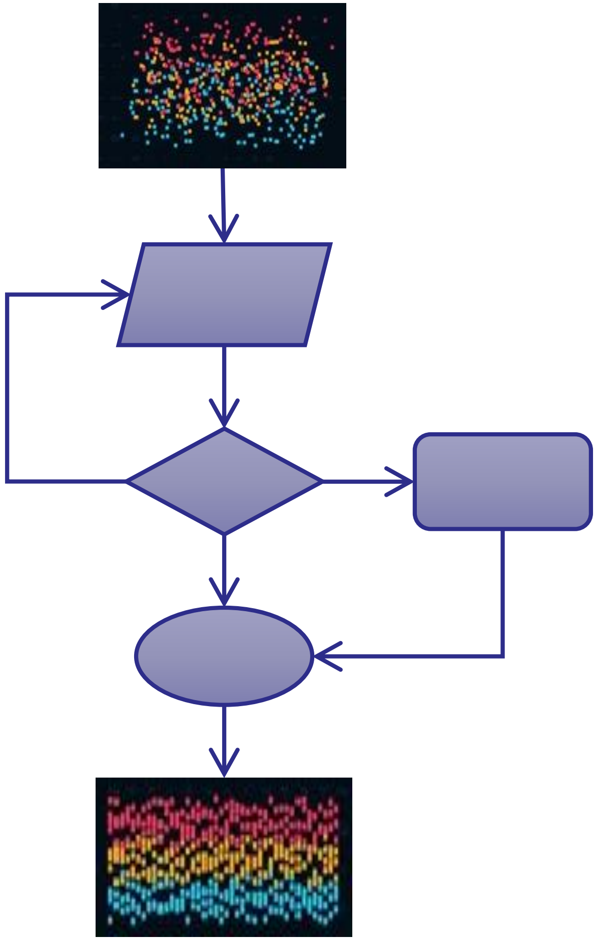


Universität Augsburg University

- **Information Theory**
  - Quantifying how much information is present in a signal
    - Likely events should have low information content
      - Likely events are uninformative
    - Less likely events should have higher information content
      - Unlikely events are more informative
  - Self-Information: $I(x) = -logP(x)$
  - Entropy: $H(x) = E_{x \sim P}[I(x)] = -E_{x \sim P}[logP(x)]$
    - Distribution of expected information

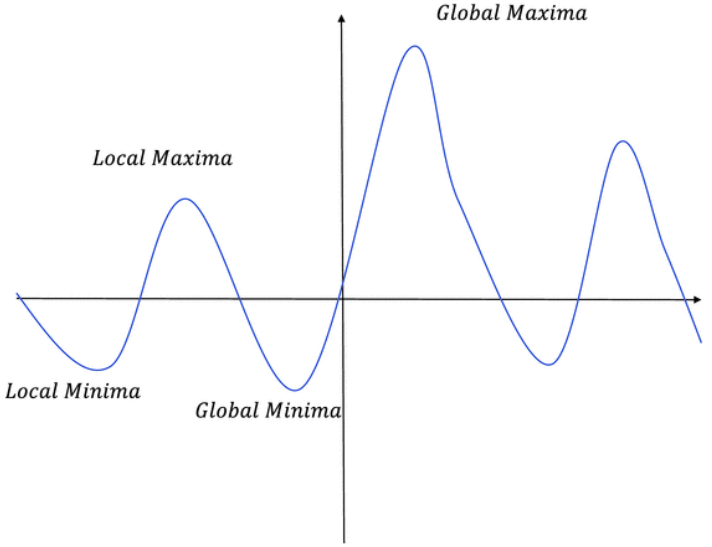
Lecture Outline

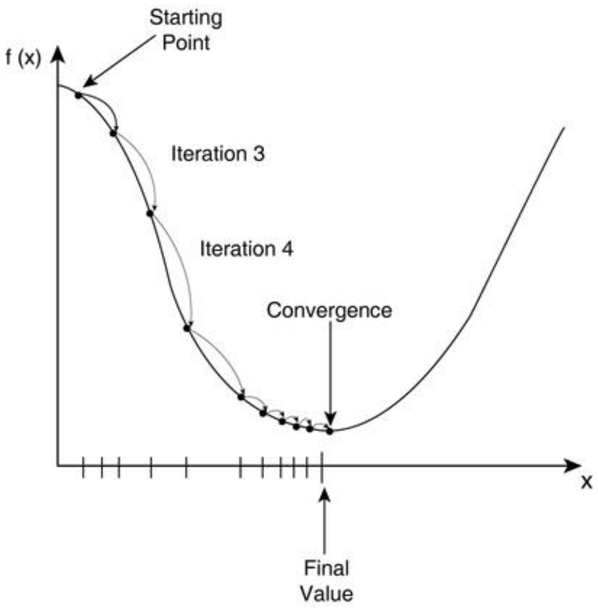
- Linear Algebra
- Probability
- **Differential Calculus**
- Machine Learning Fundamentals
- Generalisation
- Gradient Descent
- Summary



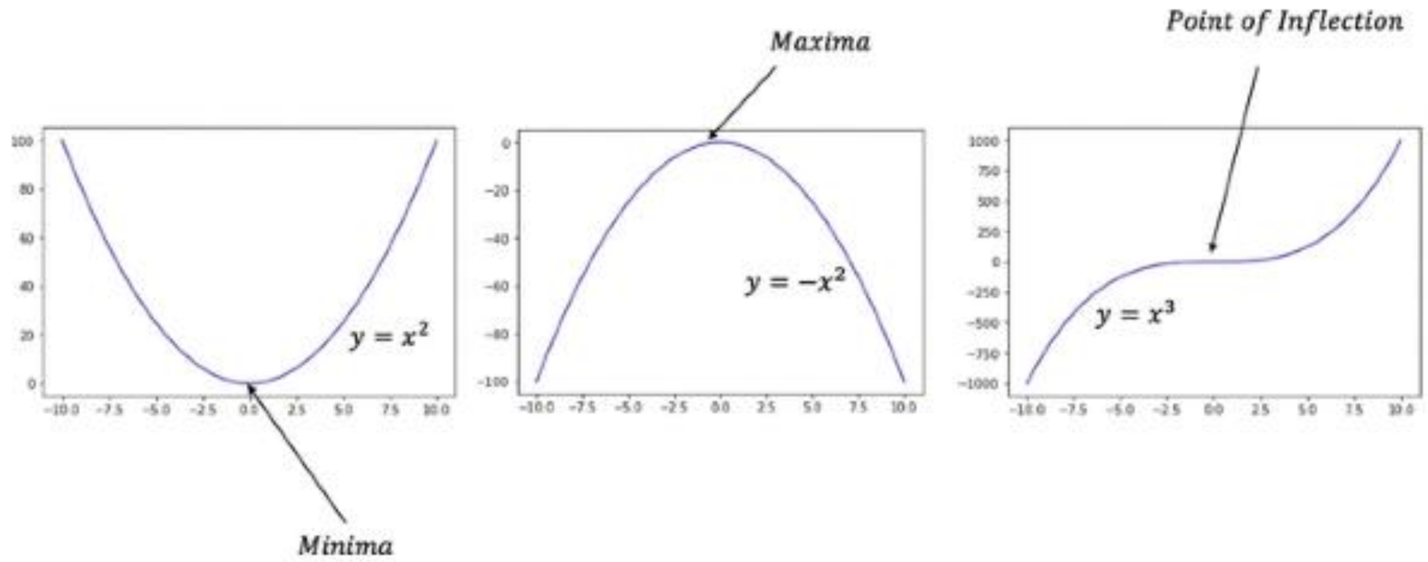
- **A good understanding of calculus is essential for machine learning**
  - Machine learning models are (normally) a function of several variables
  - In building a model we generally need to compute a cost function, we derive the models that best explain the training data by optimising this cost function
  - Optimisation refers to the task of minimising (or maximising) a function $f(\boldsymbol{x})$

- **Maxima and Minima of Functions**
  - Building machine-learning models relies on iteratively minimising a cost function





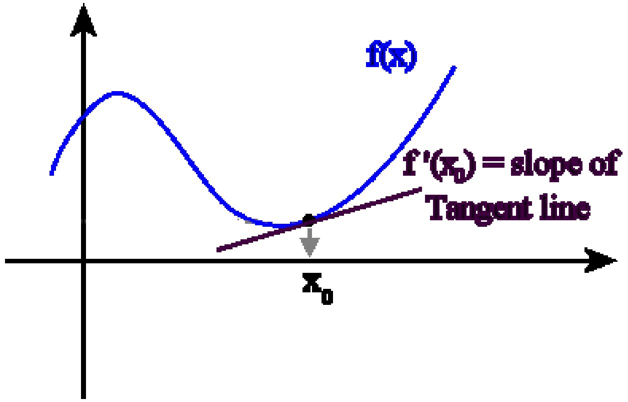
- **Rules for locating maxima/minima**
  - 1st order derivative is zero
  - Maxima: 2nd order derivative is *less* than zero
  - Minima: 2nd order derivative is *greater* than zero
  - Point of inflection: 2nd order derivative equals zero



Universität Augsburg University

- **Derivative of a function $f(t)$**
  - Rate of change of a quantity represented by a function with respect to another quantity on which the function is dependent on.

