Co-funded by the
Erasmus+ Programme
of the European Union

IoTrain
Master of Engineering in Internet of Things

Machine learning and Deep learning in IoT

University of Sumer
College of Computer Science and Information Technology

Course Information

- Basic information
  - University, semester, year
  - Teacher information
    - Short description
    - Basic contact information
- Description

  This course will provide an introduction to the necessary mathematical background and foundational principles and mechanisms to key techniques in machine learning (ML),deep learning (DL) as well as their implementation, application and evaluation. Then recent developments and emerging directions in ML-based IoT theory and applications will be briefly introduced.

Course Information

- Learning outcomes
  - Understand the fundamental concepts of machine learning and deep learning
  - Be able to define, train and use a ML and DL methods
  - Have a good understanding of the strengths and weaknesses of many popular machine learning approaches
  - Apply Machine learning algorithms to IoT problems
  - Design and implement machine learning solutions, evaluate the solution, analyse results and implication
- Main learning objectives
  - Know recent developments in the field of ML and DL
  - Understand IoT-efficient learning and applications to networked systems
- Evaluation
  - Exams
  - Labs
  - Projects

Course Information

- Requirements
  - BSc Data Science
  - BSc Mathematical and Statistic Techniques
  - BSc programming
- Bibliography

  - Machine Learning and IoT for Intelligent Systems and Smart Applications. https://www.routledge.com/Machine-Learning-and-IoT-for-Intelligent-Systems-and-Smart-Applications/P-Kumar-Umamaheswari/p/book/9781032047232
  - Hands-On Artificial Intelligence for IoT: Expert machine learning and deep learning techniques for developing smarter IoT systems, by Amita Kapoor , 2019.
  - Machine Learning Techniques for Internet of Things, P. Priakanth and S. Gopikrishnan, 2019.
  - Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). Deep Learning. MIT Press.
  - Deng, L.; Yu, D. (2014). "Deep Learning: Methods and Applications". Foundations and Trends in Signal Processing. 7 (3-4): 1-199. doi:10.1561/2000000039.
  - Deep Learning for IoT Big Data and Streaming Analytics: A Survey. 2018. https://ieeexplore.ieee.org/abstract/document/8373692
  - A project report on “Implementation of Machine Learning in IoT”. https://jecassam.ac.in/wp-content/uploads/2021/03/ML_IoT_AAIIM.pdf

Content

- Machine Learning Basics
  - Introduction and History
  - Fundamental Understanding
  - Loss Functions
- Supervised Learning
  - Decision Trees
  - K-Nearest Neighbour
  - Support Vector Machines
- Unsupervised Learning
  - Cluster Analysis
  - K-Means Clustering
  - K-Medoids
- Reinforcement Learning
  - Basics
  - Markov Decision Process
  - Q-Learning
- Deep Learning
  - Strategy
  - Convolutional Neural Network
- Machine Learning in IoT
  - Case Study I – Robot getting out
  - Case Study II - Iris classification
- Deep Learning in IoT
  - Case Study III - Handwriting recognition
  - Case Study IV – Attack Detection

Content

IoTrain
Master of Engineering in Internet of Things

- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

Machine Learning Basics

- What is machine learning?
- Why is machine learning important?
- What can machine learning be used for?
- Get to know some history of machine learning.
- Basic elements for building a machine learning system?

Machine Learning Basics

- What is machine learning?
  - Tom M. Mitchell's perspective, 2006:
    - Main research question:
      "How can we build computer systems that automatically improve with experience…"
    - A more precise definition:
      "a machine learns with respect to a particular task T, performance metric P, and type of experience E, if the system reliably improves its **performance P** at **task T**, following **experience E**."

      Task examples: prediction, decision making, pattern discovery…

Machine Learning Basics

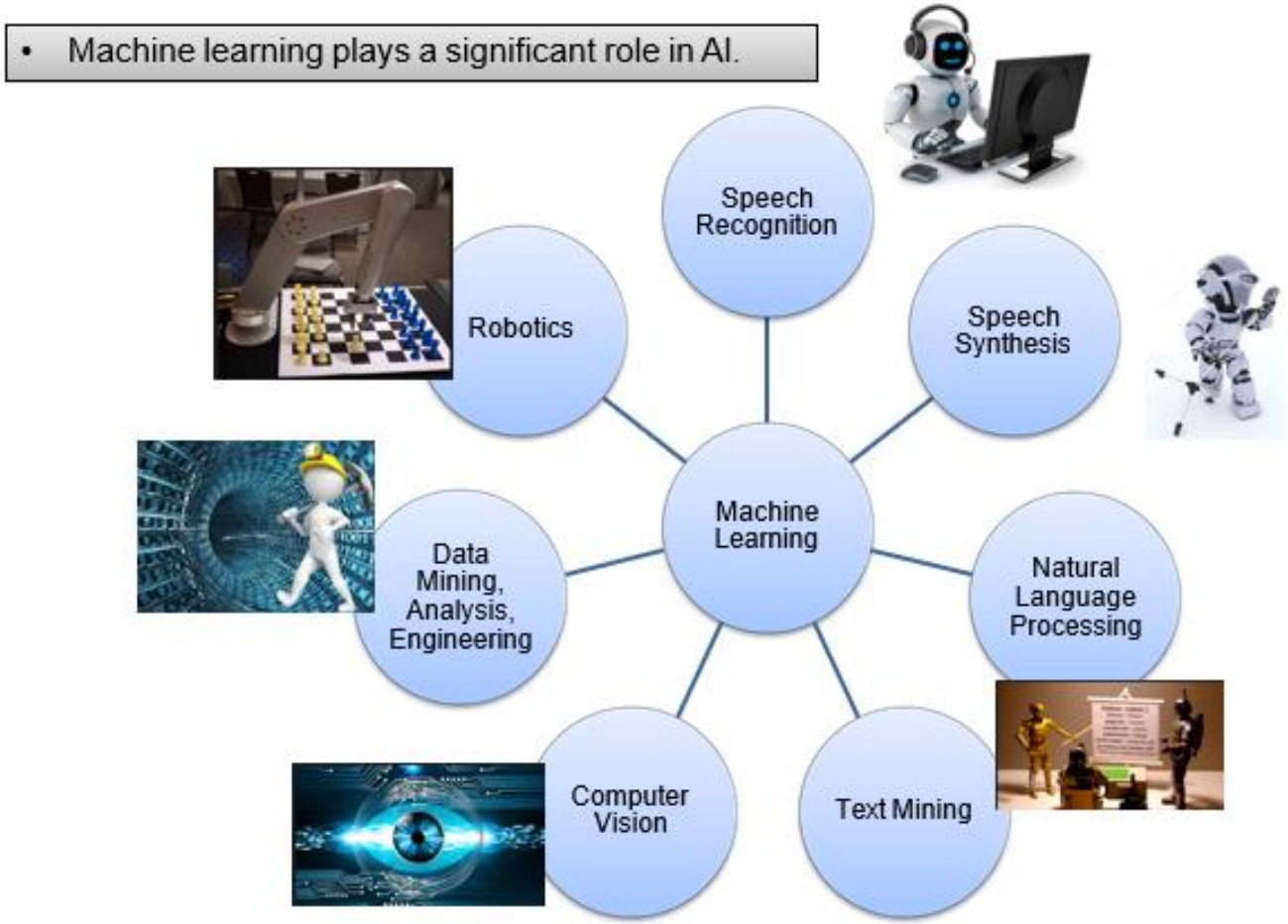
- What Can Machine Learning Do?
  - A machine learning system can be used to
    - Automate a process
    - Automate decision making
    - Extract knowledge from data
    - Predict future event
    - …

} **Task T**

Machine Learning is NOT about writing code to explicitly do the above tasks ✗	Machine Learning writes code to **make the computer learn from data** how to do the above tasks ✓

Machine Learning Basics

- Machine Learning in Artificial Intelligence



Machine Learning Basics

- Machine Learning in Data Science
  - Data is recorded on real-world phenomenons. The World is driven by data.
    - Germany's climate research centre generates 10 petabytes per year.
    - Google processes 24 petabytes per day.
    - PC users crossed over 300 billion videos in August 2014 alone, with an average of 202 videos and 952 minutes per viewer.
    - There were 223 million credit card purchases in March 2016, with a total value of £12.6 billion in UK.
    - Photo uploads in Facebook is around 300 million per day.
    - Approximately 2.5 million new scientific papers are published each year.
    - …
  - What might we want to do with that data?
    - Prediction: what can we predict about this phenomenon?
    - Description: how can we describe/understand this phenomenon in a new way?
  - Humans cannot manually handle data in such scale any more. A machine learning system can learn from data and offer insights.

Machine Learning Basics

- Machine Learning History
  - This is a young subject with pioneer research started around 1950s.
  - Please have a go with the Wiki page:

    https://en.wikipedia.org/wiki/Timeline_of_machine_learning

    This is a timeline of machine learning, explaining major discoveries, achievements, milestones and other major events in the field.

  - Here is another webpage talking about “A History of Machine Learning”.

    https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning/

Machine Learning Basics

- Key Historical Events
  - 1940s, human reasoning / logic first studied as a formal subject within mathematics (Claude Shannon, Kurt Godel et al).
  - 1950s, the Turing Test is proposed: a test for true machine intelligence, expected to be passed by year 2000. Various game-playing programs built.

    1956, Dartmouth conference coins the phrase artificial intelligence.

    1959, Arthur Samuel wrote a program that learnt to play draughts (checkers if you are American).
  - 1960s, A.I. funding increased (mainly military). Famous quote: Within a generation ... the problem of creating 'artificial intelligence' will substantially be solved."
  - 1970s, A.I. winter. Funding dries up as people realise it is hard. Limited computing power and dead-end frameworks.
  - 1980s, revival through bio-inspired algorithms: neural networks, genetic algorithms. A.I. promises the world – lots of commercial investment – mostly fails. Rule based expert systems used in medical / legal professions.

Machine Learning Basics

- Key Historical Events
  - 1990s, AI diverges into separate fields: Machine Learning, Computer Vision, Automated Reasoning, Planning systems, Natural Language processing… Machine Learning begins to overlap with statistics / probability theory.
  - 2000s, ML merging with statistics continues. Other subfields continue in parallel. First commercial-strength applications: Google, Amazon, computer games, route-finding, credit card fraud detection, etc... Tools adopted as standard by other fields e.g. biology.
  - 2010s, deep neural networks have led to significant performance improvement in speech recognition, reinforcement learning, image classification, machine translation, etc.. Yoshua Bengio, Geoffrey Hinton, and Yann LeCun 2018 ACM A.M. Turing Award for their contribution in deep neural network.
  - Future?

Machine Learning Basics

- Machine Learning Strategy
  - "Data + Model" Strategy:

- **Data X**: collection of **experience E**.

- **Function f**: Function output provides answers/solutions to **Task T**.
- **Parameters θ**: control model behaviour.
- **Objective function O(θ)**: It computes a **performance P of task T**.
- **Optimisation min O(θ)**: learning (or training).

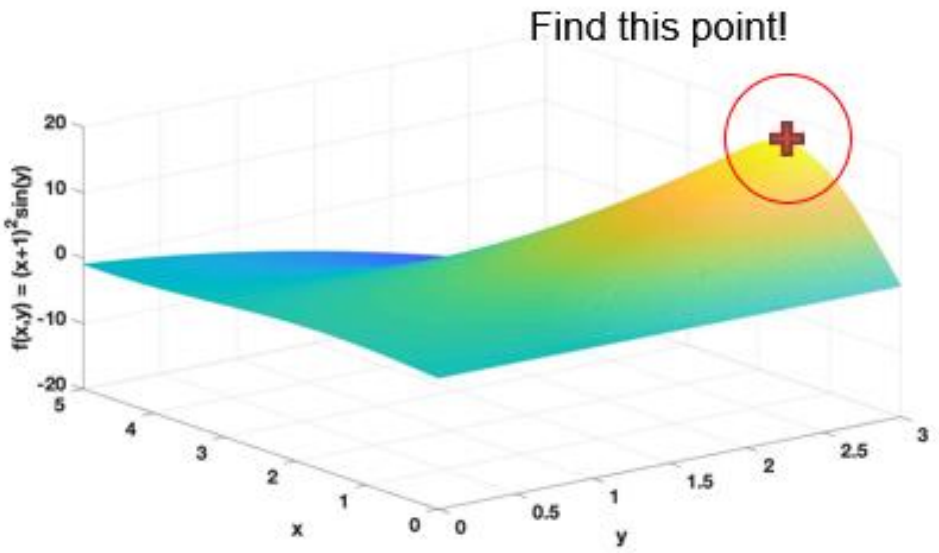
Machine Learning Basics

- What is Optimisation?
  - In English, finds the best!
    - "Optimal" means best.
    - "Optimisation" comes from the same root as "optimal".
    - When you optimise something, you are "making it best".
  - Mathematical optimisation (a branch of applied mathematics) finds the input that can given the minimum (or maximum) output value of a real- valued function.
    - *Systematically choose the values of the function input from an allowed set (who gives the best?).*
    - *Compute the output value of the function using the chosen input value (what is the best value?).*

Machine Learning Basics

- Example:

$$\max (x+1)^2 \sin(y)$$
$$-x \leq 0$$
$$x-5 \leq 0$$
$$-y \leq 0$$
$$y-3 \leq 0$$

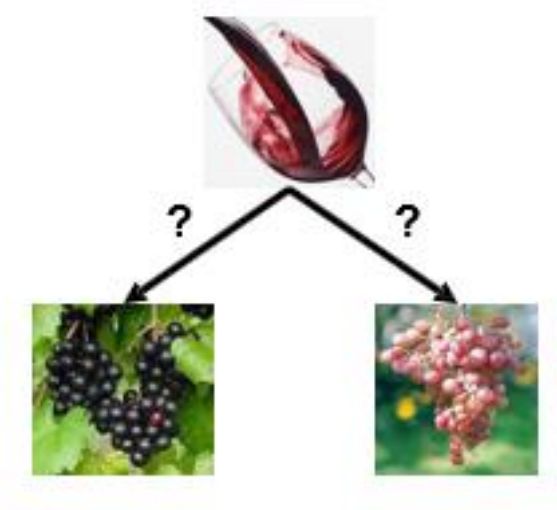


Machine Learning Basics

- Machine Learning Pipeline
  - Key stages in building a machine learning system:
    - Model construction:
      - To prepare experience (E) in proper data format.
      - To characterise a task (T) by a parametric model.
      - To characterise a performance metric (P) by an objective function.
    - Training:
      - To determine the model through optimising the objective function.
    - Evaluation:
      - To assess the determined model using a performance metric.
  - Machine learning research builds on optimisaton theory, linear algebra, probability theory…

Machine Learning Basics

- We use a wine classification example to demonstrate the machine learning pipeline.



Which type of grape is this wine made of?

Machine Learning Basics

IoTrain
Master of Engineering in Internet of Things

- Example: Wine Classification
  - Wine experts identify the grape type by smelling and tasting the wine.
  - Chemists know that the following quantities differ between wine types.

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

  - Collect the measurements. How to use these numbers?

```
14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
```

Machine Learning Basics

IoTrain
Master of Engineering in Internet of Things

- Example: Wine Classification

Task: To recognise the grape type of a given wine sample based on its measured chemical quantities!

- Collecting wine samples for each grape type.
- Characterising each wine sample with 13 chemical features.

Feature Extraction

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

Experiences

feature vectors

class labels

$$\mathbf{x}_1 = [x_{1,1}, x_{1,2}, x_{1,3}, \ldots, x_{1,12}, x_{1,13}], \quad y_1 = \text{grape type 1}$$
$$\mathbf{x}_2 = [x_{2,1}, x_{2,2}, x_{2,3}, \ldots, x_{2,12}, x_{2,13}], \quad y_2 = \text{grape type 2}$$
$$\mathbf{x}_3 = [x_{3,1}, x_{3,2}, x_{3,3}, \ldots, x_{3,12}, x_{3,13}], \quad y_3 = \text{grape type 2}$$
$$\vdots \qquad\qquad \vdots$$
$$\mathbf{x}_{30} = [x_{30,1}, x_{30,2}, x_{30,3}, \ldots, x_{30,12}, x_{30,13}], \quad y_{30} = \text{grape type 1}$$

30 bottles in total, 15 bottles for each grape type, each bottle is charaterised by 13 features.

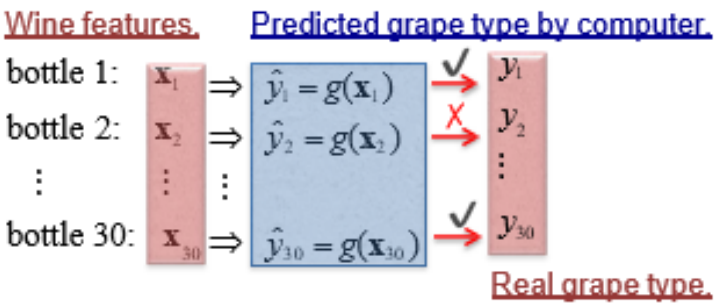
Machine Learning Basics

IoTrain
Master of Engineering in Internet of Things

Example: Wine Classification

- Design a mathematical model to predict the grape type. The model below is controlled by 14 parameters: $[w_0, w_1, w_2, \ldots, w_{13}]$

$$\hat{y} = g(\mathbf{x}) = \begin{cases} \text{type 1}, & \text{if } \sum_{i=1}^{13} w_i x_i + w_0 \geq 0 \\ \text{type 2}, & \text{if } \sum_{i=1}^{13} w_i x_i + w_0 < 0 \end{cases}$$



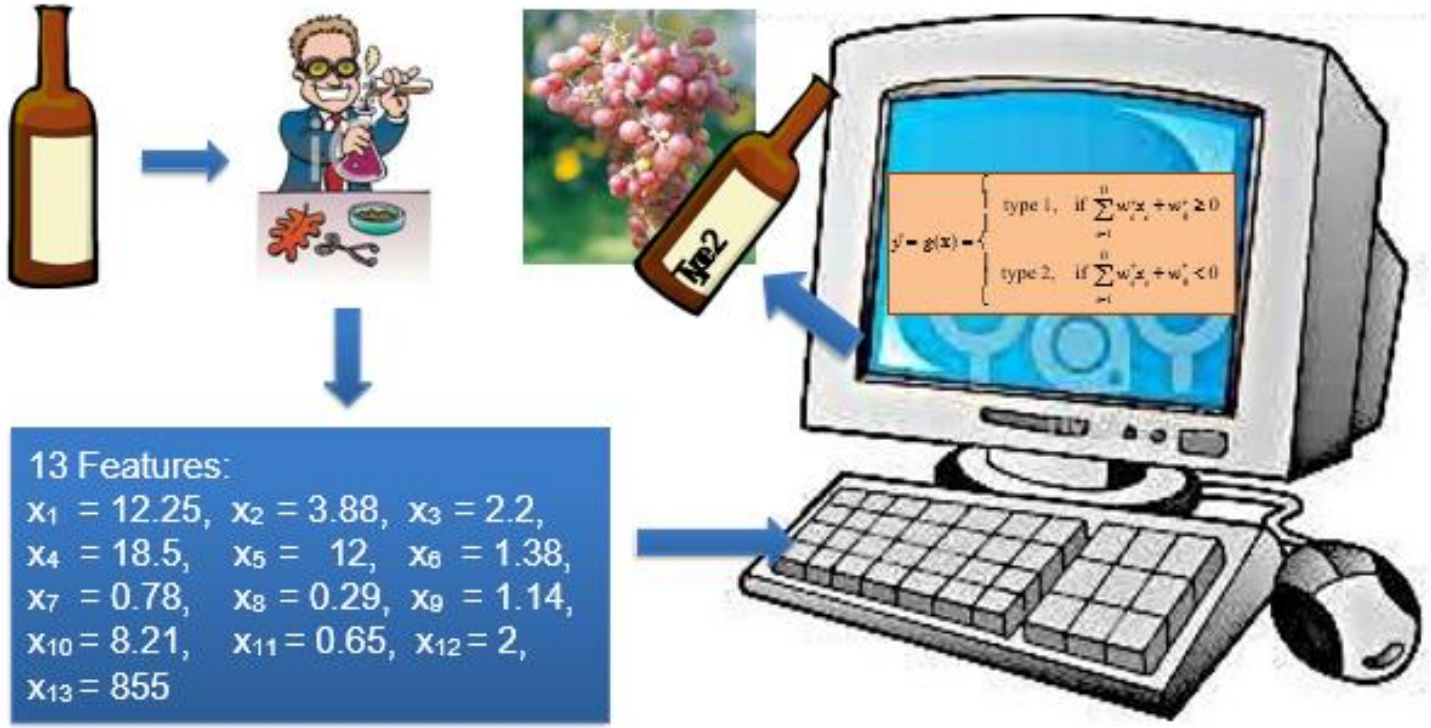
- System training is the process of finding the best model parameters by minimising a loss function.

$$[w_0^*, w_1^*, w_2^*, \ldots, w_{13}^*] = \underset{w_0, w_1, w_2, \ldots, w_{13}}{\arg\min}\ O_{loss}(w_0, w_1, w_2, \ldots, w_{13})$$

Loss: predictive error

Machine Learning Basics

- Example: Wine Classification
  - Now, test an unseen bottle of wine:



Machine Learning Basics

- Machine Learning Ingredients
  - Data (Experience)
  - Model: A piece of code (model function) with some parameters that need to be optimised.
  - Loss function: The function you use to judge how well the parameters of the model are set. It is also called error function, cost function, objective function.
  - Training algorithm: The algorithm that optimises the model parameters, using the error function to judge how well a model is performing.

Finally, the model with determined parameters is the thing you have to package up and send to a customer.

Machine Learning Basics

- Supervised Learning
  - A Wiki Definition: "*the machine learning task of learning a function that maps an input to an output based on example input-output pairs*".
    - There is a "teacher" who provides a target output for each data pattern.
    - A pair of input data pattern and its target output is called a training example.
    - Typical supervised learning tasks include classification and regression, differing from their output type.

Machine Learning Basics

- Unsupervised Learning
  - Wiki Definition: *"Unsupervised learning is a type of algorithm that learns patterns from untagged data."*
    - There is no explicit "teacher".
    - The systems form a natural "understanding" of the hidden structure from unlabelled data.
  - Typical unsupervised learning tasks include
    - Clustering: group similar data patterns together.
    - Generative modelling: estimate distribution of the observed data patterns.
    - Unsupervised representation learning: remove noise, capture data statistics, capture inherent data structure.

Machine Learning Basics

- Reinforcement Learning
  - Wiki Definition*: "Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward."*
    - There is a "teacher" who provides feedback on the action of an agent, in terms of reward and punishment.
    - Helicopter manoeuvre: example reward can be following a desired trajectory; and an example punishment can be crashing.

Machine Learning Basics

- Guess: Supervised, Unsupervised or Reinforcement?
  - Write a computer program to find out what topics a given set of news articles are talking about and display these topics to inform readers.
  - Train a computer system to control a power station by rewarding for producing power and penalizing for exceeding safety thresholds.
  - Map a sentence in English to its Chinese translation by learning from many translation examples.

Machine Learning Basics

- Deep Learning
  - A Wiki Definition: "*part of a broader family of machine learning methods based on artificial neural networks with representation learning*".
    - Refers to techniques for learning using neural networks.
    - Considered as a kind of representation (feature) learning techniques

Machine Learning Basics

- Classification and Regression
  - Write a computer program to find out what topics a given set of news articles are talking about and display these topics to inform readers. (Unsupervised)
  - Train a computer system to control a power station by rewarding for producing power and penalizing for exceeding safety thresholds. (Reinforcement)
  - Map a sentence in English to its Chinese translation by learning from many translation examples. (Supervised)

Machine Learning Basics

- Classification and Regression
  - Classification and regression are both supervised learning tasks, but differ in output types.
  - **Classification** (class output):
    - Goal: Identify which categories a data pattern belongs to.
    - Training data: Observed data patterns and their category memberships.
  - **Regression** (continuous output):
    - Goal: Estimate the relationships among the input and output variables.
    - Training data: Observed input variables and their corresponding continuous output variables.

Machine Learning Basics

- Guess: Classification or Regression?
  - Medical diagnosis: x=patient data, y=positive/negative of some pathology
  - Finance: x=current market conditions and other possible side information, y=tomorrow's stock market price
  - Image analysis: x=image pixel features, y=scene/objects contained in image
  - Robotics: x=control signals sent to motors, y=the 3D location of a robot arm end effector

Machine Learning Basics

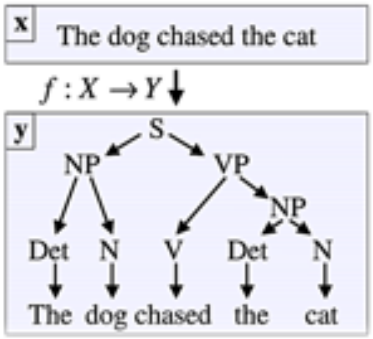
- Guess: Classification or Regression?
  - Medical diagnosis: x=patient data, y=positive/negative of some pathology (Classification)
  - Finance: x=current market conditions and other possible side information, y=tomorrow's stock market price (Regression)
  - Image analysis: x=image pixel features, y=scene/objects contained in image (Classification)
  - Robotics: x=control signals sent to motors, y=the 3D location of a robot arm end effector (Regression)

Machine Learning Basics

- Different Types of Classification
  - There are different types of classification tasks
    - **Binary** classification: classify into one of the two classes.
    - **Multi-class** classification: classify into one of the many classes.
    - **Multi-label** classification: classify into some of the many classes

      There is no constraint on how many classes a sample can belong to.
    - **Structured** classification: classify into structured classes

      Classes can be organised in sequence, tree, lattices, graph.



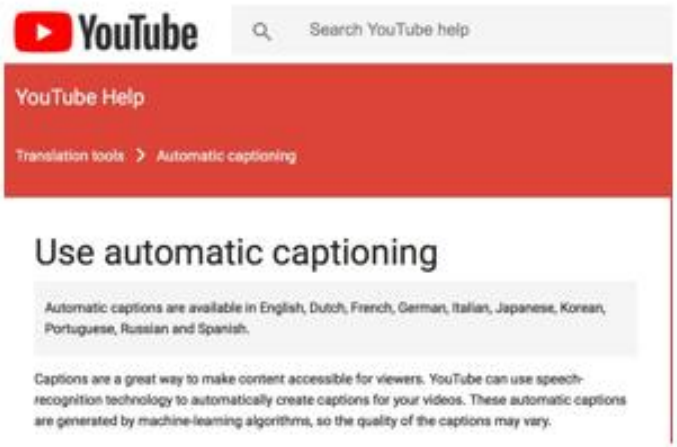
Parsing example. [Tsochantaridis, JMLR05]

Machine Learning Basics

- Successful Applications

supervised

Speech recognition and synthesis, translation



Machine Learning Basics

- Successful Applications

- Face recognition

supervised

Machine Learning Basics

- Successful Applications
  - Object recognition

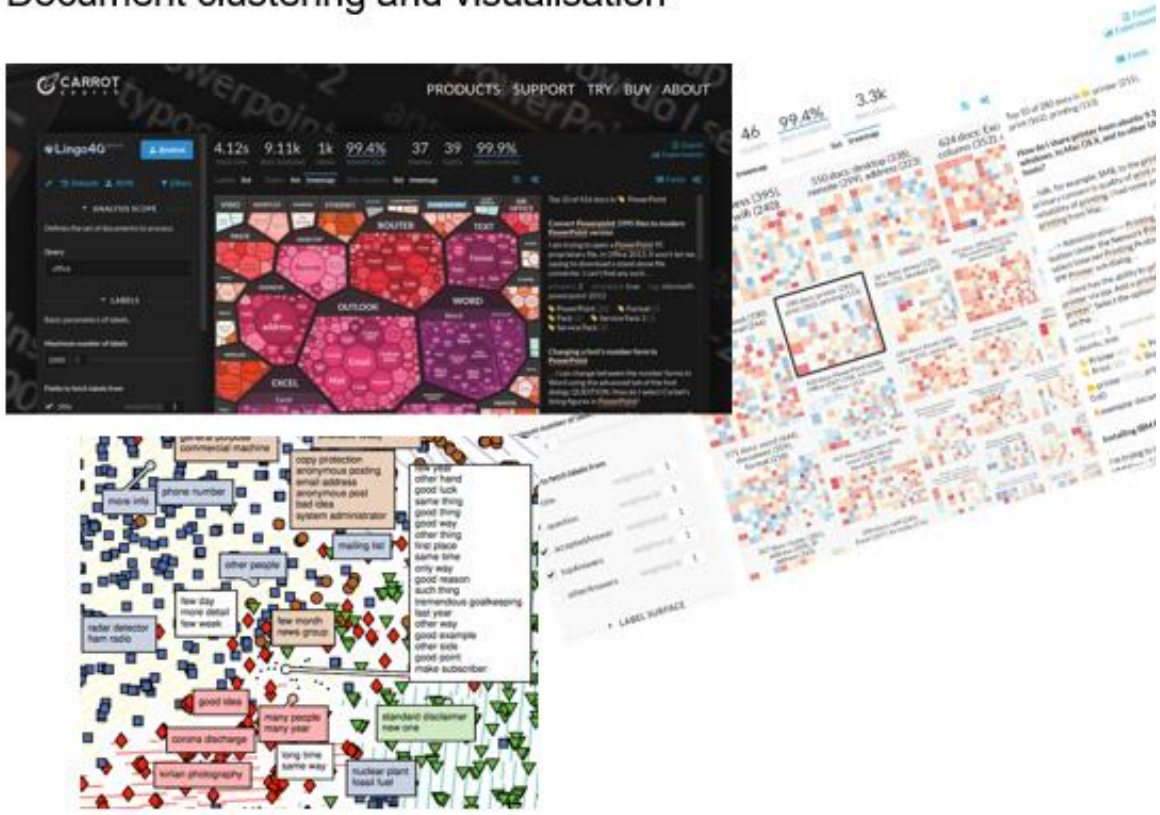
supervised

Machine Learning Basics

- Successful Applications

unsupervised

- Document clustering and visualisation



Machine Learning Basics

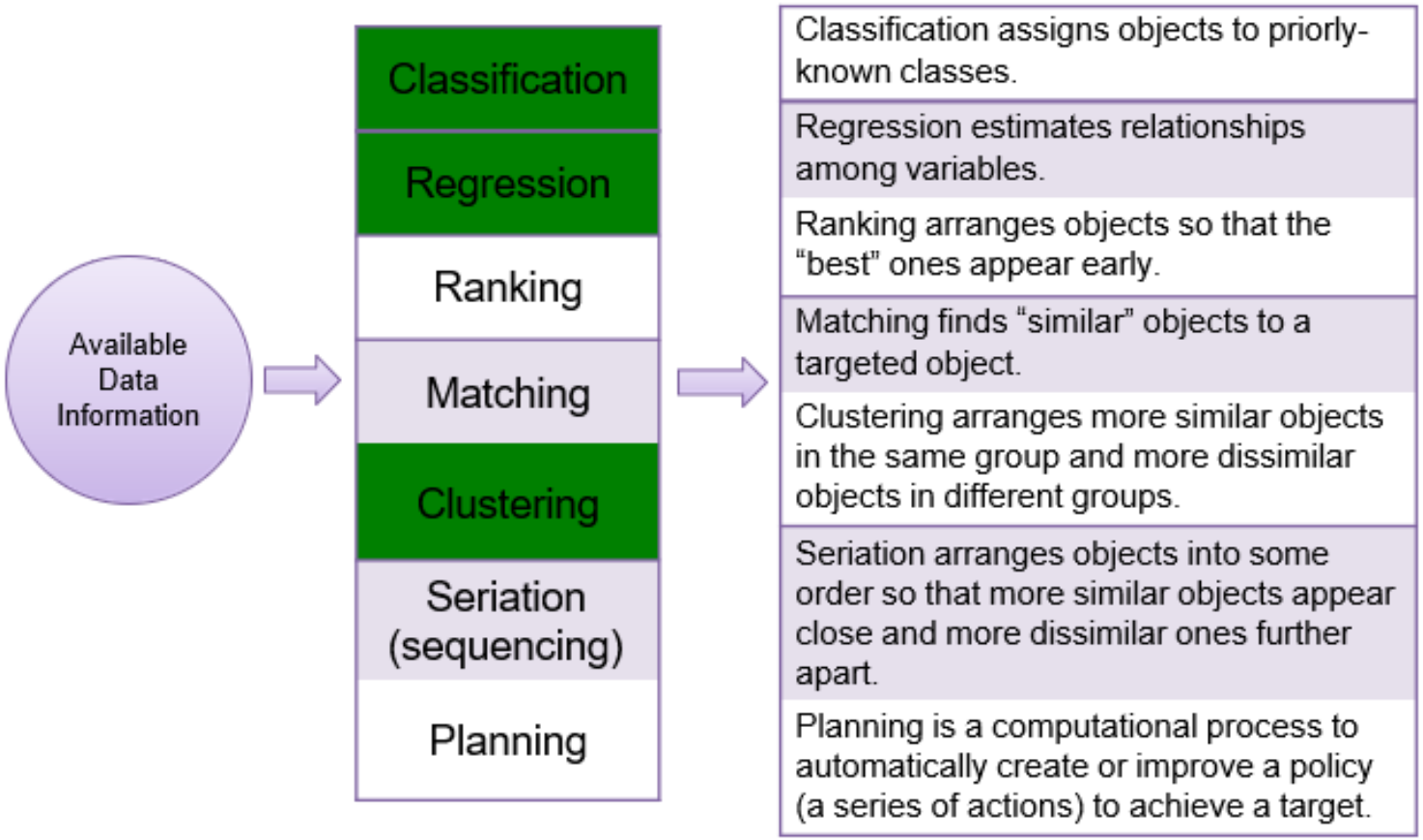
- Successful Applications

reinforcement

- Game player, self-driving cars, trading strategy.