$$\frac{df}{dt} = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}$$



- **Derivative of a function $f(t)$**

$f(x)$	$f'(x)$	$f(x)$	$f'(x)$
x^n	nx^{n-1}	e^x	e^x
$\ln(x)$	$1/x$	$\sin(x)$	$\cos(x)$
$\cos(x)$	$-\sin(x)$	$\tan(x)$	$\sec^2(x)$
$\cot(x)$	$-\mathrm{cosec}^2(x)$	$\sec(x)$	$\sec(x)\tan(x)$
$\mathrm{cosec}(x)$	$-\mathrm{cosec}(x)\cot(x)$	$\tan^{-1}(x)$	$1/(1+x^2)$
$\sin^{-1}(x)$	$1/\sqrt{1-x^2}$ for $\lvert x\rvert < 1$	$\cos^{-1}(x)$	$-1/\sqrt{1-x^2}$ for $\lvert x\rvert < 1$
$\sinh(x)$	$\cosh(x)$	$\cosh(x)$	$\sinh(x)$
$\tanh(x)$	$\mathrm{sech}^2(x)$	$\coth(x)$	$-\mathrm{cosech}^2(x)$
$\mathrm{sech}(x)$	$-\mathrm{sech}(x)\tanh(x)$	$\mathrm{cosech}(x)$	$-\mathrm{cosech}(x)\coth(x)$
$\sinh^{-1}(x)$	$1/\sqrt{x^2+1}$	$\cosh^{-1}(x)$	$1/\sqrt{x^2-1}$ for $x > 1$
$\tanh^{-1}(x)$	$1/(1-x^2)$ for $\lvert x\rvert < 1$	$\coth^{-1}(x)$	$-1/(x^2-1)$ for $\lvert x\rvert > 1$

- **Product Rule**
  - If $f(x)$ and $g(x)$ are differentiable on x then:

$$(f \cdot g)'(x) = f(x)g'(x) + g(x)f'(x)$$

- **Chain Rule**
  - If $f(x)$ and $g(x)$ are differentiable on x

$$(f \circ g)'(x) = f'\big(g(x)\big)g'(x)$$

  - If $y = g(u)$ and $u = g(x)$ the derivative of y is

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

- **Partial derivatives $z = f(x, y)$**

$$\frac{\partial z}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

$$\frac{\partial z}{\partial y} = \lim_{h \to 0} \frac{f(y+h, x) - f(x, y)}{h}$$

  – Successive Partial Derivatives

$$\frac{\partial}{\partial y}\left(\frac{\partial z}{\partial x}\right) = \frac{\partial^2 z}{\partial y \partial x}$$

$$\frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right) = \frac{\partial^2 z}{\partial x \partial y}$$

  - Note that if the second derivatives are continuous, $\frac{\partial^2 z}{\partial y \partial x} = \frac{\partial^2 z}{\partial x \partial y}$

- **Gradient of a function**
  - Vector of first order partial derivatives

$$f(\boldsymbol{x}), \text{where } \boldsymbol{x} = [x_1, x_2, \dots, x_n]^T \epsilon \mathbb{R}^{n\times 1}$$

Then,

$$\nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right]^T$$

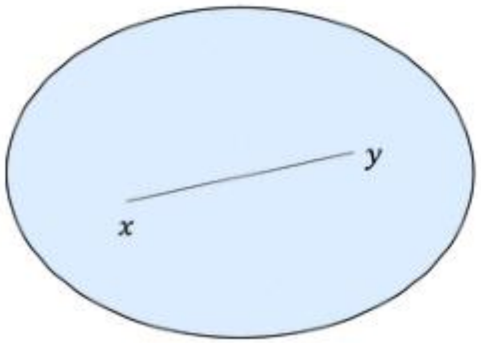
  - The gradient is important in machine-learning algorithms when we try to maximize or minimize cost functions with respect to the model parameters,

- **Hessian Matrix of a function**
  - Matrix of second order partial derivatives
    - Useful in optimisation problems
      - Especially when cost function is non linear

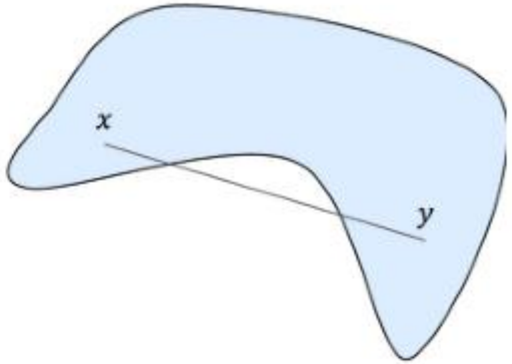
For a function: $f(x, y, z)$:

$$Hf(x,y,z) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix}$$

- **Convex Function**
  - Convex Set D
    - Given any two points x and y belonging to set D all points joining the straight line from x to y must also belong to set D



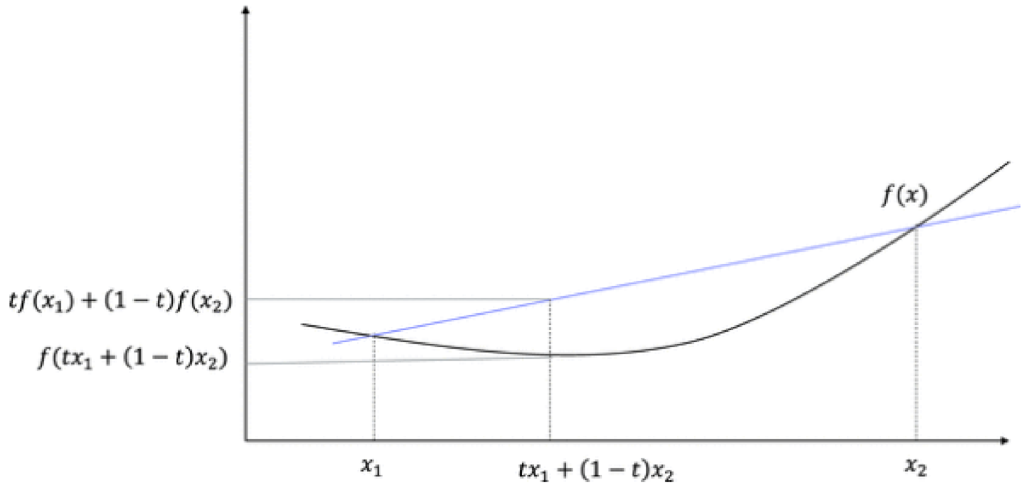
Convex Set



Concave Set

- **Convex Function**
  - A function $f(x)$ defined on a convex set D:
    - A straight line joining any two points in the function lies above or on the graph of the function

$$f(tx-(1-t)y) \leq tf(tx)+(1-t)f(y) \qquad \forall x, y \in D, \;\; \forall\, t \in [0,1]$$

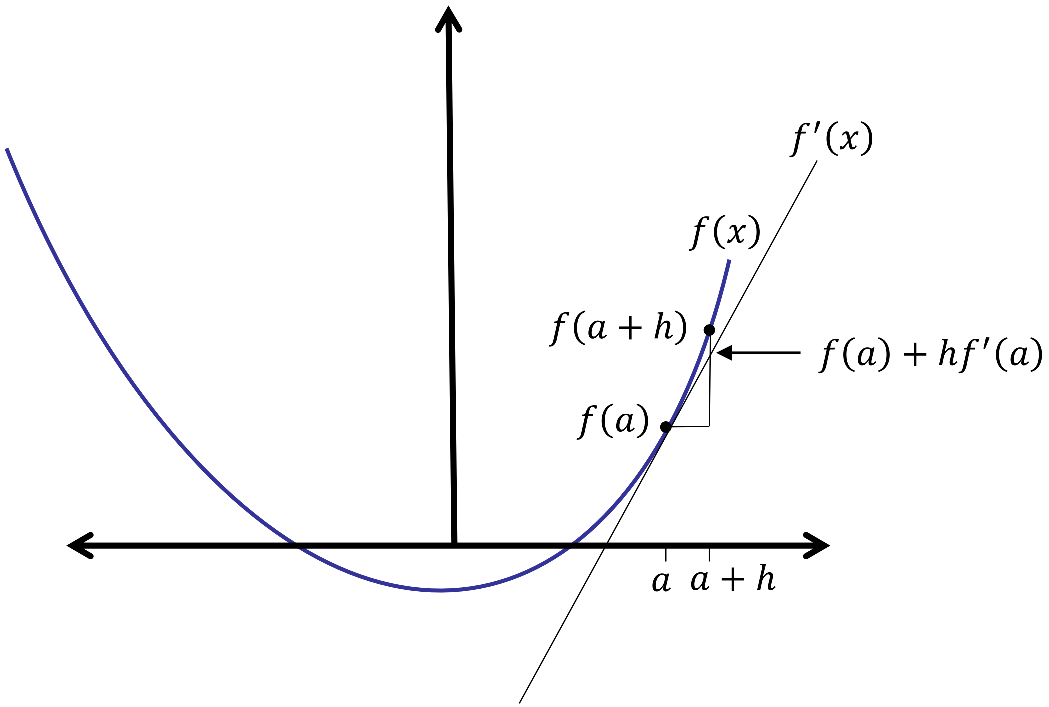


- **Convex Function**
  - Properties
    - For a convex function that is twice continuously differentiable,
    - The Hessian matrix should be positive semi-definite
    - Has the local minima as the global minima
  - Many machine learning modes are built by minimising a given cost function
    - Given the above properties convex cost functions are preferable
      - The global minima is obtainable through optimisation

- **Taylor Expansion**
  - We can approximate a point on a curve at $x = a + h$ by the corresponding point on the tangent

$$f(a+h) = f(a) + hf'(a)$$

  - For h close to 0, this is a good approximation



- **Taylor Expansion**
  - Any function can be expressed as an infinite sum by considering the value of the function, and its derivatives, at a specific point

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2!}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \cdots + \frac{1}{n!}h^n f^n(x)$$

  - Note:
    - When $f(x)$ is constant, all derivatives are zero and $f(x)$=$f(x+h)$
    - When $f(x)$ is linear, $f(x+h) = f(x) + hf'(x)$

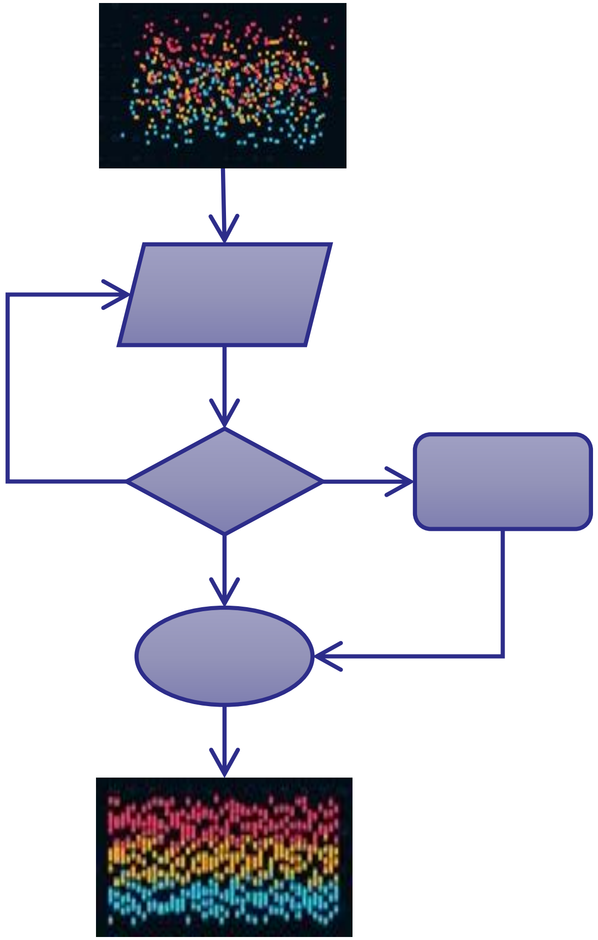
- **Taylor Expansion**
  - For multivariate functions around the point $\boldsymbol{x} \epsilon \mathbb{R}^{n\times 1}$:

$$f(\boldsymbol{x}+\Delta\boldsymbol{x}) = f(\boldsymbol{x}) + \Delta\boldsymbol{x}^T \nabla f(\boldsymbol{x}) + \Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}) \Delta\boldsymbol{x} + \cdots$$

  - Note:
    - $\Delta\boldsymbol{x}^T$ is the gradient vector
    - $\nabla^2 f(\boldsymbol{x})$ is the hessian matrix
    - In machine learning, we generally don’t expand beyond the second order as calculating the higher order terms is too cost intensive

Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- **Machine Learning Fundamentals**
- Generalisation
- Gradient Descent
- Summary



- **Machine Learning**
  - The application of statistical learning techniques to automatically identify patterns in data
  - **Start by defining a domain $\mathcal{D}$:**

$$\mathcal{D} = \{\mathcal{X}, P(X)\}$$

- Where:
  - $\mathcal{X}$ denotes a feature space
  - X denotes a set of feature vectors i.e., $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\} \in \mathcal{X}$
  - $\mathcal{X}$ is a d dimensional feature space i.e., $\boldsymbol{x} = \{x_1, x_2, \dots, x_d\}^{\boldsymbol{T}}$
  - $P(X)$ denotes a marginal probability distribution i.e., the distribution of X in $\mathcal{X}$

- **Machine Learning**
  - **Next we define a generic analysis task $\mathcal{F}$:**

$$\mathcal{F} = \{\mathcal{Y}, f(\cdot)\}$$

- Where:
  - $\mathcal{Y}$ denotes a label space
  - f denotes a predictive function or a conditional probability $P(Y|X)$
- Note, a database normally consist of two parts: features and labels
  - The feature vectors $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\} \in \mathcal{X}$
  - The corresponding labels $Y = \{y_1, y_2, \dots, y_n\} \in \mathcal{Y}$

- **Machine Learning**
  - **The goal of $\mathcal{F}$ is to learn the robust predictive function $f(\cdot)$**
    - A mapping from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$

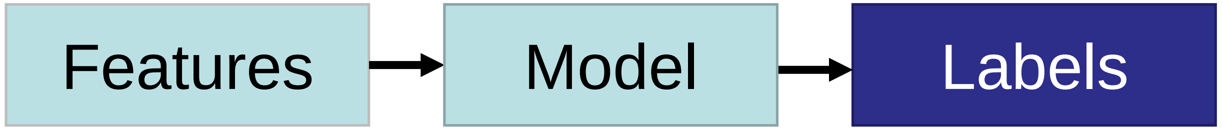
$$\mathcal{X} \xrightarrow{f(\cdot)} \mathcal{Y}$$

    - Given a test sample (unknown label), the learnt function maps the feature vector $\boldsymbol{x}_*$ onto a specific label $\boldsymbol{y}_*$

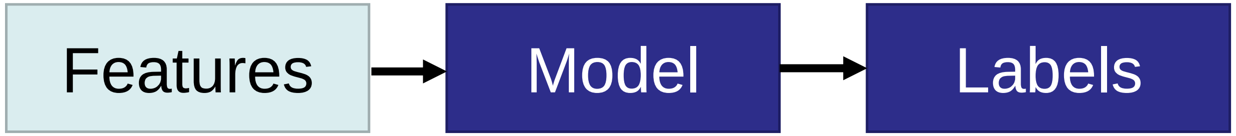
$$\boldsymbol{y}_* = f(\boldsymbol{x}_*)$$

Machine Learning Fundamentals

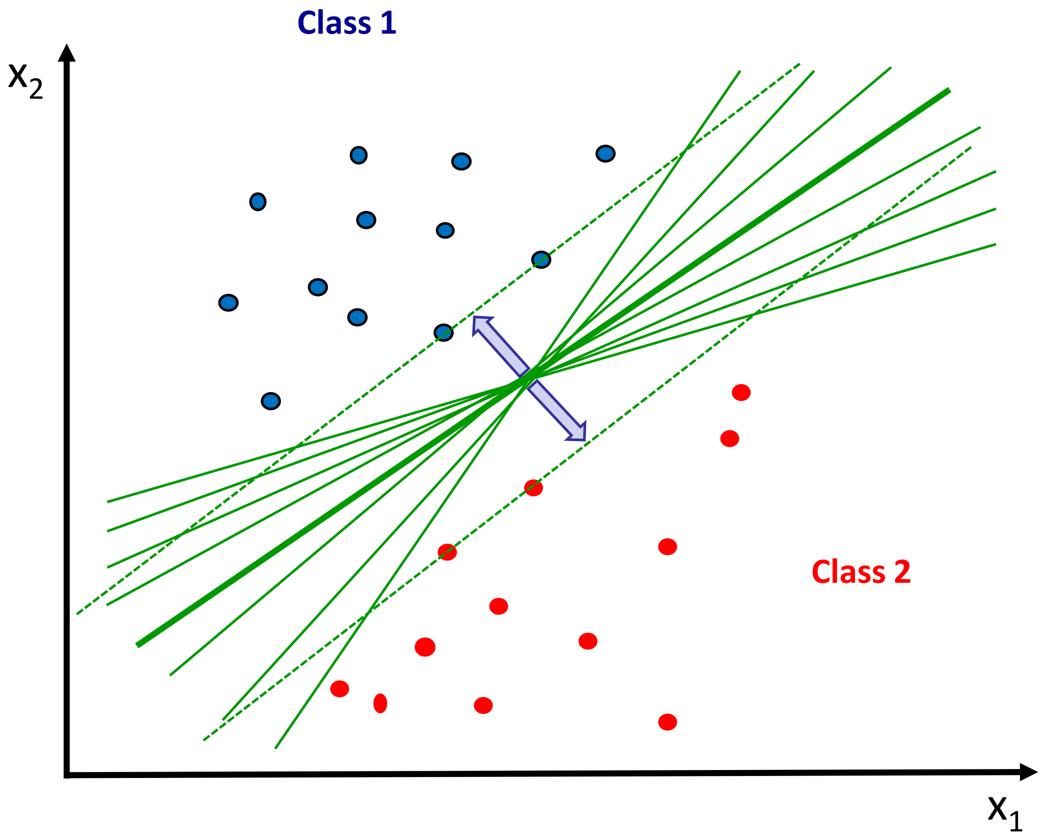
- **Supervised learning**
  - Learn a model from labels



- **Unsupervised learning**
  - Discover labels from the model



- **Machine learning algorithms**
  - Used to perform the process of pattern identification
    - Iterative techniques to find a set of optimal model parameters via the minimisation of a *cost function*
    - A **cost function** is a measure of how incorrect a model is in terms of its ability to estimate the relationship between $\mathcal{X}$ and $\mathcal{Y}$