Machine Learning Basics

- Typical Machine Learning Tasks



Machine Learning Basics

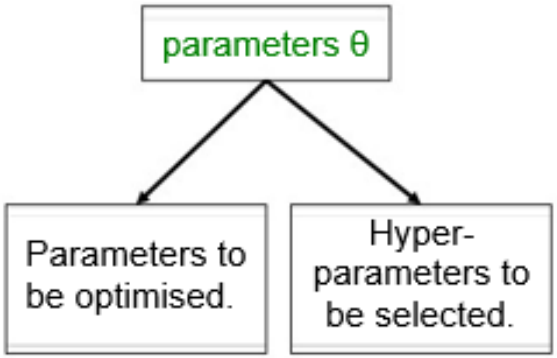
- Summary
  - Basic understanding of machine learning and its usage
  - Importance of machine learning
  - Machine learning ingredients and pipeline
  - Key machine learning tasks
    - Supervised, unsupervised, reinforcement learning
    - Classification
    - Regression
    - Other learning tasks

Machine Learning Basics- Loss Functions

- Loss Functions
  - Typical approaches of constructing losses for **regression**
    - Non-probabilistic losses
    - Probabilistic losses

Machine Learning Basics- Loss Functions

- Loss Functions
  - "Data + Model" Strategy in Supervised Learning:



Machine Learning Basics- Loss Functions

- Loss Functions
  - "Data + Model" in Supervised Learning
    - After the learning (or called training) is finished, the final product is:

A trained model:

$f(!(D_{tr}), x)$

    - The process of using the trained model on unseen data (or called query data, test data) is called inference.

$$answer = f_{(!\,(D_{tr}),x_{query})}$$

Machine Learning Basics- Loss Functions

- Loss Functions
  - Loss function is essential in training, computed using the training data.
  - It decides how good the model parameters are, how well the model fits your training data.
  - Other names: error function, cost function, or more general, objective function.
  - To train a machine learning model, you pick a loss function O(θ). Then:
    - Minimise it, if O evaluates how bad the model is.
    - Maximise it, if O evaluates how good the model is.
  - Supervised learning: $O(!, D_{tr})$

Machine Learning Basics- Loss Functions

- Loss Functions
  - Losses for Training Regression Models

Training Data: $D_{tr} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$

Feature vector: $\mathbf{x}_i \in R_d$

Target output: $\mathbf{y}_i \in R_c$ where $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots y_{ic}]$

Machine Learning Basics- Loss Functions

- Loss Functions
  - Non-probabilistic Regression Losses

Machine Learning Basics- Loss Functions

- Regression Error based Loss
  - Recall RMSE?
  - Some regression losses are simplified versions of RMSE computed using training samples.

The prediction for each training sample is computed by $\hat{\mathbf{y}}_i = f\left(\theta, \mathbf{x}_i\right)$.

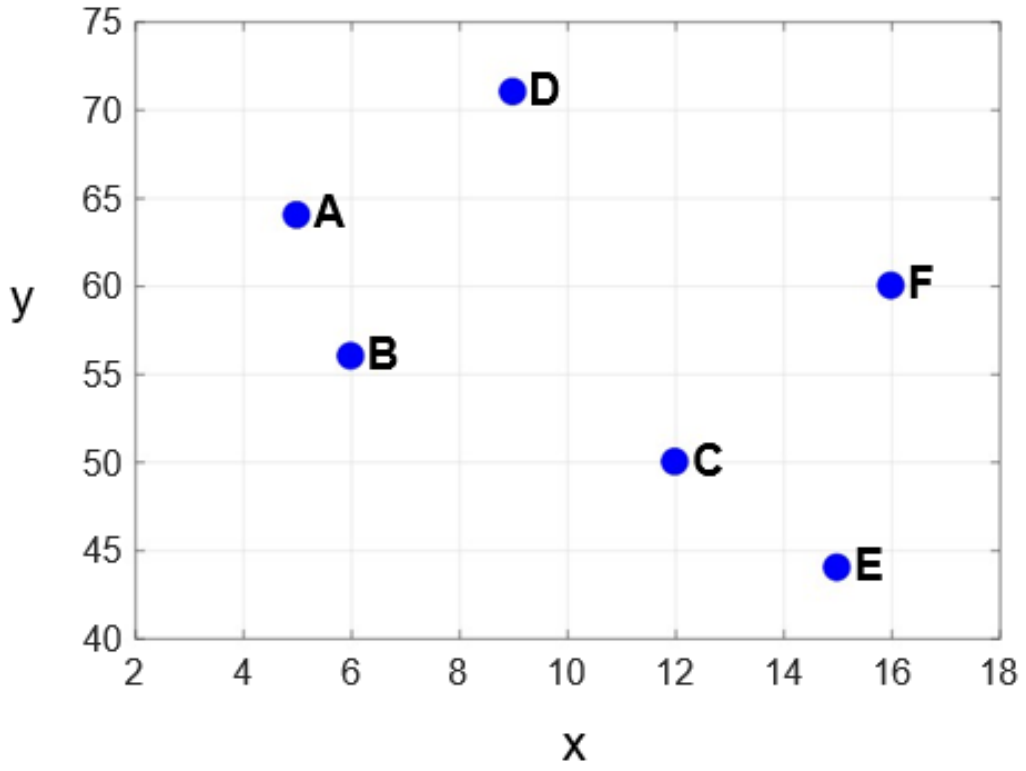
- **Sum-of-squares error loss:** $O\left(\theta\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{c}\left(\hat{y}_{ij} - y_{ij}\right)^2$

- **Mean-squared error loss:**

  A single-output example (*c=1*): $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2$

Machine Learning Basics- Loss Functions

- A Regression Example
  - Fit a **linear model** using the following six training data samples.



x	y
5	64
6	56
12	50
9	71
15	44

$$\hat{y} = f(x) = w_0 + w_1 x$$

Machine Learning Basics- Loss Functions

- A Regression Example
  - Sum-of-squares error loss for training the linear model (finding the best w0 and w1):

$$O = \frac{1}{2}\left[\left(\hat{y}_1 - y_1\right)^2 + \left(\hat{y}_2 - y_2\right)^2 + \left(\hat{y}_3 - y_3\right)^2 + \left(\hat{y}_4 - y_4\right)^2 + \left(\hat{y}_5 - y_5\right)^2 + \left(\hat{y}_6 - y_6\right)^2\right]$$

$$= \frac{1}{2}\sum_{i=1}^{6}\left(\hat{y}_i - y_i\right)^2$$

$$= \frac{1}{2}\sum_{i=1}^{6}\left(w_1 x_i + w_0 - y_i\right)^2$$

Machine Learning Basics- Loss Functions

- A Regression Example
  - Incorporate the values of x_i and y_i to the error function:

$$O(w_1, w_0) = \frac{1}{2}\Big[(5w_1 + w_0 - 64)^2 + (6w_1 + w_0 - 56)^2 + (12w_1 + w_0 - 50)^2 + (9w_1 + w_0 - 71)^2 + (15w_1 + w_0 - 44)^2 + (16w_1 + w_0 - 60)^2\Big]$$

$x_1 = 5, y_1 = 64$
$x_2 = 6, y_2 = 56$
$x_3 = 12, y_3 = 50$
$x_4 = 9, y_4 = 71$
$x_5 = 15, y_5 = 44$
$x_6 = 16, y_6 = 60$

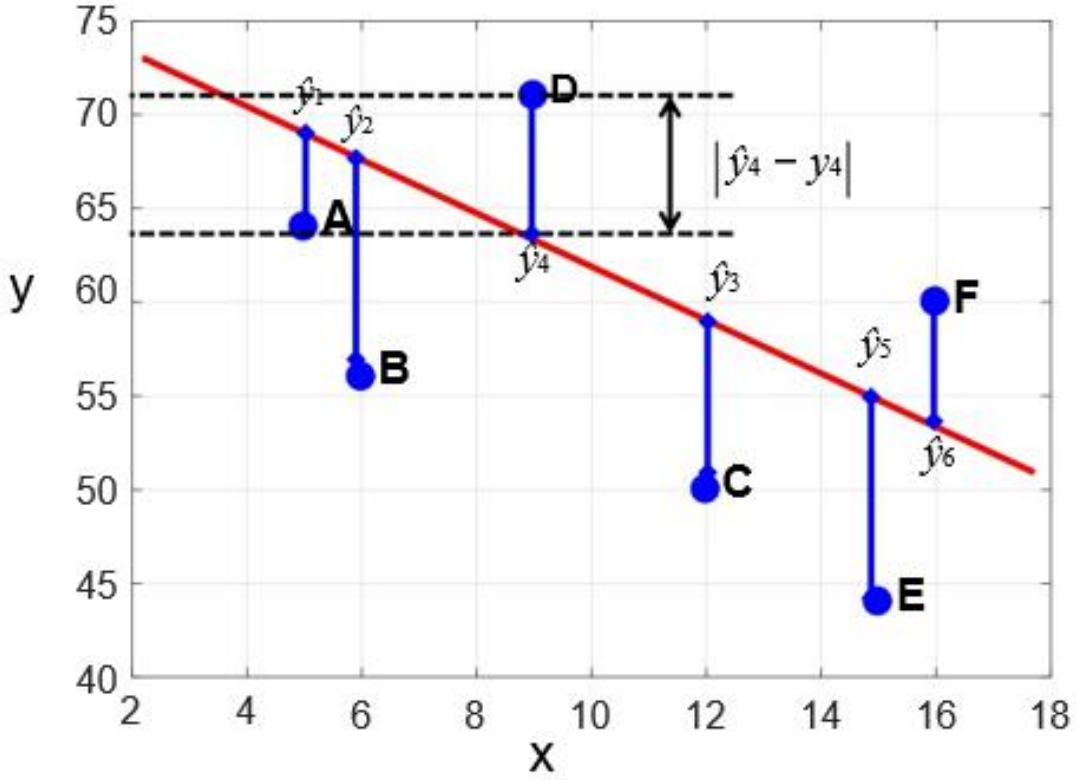
  - We want to minimize this training loss:

$$\min O(w_1, w_0)$$

Machine Learning Basics- Loss Functions

IoTrain
Master of Engineering in Internet of Things

- A Regression Example
  - Geometrically, to minimise this regression error loss enables you to find the best red line to have the shortest blue distances on average.



Machine Learning Basics- Loss Functions

- Linear Least Squares (LLS)
  - To train a linear model by minimising the sum-of-squares error.

- Single-output case:

$$\min O(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2$$

- Multi-output case:

$$\min O(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{c}\left(y_{ij} - \mathbf{w}_j^T \tilde{\mathbf{x}}_i\right)^2$$

Machine Learning Basics- Loss Functions

- Regularised Linear Least Squares
  - A regularisation term can be added to the error function. For instance, in the single-output case, we have

$$\min O_\lambda(\mathbf{w}) = \text{sum of suqares error} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T\tilde{\mathbf{x}}_i\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$\mathbf{w}^T\mathbf{w} = \sum_{j=1}^{d+1} w_j^2$$

  - Other type of regulariser:

λ is a positive real-valued number set by the user.

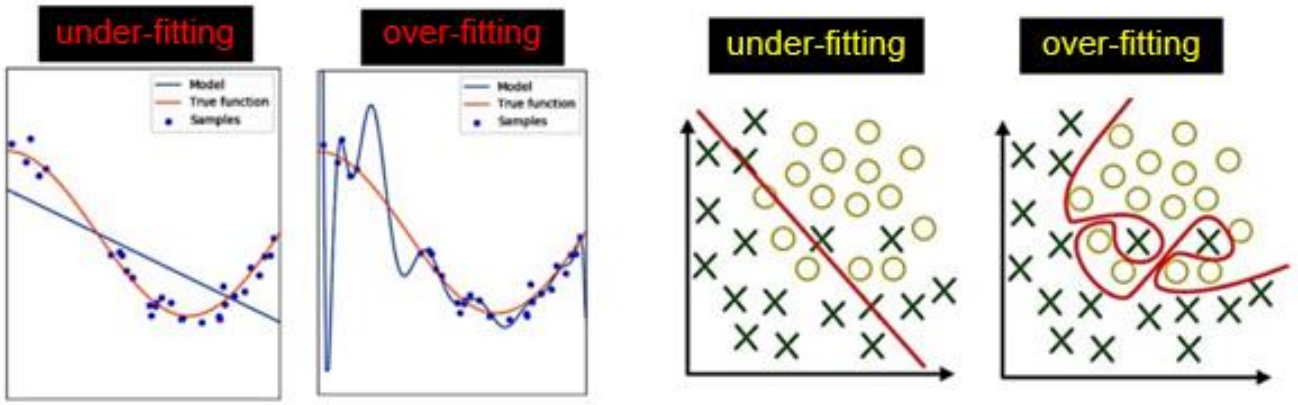
$$\min O_\lambda(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T\tilde{\mathbf{x}}_i\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d+1}|w_j|^q$$

Here q is a positive integer set by the user.
- The case of $q=1$ (l_1-regularisation) for regression is known as lasso.
- The case of $q=2$ (l_2-regularisation) for regression is known as ridge regression.

  - Regularisation prevents the model from over-fitting to training data.
  - When λ is too large, it will lead to under-fitting though.

Machine Learning Basics- Loss Functions

- Why Regularisation?
  - Over-fitting: Fit too closely to a particular set of data (e.g., training data), and may therefore fail to fit new data.
  - Under-fitting: Cannot capture the underlying trend of the training data.
  - Prevent over-fitting, e.g., $O_\lambda(\mathbf{w})$ gives less emphasis to the sum of squares error of the training data.



Machine Learning Basics- Loss Functions

- Loss Functions
  - Probabilistic Regression Losses

Machine Learning Basics- Loss Functions

- Likelihood
  - **Likelihood:** Given the observed data, it is the conditional probability assumed for the observed data given some parameter values.

$$\text{Likelihood}(!!\text{data}) = p(\text{data}!!)$$

  - **Log likelihood:** Take the natural logarithm of the likelihood.

Machine Learning Basics- Loss Functions

- Likelihood Maximization
  - A model can be trained by **maximising** the likelihood (or log likelihood) function of the training samples.
  - Assume independence between samples.
    - **Maximum likelihood estimator** (MLE):

$$\max_{\theta} = \prod_{i=1}^{N} p(\mathbf{x}_i, y_i \,|\, \theta)$$

    - **Log likelihood maximisation**:

$$\max_{\theta} = \sum_{i=1}^{N} \log p(\mathbf{x}_i, y_i \,|\, \theta)$$

Machine Learning Basics- Loss Functions

- Example: MLE for Linear Regression
  - Likelihood of N training samples:

$$L = \prod_{i=1}^{N} N\left(y_i \middle| \mathbf{w}^T \tilde{\mathbf{x}}_i, \sigma^2\right)$$

Assume a Gaussian distribution for likelihood estimation.

$$p(\text{data}_i \mid \theta) = p(y_i \mid \theta) = N\left(y_i \mid \mathbf{w}^T \tilde{\mathbf{x}}_i, \sigma^2\right)$$

  - Log-likelihood:

$$O = \ln(L) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta\frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i\right)^2$$

where $\beta^{-1} = \sigma^2$

Sum-of-squares error function!

  - In this case, an MLE is equivalent to a sum-of-squares error minimizer.

Machine Learning Basics- Loss Functions

- Loss Functions
  - Typical approaches of constructing losses for **classification**
    - Non-probabilistic losses
    - Probabilistic losses

Machine Learning Basics- Loss Functions

- Loss Functions for Classification
  - Non-probabilistic losses

Machine Learning Basics- Loss Functions

- Losses for Training Classification Models

Samples: $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$

Feature vector: $\mathbf{x}_i \in R^d$

Class label: $y_i \in \{1,2,\ldots,c\}$

Machine Learning Basics- Loss Functions

- Sum-of-squares Loss for Classification
  - You can train a discriminant function by minimising the sum-of-squares loss.
  - This results in the least squares approach for classification.

Machine Learning Basics- Loss Functions

- Least Squares for Binary Classification

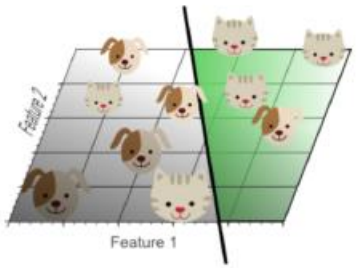
  - Binary classification: **+1/-1 label coding**

    y=1 class A, *y=-1* class B

  - Discriminant function: $\hat{y} = \begin{cases} 1, & f(\mathbf{x}) \geq 0 \\ -1, & f(\mathbf{x}) < 0 \end{cases}$

  - Minimise sum-of-squares loss in training (*N* training samples):

$$O(\theta) = \frac{1}{2}\sum_{i=1}^{N}\left(f(\mathbf{x}_i) - y_i\right)^2$$



Machine Learning Basics- Loss Functions

- Least Squares for Multi-class Classification
  - Multi-class classification: **+1/-1 label coding**
    - y_{ij}=*1: The i-th sample is* from the *j*-th class.
    - y_{ij}=*-1: The i-th sample is* ***NOT*** from the *j*-th class.
  - Discriminant function for the *j*-th class:

$$\hat{y}_j = \begin{cases} 1, & f_j(\mathbf{x}) \geq 0 \\ -1, & f_j(\mathbf{x}) < 0 \end{cases}$$

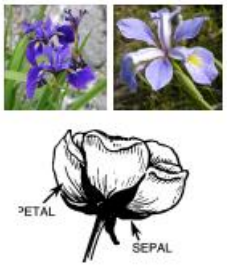
  - Minimise sum-of-squares loss in training (*N* training samples):

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{c}\left(y_{ij} - f_j\left(\mathbf{x}_i\right)\right)^2$$

Machine Learning Basics-Loss Functions

IoTrain
Master of Engineering in Internet of Things

- Example: Iris Classification



Iris Data: https://archive.ics.uci.edu/ml/datasets/iris

Measured sepal length and petal width of 150 samples of irises belonging to two types: versicolour and virginica. Predict the flower type of a query iris sample.

60 training samples with 30 samples from each class.

$\mathbf{X}_{tr}$ 60x2 double

	1	2
1	5.7000	1
2	5.5000	1.1000
3	5.5000	1.3000
4	7	1.4000
5	6.3000	2.4000
6	6.5000	1.5000
7	6.9000	2.3000
8	6.7000	2.3000
9	5.6000	1.1000
10	6.7000	2.1000
11	7.4000	1.9000
12	6.5000	2
13	6	1.6000
14	6.8000	1.4000
15	6.7000	1.7000
16	5.6000	1.3000
17	6	1.6000
18	7.3000	1.8000
19	7.6000	2.1000
20	5.1000	1.1000

$\mathbf{Y}_{tr}$ 60x1 double

	1
1	-1
2	-1
3	-1
4	-1
5	1
6	-1
7	1
8	1
9	-1
10	1
11	1
12	1
13	-1
14	-1
15	-1
16	-1
17	-1
18	1
19	1
20	-1

```
classID =
    '1:Virginica'    '-1:Versicolour'
>> featureID
featureID =
    'sepal_length'    'petal_width'
```

Prediction model:

$$f = w_0 + w_1 x_1 + w_2 x_2$$

$$\hat{y} = \begin{cases} \text{Virginica} & \text{if } f \geq 0 \\ \text{Versicolour} & \text{if } f < 0 \end{cases}$$

Training:

$$\min_{w_0, w_1, w_2} \frac{1}{2} \sum_{i=1}^{N} \left(f(\mathbf{x}_i) - y_i \right)^2$$

Machine Learning Basics- Loss Functions

- A More General Setting
  - **Thresholding by T** the discriminant function.

$$\hat{y} = \begin{cases} \text{class 1}, & f \geq T, \\ \text{class 2}, & f < T. \end{cases}$$

Usual setup: $T = \frac{a+b}{2}$

  - **a/b label coding:**
    - Binary classification: single-output $y \in \{a, b\}$
    - Multiclass classification: multi-output $\mathbf{y} = [y_1, y_2, \ldots y_c]$, where $y_i \in \{a, b\}$

Binary:
- A number *a* indicates the input is from class 1.
- A number *b* indicates the input is from class 2.

Multi-class:
- A number *a* indicates the input is not from class i.
- A number *b* indicates the input is from class i.

Machine Learning Basics- Loss Functions

- A More General Setting

  - Tips on setting:
    - Theoretically, you can choose any number for *a* and *b*, as long as *a* is not equal to *b*.
    - In reality, for some data, the performance can be sensitive to the choice of *a, b*.
    - Usually, we use $\boxed{\begin{matrix} a=+1 \\ b=-1 \end{matrix}}$ or $\boxed{\begin{matrix} a=1 \\ b=0 \end{matrix}}$
    - Usually, we use fixed threshold as $T=\frac{a+b}{2}$. For example, a=+1, b=-1, T=0; or a=1, b=0. T=0.5.
    - Sometimes, the threshold T is treated as a hyperparameter to tune.

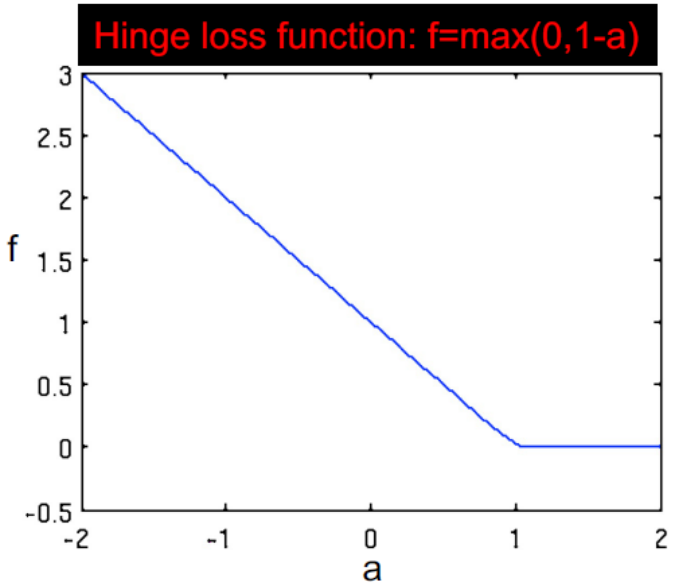
Machine Learning Basics-Loss Functions

- Hinge Loss for Classification
  - Hinge loss assesses classification error.
  - Given **+1/-1 label coding**, hinge loss over N training samples is

$$O(\theta) = \sum_{i=1}^{N} \max\left(0, 1 - y_i f\left(\theta, x_i\right)\right)$$

class label $y_i \in \{-1, +1\}$

your prediction model: $f\left(\theta, x_i\right)$



Machine Learning Basics- Loss Functions

- Hinge Loss for Classification
  - Hinge loss assesses classification error.
  - Given **+1/-1 label coding**, hinge loss over N training samples is

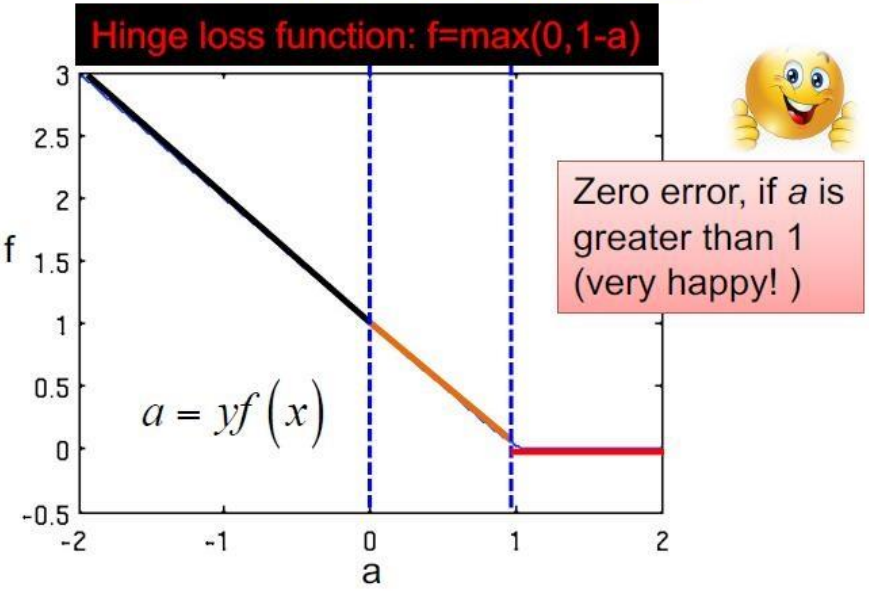
$$O(\theta) = \sum_{i=1}^{N} \max\left(0, 1 - y_i f\left(\theta, x_i\right)\right)$$

class label $y_i \in \{-1, +1\}$

your prediction model: $f\left(\theta, x_i\right)$

Small error, if *a* is less than 1 but still positive (not too bad…)

Large error, if *a* is negative (very unhappy!)



Zero error, if *a* is greater than 1 (very happy!)

Machine Learning Basics- Loss Functions

- Hinge Loss for Classification

$$O(\theta) = \sum_{i=1}^{N} \max\left(0, 1 - y_i f\left(\theta, x_i\right)\right)$$

- Let *f* be a linear model and add a regularisation term to the loss. This results in **support vector machine**.

$$\min_{(\mathbf{w}, w_0) \in \Re^{d+1}} C \sum_{i=1}^{N} \max\left(0, 1 - y_i\left(\mathbf{w}^T \mathbf{x}_i + w_0\right)\right) + \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

regularisation parameter (hyper-parameter)

hinge loss

regularisation term

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

Machine Learning Basics- Loss Functions

- Example: Hinge Loss for Binary Classification
  - A linear basis function model is trained to return the following prediction for the 3 samples as below:

A	y	f
x_1	1	10
x_2	1	0.9
x_3	-1	0.1

$$O(\theta) = \sum_{i=1}^{N} \max\left(0, 1 - y_i f\left(\theta, x_i\right)\right)$$

  - Compute this model's hinge loss using these 3 samples.

$$O = \max(0, 1 - 1 \times 10) + \max(0, 1 - 1 \times 0.9) + \max(0, 1 - (-1) \times 0.1)$$
$$= 0 + 0.1 + 1.1 = 1.2$$

Machine Learning Basics- Loss Functions

- Loss Functions for Classification

  - Probabilistic losses
    - Cross entropy loss based on class posterior. p(class k!**x**)

Machine Learning Basics- Loss Functions

- Cross Entropy
  - Cross entropy measures distance between probability distributions.
  - Its discrete version can be used to examine the distance between the predicted class probabilities (posterior) and the true probabilities.

$$H(p,q) = -\left[p(1)\log\left(q(1)\right) + p(0)\log\left(q(0)\right)\right] \text{ (two classes)}$$

$$H(p,q) = -\left[p(1)\log\left(q(1)\right) + p(2)\log\left(q(2)\right) + \cdots + p(c)\log\left(q(c)\right)\right] \text{ (multiple classes)}$$

Machine Learning Basics- Loss Functions

- Cross Entropy Loss

- **Binary classification**: $H(p,q) = -\left[p(1)\log(q(1)) + p(0)\log(q(0))\right]$
- **0/1 label coding** for a sample (***x***, *y*).
  - If y=1, x is from class 1, which means *p(1) =100%* and *p(0) =0%.*
  - If y=0, x is from class 0, which means *p(1) =0%* and *p(0) = 100%.*
  - Therefore, ***p(1) =y*** and ***p(0) =1-y***.
- Cross entropy loss computed over N training samples is

$$O = -\sum_{i=1}^{N}\left[y_i \log_b\left(p\left(c_1|\mathbf{x}_i\right)\right) + \left(1-y_i\right)\log_b\left(p\left(c_2|\mathbf{x}_i\right)\right)\right]$$

You can use natural log (*ln, b=e*) or log base 2 (*b=2*).

Machine Learning Basics- Loss Functions

- Cross Entropy Loss

- **Multi-class classification**

$y_{ik}=1$ means that the i-th sample belongs to class k, $y_{ik}=0$ otherwise.

  - 1-of-K label coding scheme: $\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1c} \\ y_{21} & y_{22} & \cdots & y_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nc} \end{bmatrix}$, where $y_{ik} \in \{0,1\}$

  - Cross entropy loss computed over N training samples is

$$O = -\sum_{i=1}^{N}\sum_{k=1}^{c} y_{ik} \log_b \left(p\left(c_k \middle| \mathbf{x}_i\right)\right)$$

Machine Learning Basics- Loss Functions

- Cross Entropy
  - Different models give you different ways to formulate $p\left(c_k \middle| \mathbf{x}\right)$
  - Logistic regression: A linear classification model trained using cross-entropy loss.

Machine Learning Basics- Loss Functions

- Loss Functions for Classification

  - Classification Losses based on likelihood.

Machine Learning Basics- Loss Functions

- Likelihood Maximization for Classification
  - One way to train the model is to maximise the likelihood (or log likelihood) function.
    - Maximum likelihood estimator (MLE):

$$\max_{\theta} p(\text{data} \mid \theta)$$

    - Often it is more convenient to use log likelihood:

$$\max_{\theta} \log p(\text{data} \mid \theta)$$

Machine Learning Basics- Loss Functions

- Assumption on Class Label Distribution
  - Binary classification: Assume the class label follows Bernoulli distribution.

$$p\left(y\middle|\theta\right)=\theta^{y}\left(1-\theta\right)^{1-y}=\begin{cases}\theta, & \text{if } y=1,\\ 1-\theta, & \text{if } y=0.\end{cases}$$

  - Multi-class classification: Assume the class label follows categorical (multinomial) distribution.

$$p\left(\mathbf{y}\middle|\theta_1,\theta_2,\ldots\theta_c\right)=\prod_{k=1}^{c}\theta_k^{y_k}$$

1-of-K coding scheme:

$\mathbf{y}=\left[y_1,y_2,\ldots y_k\right]^T$

$y_k=1$ if the sample is from class k

$y_k=0$ otherwise

Machine Learning Basics- Loss Functions

- Likelihood of An Individual Sample
  - Consider the *i*-th training sample ($\mathbf{x}_i$, y_i)

- **Binary classification**:

$$p(\mathbf{x}_i, y_i \mid \theta) = \theta(\mathbf{x}_i)^{y_i}\left(1 - \theta(\mathbf{x}_i)\right)^{1-y_i}$$

- **Multi-class classification**: Assume the class label follows categorical (multinomial) distribution (generalise Bernoulli distribution to more than two options):

$$p\left(\mathbf{x}_i, \mathbf{y}_i \mid \theta\right) = \prod_{k=1}^{c} \theta_k(\mathbf{x}_i)^{y_{ik}}$$

Machine Learning Basics- Loss Functions

- Likelihood of N Training Samples
  - Given N training samples and assume sample independence.