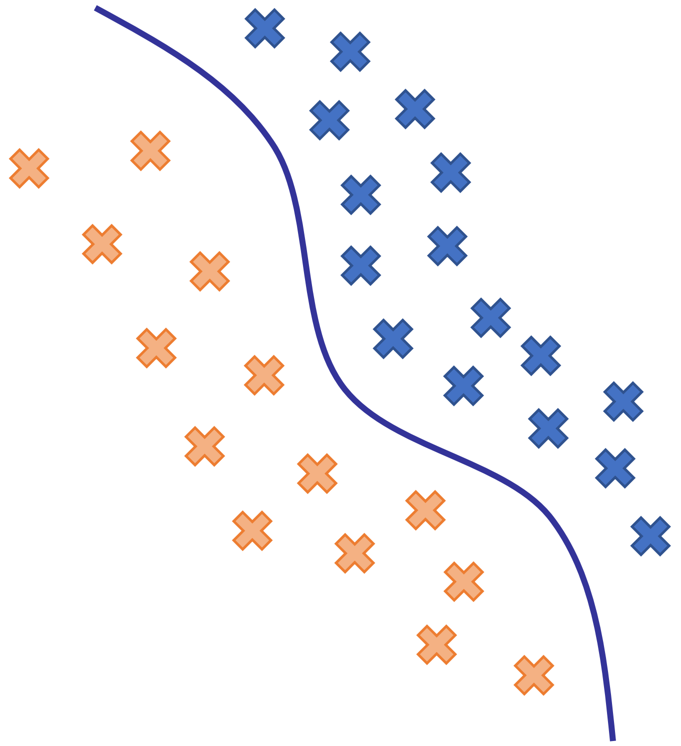
- **Machine learning algorithms**
  - Set of algorithms that can built mathematical models of data
  - Two main classes of machine learning algorithms:
    - **Discriminative Models:** Algorithms that directly learn a decision function from training data

$$\mathcal{X} \xrightarrow{f(\cdot)} \mathcal{Y}$$

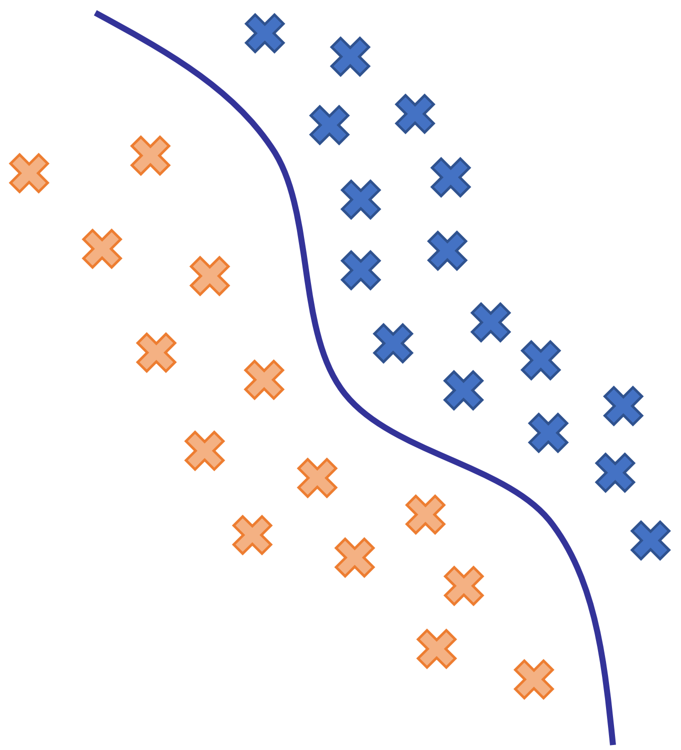
    - **Generative Models:** Algorithms that estimate the joint probability function between the features and the label, detection is then based on Bayes Rule

$$p(X, Y)$$

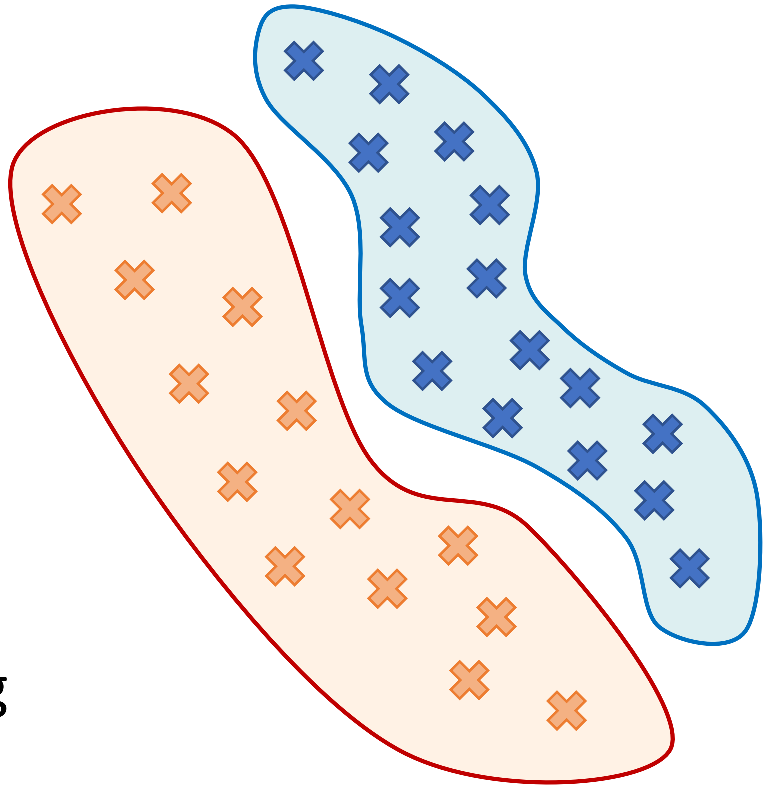
- **Discriminative Models**
  - Learn the (hard or soft) decision boundary between classes of interest
  - Assume some functional form for $p(Y|X)$
  - Estimate parameters for functional representation directly from training data
  - **Advantages**
    - Directly learn core decision objective
    - Higher accuracies with limited training samples



- **Discriminative Models**
  - Examples of discriminative models:
    - Random Forest
    - K-Nearest Neighbours
    - Support Vector Machines
    - Neural Networks
      - Deep Learning



- **Generative Models**
  - Model the joint probability function $p(X, Y)$ between the label and the features
  - Assume some functional form for $p(X|Y)$ and $p(Y)$ and estimate their parameters directly using the training data
  - Use Bayes Rule to calculate $p(Y|X)$
  - **Advantages**
    - Generative assumption helps prevent overfitting
    - Detect changes is testing data distribution
      - Potential to update models accordingly



Universität Augsburg University

• Generative Models

– Model the joint probability function $p(X,Y)$ between the label and the features