    - **Binary classification**:

$$L = \prod_{i=1}^{N} p(\mathbf{x}_i, y_i \mid \theta) = \prod_{i=1}^{N} \theta(\mathbf{x}_i)^{y_i} \left(1 - \theta(\mathbf{x}_i)\right)^{1-y_i}$$

    - **Multi-class classification**:

$$L = \prod_{i=1}^{N} p\left(\mathbf{x}_i, \mathbf{y}_i \mid \theta\right) = \prod_{i=1}^{N} \prod_{k=1}^{c} \theta_k(\mathbf{x}_i)^{y_{ik}}$$

Machine Learning Basics- Loss Functions

- Example
  - You can model your θ using a linear model.
    - Binary classification: Apply logistic sigmoid function to a linear model.

$$\theta(\mathbf{x}) = \frac{1}{1+\exp\left(-\mathbf{w}^T\tilde{\mathbf{x}}\right)}$$

    - Multi-class classification: Apply softmax function to a linear model.

$$\theta_k(\mathbf{x}) = \frac{\exp\left(\mathbf{w}_k^T\tilde{\mathbf{x}}\right)}{\sum_{j=1}^{c}\exp\left(\mathbf{w}_j^T\tilde{\mathbf{x}}\right)} \quad ,k = 1,2,\ldots c$$

This gives you again logistic regression.

Machine Learning Basics- Loss Functions

- Negative Log Likelihood Loss
  - Classification loss: negative log likelihood.
  - Negative log likelihood loss is equal to the cross-entropy loss, if

$$\theta(\mathbf{x}) = p(c_1|\mathbf{x})$$

$$\theta_k(\mathbf{x}) = p(c_k|\mathbf{x})$$

Machine Learning Basics- Loss Functions

- Example 1: Cross-Entropy Loss for Binary Classification
  - A logistic regression model returns the following posterior class probabilities for the 3 samples as below:

A	y	$p(c_1\|x)$	$p(c_2\|x)$
x_1	1	0.8	0.2
x_2	1	0.9	0.1
x_3	0	0.2	0.8

$$O = -\sum_{i=1}^{N}\left[y_i \log_b\left(p\left(c_1|\mathbf{x}_i\right)\right) + \left(1-y_i\right)\log_b\left(p\left(c_2|\mathbf{x}_i\right)\right)\right]$$

  - Compute this model's cross-entropy loss using these 3 samples.

$$O_A = -\left(1\times\ln 0.8 + 0\times\ln 0.2\right) - \left(1\times\ln 0.9 + 0\times\log 0.1\right) - \left(0\times\ln 0.2 + 1\times\ln 0.8\right)$$
$$= -\left(\ln 0.8 + \ln 0.9 + \ln 0.8\right) = 0.55$$

Machine Learning Basics- Loss Functions

- Example 2: Negative log likelihood Loss for Multi-class Classification
  - A 3-class classification model is trained by MLE assuming categorical distribution. The ground truth labels and estimated theta function for the following 4 samples are provided:

	y	$\Theta_1(x)$	$\Theta_2(x)$	$\Theta_3(x)$
x_1	1	0.7	0.2	0.1
x_2	3	0.5	0.3	0.2
x_3	3	0.1	0.1	0.8
x_4	2	0.3	0.6	0.1

$$L = \prod_{i=1}^{N} \prod_{k=1}^{c} \theta_k \left(\mathbf{x}_i \right)^{y_{ik}}$$

  - Compute this model's Negative log likelihood loss using these 4 samples.

$$L = \left(0.7^1 \times 0.2^0 \times 0.1^0\right) \times \left(0.5^0 \times 0.3^0 \times 0.2^1\right) \times \left(0.1^0 \times 0.1^0 \times 0.8^1\right) \times \left(0.3^0 \times 0.6^1 \times 0.1^0\right)$$

$$= 0.7 \times 0.2 \times 0.8 \times 0.6 = 0.0672$$

$$-\ln(L) = 2.7$$

Machine Learning Basics- Loss Functions

- Summary
  - Regression losses:
    - Sum of squares error
    - Mean squared error
  - Classification losses:
    - Sum of squares error
    - Hinge loss
    - Cross entropy loss
    - Likelihood and log likelihood based
  - Linear least squares (LLS) approach for classification and regression
  - Regularization, regularized LLS

Content

- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

Supervised Learning: Decision Trees

- Decision Trees
  - A decision tree is a tree with the following properties:
    - An inner node represents an attribute
    - An edge represents a test on the attribute of the father node
    - A leaf represents one of the classes
  - Construction of a decision tree
    - Based on the training data
    - Top-Down strategy

Supervised Learning: Decision Trees

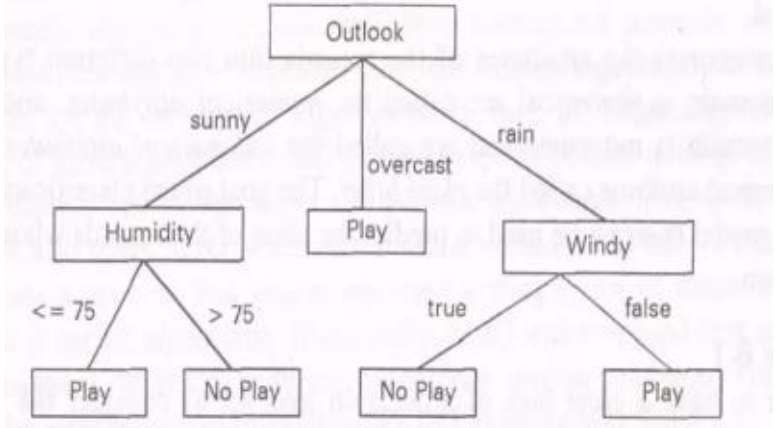
- Example
  - The data set has five attributes.
  - There is a special attribute: the attribute *class* is the class label.
  - The attributes, *temp* and *humidity* are numerical attributes.
  - Other attributes are categorical, that is, they cannot be ordered.
  - Based on the training data set, we want to find a set of rules to know what values of *outlook, temperature*, *humidity* and *wind*, determine whether or not to play golf.

Training Data Set

OUTLOOK	TEMP(F)	HUMIDITY(%)	WINDY	CLASS
sunny	79	90	true	no play
sunny	56	70	false	play
sunny	79	75	true	play
sunny	60	90	true	no play
overcast	88	88	false	no play
overcast	63	75	true	play
overcast	88	95	false	play
rain	78	60	false	play
rain	66	70	false	no play
rain	68	60	true	no play

Supervised Learning: Decision Trees

- Example
  - We have five leaf nodes.
  - In a decision tree, each leaf node represents a rule.
  - Rules:
    - Rule 1: If it is sunny and the humidity is not above 75%, then play.
    - Rule 2: If it is sunny and the humidity is above 75%, then do not play.
    - Rule 3: If it is overcast, then play.
    - Rule 4: If it is rainy and not windy, then play.
    - Rule 5: If it is rainy and windy, then don't play.



Supervised Learning: Decision Trees

- Classification
  - The classification of an unknown input vector is done by traversing the tree from the root node to a leaf node.
  - A record enters the tree at the root node.
  - At the root, a test is applied to determine which child node the record will encounter next.
  - This process is repeated until the record arrives at a leaf node.
  - All the records that end up at a given leaf of the tree are classified in the same way.
  - There is a unique path from the root to each leaf.
  - The path is a rule which is used to classify the records.

Supervised Learning: Decision Trees

- Classification
  - The *accuracy* of the classifier is determined by the percentage of the test data set that is correctly classified.
  - We can see that for *Rule 1* there are two records of the test data set satisfying *outlook=sunny* and *humidity<75*, and only one of these is correctly classified as play.
  - Thus, the accuracy of this rule is 0.5. Similarly, the accuracy of Rule 2 is also 0.5. The accuracy of Rule 3 is 0.66.

Supervised Learning: Decision Trees

- Iterative Dichotomizer (ID3)
  - Quinlan (1986)
  - Each node corresponds to a splitting attribute.
  - Each arc is a possible value of that attribute.
  - At each node the splitting attribute is selected to be the most informative among the attributes not yet considered in the path from the root.
  - *Entropy* is used to measure how informative is a node.
  - The algorithm uses the criterion of *information gain* to determine the goodness of a split.
    - The attribute with the greatest information gain is taken as the splitting attribute, and the data set is split for all distinct values of the attribute.

Supervised Learning: Decision Trees

- Algorithm for Decision Tree Induction
  - Basic algorithm (a greedy algorithm)
    - Tree is constructed in *a top-down recursive divide-and-conquer manner*.
    - At start, all the training examples are at the root.
    - Attributes are categorical (if continuous-valued, they are discretized in advance)
    - Examples are partitioned recursively based on selected attributes
    - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., *information gain*)
  - Conditions for stopping partitioning
    - All samples for a given node belong to the same class.
    - There are no remaining attributes for further partitioning – *majority voting* is employed for classifying the leaf.
    - There are no samples left.

Supervised Learning: Decision Trees

- Attribute Selection Measure: Information Gain (ID3/C4.5)
  - Select the attribute with the highest information gain.
  - S contains s_i tuples of class C_i for i = {1,…,m}
  - *information* measures info required to classify any arbitrary tuple

$$I(s_1,s_2,\ldots,s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s} \quad \ldots\ldots \textit{information is encoded in bits.}$$

  - *entropy* of attribute A with values $\{a_1,a_2,\ldots,a_v\}$

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j}+\ldots+s_{mj}}{s} I(s_{1j},\ldots,s_{mj})$$

  - *information gained* by branching on attribute A

$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)$$

Supervised Learning: Decision Trees

- Attribute Selection Measure: Information Gain (ID3/C4.5) - Example

  - Class P: buys_computer = "yes"
  - Class N: buys_computer = "no"
  - I(p, n) = I(9, 5) =0.940
  - Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes's and 3 no's. Hence

$$Gain(age) = I(p,n) - E(age) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Since, age has the highest information gain among the attributes, it is selected as the test attribute.

Supervised Learning: Decision Trees

- Advantages and shortcomings of decision tree classification
  - A decision tree construction process is concerned with identifying the splitting attributes and splitting criterion at every level of the tree.
  - Major strengths are:
    - Decision tree able to generate understandable rules.
    - They are able to handle both numerical and categorical attributes.
    - They provide clear indication of which fields are most important for prediction or classification.
  - Weaknesses are:
    - The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field is examined before its best split can be found.
    - Some decision tree can only deal with binary-valued target classes.

Supervised Learning: k-NN

- K-nearest neighbour (k-NN)
  - History k-NN
  - k-NN classification
  - k-NN regression
  - Instance-based learning

k-NN

- History
  - k-NN is the simplest of all machine learning algorithms.
  - The philosophy behind k-NN dates back very early (965-1040) by a scientist in the optics and perception.
  - An interesting article on k-NN history: http://37steps.com/4370/nn-rule-invention/
  - Early academic works on k-NN start in 1960s:
    - G. S. Sebestyen, Decision-making processes in pattern recognition, First edition, 1962.(known as the proximity algorithms)
    - N. Nilsson, Learning machines: foundations of trainable pattern classifying systems, First edition, 1965. (known as the minimum distance classifier)
    - T. Cover and P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory, 3(1):21-27, 1967.

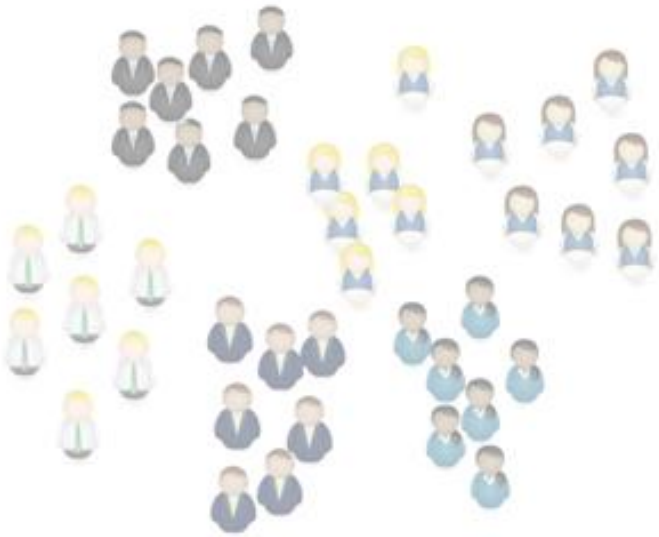
k-NN

- k-NN Classification

Example data: $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$

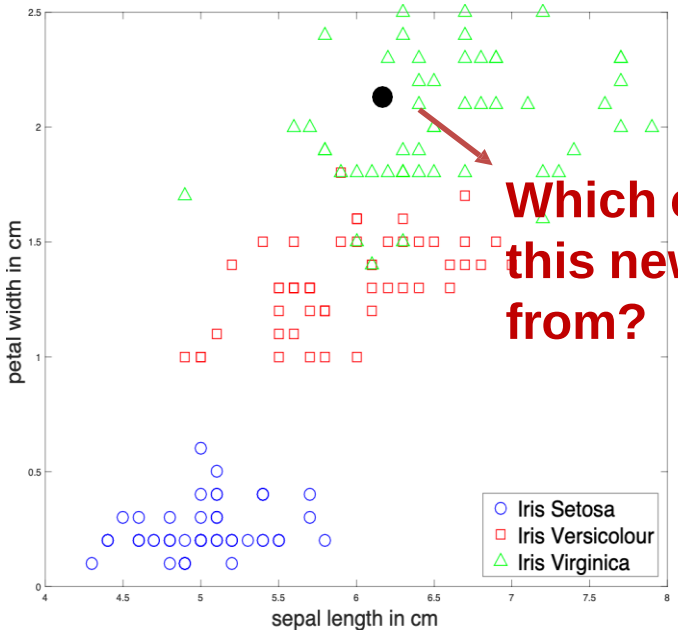
Feature vector: $\mathbf{x}_i \in R^d$

Class label: $y_i \in \{1,2,\ldots,c\}$



k-NN

- Iris Classification
  - Iris Data: https://archive.ics.uci.edu/ml/datasets/iris
    - The dataset contains 150 iris samples from 3 classes (setosa, versicolour and virginica), each characterised by its sepal length and petal width measured.
    - Predict the flower type of a query iris sample.



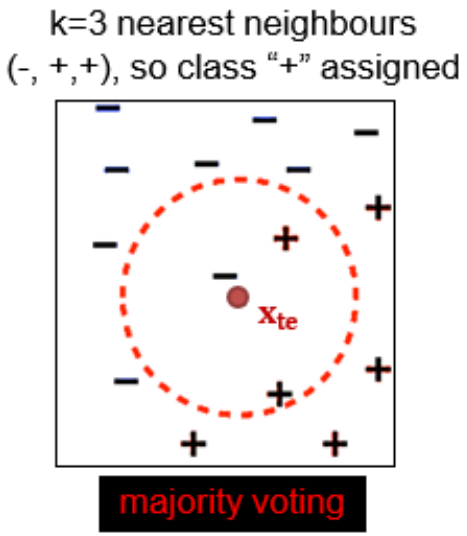
Which class is this new sample from?

150 samples
2-dimensional feature vector
3 classes

k-NN

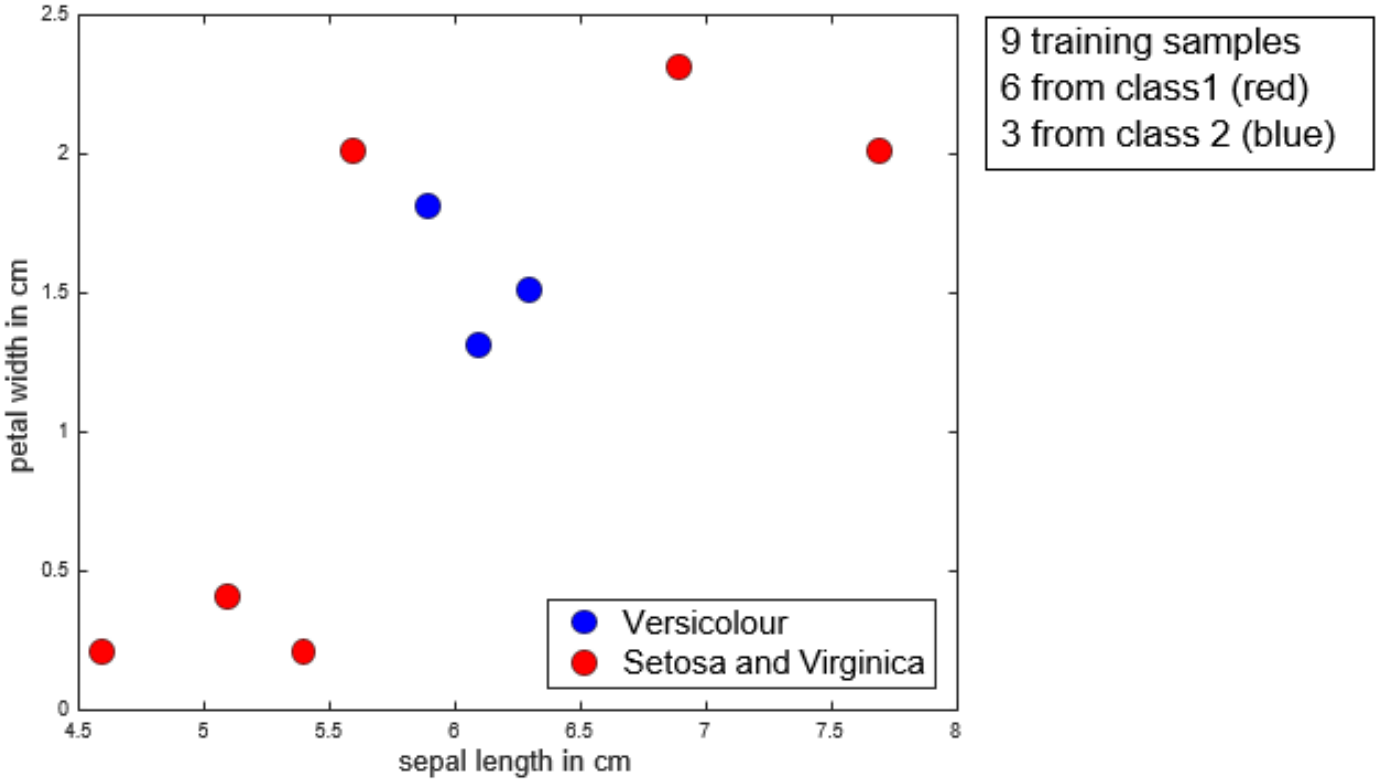
- k-NN Classification Rule
  - Given a set of training samples {features, class label}.
  - Each sample corresponds to a data point in the feature space.
  - k-NN classification rule:

Testing point x_{te}
For each training data point x_{tr}
measure distance(x_{te}, x_{tr})
End
Sort distances
Select k nearest points
Assign most common class



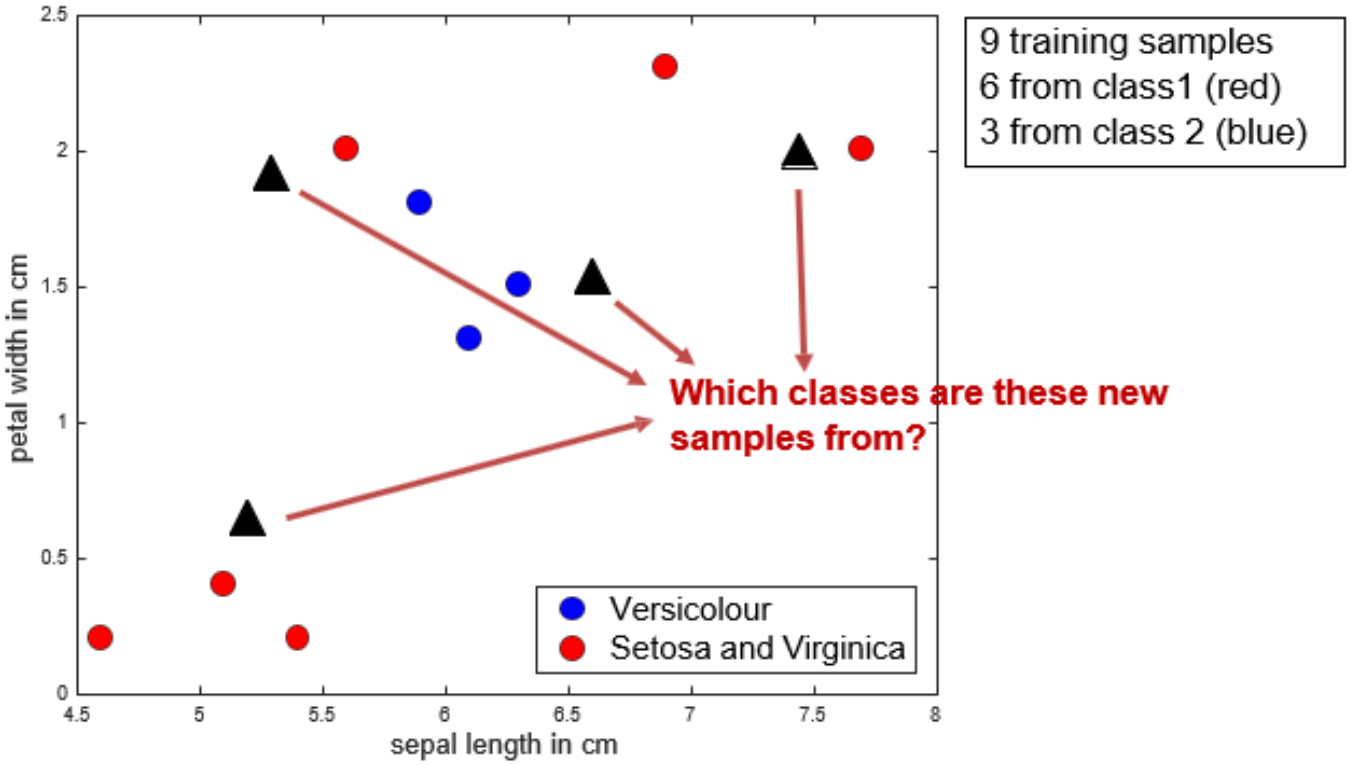
k-NN

- Example: 1-NN for Binary Classification



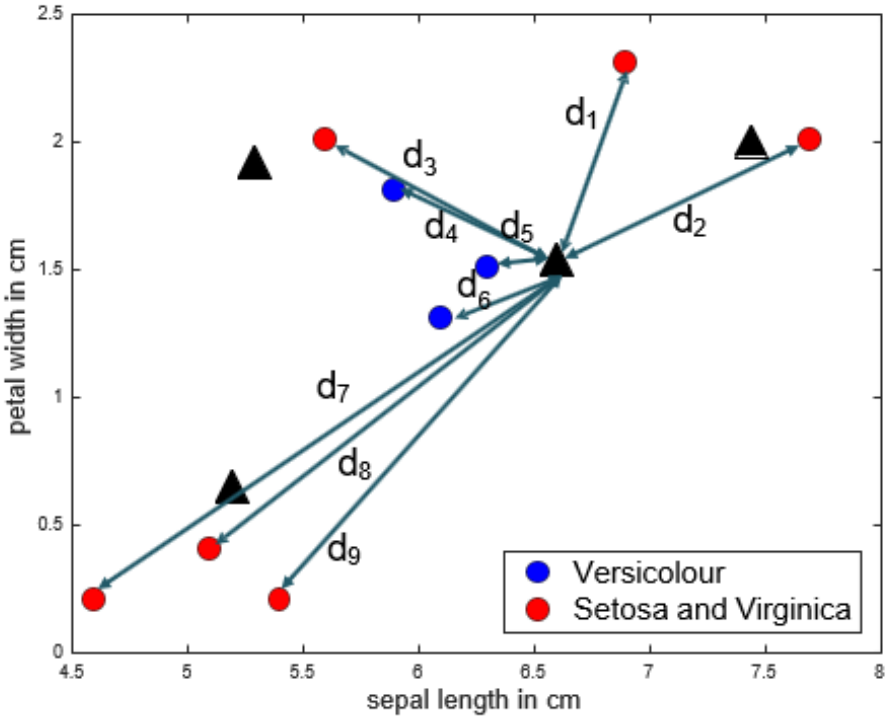
k-NN

- Example: 1-NN for Binary Classification



k-NN

- Example: 1-NN for Binary Classification

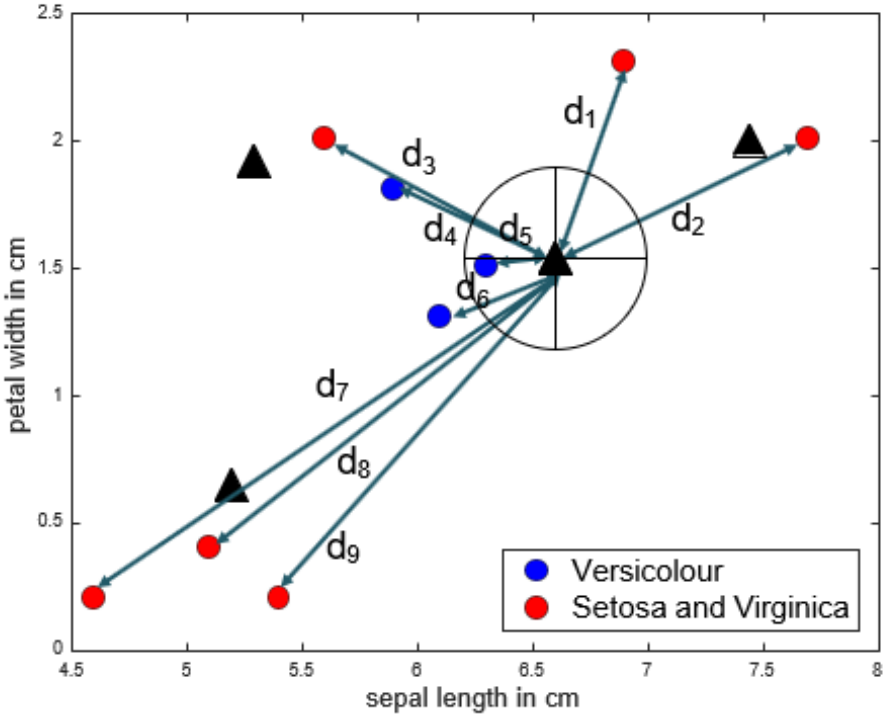


9 training samples
6 from class1 (red)
3 from class 2 (blue)

Step 1: Calculate distances to the 9 training samples.

k-NN

- Example: 1-NN for Binary Classification



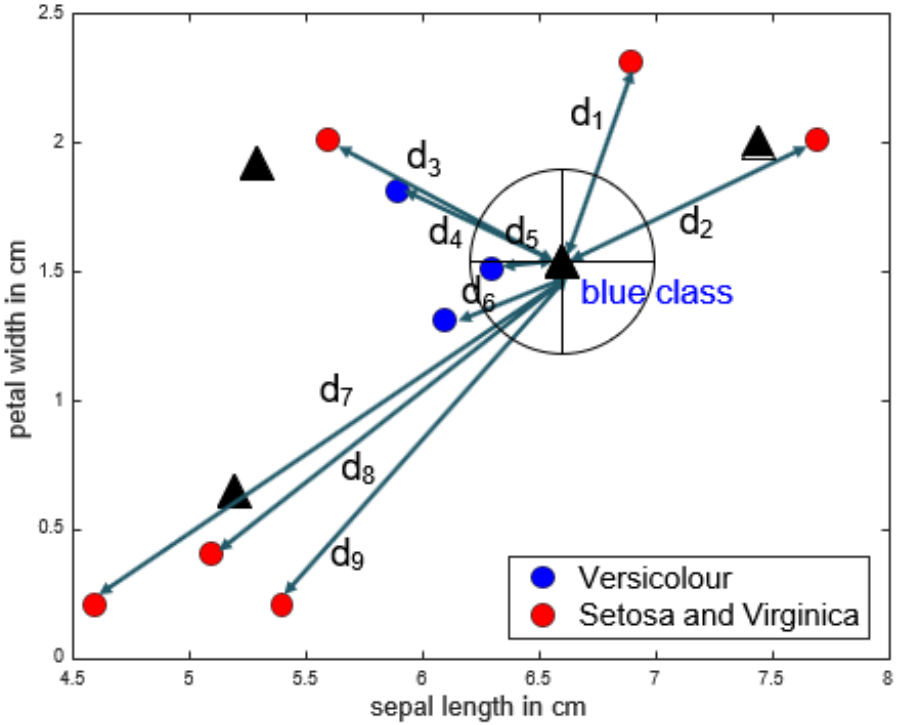
9 training samples
6 from class1 (red)
3 from class 2 (blue)

Step 1: Calculate distances to the 9 training samples.

Step 2: Sort the distances and find the nearest neighbour (k=1).

k-NN

- Example: 1-NN for Binary Classification



9 training samples
6 from class1 (red)
3 from class 2 (blue)

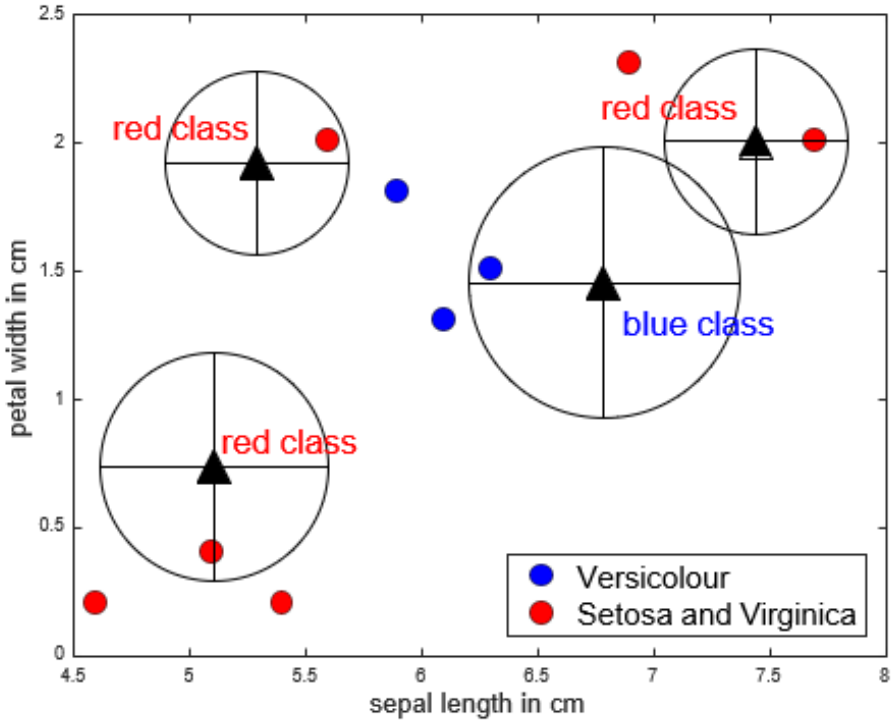
Step 1: Calculate distances to the 9 training samples.

Step 2: Sort the distances and find the nearest neighbour (k=1).

Step 3: Assign class.

k-NN

- Example: 1-NN for Binary Classification



9 training samples
6 from class1 (red)
3 from class 2 (blue)

Do these for other query samples.

k-NN

- k-NN Regression

Example data: $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{n}$

Feature vector: $\mathbf{x}_i \in R^d$

Target output: $\mathbf{y}_i \in R^k$

k-NN

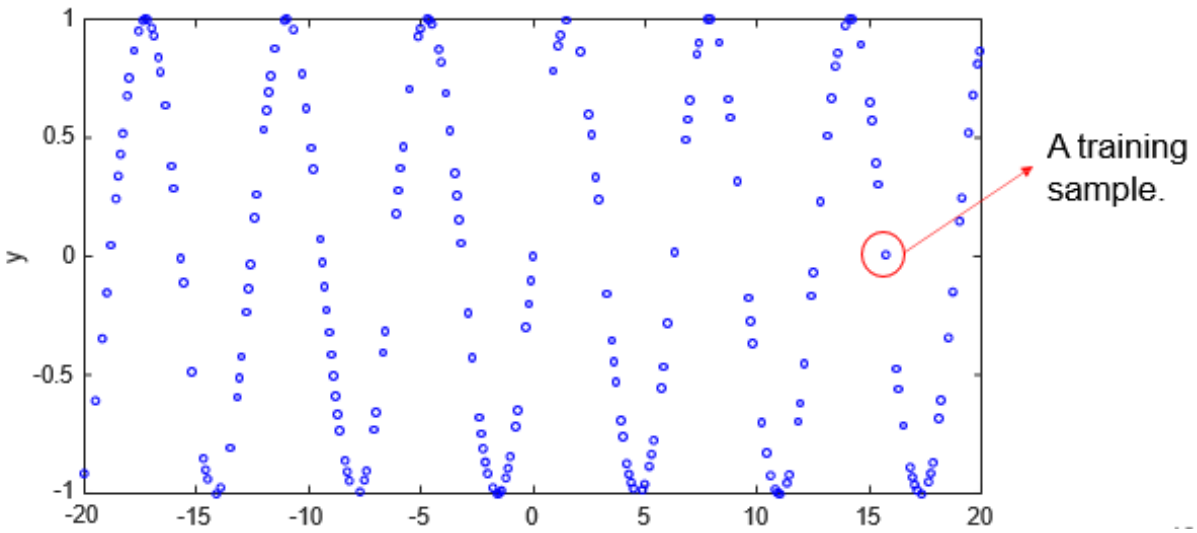
- k-NN Regression Rule
  - Given a set of training samples {features, output}.
  - Each sample corresponds to a data point in the feature space.
  - k-NN regression rule:

Testing point x_{te}
For each training data point x_{tr}
measure distance(x_{te}, x_{tr})
End
Sort distances
Select k nearest points: x_{NN1}, x_{NN2}, ...x_{NNk}
Calculate y_{te} by averaging the output of these nearest points: $y_{te} = (y_{NN1} + y_{NN2} + ... + y_{NNk})/k$

k-NN

- Regression Example 1
  - Training samples: 200 input-output pairs (blue circles).
  - Predict the single-output y from the single-input x.

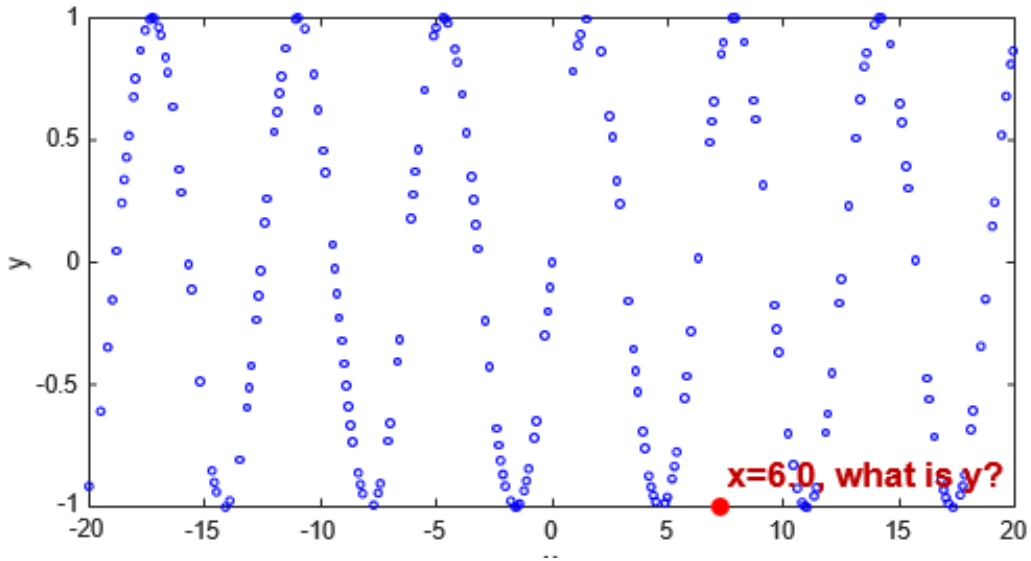
Each training sample is generated by first choosing a random input x between -20 and 20, and then calculating the output by y=sin(x).



k-NN

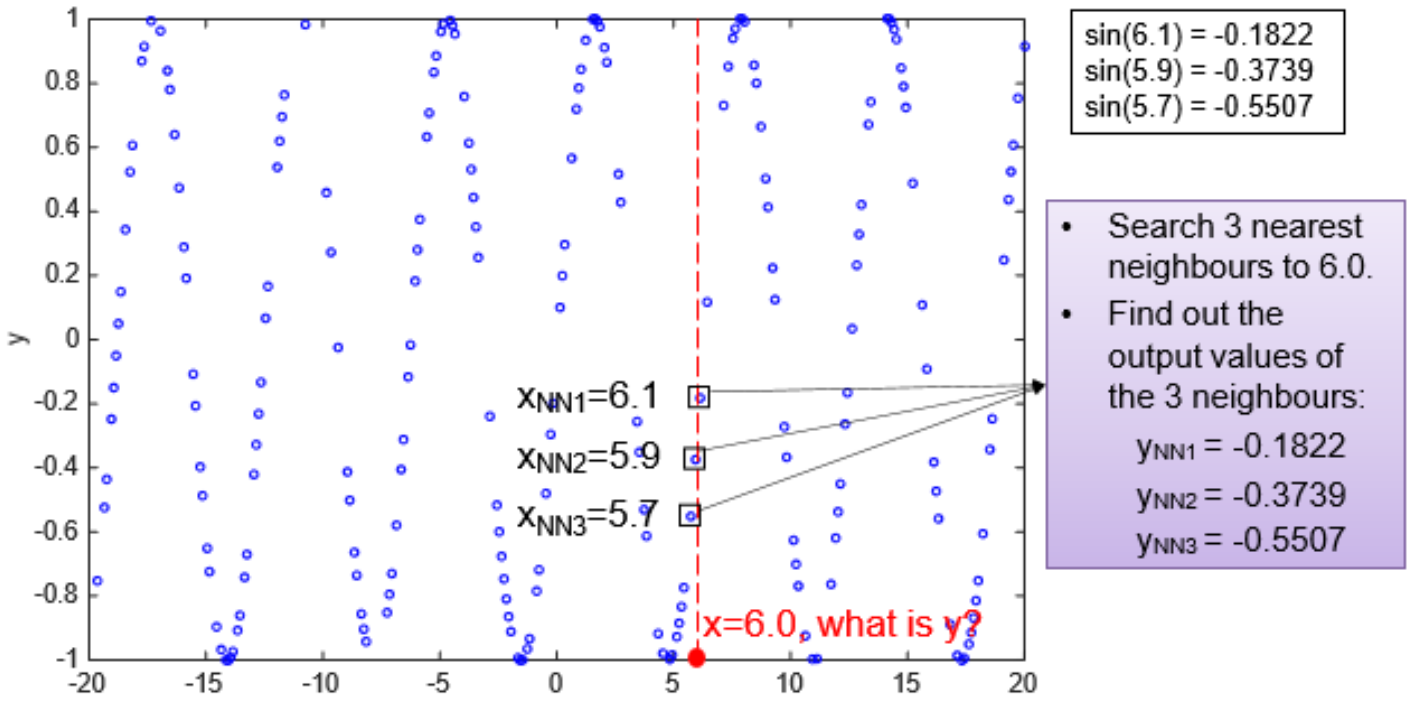
- Regression Example 1
  - Training samples: 200 input-output pairs (blue circles).
  - Predict the single-output y from the single-input x.

Each training sample is generated by first choosing a random input x, and then calculating the output by y=sin(x).



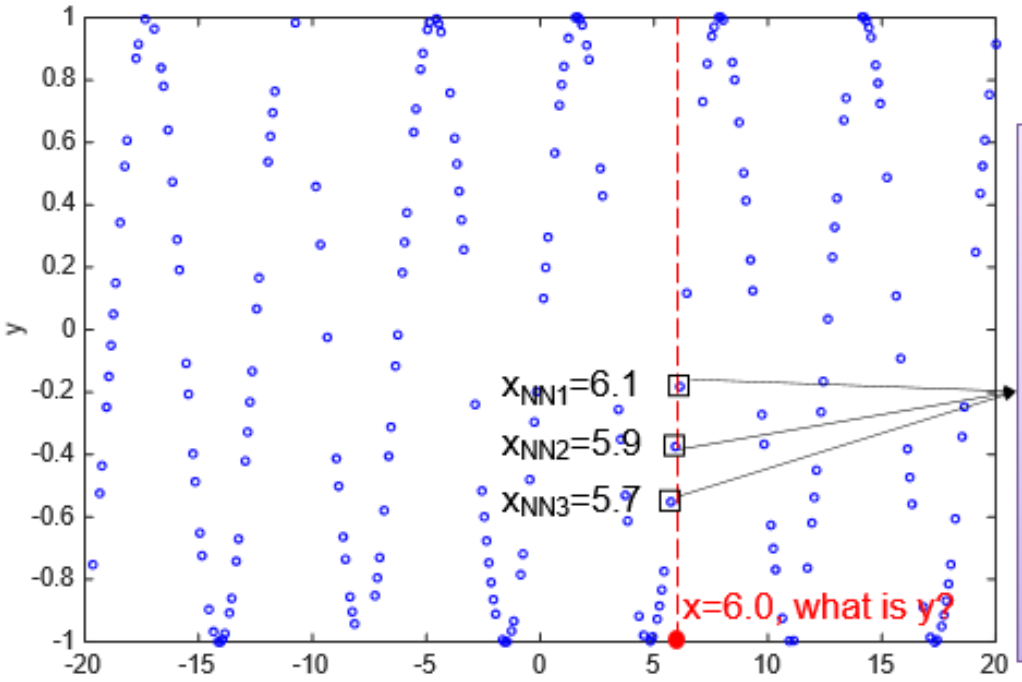
k-NN

- Regression Example 1
  - Build a 3-NN regression model



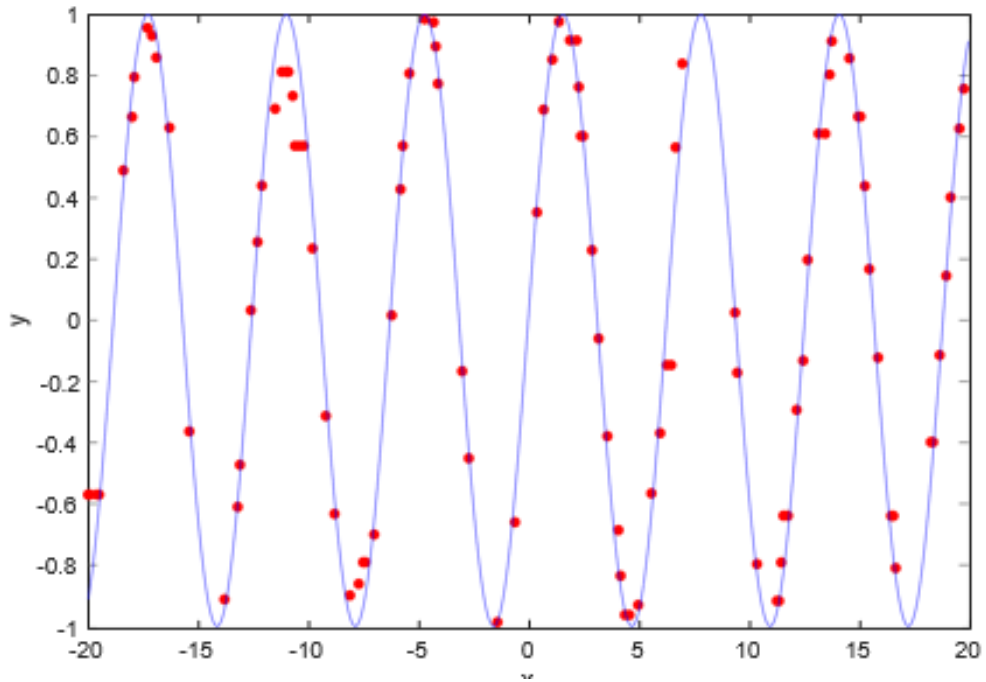
k-NN

- Regression Example 1
  - Build a 3-NN regression model



k-NN

- Regression Example 1
  - Build a 3-NN regression model



Predict for new samples: red circles represent (x, predicted y).

Question: Is the model reliable for predicting output for input values outside [-20, 20]?

k-NN

- Regression Example 2
  - ORL Database of Faces: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
    - The dataset contains face images of 40 people, with 10 images per person.
    - Each image contains 32x32=1024 pixels, with 256 grey levels per pixel.

```
Columns 1 through 25
  75    83    81    75    60    69    71    79    76    91   148   151   142   145   155   151   156   160   189    46    42    43    42    42    44
Columns 26 through 50
  46    43    47    46    48    50    49   101   109   115   129   142   145   143   144   151   156   157   162   161   159   161   158   156   153
Columns 51 through 75
 153   151    75    30    41    43    44    44    43    46    49    46    48    51   128   139   155   159   153   148   149   147   162   171   172
Columns 76 through 100
 170   171   172   173   182   180   178   173   173   168   164   161   121    62    37    41    42    47    48    46    49   159   164   161   158
Columns 101 through 125
 152   155   143   153   170   177   182   161   163   167   176   179   180   182   183   185   185   184   179   178   171   158    44    29    36
Columns 126 through 150
  41    42    46   167   168   165   160   161   154   146   173   178   143   147   163   162   165   173   177   179   183   190   190   190   193
Columns 151 through 175
 192   189   189   188   180   157   154   160   165   166   170   169   165   166   146   145   160   186   131   123   160   180   171   165   171
Columns 176 through 200
 171   177   183   190   189   195   197   197   194   195   194   190   183   172   157   164   177   168   170   170   168   164   152   148   175
```

k-NN

- Regression Example 2
  - ORL Database of Faces: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
    - The dataset contains face images of 40 people, with 10 images per person.
    - Each image contains 32x32=1024 pixels, with 256 grey levels per pixel.
  - A regression task: Guess the right face from the left face (image completion).
    - Input variables (x): the 512 pixels of the left side of an image (given).
    - Output variables (y): the other 512 pixels of the image (to be estimated).

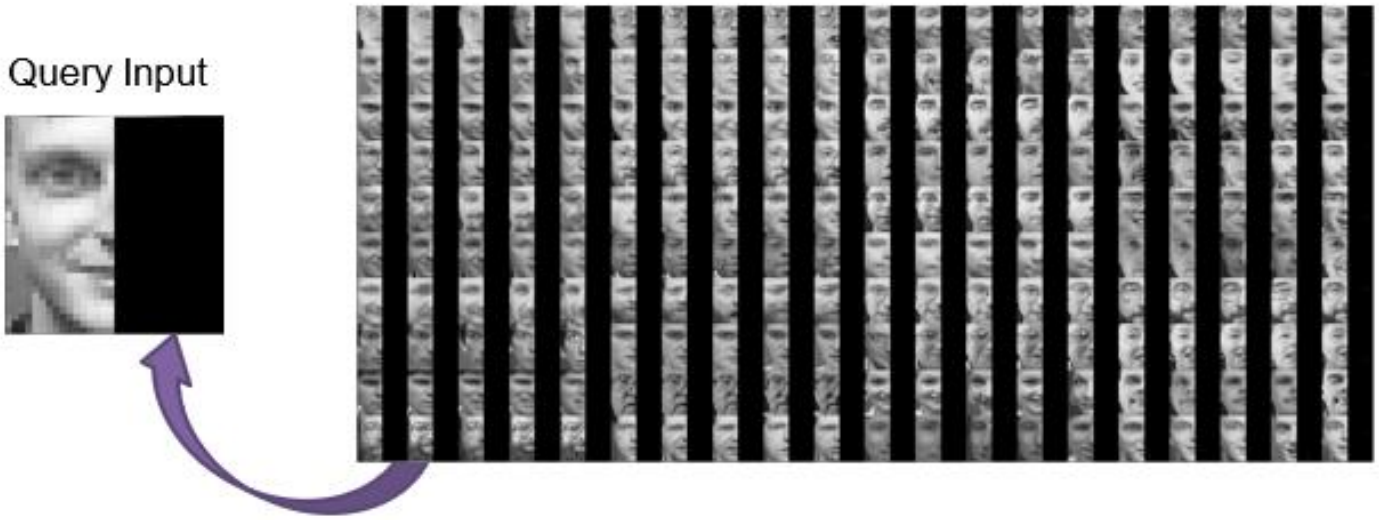
$$\mathbf{x} = [x_1, x_2, x_3, \ldots x_{512}]$$

$$\mathbf{y} = [y_1, y_2, y_3, \ldots y_{512}]$$

Given ***x***, guess ***y***!

k-NN

- Regression Example 2
  - Build a 3-NN regression model using 200 training images, containing 5 example images (with both left and right faces) for each of the 40 people.



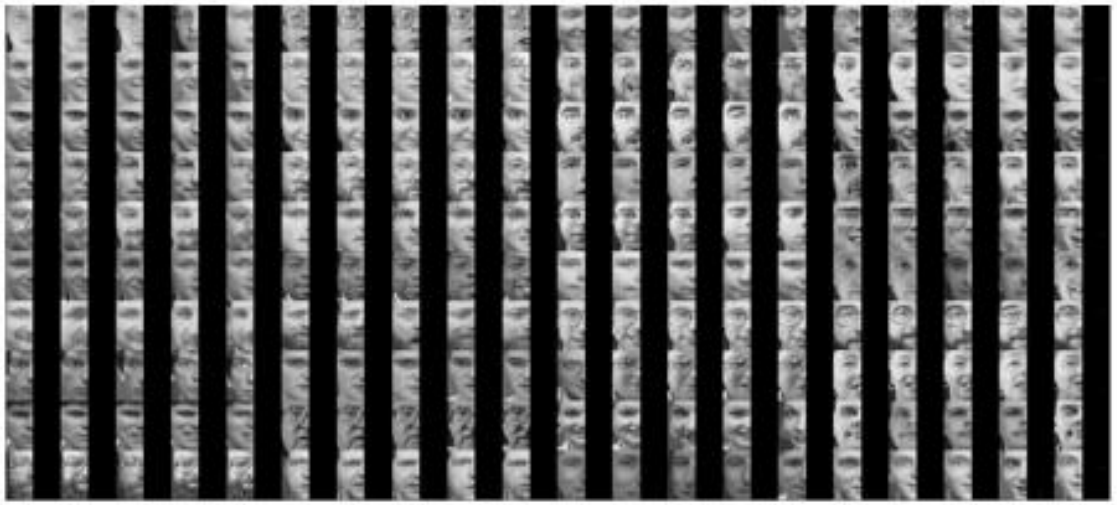
Information carried by the training images informs how to infer right face from left.

k-NN

- Regression Example 2
  - Build a 3-NN regression model using 200 training images, containing 5 example images (with both left and right faces) for each of the 40 people.

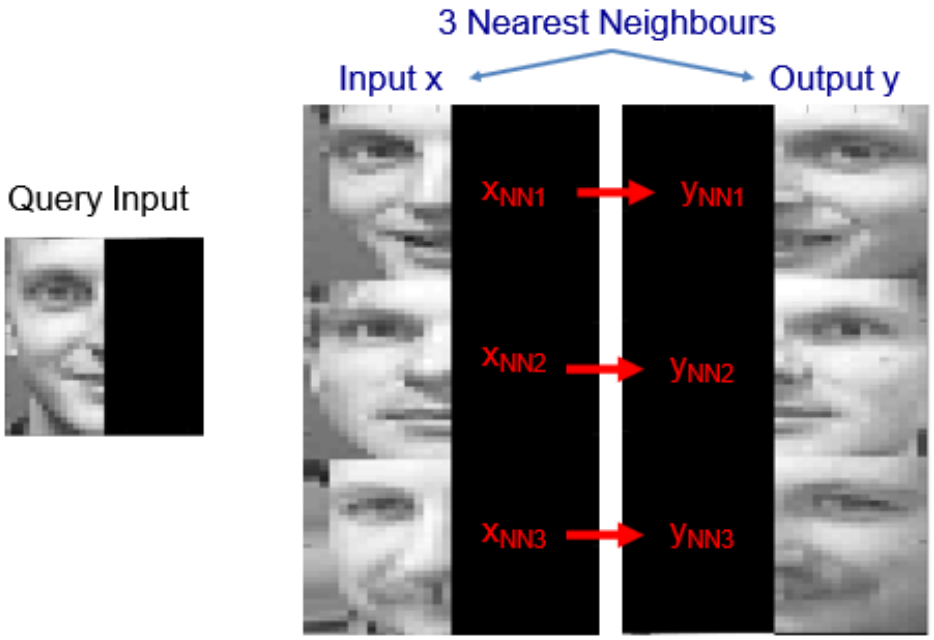
Query Input

Compare Euclidean distances using image pixels, and search the 3 most similar left faces.



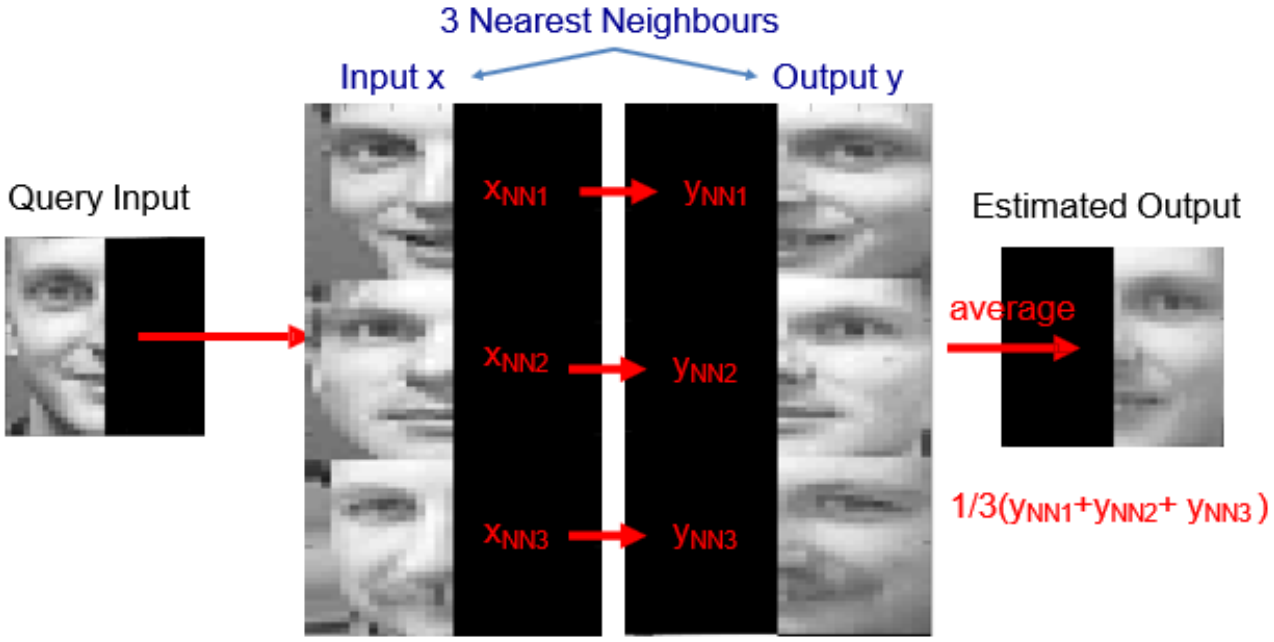
k-NN

- Regression Example 2
  - Build a 3-NN regression model using 200 training images, containing 5 example images (with both left and right faces) for each of the 40 people.



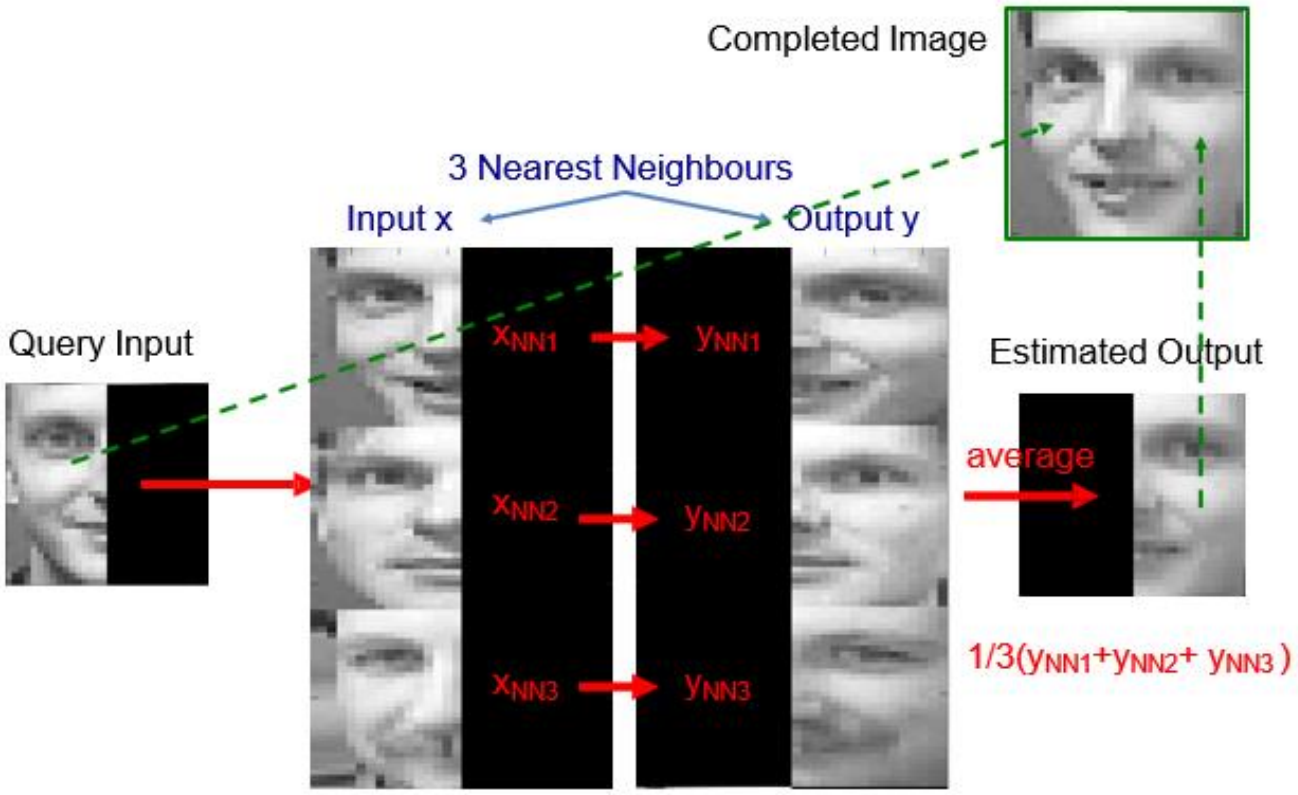
k-NN

- Regression Example 2
  - Build a 3-NN regression model using 200 training images, containing 5 example images (with both left and right faces) for each of the 40 people.



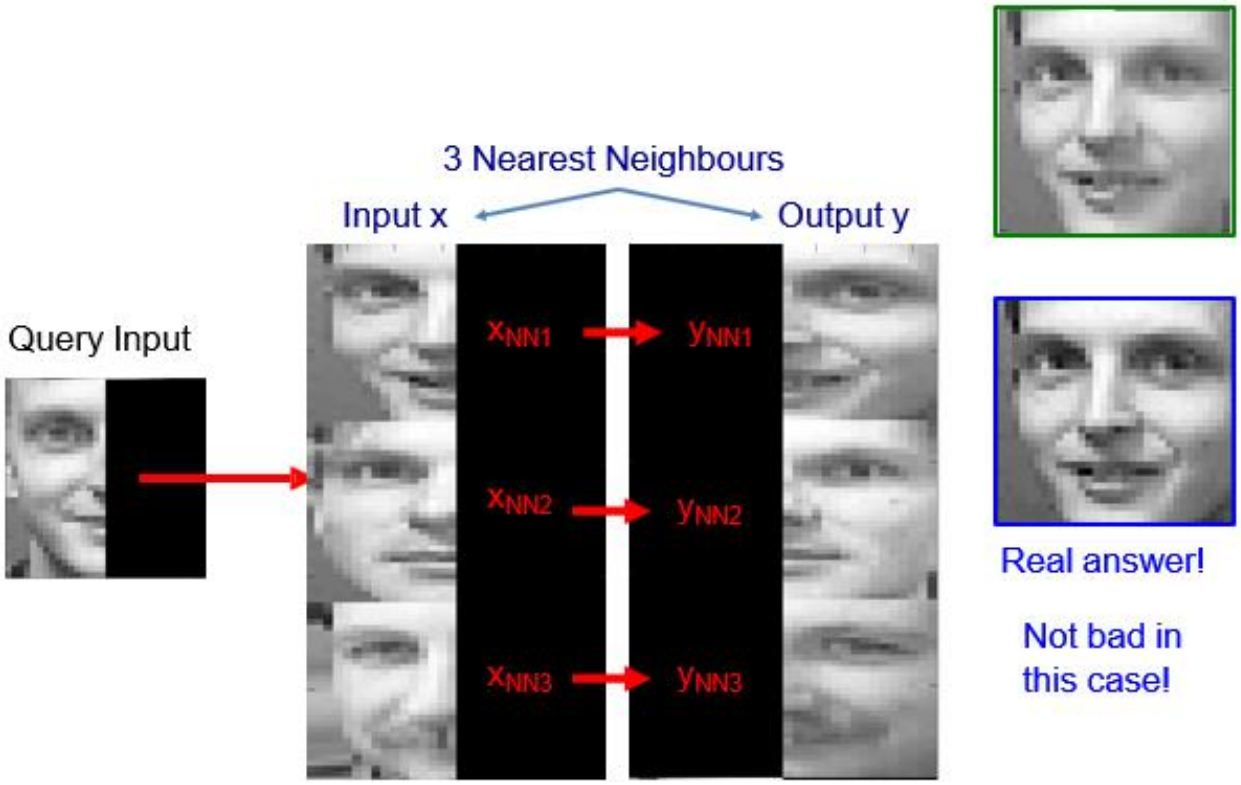
k-NN

- Regression Example 2



k-NN

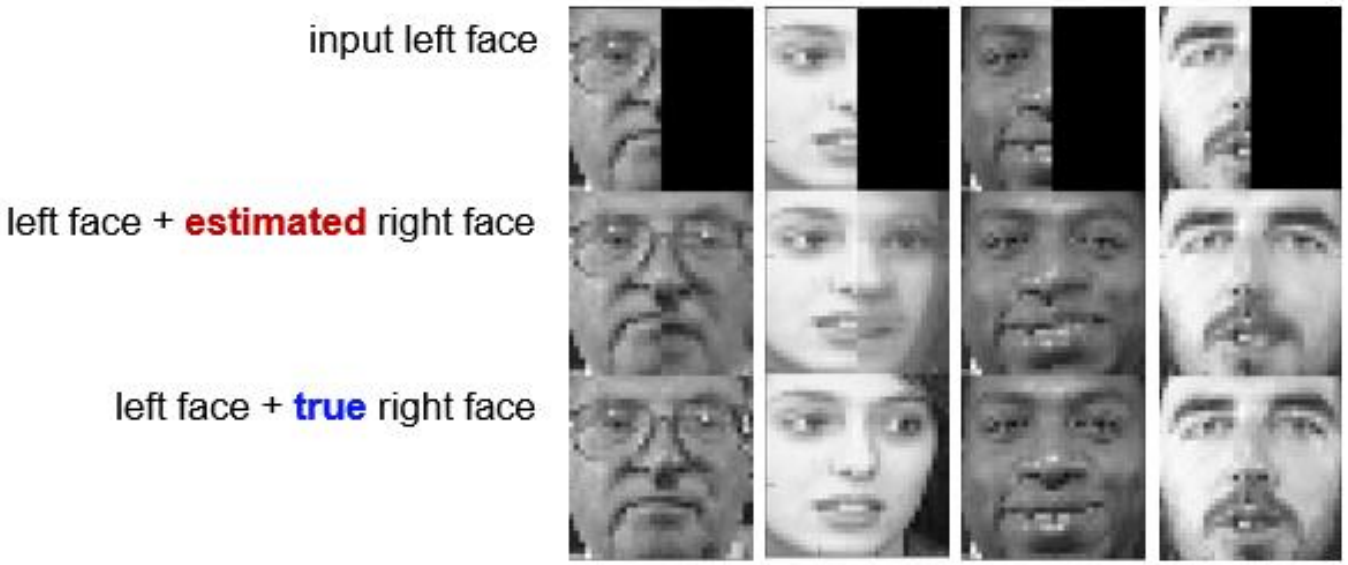
- Regression Example 2



k-NN

- Regression Example 2

  - More testing images!



k-NN

- Instance-based Learning
  - Many algorithms are developed to predict output based on the similarity (or distance) of the query to its nearest neighbour(s) in the training set.
  - Representative algorithm: k-NN.
  - Aspects to be considered:
    - How to compute the distance?
    - How to choose number of neighbours (k)?
    - How to infer the output from neighbouring points?