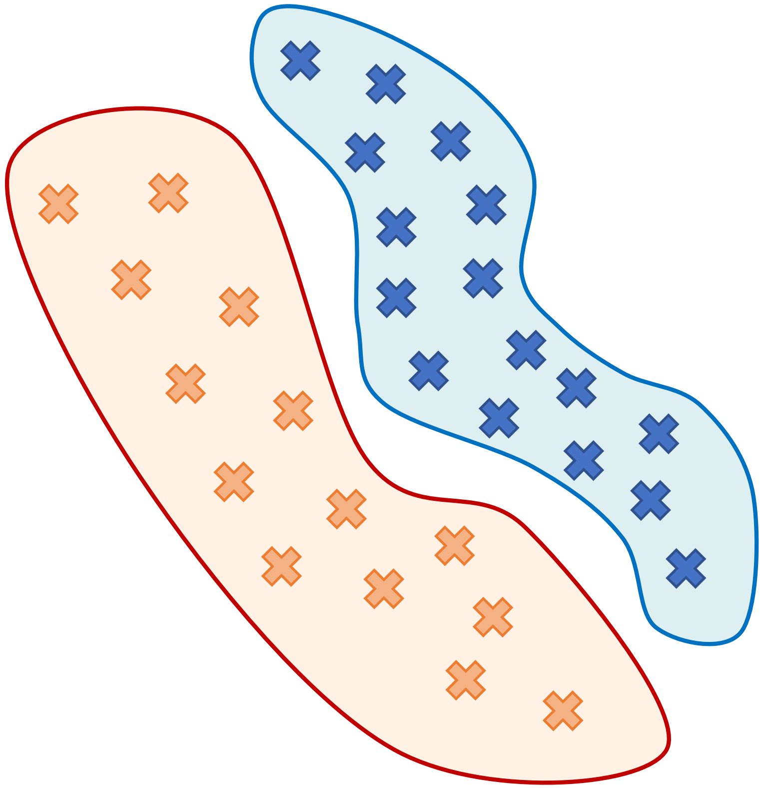
– However, we want to model $p(Y|X)$

- Conditional Probability Definition

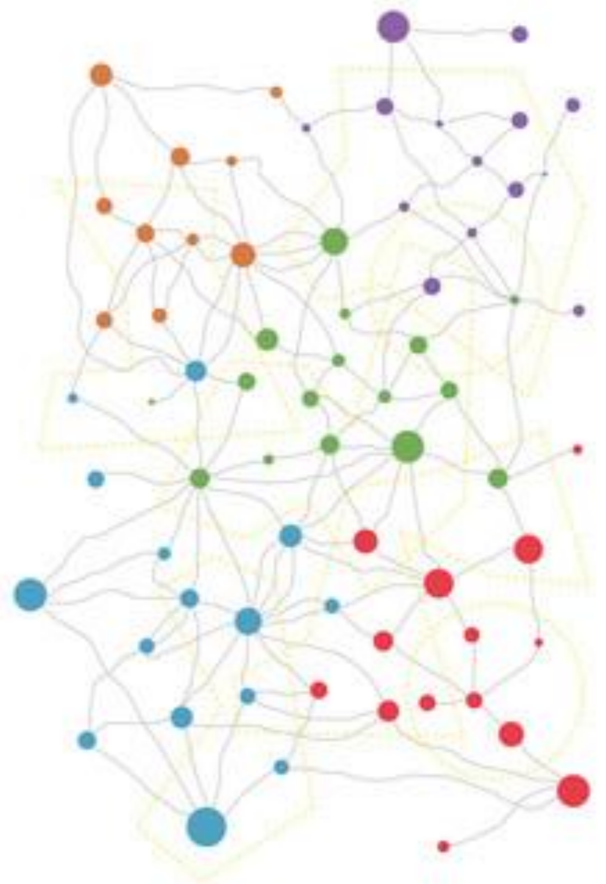
$$p(X|Y) = \frac{p(X,Y)}{p(Y)}$$

- Bayes Rule

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

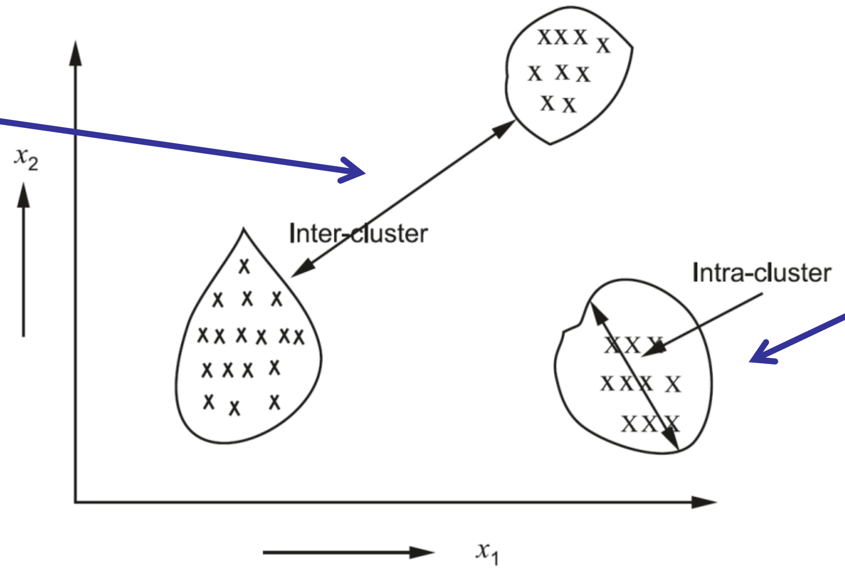


- Generative models often based on the concept of **clustering**
  - The process of grouping together sets of feature vectors
  - It generate partitions consisting of cohesive groups or clusters from a given collection of vectors
  - Feature vectors with similar (statistical) properties are grouped together while feature vectors with different properties are placed in separate groups



- **Clustering: the basic idea**
  - The distance between any two points belonging to the same cluster is smaller than that between any two points belonging to different clusters.

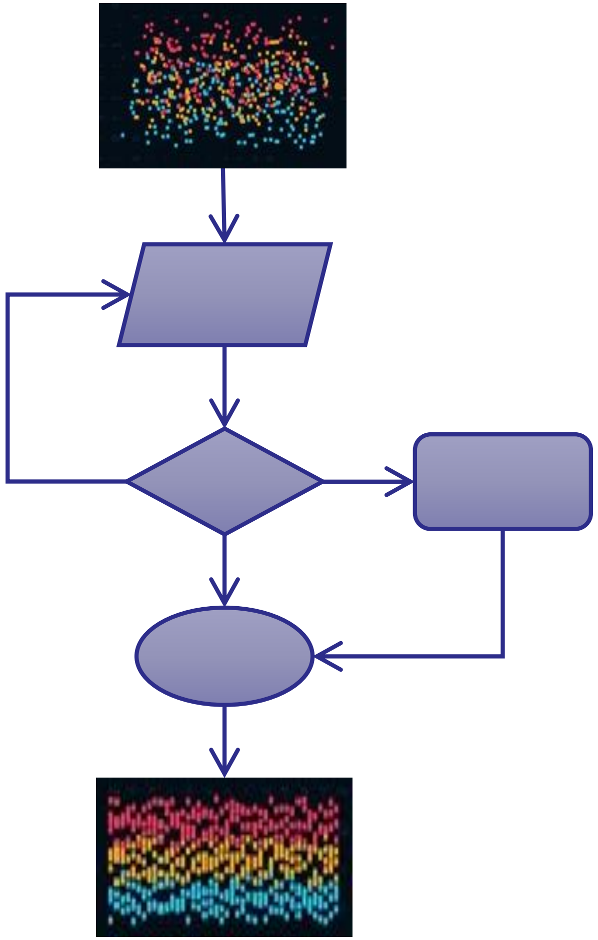
The inter-cluster cluster distance is large, hence the feature similarity is low



The intra-cluster cluster distance is small, hence the feature similarity is high

Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- Machine Learning Fundamentals
- **Generalisation**
- Gradient Descent
- Summary



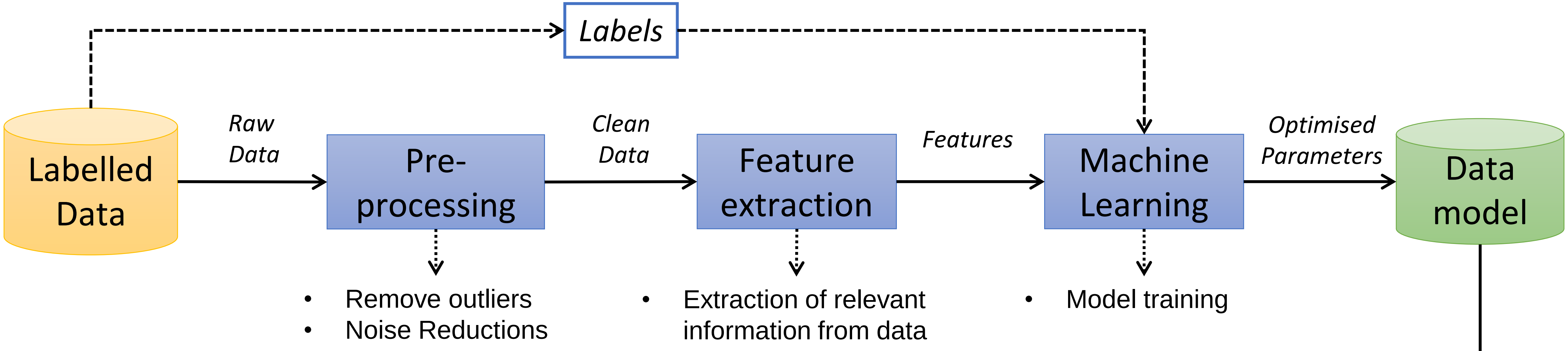
Generalisation

- **Aim**
  - Minimize/maximise a cost function of the model parameters given the data by using different optimization techniques.
- **Solution**
  - Set the derivative or gradient of the cost function to zero and solve for the model parameters
  - Not always possible
    - All solutions might not have a closed-form solution
    - The closed-form solution might be computationally expensive
    - A need for iterative methods for complex optimization problems

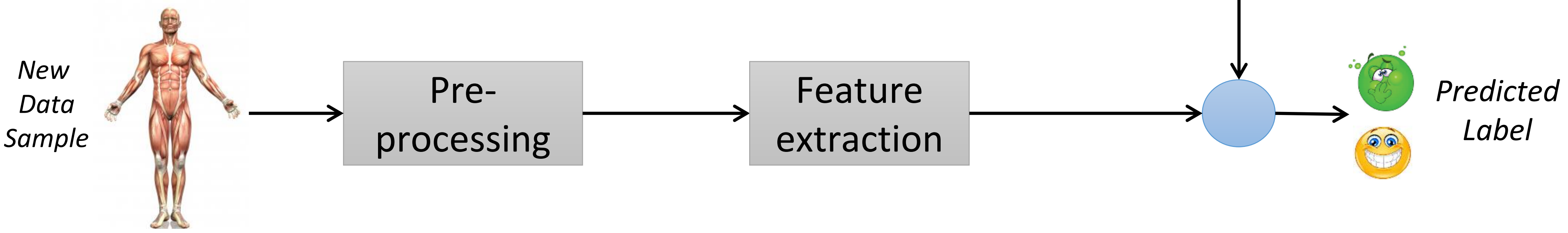
- **Generalisation of a supervised model**
  - In machine learning we want to minimise both the **training error** and the **generalization error**
    - The iterative process of training a machine learning algorithm minimises the training error
      - The difference between the actual and predicted label values in the training data, the subset of data instances used in the optimisation process
    - Introduces issues relating to the statistical concept of *sampling error*
      - As the number of data samples used in training is finite, therefore it cannot not include all members of the population of interest
    - There is a need to ensure the **generalisability** of a model
      - The model's ability to adequately label new test data samples
        - » Data **not used** during the training/optimisation phase
      - The generalization error is the error on these new data instances

Generalisation

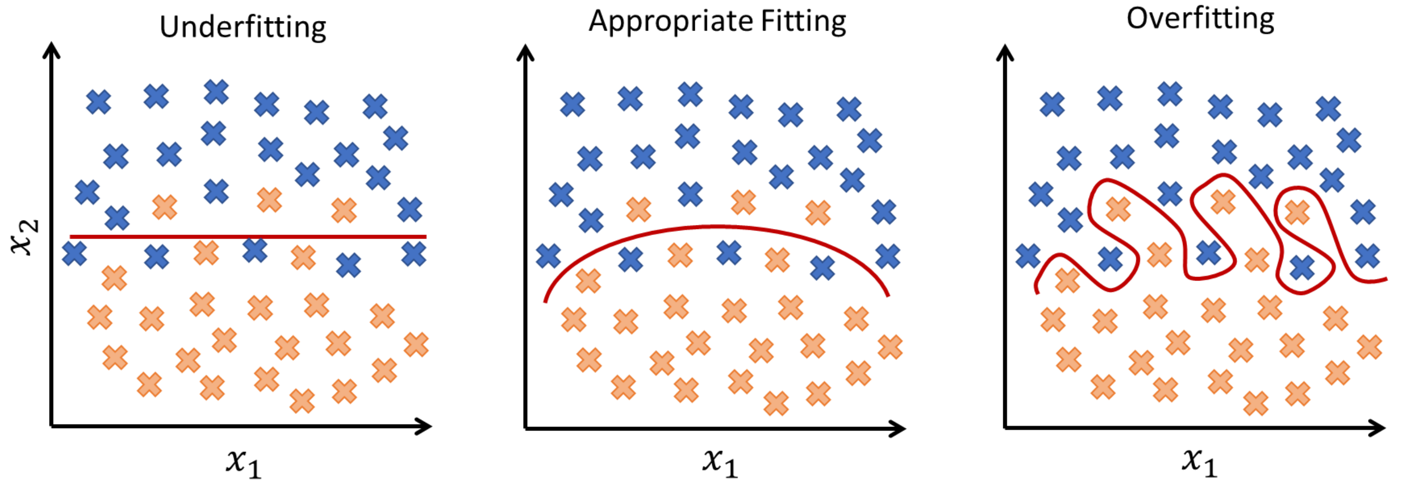
- **Training phase** – *minimise training errors*



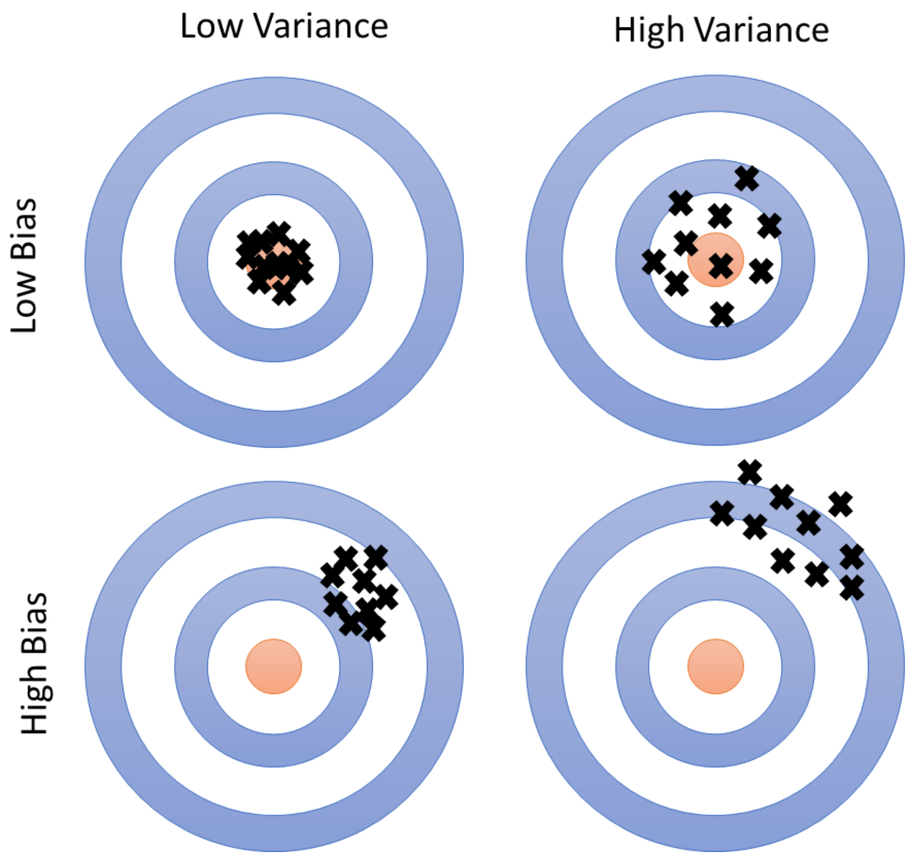
- **Test phase** – *minimise generalisation errors*



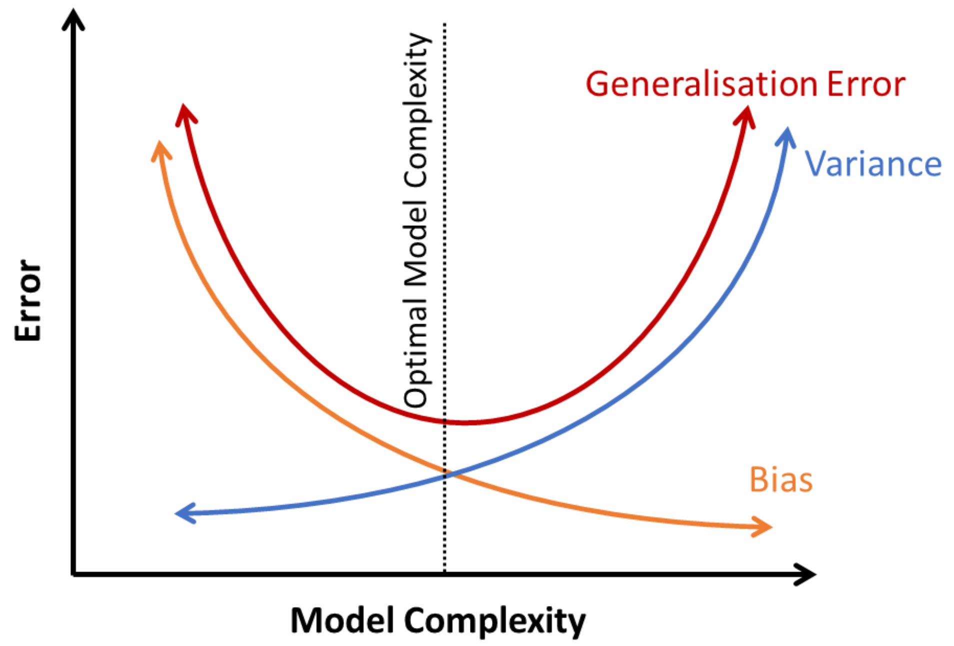
- **Generalisation Errors**
  - **Underfitting** – the model is too simple
    - The model has high bias and lacks sensitivity to the variation in data
  - **Overfitting** – the model is too complex
    - Model attempts to account for all the variation in the training data



- **Bias and Variance Errors**
  - **Bias:** on average, how much are do the predicted values differ from the actual values
  - **Variance:** how different will the predictions of the model be using different samples taken from the same population



- **Minimising Generalisation errors**
  - A trade-off of between bias and variance errors and the effect of model complexity
    - Increase in model complexity results in an initial decreases in generalisation error due to a decrease in model bias
    - As model becomes more complex generalisation errors increases due an increase in model variance