Reference: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_409

k-NN

- Things to consider when use k-NN:
  - Distance calculation
  - Effect of neighbour number k
  - Effect of training samples
  - Fast neighbour search

k-NN

An important quantity to compute in a k-NN model is distance!

k-NN

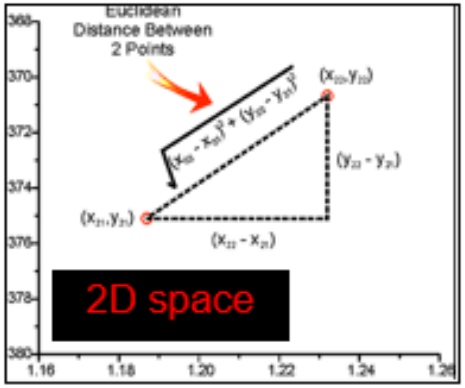
- Distance Measure
  - The most commonly used distance is Euclidean distance:

Given two d-dimensional data points

$$\mathbf{p} = [p_1, p_2, \ldots, p_d], \mathbf{q} = [q_1, q_2, \ldots, q_d]$$

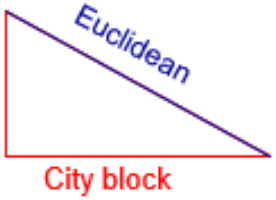
$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_d - q_d)^2}$$

$$= \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2} = \|\mathbf{p} - \mathbf{q}\|_2$$



  - Minkowski Distance:

$$d(\mathbf{p},\mathbf{q}) = \left[|p_1 - q_1|^t + |p_2 - q_2|^t + \ldots + |p_d - q_d|^t \right]^{\frac{1}{t}} = \left[\sum_{i=1}^{d} |p_i - q_i|^t \right]^{\frac{1}{t}}$$



t=2: Euclidean distance

t=1: Manhattan (city block) distance

k-NN

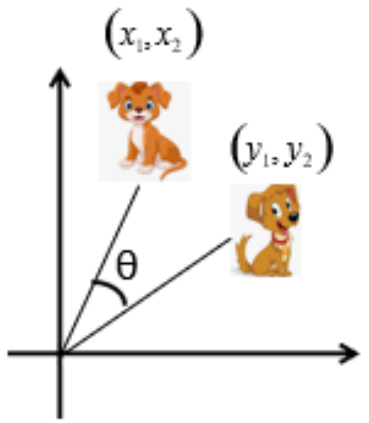
- Similarity Measure
  - You can also use a similarity measure!
  - The k nearest neighbours have the highest similarity values.

Inner product and cosine similarity are good similarity measures for capturing co-occurrence pattern:

$$s_{\text{inner}}(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{d} p_i q_i = \mathbf{p}^T \mathbf{q},$$

$$s_{\cos}(\mathbf{p},\mathbf{q}) = \frac{\sum_{i=1}^{d} p_i q_i}{\sqrt{\sum_{i=1}^{d} p_i^2}\sqrt{\sum_{i=1}^{d} q_i^2}} = \frac{\mathbf{p}^T \mathbf{q}}{\|\mathbf{p}\|_2 \|\mathbf{q}\|_2}$$

Cosine example:



$$\cos(\theta) = \frac{x_1 y_1 + x_2 y_2}{\sqrt{x_1^2 + x_2^2}\sqrt{y_1^2 + y_2^2}}$$

- Convert cosine to distance:

$$d(\mathbf{p},\mathbf{q}) = 1 - s_{\cos}(\mathbf{p},\mathbf{q})$$

k-NN

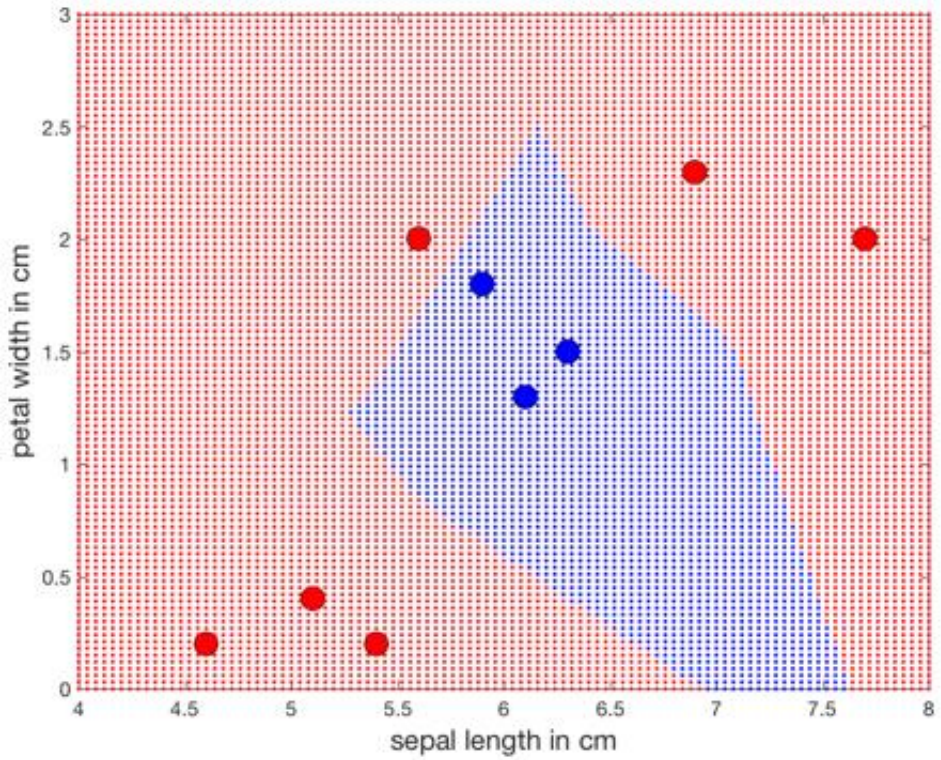
Training Samples
in k-NN

k-NN

- Effect of Training Samples
  - Small number of training samples:
    - Insufficient information
  - Large number of training samples:
    - More information
    - But, time and memory cost consuming (distance calculation, sorting)
  - Noisy training samples:
    - Inaccurate prediction

k-NN

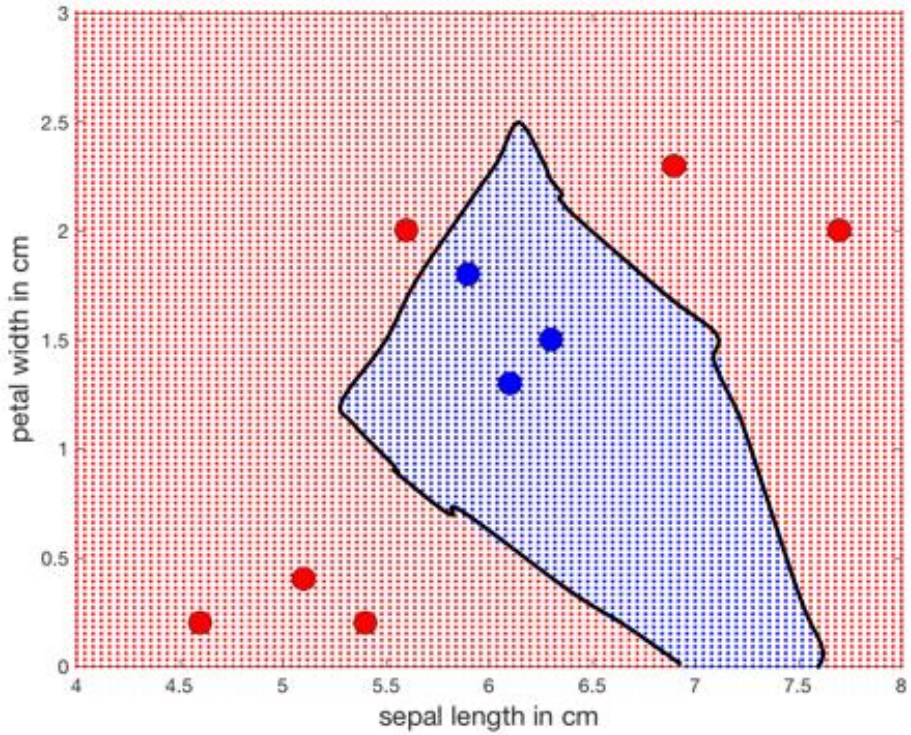
- A binary 1-NN classifier trained by 9 iris samples!



- Imagine to apply the 1-NN rule to every single point in the space. The whole space will be divided into red and blue areas. The left figure displays such effect.

k-NN

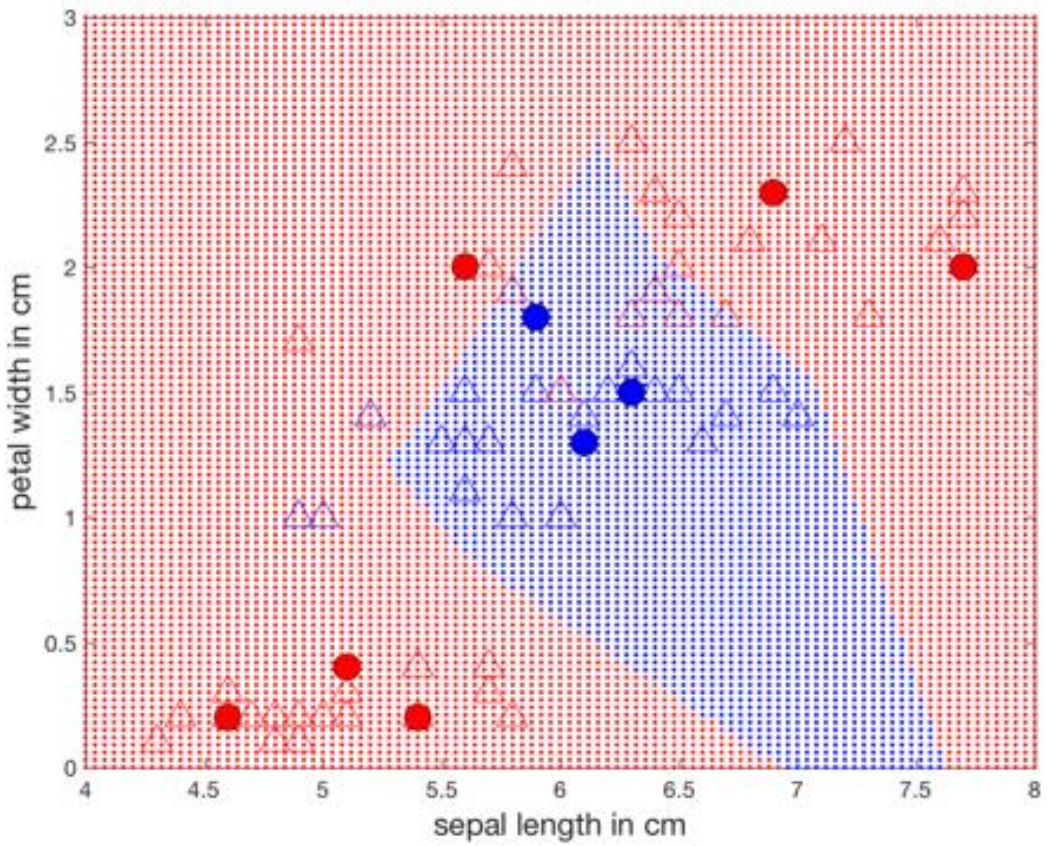
- A binary 1-NN classifier trained by 9 iris samples!



- This is equivalent to using the nonlinear separation boundary (highlighted by the black line) to partition the data space to two classes.

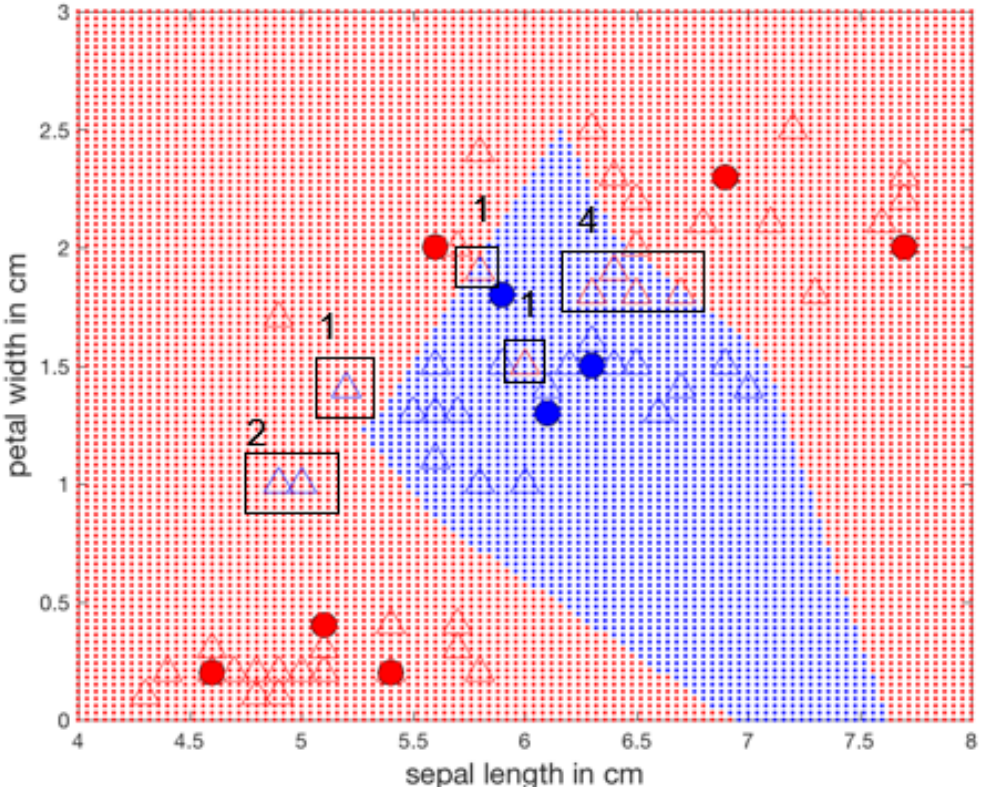
k-NN

- Testing it using 60 new samples!



k-NN

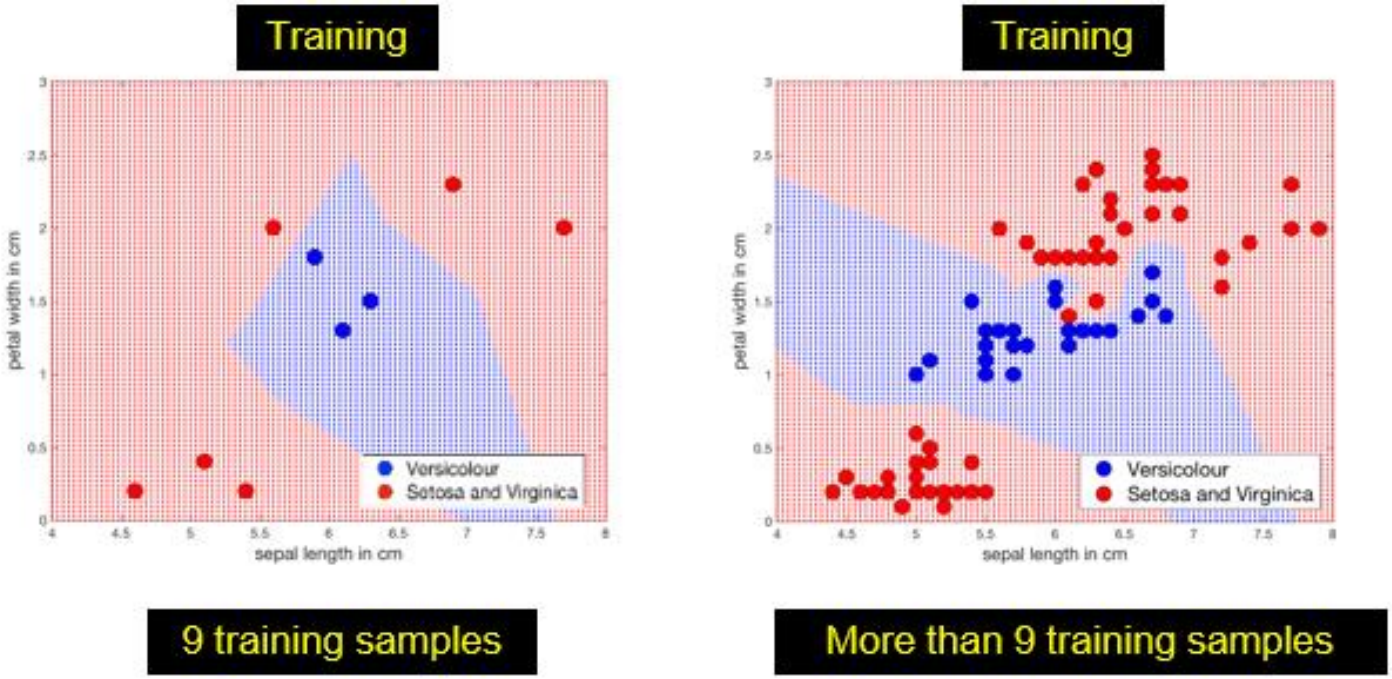
- Testing it using 60 new samples!



- 9 out 60 testing samples are classified into the wrong class!
  error rate = 9/60=15%
  accuracy = 51/60=85%
- The 9 training samples do not summarise well the data distribution in each class.
- Insufficient training information!

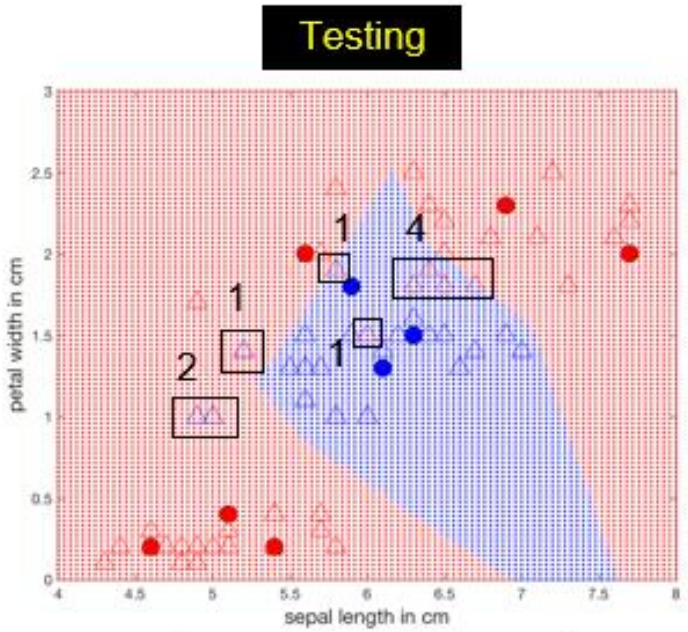
k-NN

- Increase the number of training samples



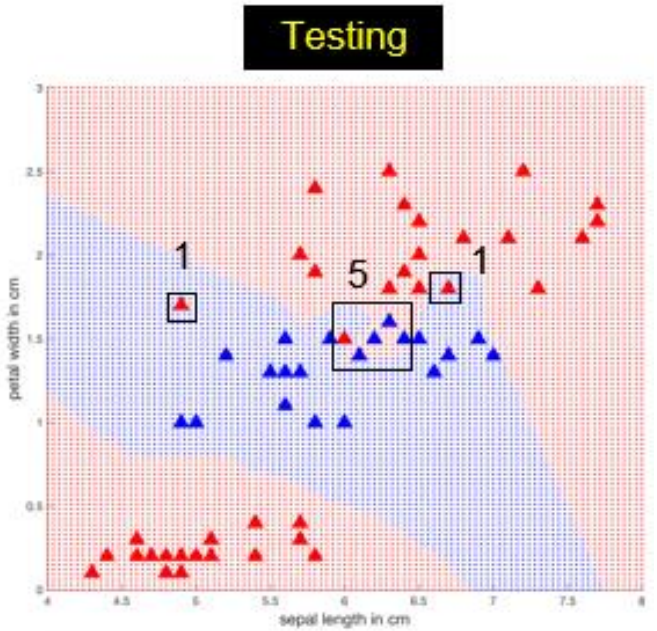
k-NN

- Increase the number of training samples



9 training samples

9 out 60 testing samples are classified into the wrong class!
error rate = 9/60=15%
accuracy = 51/60=85%

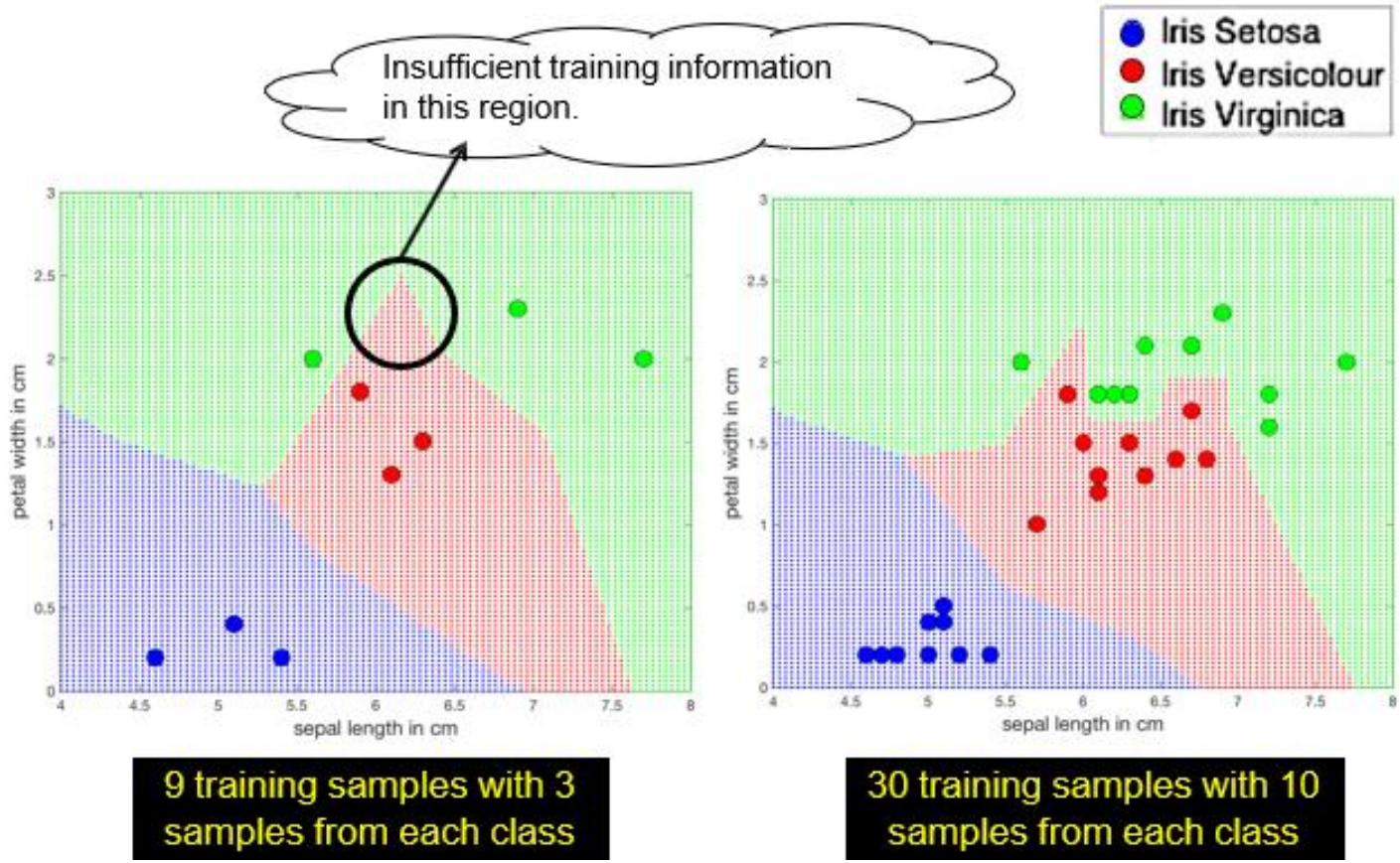


More than 9 training samples

7 out 60 testing samples are classified into wrong class!
error rate = 7/60 = 12%
accuracy = 53/60 = 88%

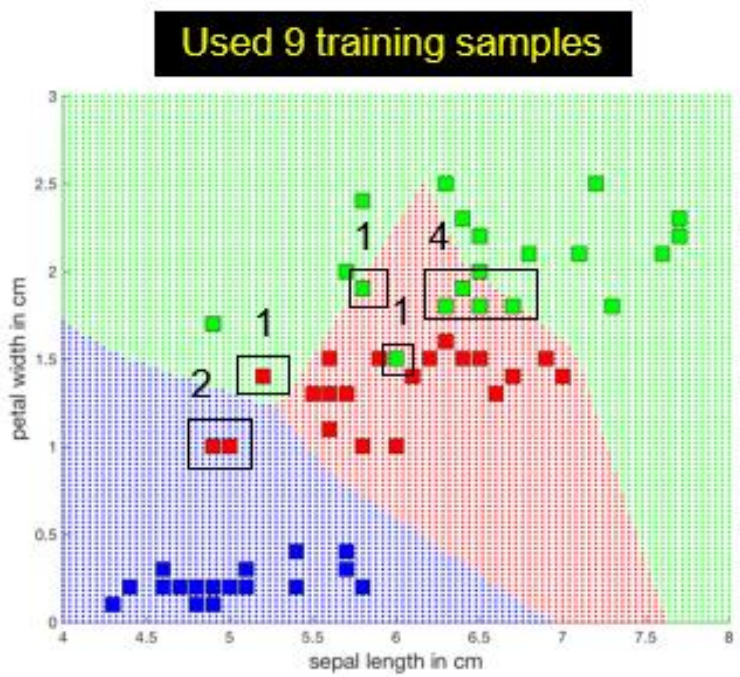
k-NN

- A 3-class 1-NN example



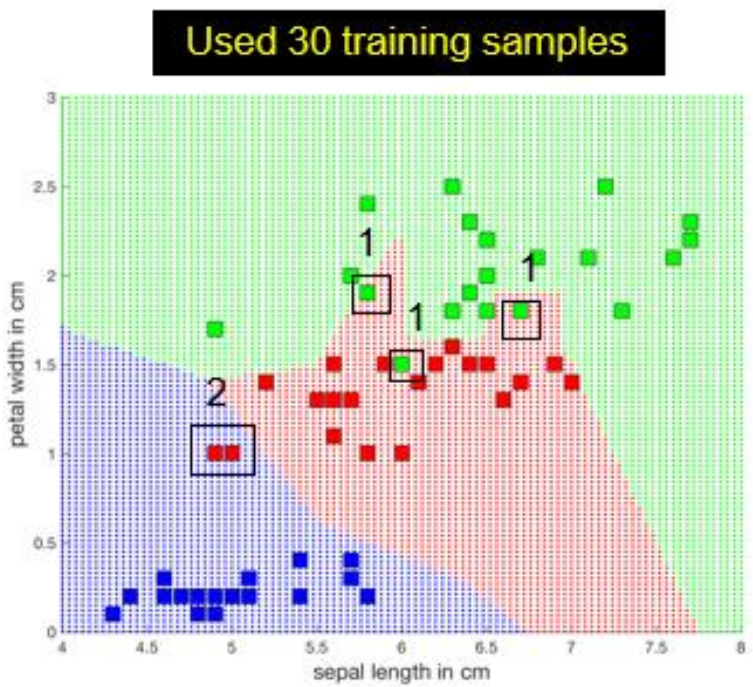
k-NN

- Testing with 60 new samples



testing error rate = 9/60 = 15%

testing accuracy = 51/60 = 85%



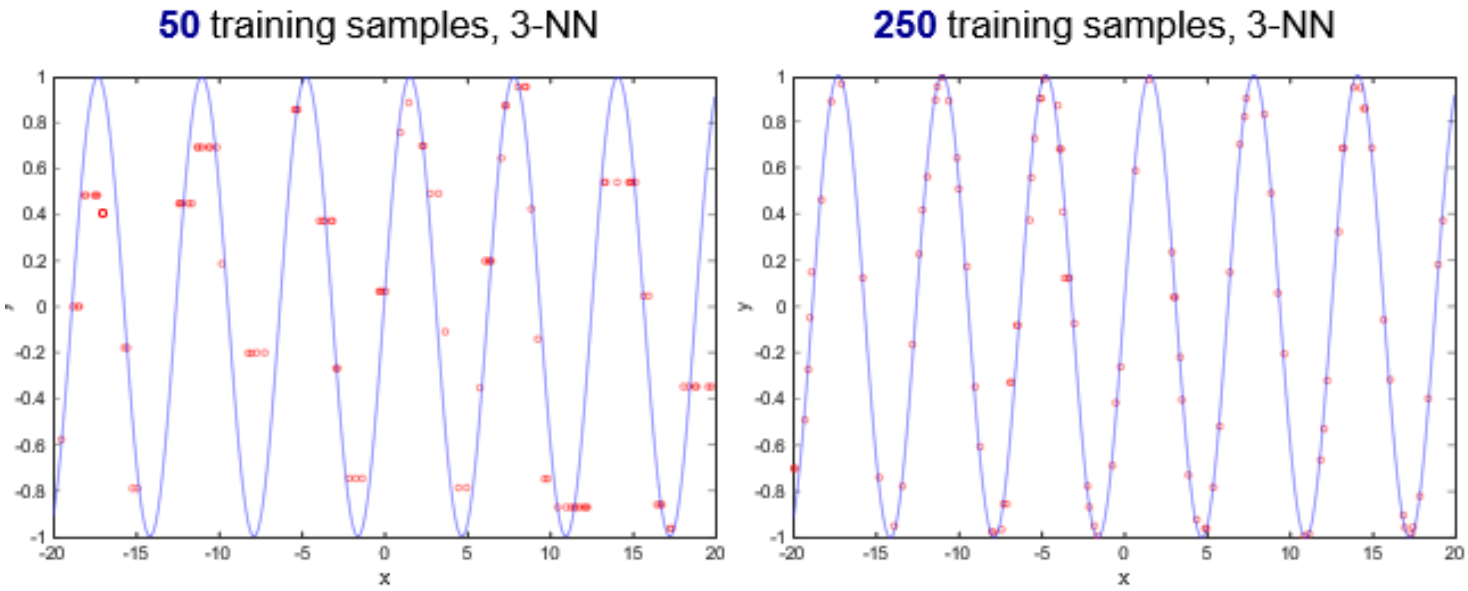
testing error rate = 5/60 = 8%

testing accuracy = 55/60 = 92%

k-NN

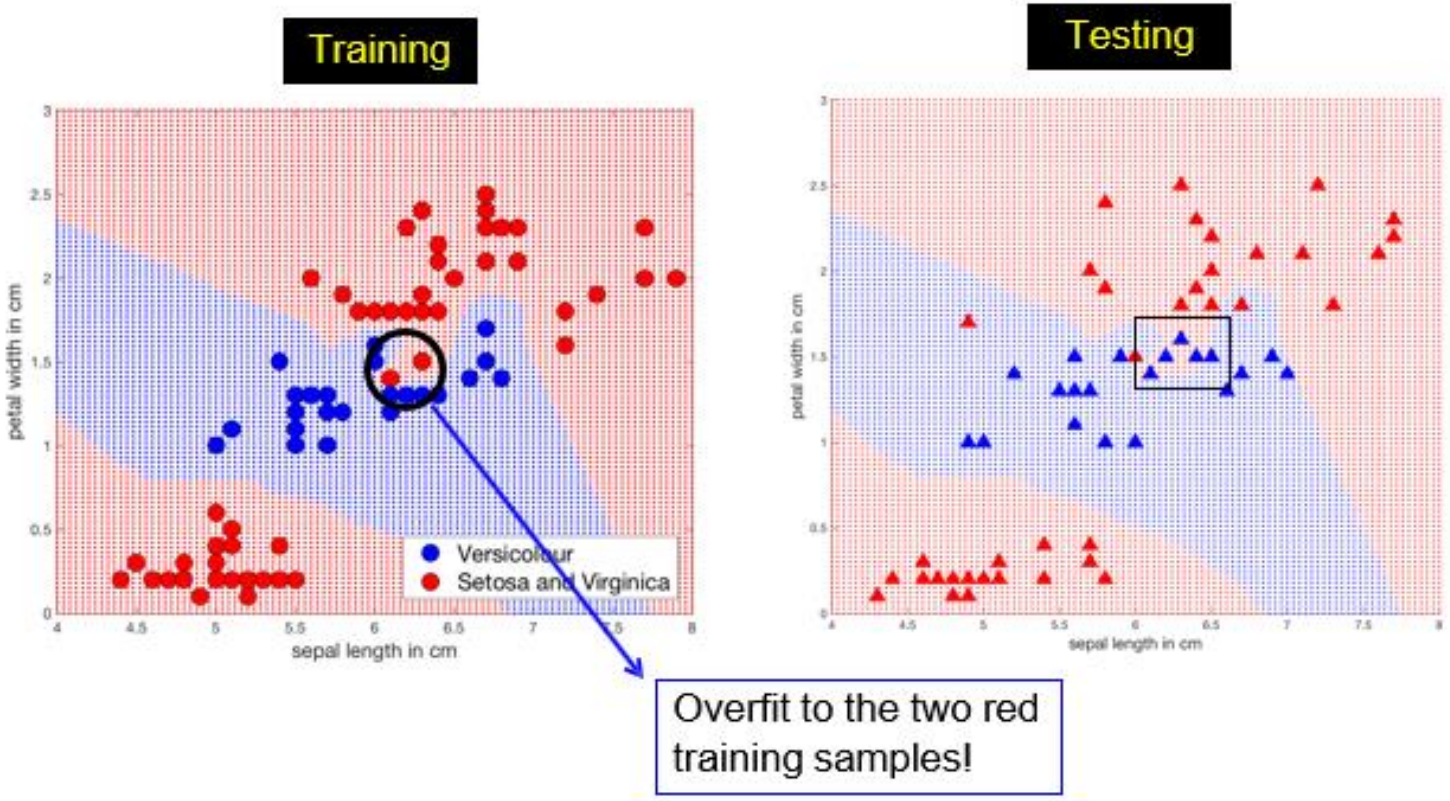
- Regression Example 1

More training samples provide more information to predict better.



k-NN

- Effect of Noisy Training Samples

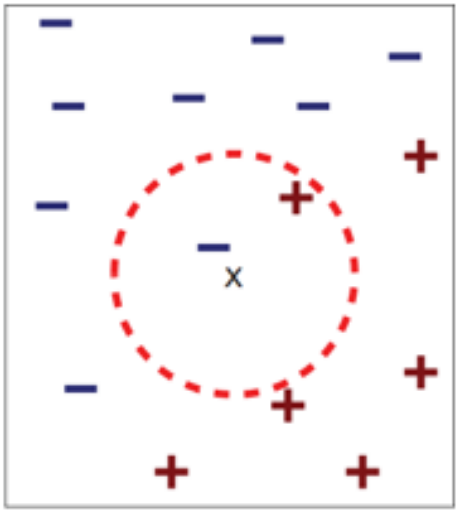


k-NN

Neighbour Number k in k-NN

k-NN

- Neighbour Number k
  - Hyper-parameter: neighbour number k.
  - The process of determining the neighbour number k is called
  - hyper-parameter selection or model selection.
  - It is not a good idea to set k as an even number for binary classification.