Generalisation

- **Data Partitioning**
  - System must generalise well to unseen data
  - Use 3 non-overlapping partitions

optimisation

TRAIN learn the model
60% of data

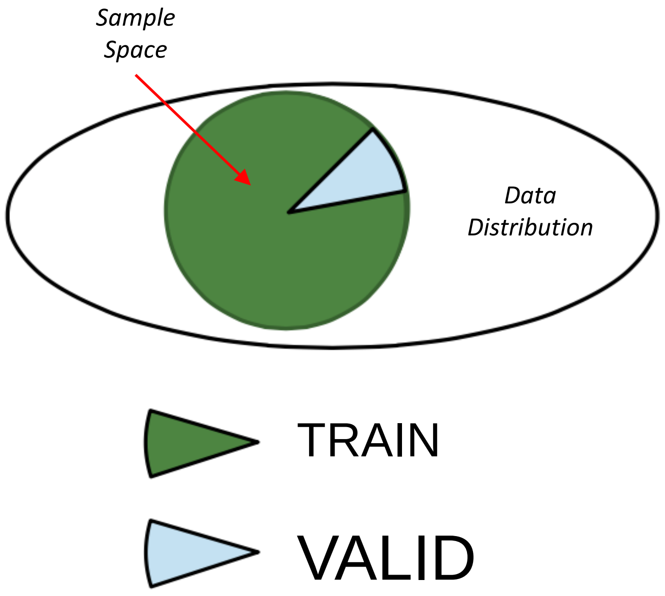
VALID test the model
20% of data

feature set
normalisation
machine learning parameters
…

Test model trained with best configuration found on **TRAIN+VALID**

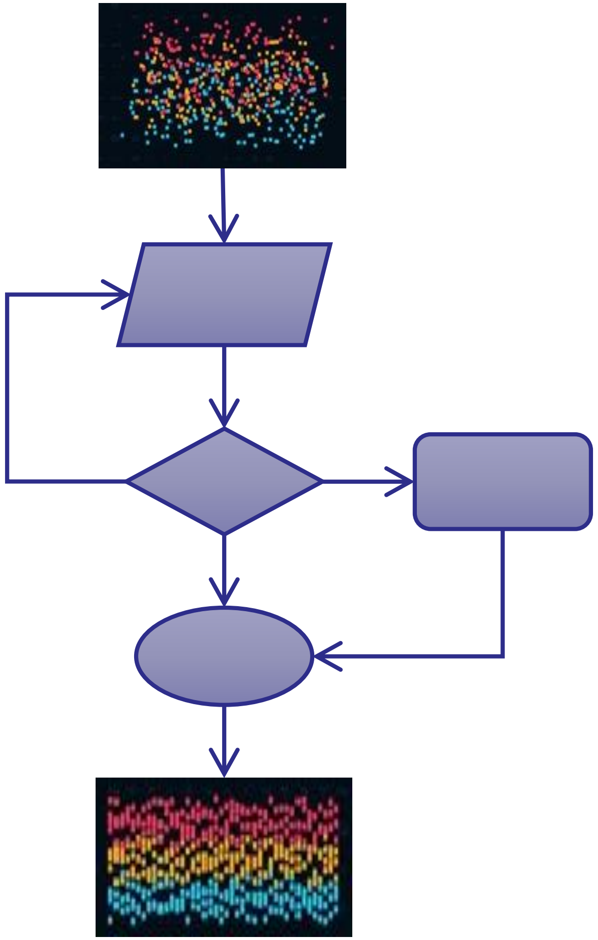
TEST: evaluate system
20% of data

- **Data partitioning**
  - Large dataset: percentage split
    - TRAIN 60%, VALID 20%, TEST 20%
  - Small dataset: cross-validation training
    - Randomise speaker ID or instances
    - Divide dataset into *k* equal folds
    - Train on all (*k* – 1) folds and validate on fold *k*
    - Repeat the procedure *k* times to cover all the data



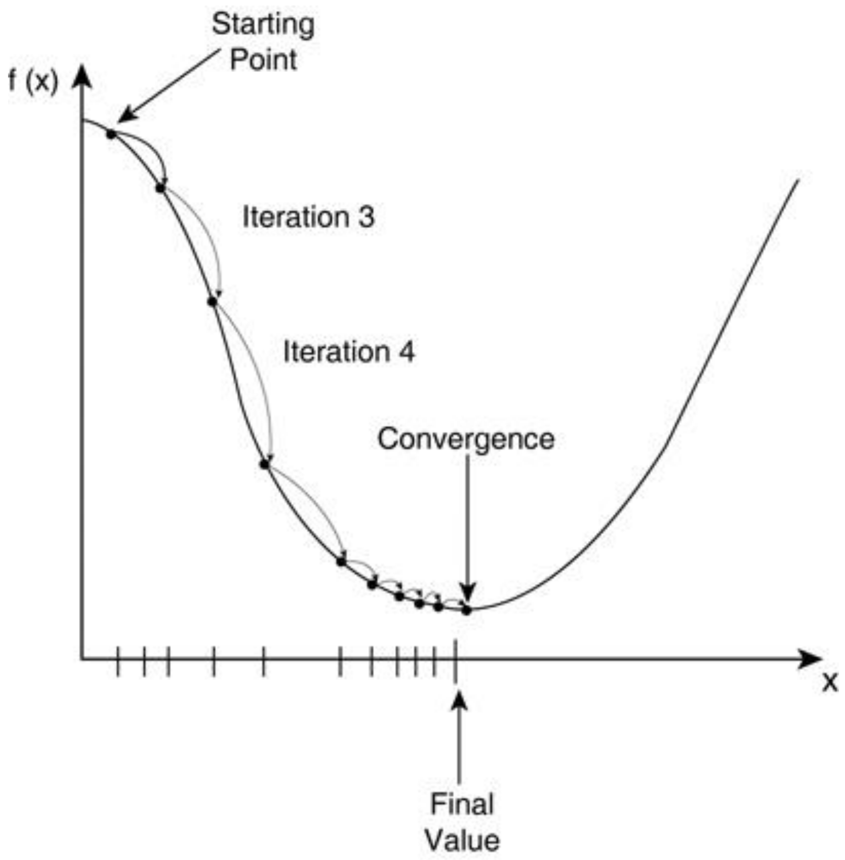
Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- Machine Learning Fundamentals
- Generalisation
- **Gradient Descent**
- Summary



- **Gradient Descent Algorithms**
  - Arguably the most widely used optimisation technique
  - Iterative solution
    - Uses the negative gradient of the cost function to determine the direction they parameters need updating

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)})$$

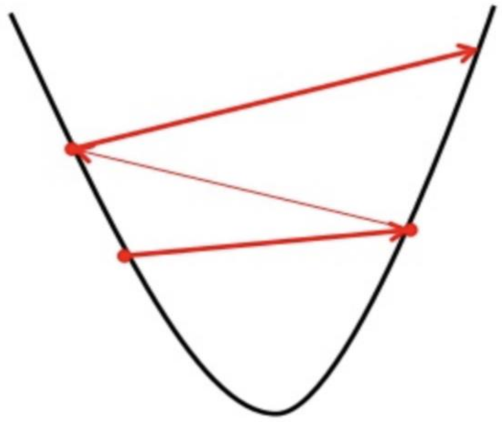


- **Gradient Descent Algorithms**

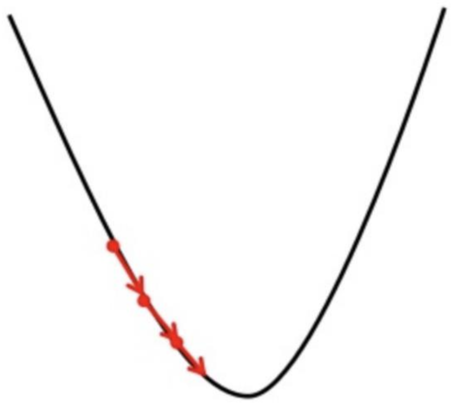
$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)})$$

  - η is the learning rate
  - A constant that defines the size of the gradient descent step
  - Size is important:
    - To large and the update function might oscillate over the minima
    - To small and convergence is slow

Big learning rate



Small learning rate



Source: https://towardsdatascience.com/

- **Multivariate Gradient Descent Algorithms**
  - Lets consider a cost function $C(\theta)$ where $\theta \in \mathbb{R}^{n \times 1}$
  - At every iteration we want to update θ to $\theta + \Delta\theta$ such that $C(\theta + \Delta\theta)$ is less than $C(\theta)$
  - Achieved by assuming linearity and using a *Taylor series expansion* we get:

$$\boxed{C(\theta + \Delta\theta) = C(\theta) + \Delta\theta^T \nabla C(\theta)}$$

  - Need to choose $\Delta\theta$ such that $C(\theta + \Delta\theta)$ is less than $C(\theta)$

- **Multivariate Gradient Descent Algorithms**
  - Need to choose $\Delta\theta$ such that $C(\theta + \Delta\theta)$ is less than $C(\theta)$

$$C(\theta + \Delta\theta) = C(\theta) + \Delta\theta^T \nabla C(\theta)$$

– To get the minimum value of the dot product $\Delta\theta^T \nabla C(\theta)$, the direction of $\Delta\theta$ should be the opposite of $\nabla C(\theta)$

$$\Delta\theta \propto -\nabla C(\theta)$$

Hence

$$\Delta\theta = -\eta \nabla C(\theta)$$

$$\theta + \Delta\theta = \theta - \eta \nabla C(\theta)$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)})$$

- **Gradient Descent strategies**
  - **Batch gradient descent**
    - Computes the gradient of the cost function with respect to the entire training dataset