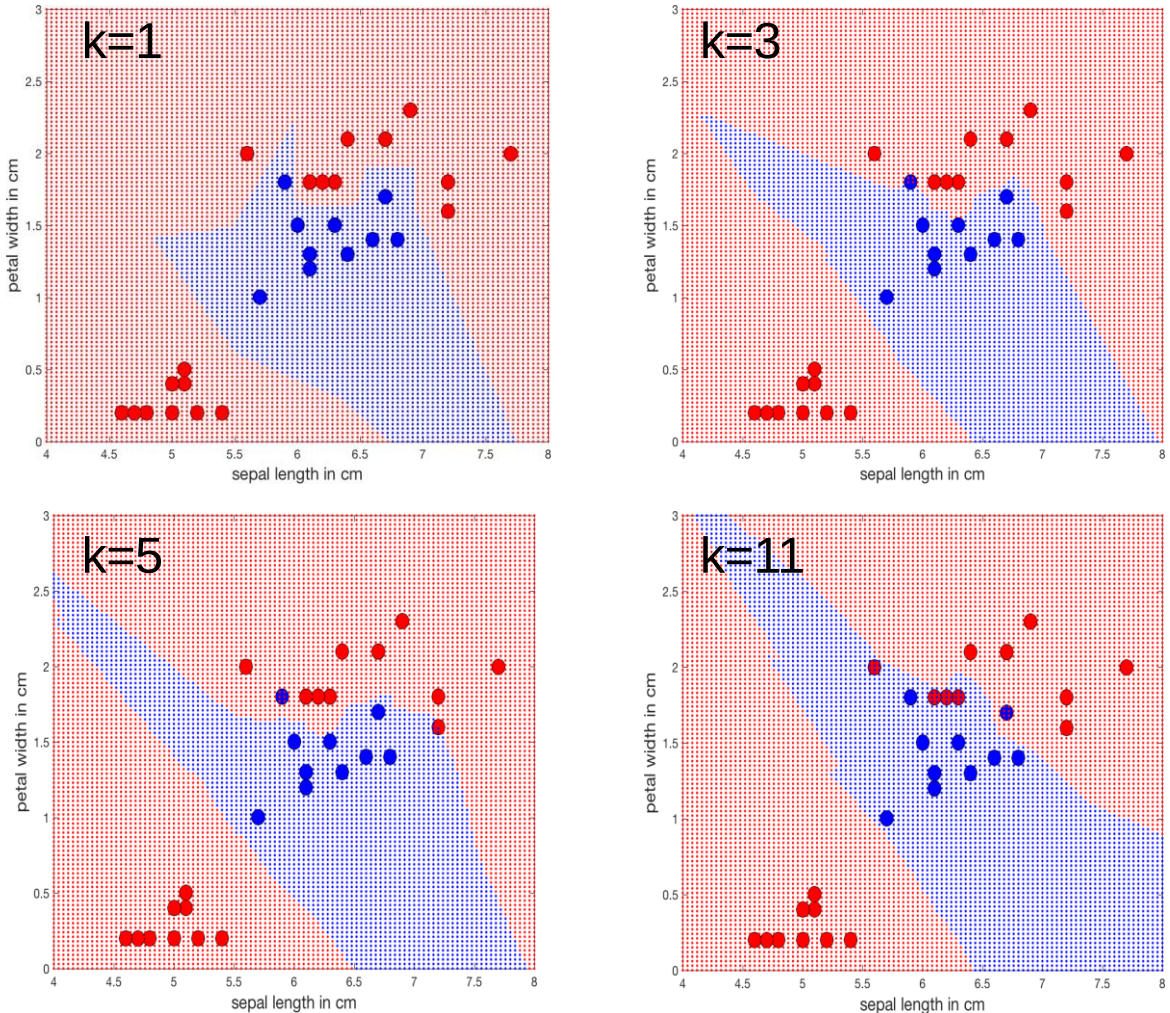
Not possible to reach a decision!..

k-NN

- Effect of Neighbour Number k
  - Hyper-parameter: neighbour number k.
  - Small k: We may model noise!
  - Large k: Neighbours will include too many samples from other classes, which can negatively affect the prediction.

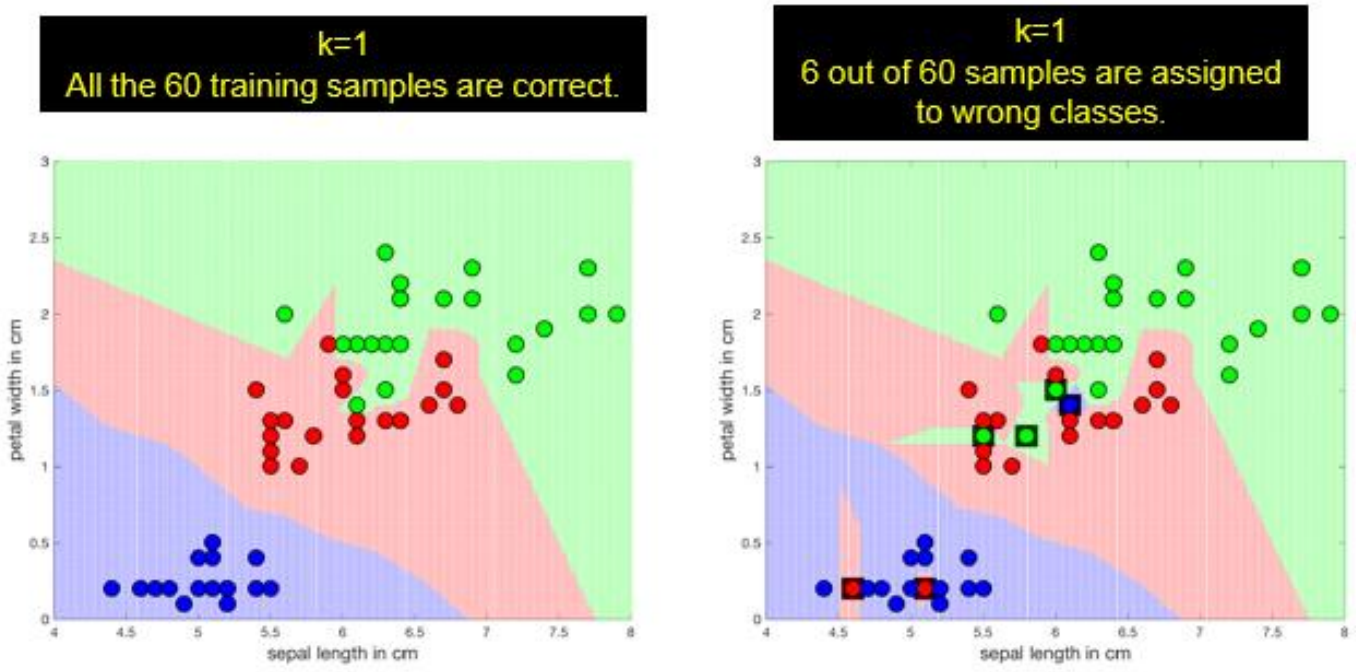
k-NN

- Binary k-NN trained with 20 samples with varying neighbour number k.



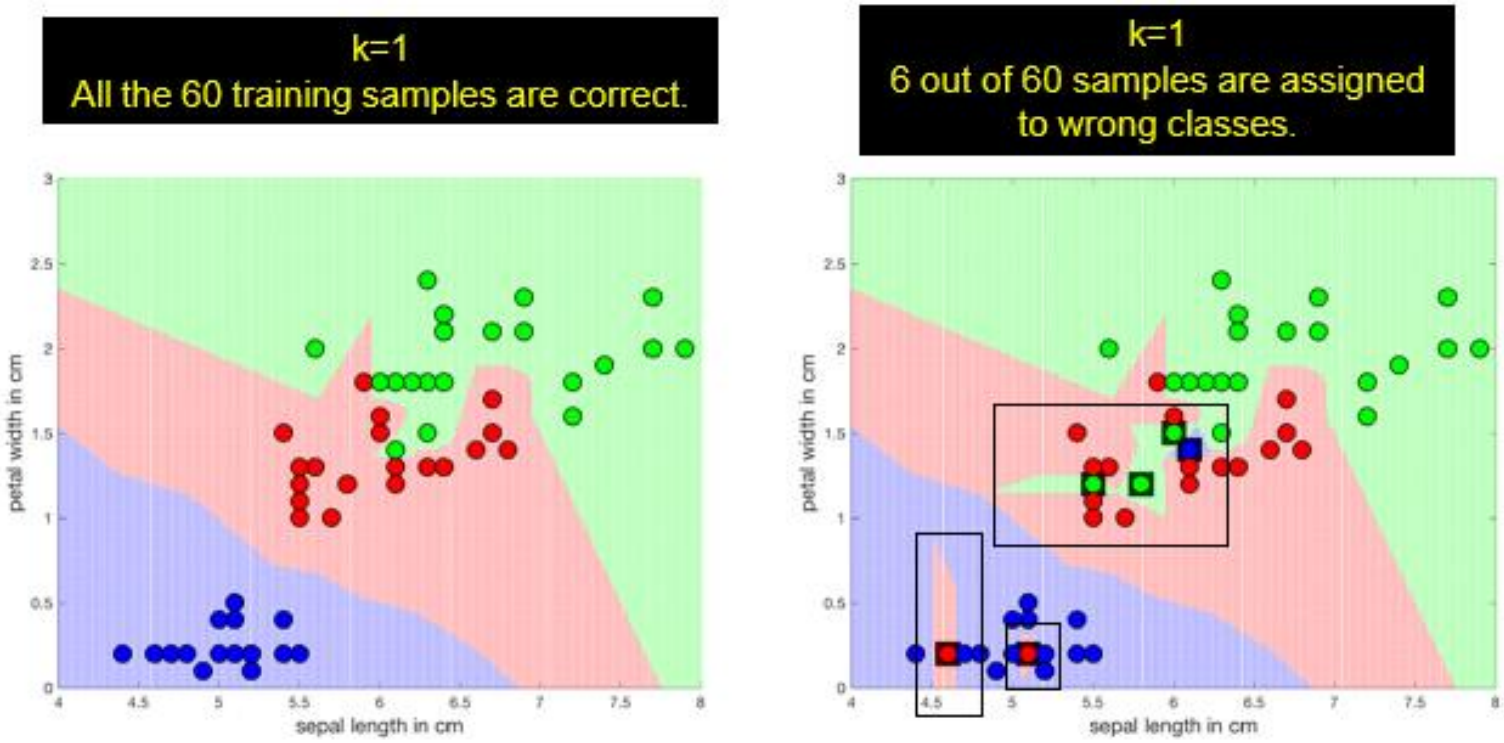
k-NN

- A 3-class k-NN case trained with noisy samples. Here, 6 out of 60 training samples were assigned randomly to a wrong class.



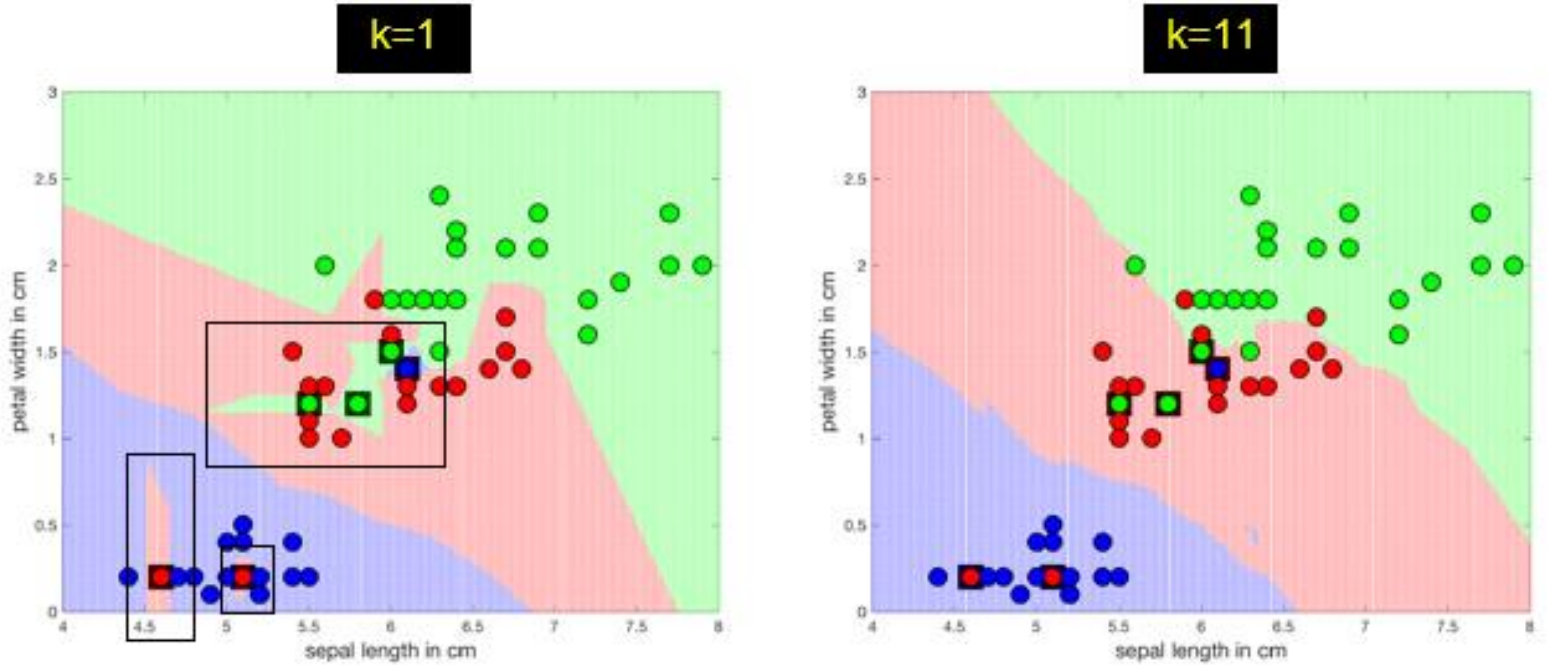
k-NN

- A 3-class k-NN case trained with noisy samples. Here, 6 out of 60 training samples were assigned randomly to a wrong class.



k-NN

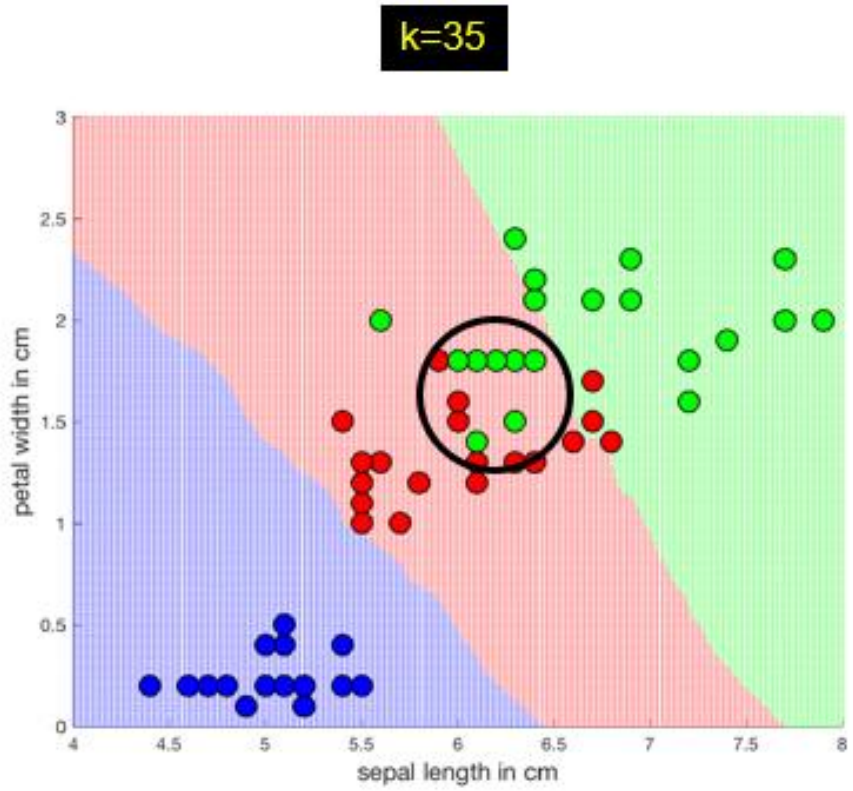
- Increase the neighbour number k



k=1 case is much more sensitive to those noisy samples than the k=11 case.

k-NN

- Too large k: 3-class k-NN



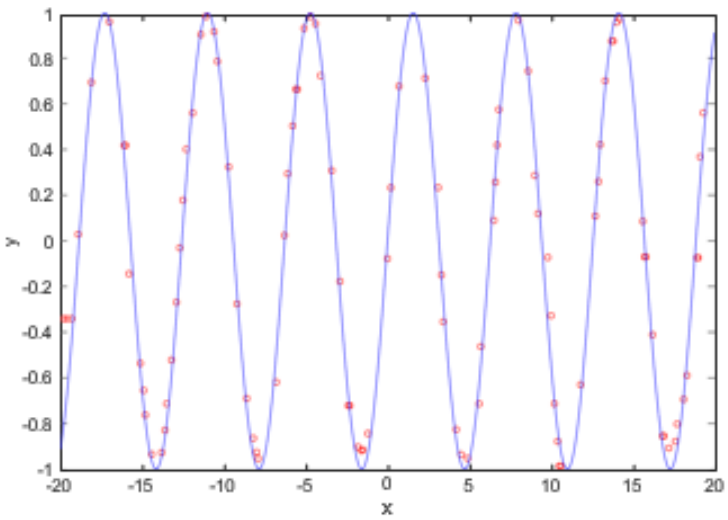
Many samples from other classes are identified as neighbours. The decision can become unreliable.

k-NN

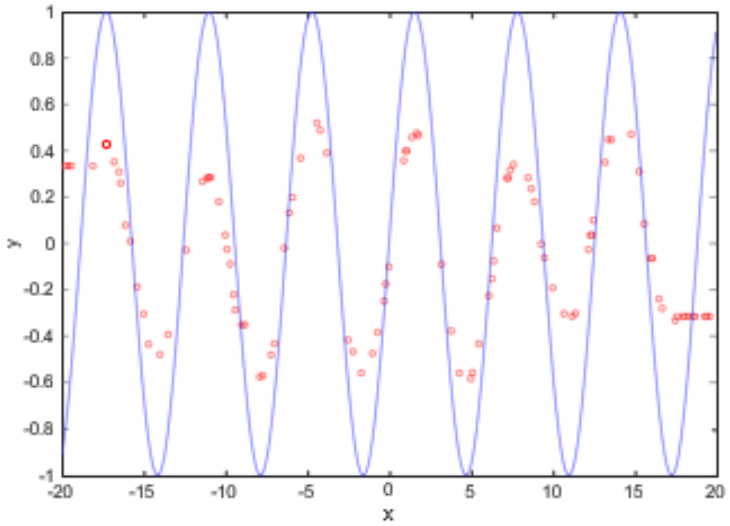
- Too large k: Regression Example 1

Including too many neighbours may introduce noise.

250 training samples, 5-NN



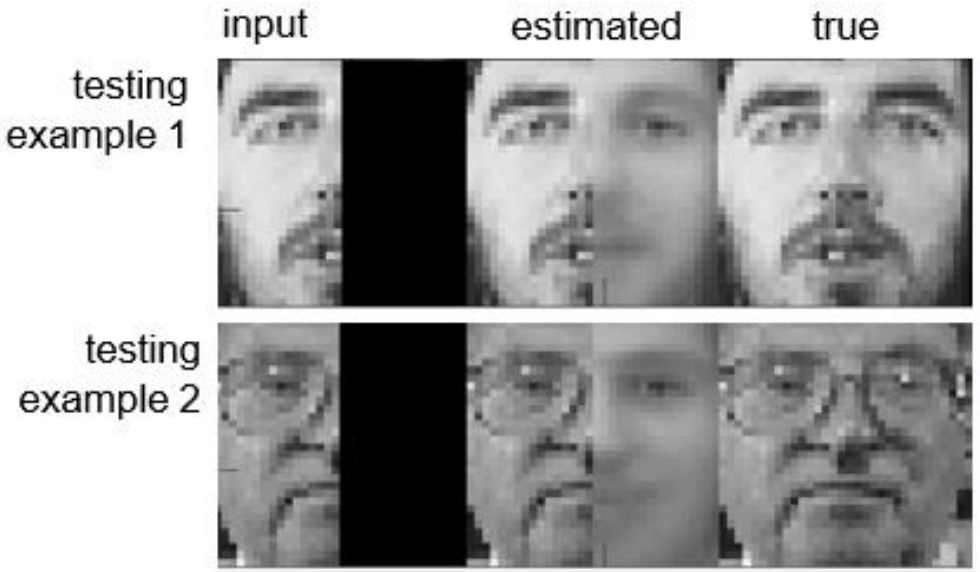
250 training samples, 25-NN



k-NN

- Too large k: Regression Example 2

  - Some testing images for 50-NN regression.



Too high neighbour number includes bad images (wrong faces in our case) in estimation, resulting in bad results.

k-NN

- Neighbour Search Algorithm
  - Study fast computation of nearest neighbours is an active research area in machine learning.
  - Naïve Approach: brute-force compute distances between all pairs of samples and sort.
  - Tree-based methods:
    - Basic idea: Knowing A is very distant from B, and B is very close to C, it is certain that A and C are very distant, without having to explicitly calculate their distance. Apply this to reduce the number of distance calculations.
    - Variaties: K-D tree, Ball tree, etc.

k-NN tool in Python provided by Scikit-learn:

https://scikit-learn.org/stable/modules/neighbors.html

k-NN

- Summary
  - k-NN is the simplest machine learning algorithm.
  - It can be used for both classification and regression.
  - ***No*** explicit ***training***.
    - A ***non-parametric*** method: no parameter to be optimised
  - The algorithm relies on a distance measure.
  - More training data provides more information to learn, but results in high memory cost (needs to store all the training data).
  - The neighbour number k is a hyper-parameter to be selected by the user.
    - Too small: sensitive to noise
    - Too large: inaccurate prediction

Supervised Learning: SVM

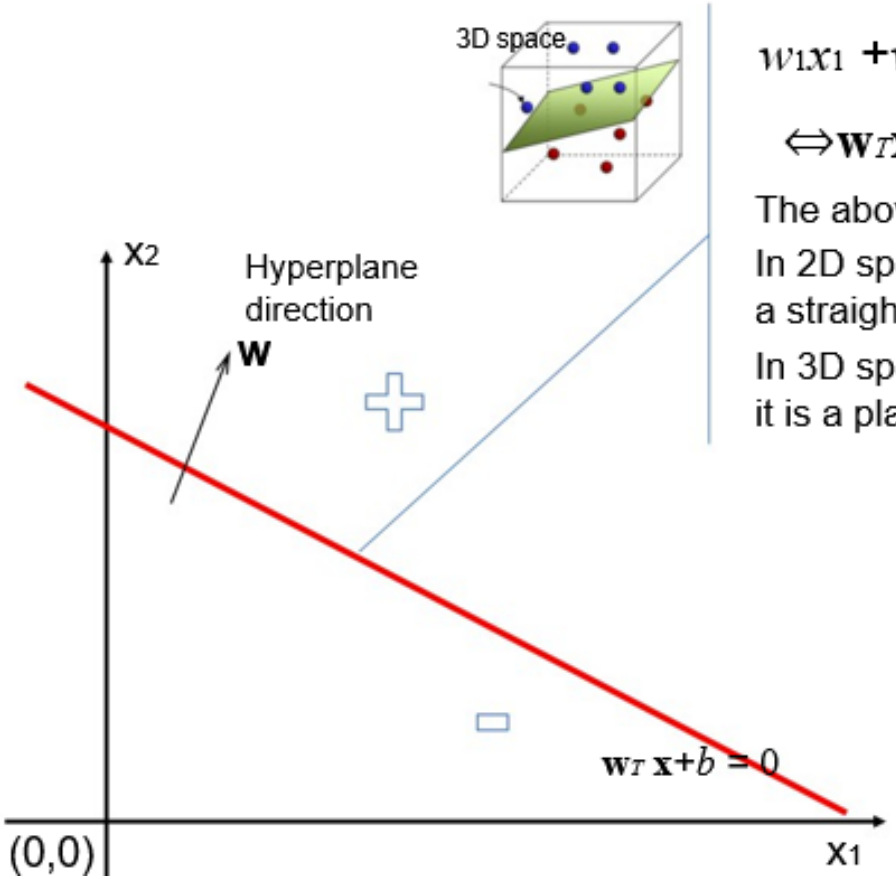
- Support Vector Machines (SVM)
  - SVM history and basic concepts
  - Core idea: hard-margin SVM for linearly separable cases

SVM

- SVM History and Library
  - Vapnik and Lerner (1963) introduced the generalised portrait algorithm. The algorithm implemented by SVMs is a nonlinear generalisation of the generalised portrait algorithm.
  - Support vector machine was first introduced in 1992:
    - Boser et al. A training algorithm for optimal margin classifiers. Proceedings of the 5-th Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
  - More on SVM history: http://www.svms.org/history.html
  - Centralised website: http://www.kernel-machines.org
  - Popular textbook:
    - N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, 2000. http://www.support-vector.net
  - Popular library: LIBSVM, MATLAB SVM, scikit-learn

SVM

- Revisit Hyperplane



$$w_1x_1 + w_2x_2 + \ldots + w_d x_d + b = 0$$

$$\Leftrightarrow \mathbf{w}_T\mathbf{x} + b = 0$$

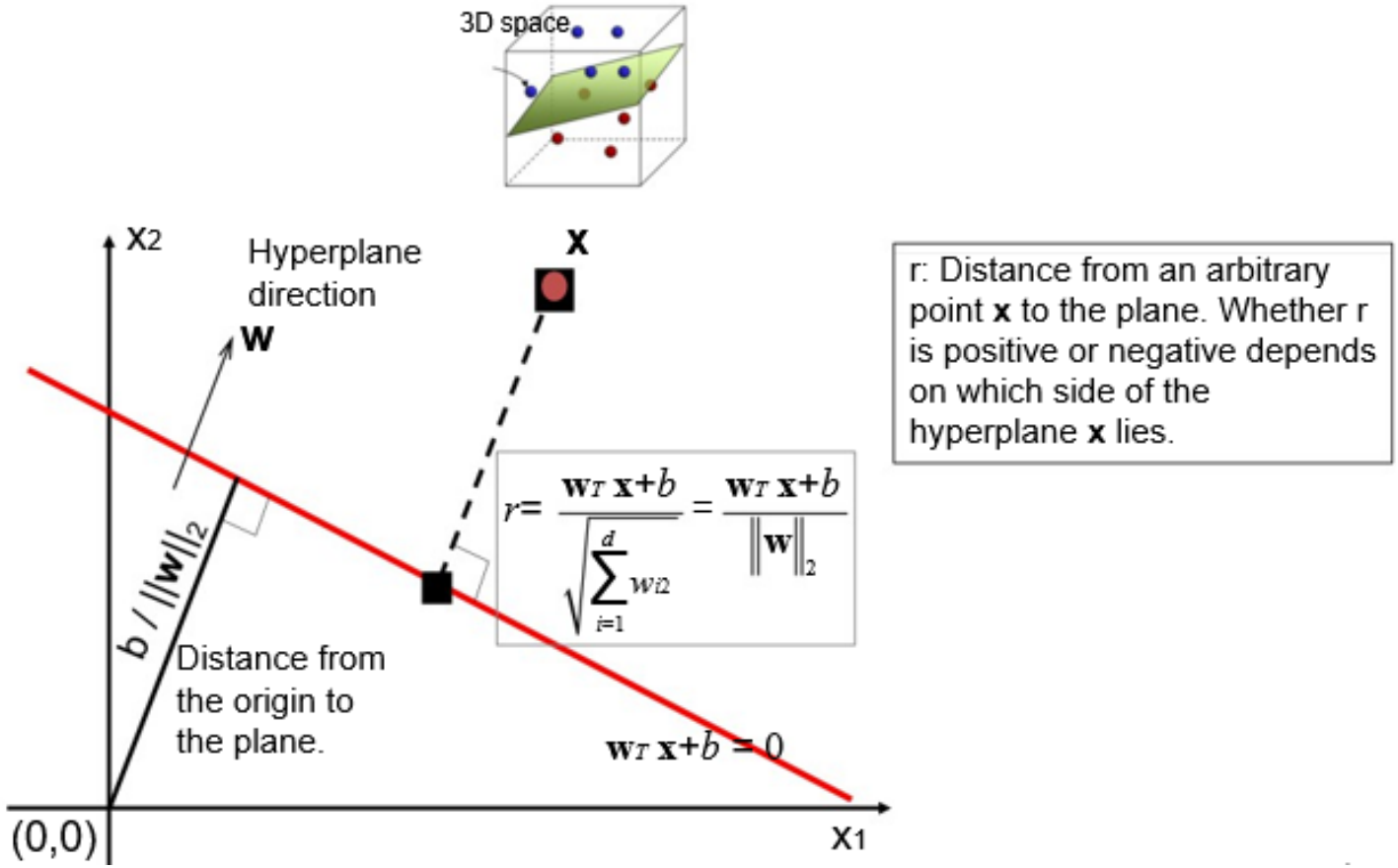
The above is called a hyperplane.