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)})$$

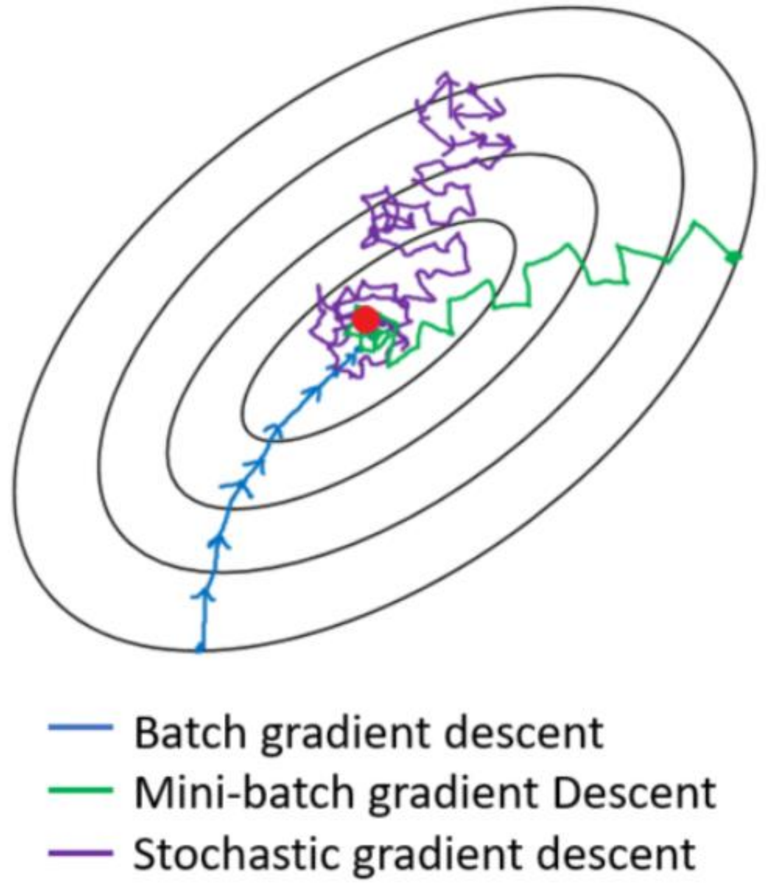
  - **Stochastic gradient descent (SGD)**
    - Performs a parameter update on each training example

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)}; x^{(i)}; y^{(i)})$$

  - **Mini-Batch**
    - Performs an update for every mini-batch of n training examples

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)}; x^{(i:i+n)}; y^{(i:i+n)})$$

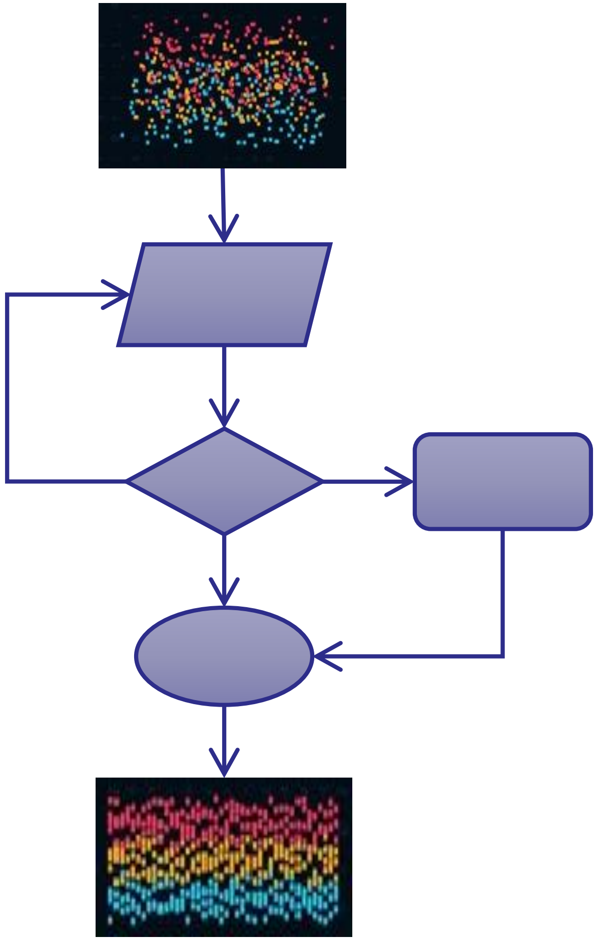
- **Gradient Descent strategies**
  - **Batch gradient descent**
    - Computational heavy, smoothest trajectory to convergence, should converge
  - **Stochastic gradient descent (SGD)**
    - Computational light, noisy convergence trajectory, may not converge
  - **Mini-Batch**
    - Trade-off between above methods
    - Batching adds noise to learning process which can improve generalisation



Source: https://towardsdatascience.com/

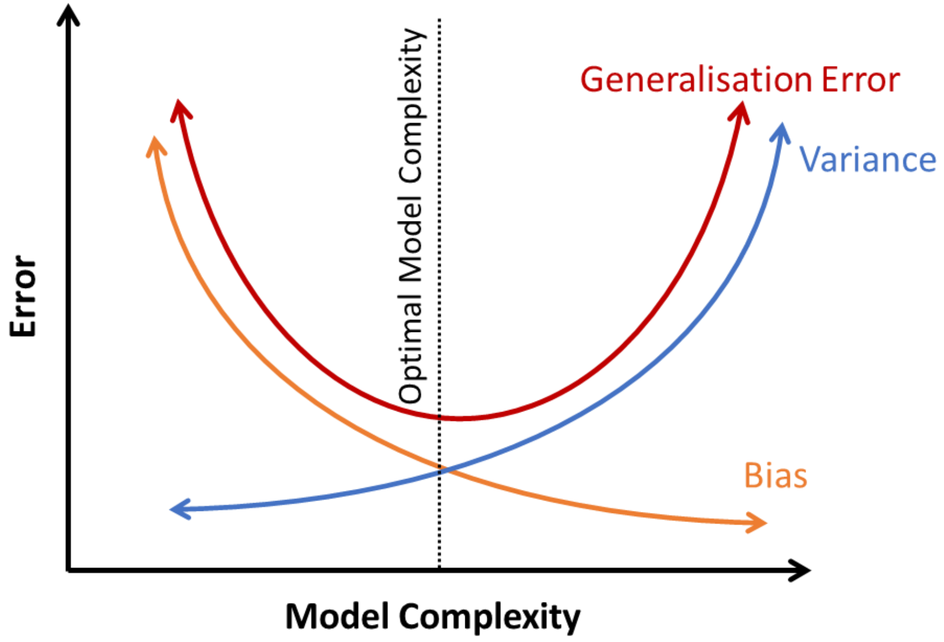
Lecture Outline

- Linear Algebra
- Probability
- Differential Calculus
- Machine Learning Fundamentals
- Generalisation
- Gradient Descent
- Summary



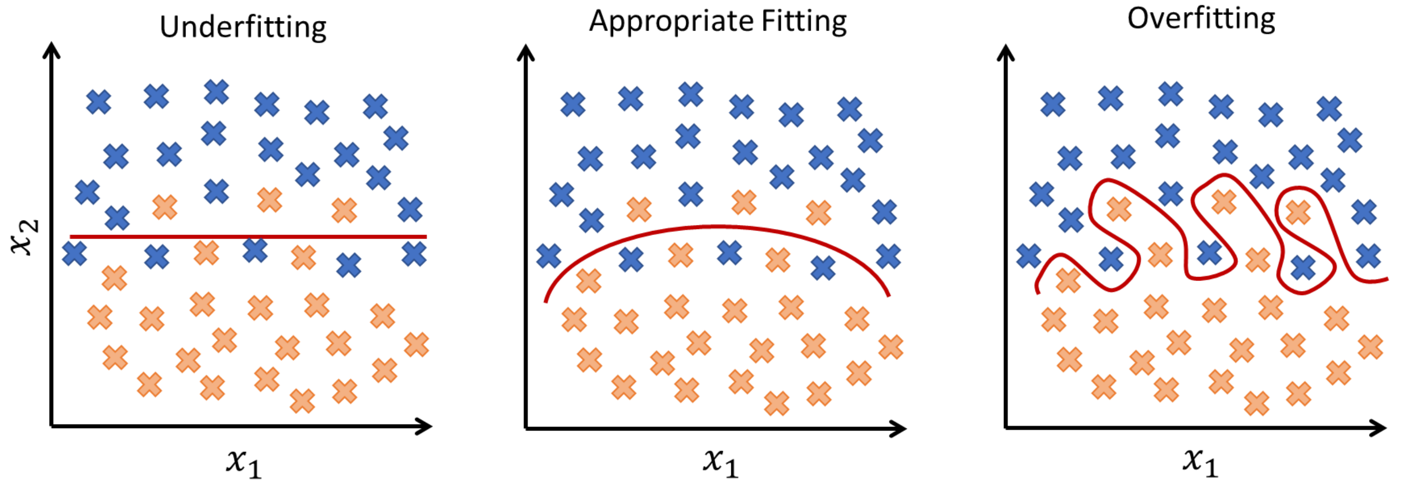
- **Machine Learning, the basic idea**
  - Minimise the cost function of the model parameters given the data by using different optimization techniques
    - Set the derivative or gradient of the cost function to zero and solve for the model parameters
  - Not always possible
    - A need for iterative methods for complex optimization problems
  - **Solution must be generalisable**
    - The model must be able to robustly label *new* data samples
    - In machine learning we want to minimise both the training error and the generalization error
      - **Training and Testing phase**

- **Minimising Generalisation errors**
  - A trade-off of between bias and variance errors and the effect of model complexity
    - Increase in model complexity results in an initial decreases in generalisation error due to a decrease in model bias
    - As model becomes more complex generalisation errors increases due an increase in model variance



Universität Augsburg University

- **Generalisation Errors**
  - **Underfitting** – the model is too simple
    - The model has high bias and lacks sensitivity to the variation in data
  - **Overfitting** – the model is too complex
    - Model attempts to account for all the variation in the training data



- **Gradient Descent Algorithms**
  - Iterative solution which uses the negative gradient of the cost function to determine the direction they parameters need updating

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)})$$

where η is the learning rate

  - **Gradient Descent strategies**
    - Batch gradient descent
    - Stochastic gradient descent (SGD)
    - Mini-Batch