In 2D space ($w_1x_1+w_2x_2+b=0$), it is a straight line.

In 3D space ($w_1x_1+w_2x_2+w_3x_3+b=0$), it is a plane.

SVM

- Distance to Hyperplane

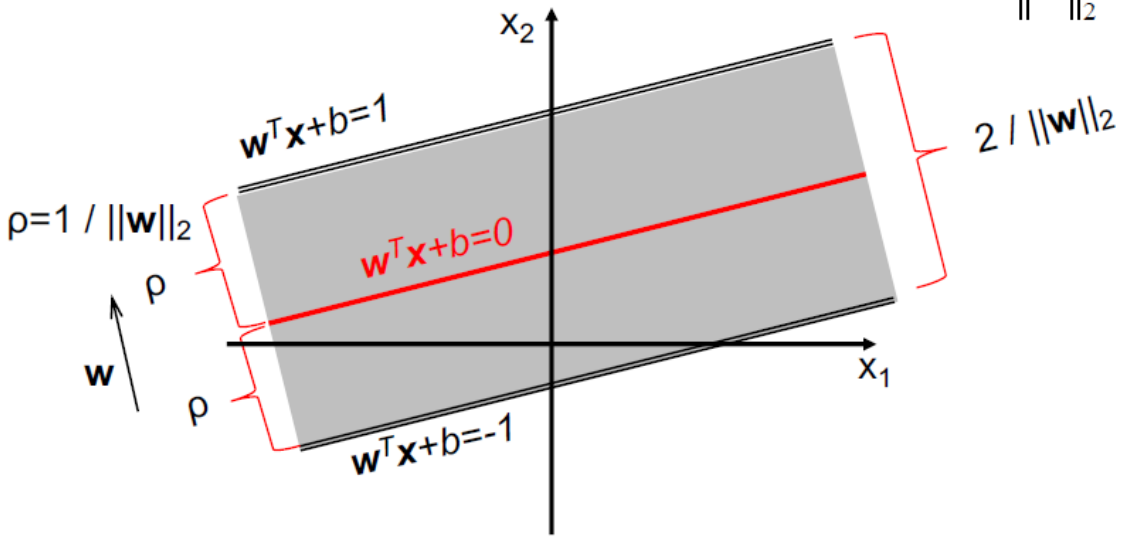


r: Distance from an arbitrary point **x** to the plane. Whether r is positive or negative depends on which side of the hyperplane **x** lies.

- Parallel Hyperplanes
  - We focus on two parallel hyperplanes:

$$\begin{cases} \mathbf{w}^T\mathbf{x} + b = 1 \\ \mathbf{w}^T\mathbf{x} + b = -1 \end{cases}$$

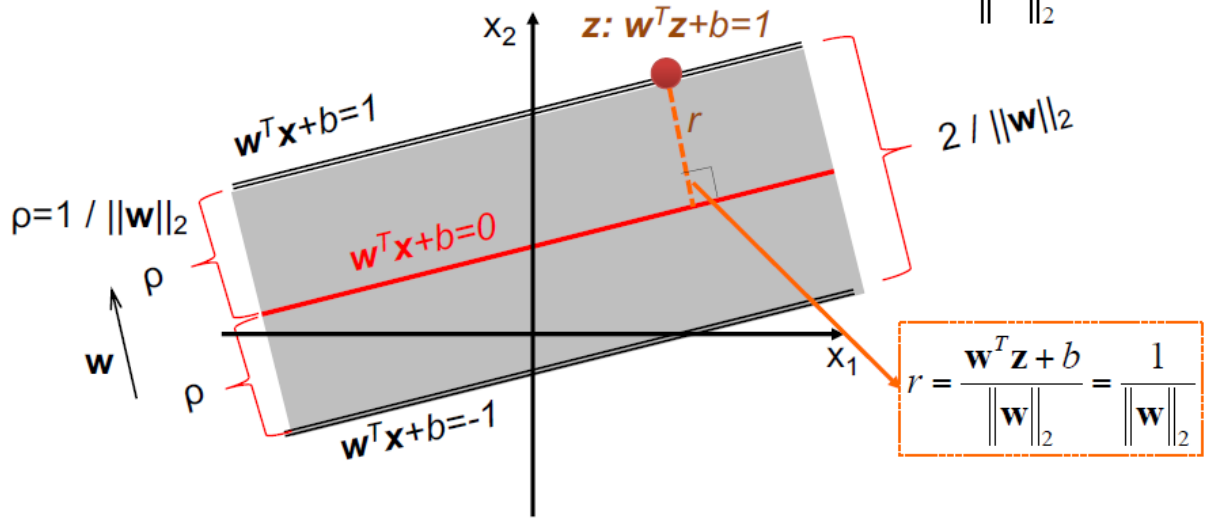
  - Geometrically, distance between these two planes is $\frac{2}{\|\mathbf{w}\|_2}$



- Parallel Hyperplanes
  - We focus on two parallel hyperplanes:

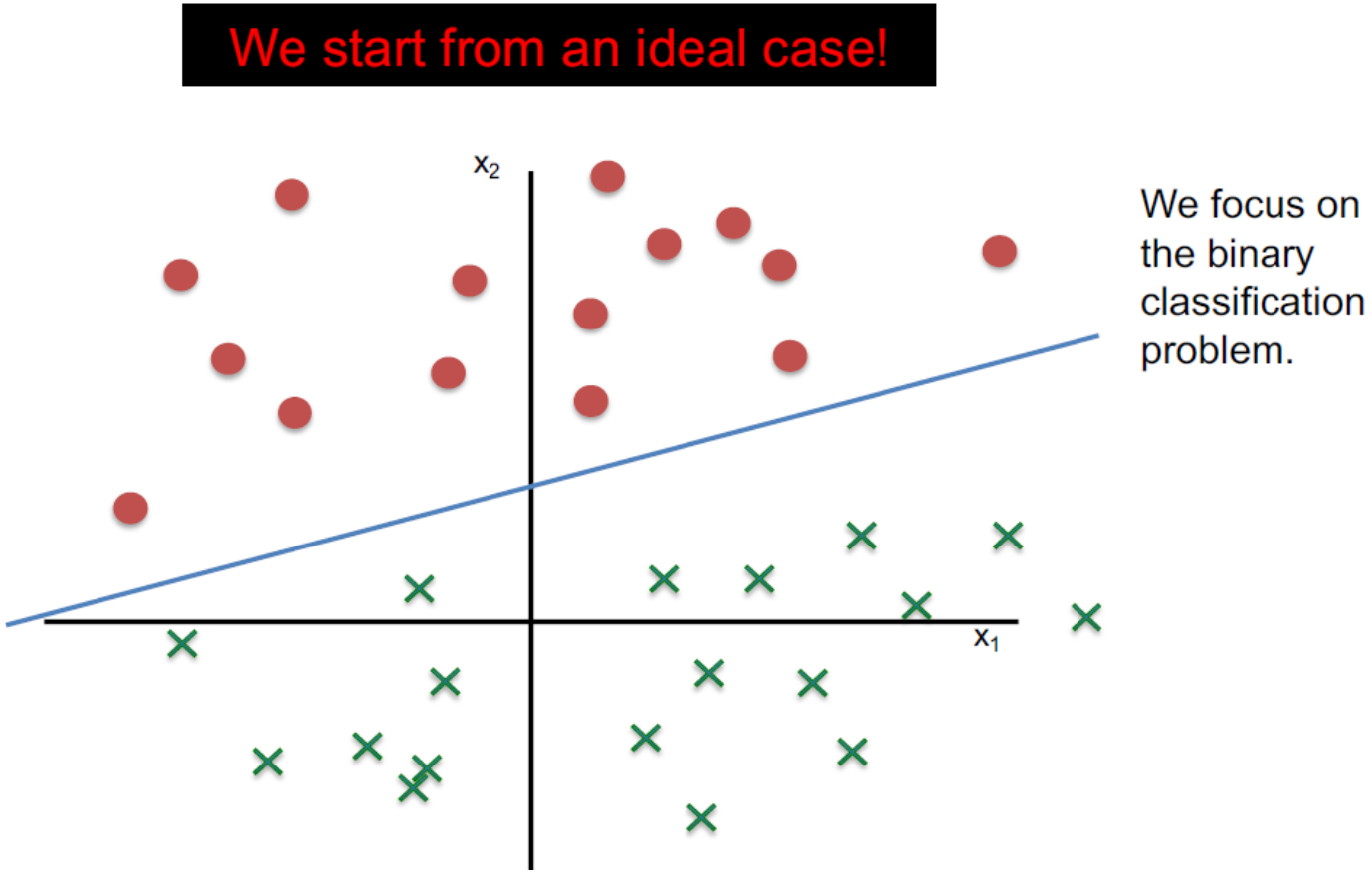
$$\begin{cases} \mathbf{w}^T\mathbf{x}+b=1 \\ \mathbf{w}^T\mathbf{x}+b=-1 \end{cases}$$

  - Geometrically, distance between these two planes is $\frac{2}{\|\mathbf{w}\|_2}$



SVM

- Linearly Separable Case



Separation Margin

- Given two parallel hyperplanes below, we separate two classes of data points by preventing the data points from falling into the margin:

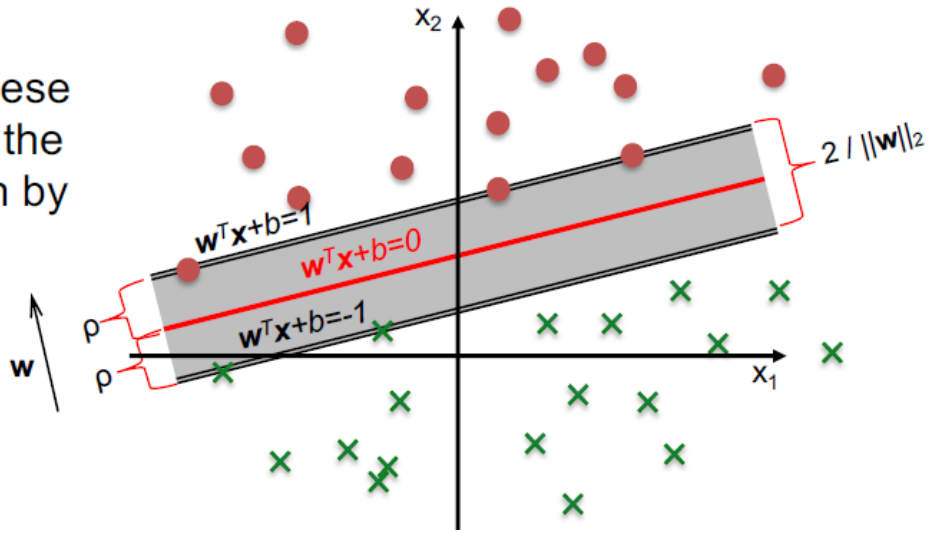
$$\begin{cases} \mathbf{w}^T\mathbf{x}+b \geq 1, & \text{if } y=1, \\ \mathbf{w}^T\mathbf{x}+b \leq -1, & \text{if } y=-1. \end{cases}$$

equivalent expression

$$y\left(\mathbf{w}^T\mathbf{x}+b\right) \geq 1$$

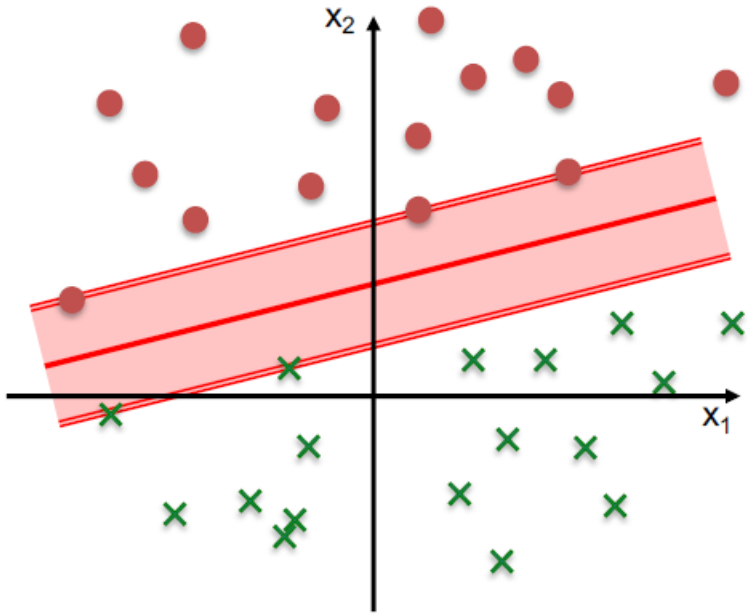
- The region bounded by these two hyperplanes is called the separation "margin", given by

$$\rho = \frac{2}{\|\mathbf{w}\|_2} = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$$



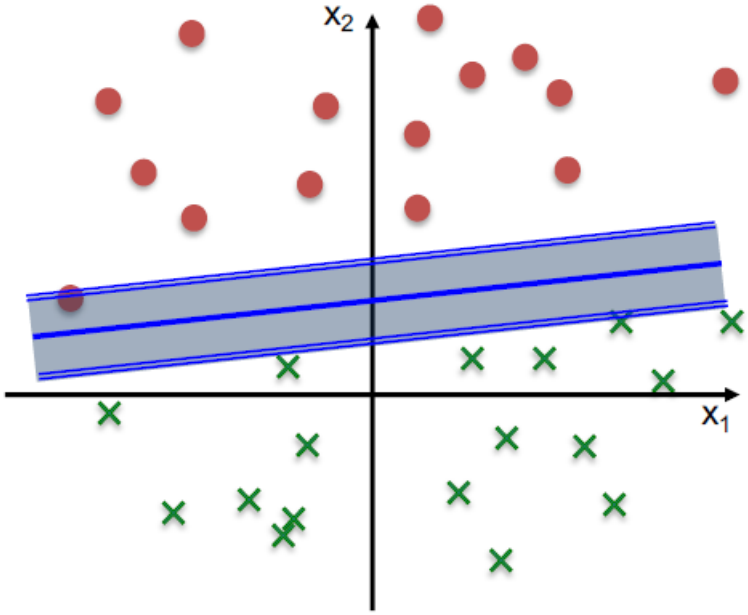
SVM

- Core Idea of SVM
  - The aim of SVM is simply to find an optimal hyperplane to separate the two classes of data points with the widest margin.



SVM

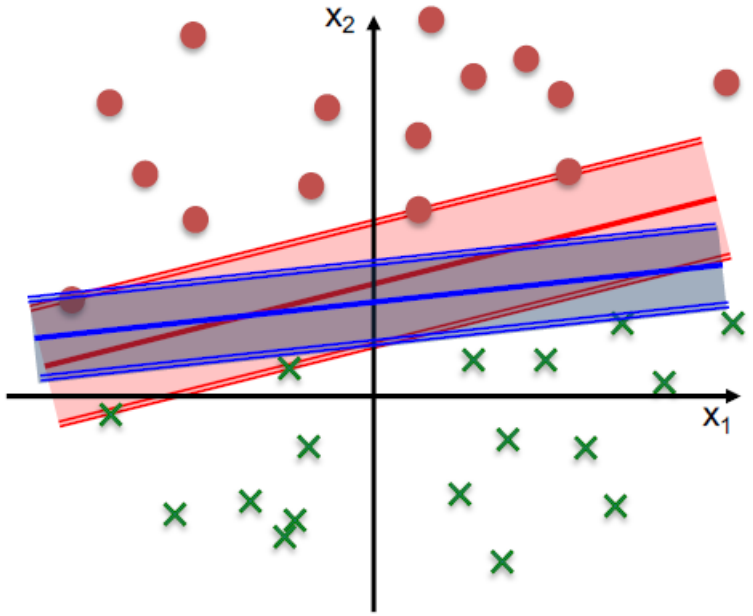
- Core Idea of SVM
  - The aim of SVM is simply to find an optimal hyperplane to separate the two classes of data points with the widest margin.



SVM

- Core Idea of SVM
  - The aim of SVM is simply to find an optimal hyperplane to separate the two classes of data points with the widest margin.

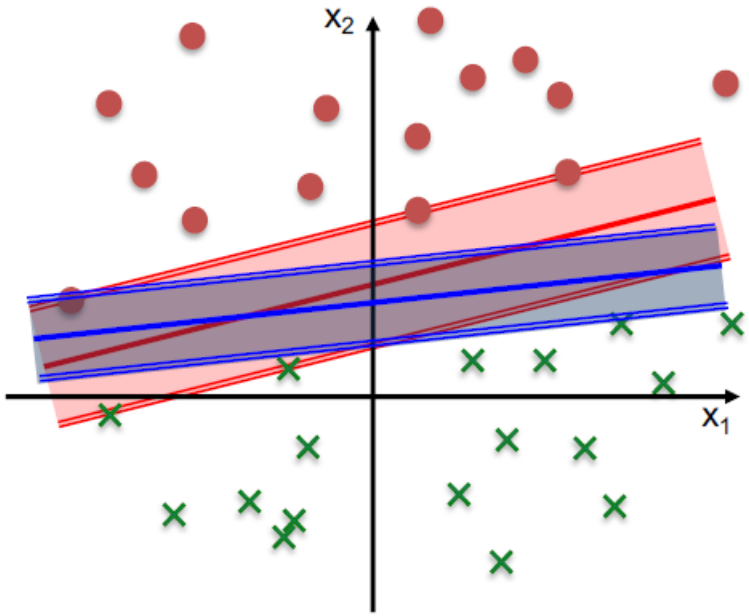
Which one is the better hyperplane?



SVM

- Hard-margin SVM
  - **Hard-margin SVM** finds an optimal hyperplane to fully separate the two classes of data points with the widest margin, by solving the following constrained optimisation problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i \in \{1,\ldots,N\} \end{aligned}$$



SVM

- Hard-margin SVM
  - **Hard-margin SVM** finds an optimal hyperplane to fully separate the two classes of data points with the widest margin, by solving the following constrained optimisation problem:

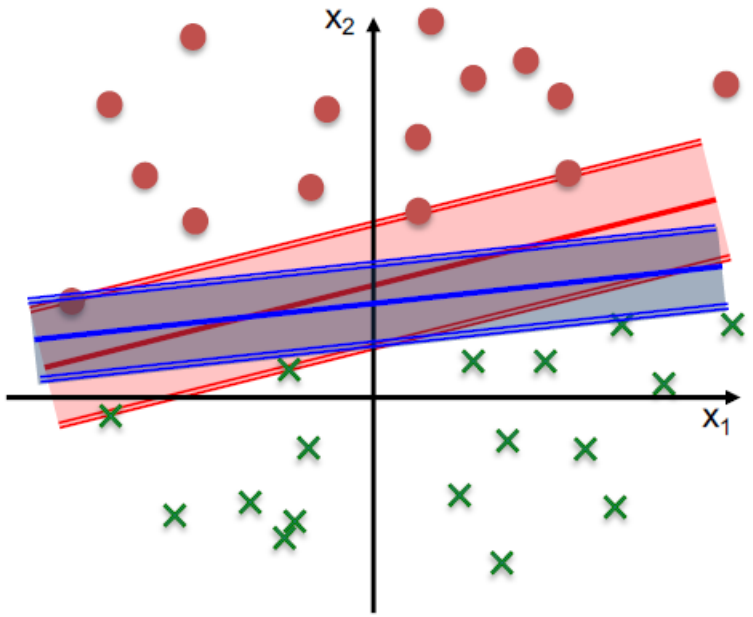
$$\text{margin}: \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$$

Margin maximisation

$$\min_{\mathbf{w},b} \ \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

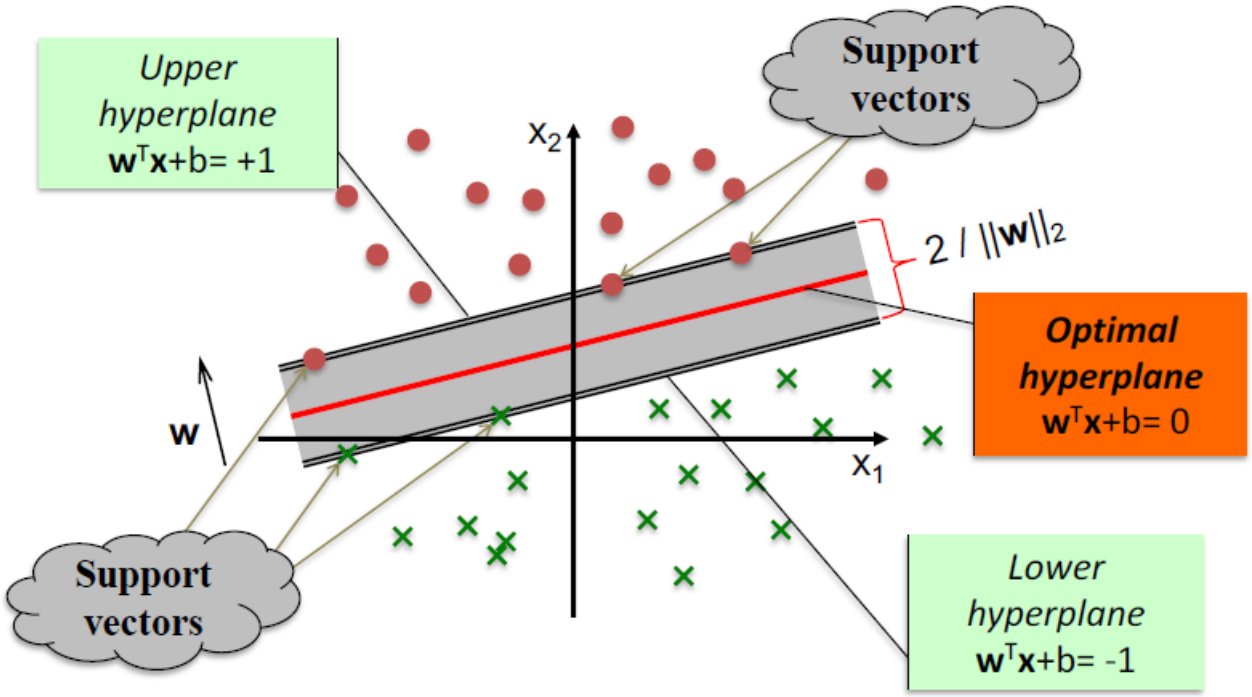
$$\text{s.t.} \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i \in \{1,\ldots,N\}$$

Stopping training samples from falling into the margin.



SVM

- Support Vectors
  - Support vectors: training points that satisfy $y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) = 1$
  - These points are the most difficult to classify and are very important for the location of the optimal hyperplane:



- Hard-margin SVM Training
  - Hard-margin SVM training: the process of solving the following constrained optimisation problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i \in \{1,\ldots,N\} \end{aligned}$$

How to derive the dual form can be found in the SVM notes as optional reading materials.

  - The above problem is solved by solving a dual problem as shown below.

$$\text{Dual problem} \begin{cases} \max\limits_{\lambda \in \Re^N} \sum\limits_{i=1}^{N} \lambda_i - \frac{1}{2}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad \sum\limits_{i=1}^{N} \lambda_i y_i = 0 \\ \qquad \lambda_i \geq 0 \end{cases}$$

SVM

- Quardratic Programing (QP)
  - The dual problem is called a quardratic programing (QP) problem in optimisation
  - There are many QP solvers available: https://en.wikipedia.org/wiki/Quadratic_programming
  - The new variables $\{\lambda_i\}_{i=1}^{N}$ are called Lagrangian multipliers. They should be positive numbers.
  - A fixed relationship exists between $\boldsymbol{w}$, b and $\{\lambda_i\}_{i=1}^{N}$

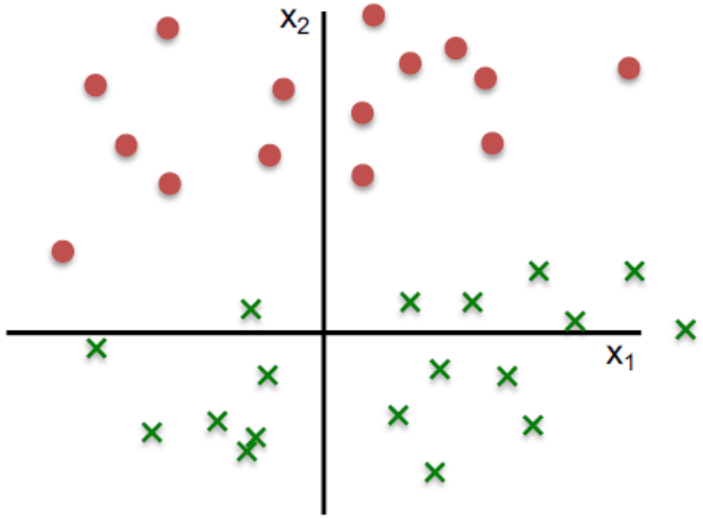
One way to solve the QP problem for SVM can be found in the SVM notes as optional reading materials.

SVM

- Support Vector Machines (SVM)
  - Soft-margin SVM for linear non-separable cases
  - Kernel SVM for nonlinear cases

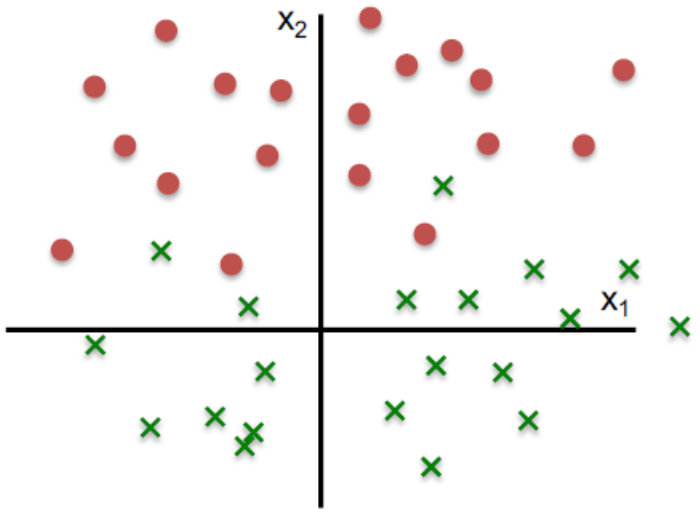
SVM

- We can handle simple cases like this:



separable data patterns

What if the data points look like this?



non-separable data patterns

In practice, no datasets are ideally linearly separable. This means that there always will be some data points misclassified by a linear hyperplane.

SVM

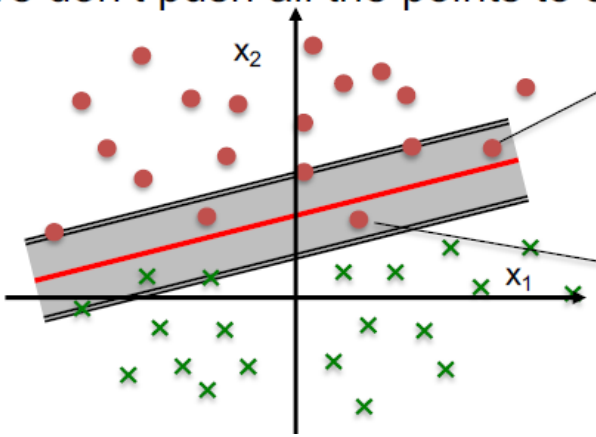
- Soft-margin SVM
  - We use the slack variable $\xi_i \geq 0$ (i=1,2,…N), each of which measures the deviation of the i-th point from the ideal situation, to relax the hard-margin SVM constraints as:

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i, & \text{if } y_i = 1, \\ \mathbf{w}^T\mathbf{x}_i + b \leq -(1 - \xi_i), & \text{if } y_i = -1. \end{cases}$$

equivalent expression

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$$

  - We don't push all the points to stay outside the margin any more.



Point within region of separation, but still in the right side: $0<\xi_i\leq 1$

Point in the wrong side of the decision boundary: $\xi_i>1$

Soft-margin SVM

- In addition to maximising the margin as before, we need to keep all slacks ξ_i as small as possible to minimise the classification errors. The modified SVM optimisation problem becomes:

$$\min_{(\mathbf{w},b)\subset\Re^{d+1},\ \{\xi_i\}_{i=1}^N} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad \left.\begin{array}{l} y_i\left(\mathbf{w}^T\mathbf{x}+b\right) \geq 1-\xi_i \\ \xi_i \geq 0 \end{array}\right\} \forall i \in \{1,\ldots,N\}$$

C≥0 is a user defined parameter, which controls the **regularisation**. This is the trade-off between complexity and nonseparable patterns.

- The above constrained optimisation problem can be converted to a QP problem.

Dual problem

$$\max_{\lambda\subset\Re^N} \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\lambda_i y_i = 0$$

$$0 \leq \lambda_i \leq C$$

Soft-margin SVM

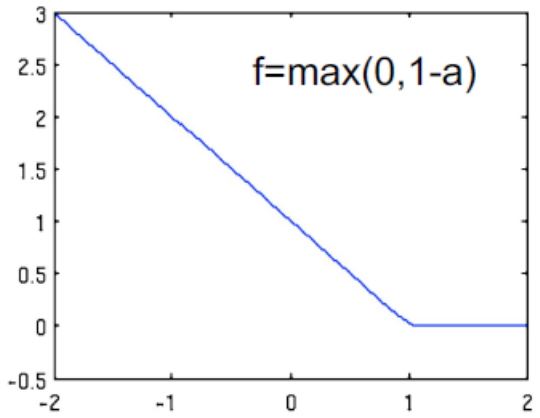
SVM

- Another SVM Formulation

hard-margin SVM

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{s.t.} \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i \in \{1,\ldots,N\}$$



- Use hinge loss:

$$\min_{(\mathbf{w},\,b)\in\Re^{d+1}} C\sum_{i=1}^{N}\max\left(0,1-y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right) + \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

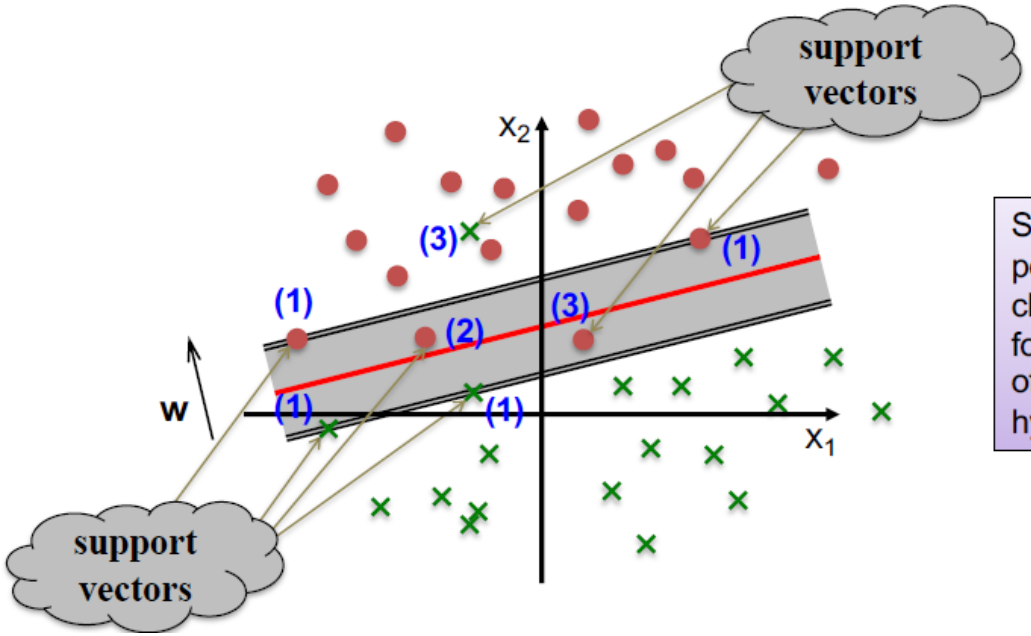
regularisation parameter (hyper-parameter)

hinge loss

regularisation term

SVM

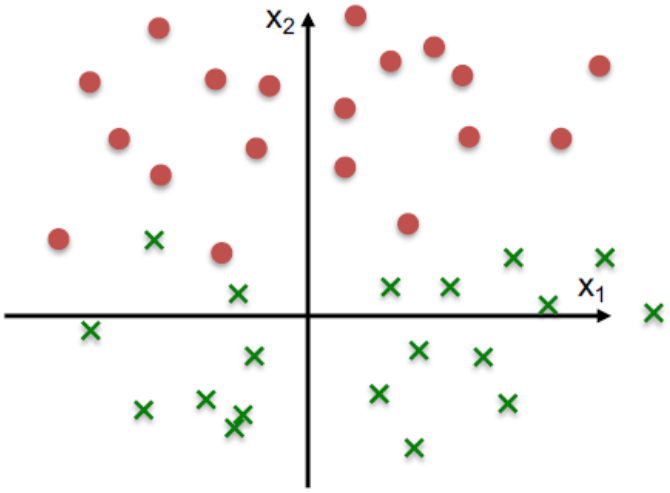
- Support Vectors
  - Support vectors: training points that satisfy $y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \leq 1$
  - These points either distribute along one of the two parallel hyperplane (1), or fall within the margin (2), or stay in the wrong side of the separating hyperplane (3).



Support vectors represent points that are difficult to classify and are important for deciding the location of the separating hyperplane.

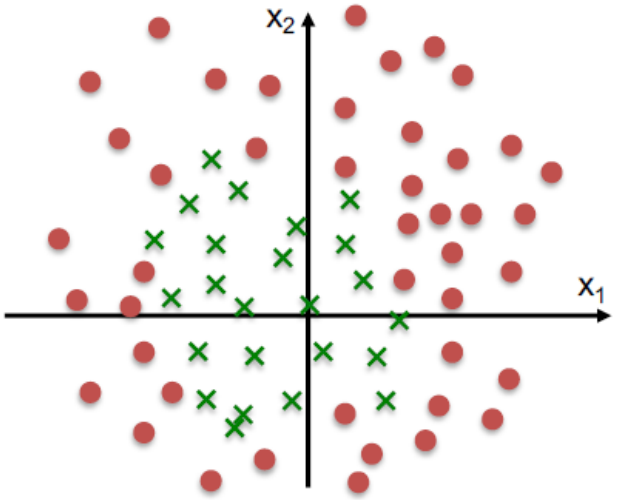
SVM

- We can handle linear cases like this:



linear data patterns

What if the data points look like this?



non-linear data patterns

SVM

- Kernel SVM

  - The SVM dual problem in the original space:

$$\text{Dual problem} \begin{cases} \max\limits_{\lambda \sqsubset \Re^N} \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j . y_i . y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad \sum_{i=1}^{N} \lambda_i . y_i = 0 \\ \quad 0 \le \lambda_i \le C \end{cases}$$

Move from the original space to a kernel induced space.

  - Kernel SVM with the modified dual problem:

$$\text{Dual problem} \begin{cases} \max\limits_{\lambda \sqsubset \Re^N} \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j . y_i . y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \\ \text{s.t.} \quad \sum_{i=1}^{N} \lambda_i . y_i = 0 \\ \quad 0 \le \lambda_i \le C \end{cases}$$

$$\max\limits_{\lambda \sqsubset \Re^N} \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j . y_i . y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \lambda_i . y_i = 0$$

$$0 \le \lambda_i \le C$$

- SVM Decision Function
  - After solving the QP problem for SVM, we compute the optimal values for the multipliers $\left\{\lambda_i^*\right\}_{i=1}^N$
  - Linear SVM decision function:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i^* y_i \mathbf{x}_i^T \mathbf{x} + b$$

Bias parameter b can be estimated from support vectors.

  - Nonlinear SVM decision function:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Example

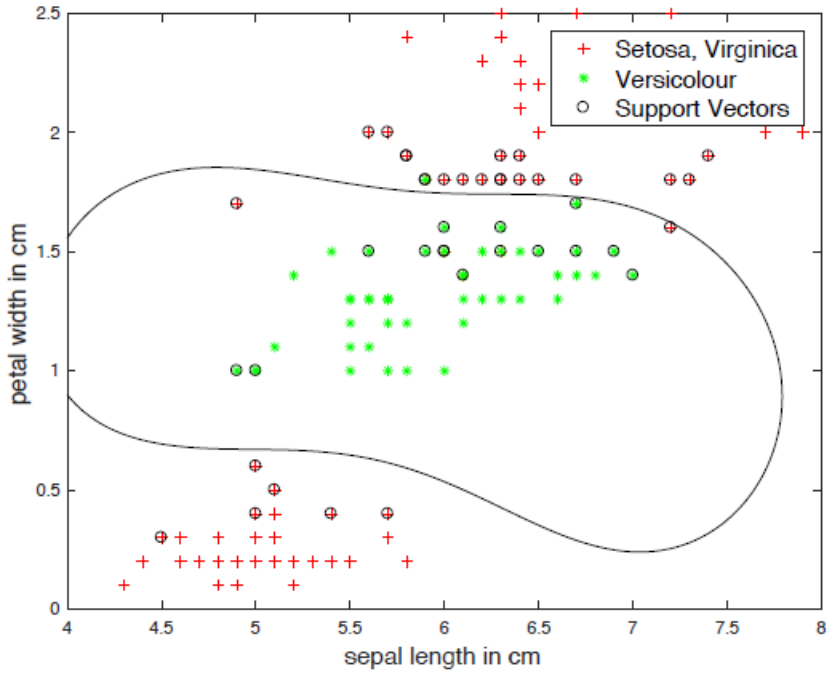
Polynomial: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i^* y_i \left(\mathbf{x}^T \mathbf{x}_i + 1\right)^p + b$

Gaussion: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i^* y_i \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|_2^2}{2\sigma^2}\right) + b$

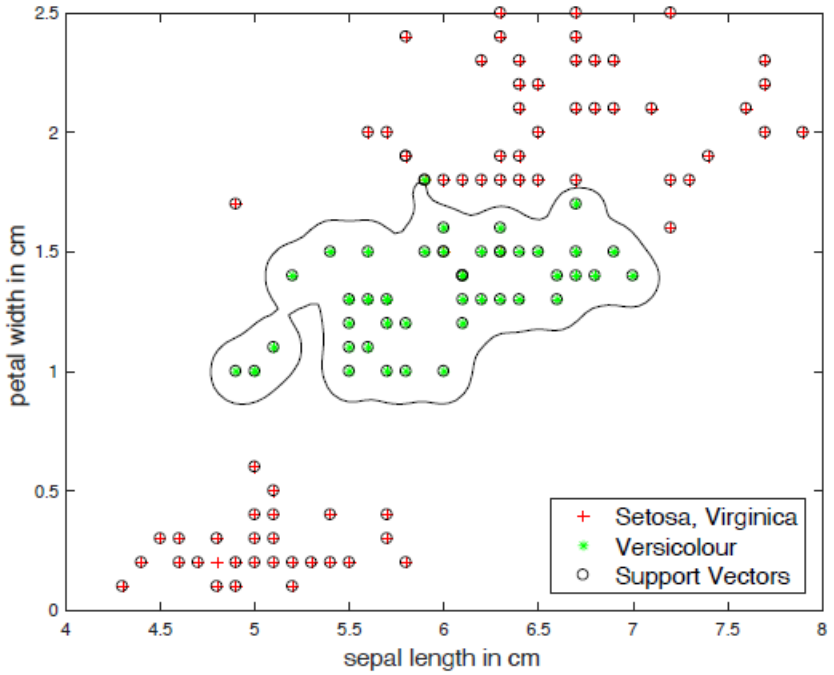
SVM

- Iris Classification Example:

Soft-margin SVM with Gaussian kernel (σ=1, C=1)



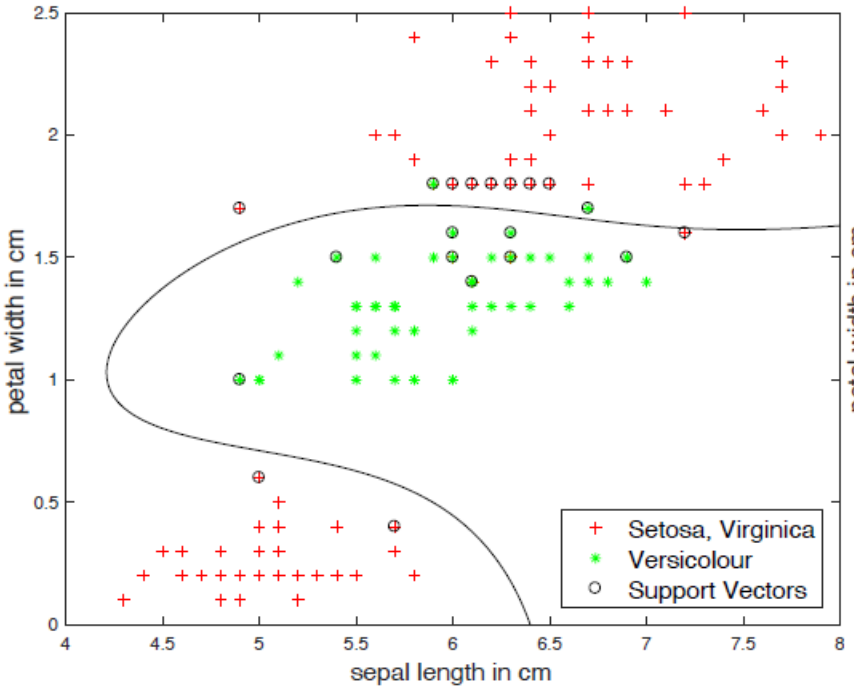
Soft-margin SVM with Gaussian kernel (σ=0.1, C=1)



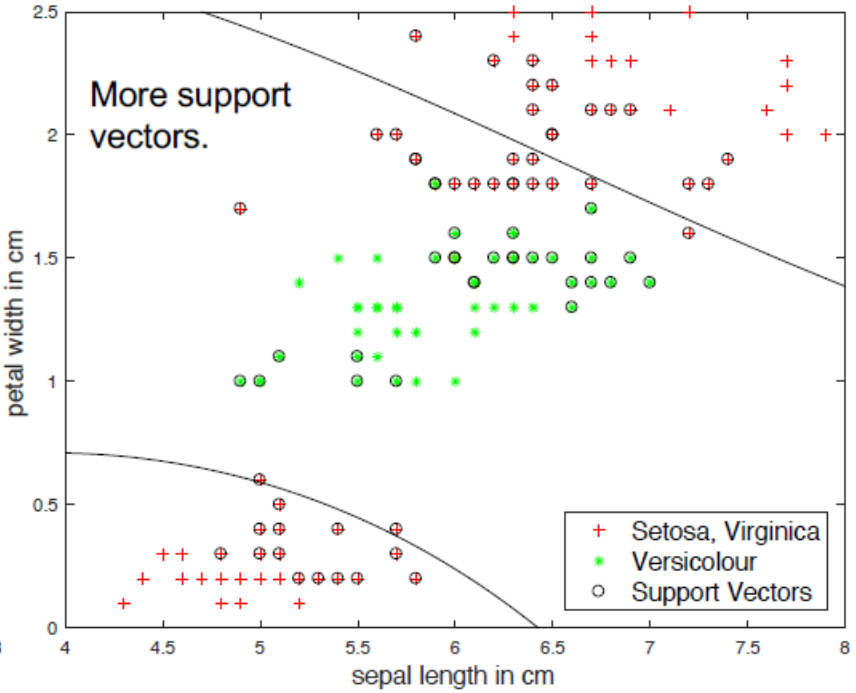
SVM

- Iris Classification Example:

Soft-margin SVM with polynomial kernel (p=1, C=1)



Soft-margin SVM with polynomial kernel (p=1, C=0.01)



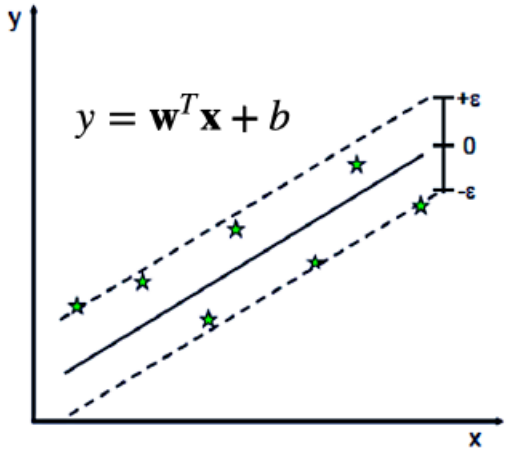
SVM

- Support Vector Regression
  - One simple way to formulate a linear SVM regressor:

$$\min_{(\mathbf{w},b)\in R^{d+1}} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{subject to} \left| \mathbf{w}^T\mathbf{x}_i + b - y_i \right| \leq \epsilon, i = 1,2,\ldots,N$$

The parameter ϵ is a hyperparameter.



  - It can be extended to fit nonlinear data patterns by using kernel trick.
  - Scikit Learn provides SVR tool.

    https://scikit-learn.org/stable/modules/svm.html

SVM

- Support Vector Machines (SVM)
  - ROC analysis
  - Comparison of machine learning models
  - No free lunch theorem

SVM

- Classification Performance Evaluation
  - Given a binary classifier, we vary its decision threshold.

$$\hat{y} = \begin{cases} 1, & f(\mathbf{x}) \geq T \\ -1, & f(\mathbf{x}) < T \end{cases}$$

  - Your model can be a logistic regression model, an artificial neural network, or an SVM classifier, etc.
  - You can evaluate the classification performance without fixing T, but observing **performance change over a varying value of T**.

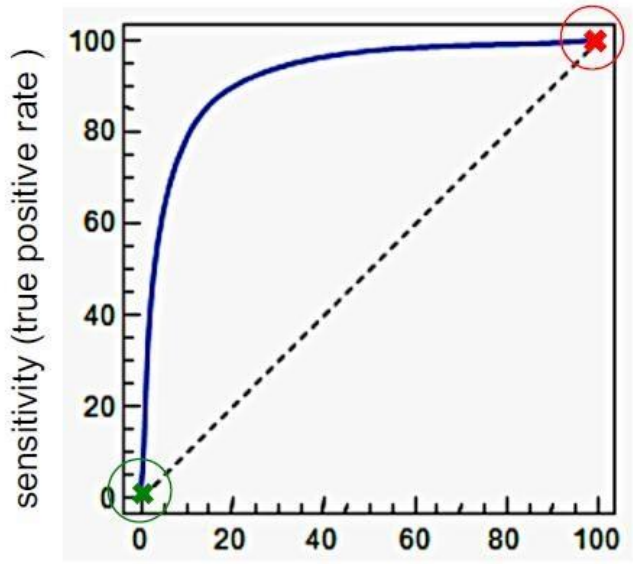
SVM

- ROC Analysis
  - For each threshold value, we can collect a pair of measures:
    - (1- specificity, sensitivity), sensitivity is another name of recall
    - or (false positive rate, true positive rate), (FPR, TPR)
  - Receiver operating characteristic (ROC) curve is a graphical plot that illustrates pairs of the above collected measures, as the decision threshold varies.
    - 1- specificity in x-axis and sensitivity in y-axis
    - or FPR in x-axis and TPR in y-axis

SVM

- ROC Analysis
  - Example of an ROC curve (in percentage)

From https://www.medcalc.org/manual/roc-curves.php



Classify everybody into **positive** class. Correspondingly, the capability of correctly classifying samples into negative class drops to zero.

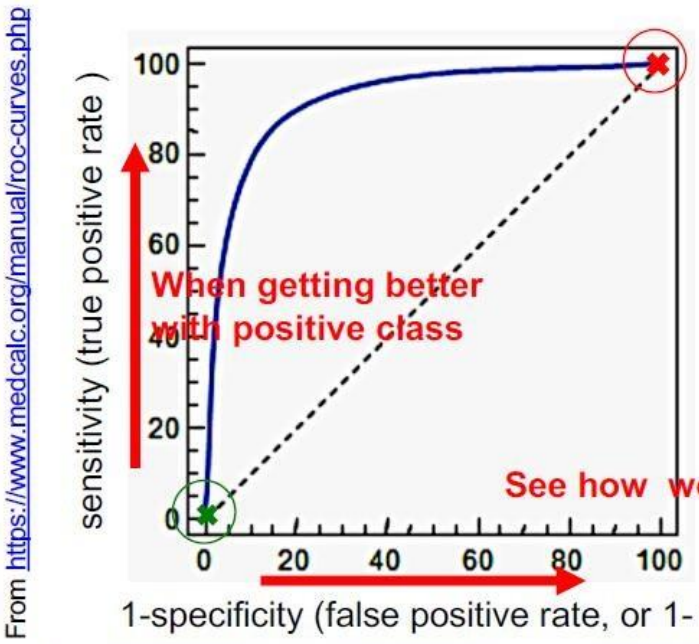
$$\text{sensitivity}=\text{TPR}=\frac{\text{TP}}{\text{the number of real positives}}$$

$$\text{specificity}=\text{TNR}=1-\text{FPR}=\frac{\text{TN}}{\text{the number of real negatives}}$$

Classify everybody into **negative** class. Correspondingly, the capability of correctly classifying samples into positive class drops to zero.

SVM

- ROC Analysis
  - Example of an ROC curve (in percentage)



Classify everybody into **positive** class. Correspondingly, the capability of correctly classifying samples into negative class drops to zero.

$$\text{sensitivity=TPR=}\frac{\text{TP}}{\text{the number of real positives}}$$

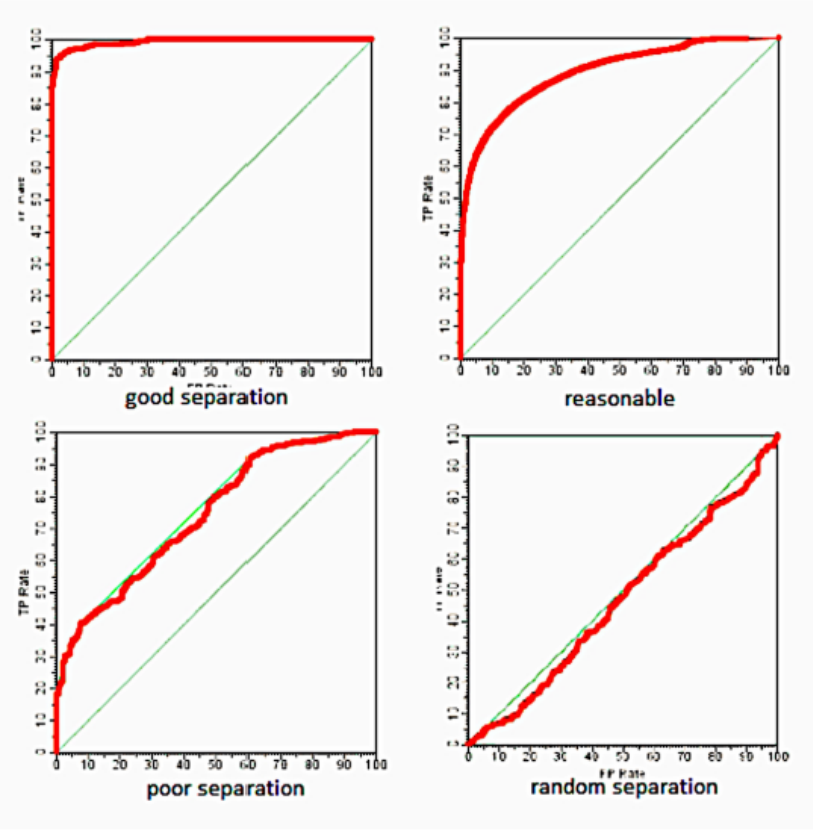
$$\text{specificity=TNR=1-FPR=}\frac{\text{TN}}{\text{the number of real negatives}}$$

See how worse becomes with negative class!

Classify everybody into **negative** class. Correspondingly, the capability of correctly classifying samples into positive class drops to zero.

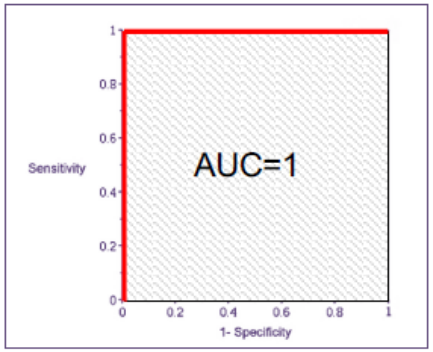
SVM

- ROC Analysis
  - Example of an ROC curve



We wish to rapidly getting better with positive class and slowly getting worse with negative class.

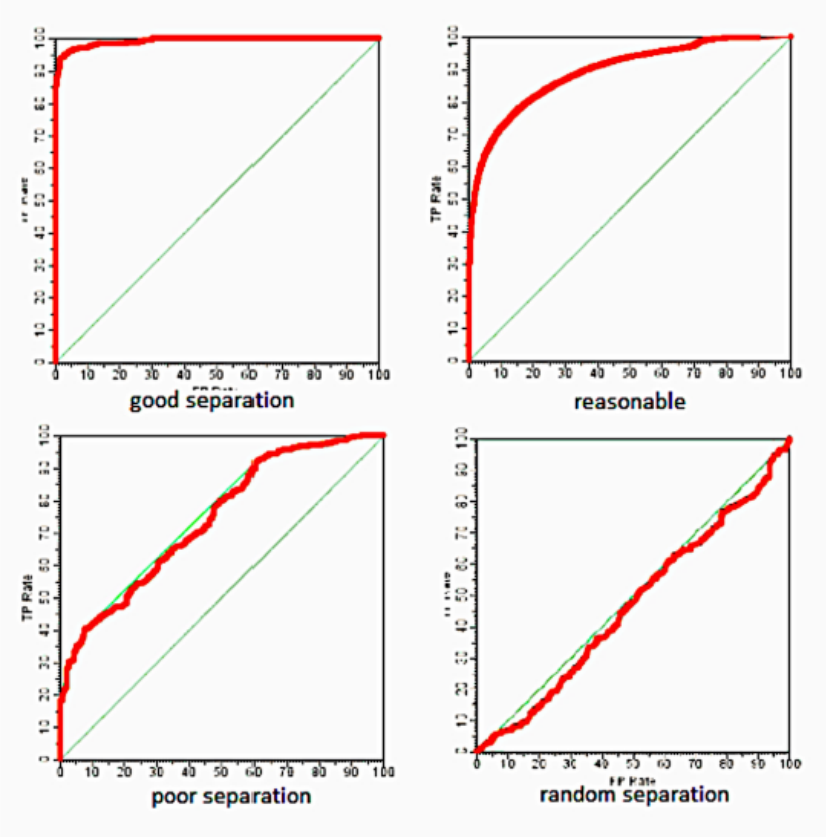
Area under the ROC curve (AUC, A_z) measures the balance. The higher the better. Maximum is 1.



ROC implementation: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

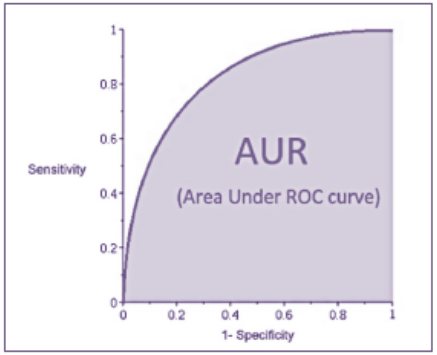
SVM

- ROC Analysis
  - Example of an ROC curve



We wish to rapidly getting better with positive class and slowly getting worse with negative class.

Area under the ROC curve (AUC, A_z) measures the balance. The higher the better.



ROC implementation: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

SVM

- Comparison: k-NN
- Pros (+):
  - A simple intuitive method with no training.
  - Can be used for both classification and regression.
  - Can handle both linear and nonlinear data patterns.
  - Easy to implement for multi-class classification.
  - There are only two things to decide, which can be relatively easy.
    - What hyper-parameter value *k* to use?
    - What distance measure to use?
- Cons (-):
  - Slow with large-scale data.
    - The computational complexity of distance calculation suffers from curse of dimensionality.
    - The computational complexity of neighbour search suffers from training data size.
  - High memory cost (needs to store all the training data)
  - Not good at dealing with imbalanced data.
  - Sensitive to outliers.

SVM

- Comparison: (Regularised) Linear Least Squares
- Pros (+):
  - The most widely used modelling method.
  - It is what most people mean when they say they have used “regression”, “linear regression” or “least squares” to fit a model to their data.
  - Can be used for both regression and classification.
  - Makes efficient use of the data. Good results can be obtained with relatively small data.
  - Easy to explain and understand.
  - Low computational cost: fast training and prediction.
  - Low memory.
  - In the least squares case, there is no hyper-parameter to set. In the regularised case, there are two things to decide: (1) the form of regularisation term (e.g., l_2, l_1), and regularisation parameter.
- Cons (-):
  - Limited model expressive power, cannot deal with nonlinear data patterns.
  - Sensitive to outliers.
  - When being used in classification, the performance is sometimes sensitive to how the target output is set, and it does not offer probabilistic interpretation.

SVM

- Comparison: SVM
- Pros (+):
  - Usually provide competitive performance. It is a good choice when we have no idea on the data.
  - Unlike ANN, it is not solved for local optima.
  - Scale relatively well to high dimensional data.
  - It can generalise.
  - Can handle both linear and nonlinear data patterns.
- Cons (-):
  - Need to choose a “good” kernel and select hyper-parameters, which are not easy.
  - Long training time for large-scale data.
  - Difficult to interpret the learned model.

SVM

- No Free Lunch Theorem
  - Questions:

> - If we are interested solely in the generalisation performance, is there any reason to prefer one classifier or learning algorithm over another?
> - If we make no prior assumptions about the nature of the classification task, can we expect any classification method to be superior or inferior overall?

  - No Free Lunch Theorem (D. Wolpert, 1996): The answer is NO!
  - There is no context-independent or problem-independent reason to favour one method over another. You **always** need to
    - Understand the data/task/problem to be processed/solved first!
    - Experiment with different machine learning models!

SVM

- Summary
  - Basic concepts
    - Geometric concepts relevant to hyperplane
    - Separation margin
    - Support vector
  - SVM models
    - Hard-margin vs soft-margin SVM classifier
    - Linear vs kernel SVM classifier
    - Support vector regressor
  - ROC Analysis
  - Comparison
  - No free lunch theorem

Content

- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

Unsupervised Learning: Cluster Analysis

- Cluster Analysis
  - Clustering basics
  - Between-cluster distance

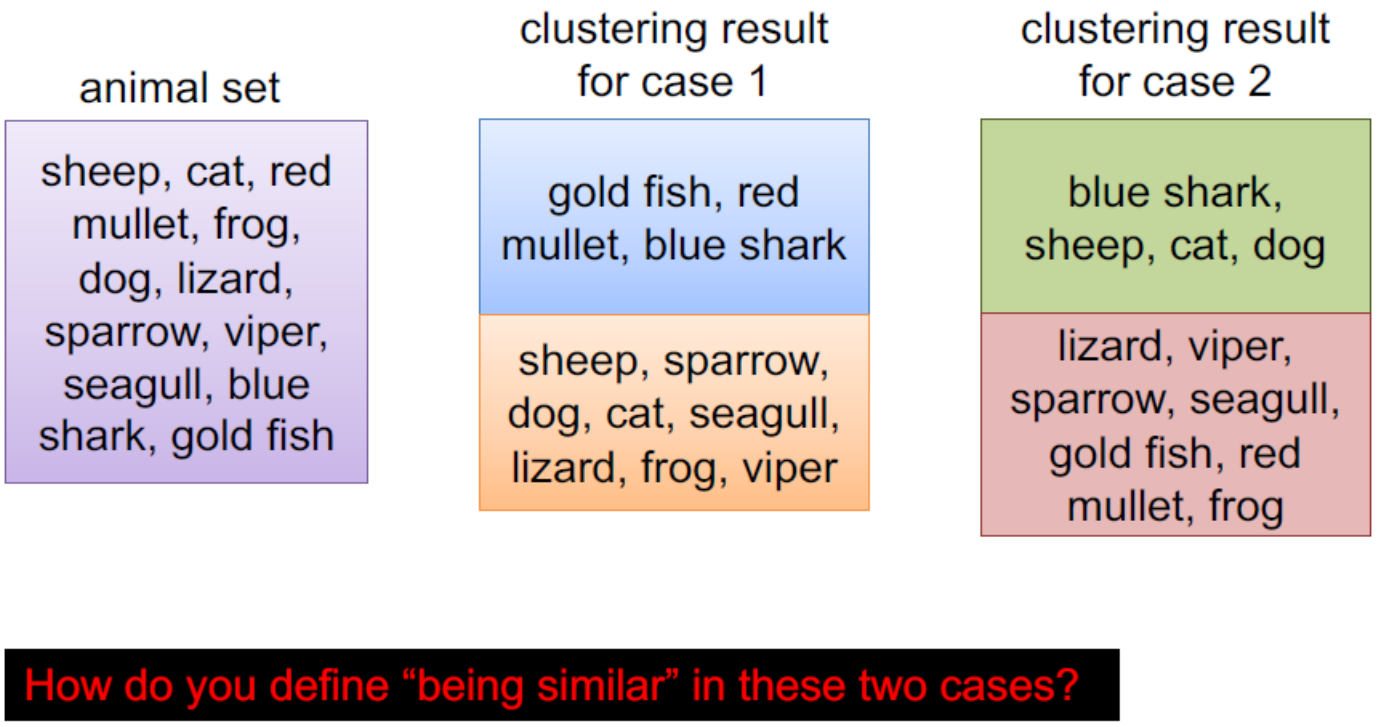
Cluster Analysis

- Cluster analysis (known as clustering) is the task of grouping a set of objects so that *objects in the same group are more similar to each other than to those in other groups.* We call each group a cluster.
- Clustering is unsupervised learning. There are no predefined classes and training data
- Need to define “similar”!
  - The objects are usually characterised by a set of features, corresponding to a set of data points in a high-dimensional space.
  - Distance (or similarity) values are computed between pairs of data points, serving as a way to define “similar”.

Examples:
- Euclidean distance
- Cosine similarity
- Minkowski distance
- Manhattan distance

Cluster Analysis

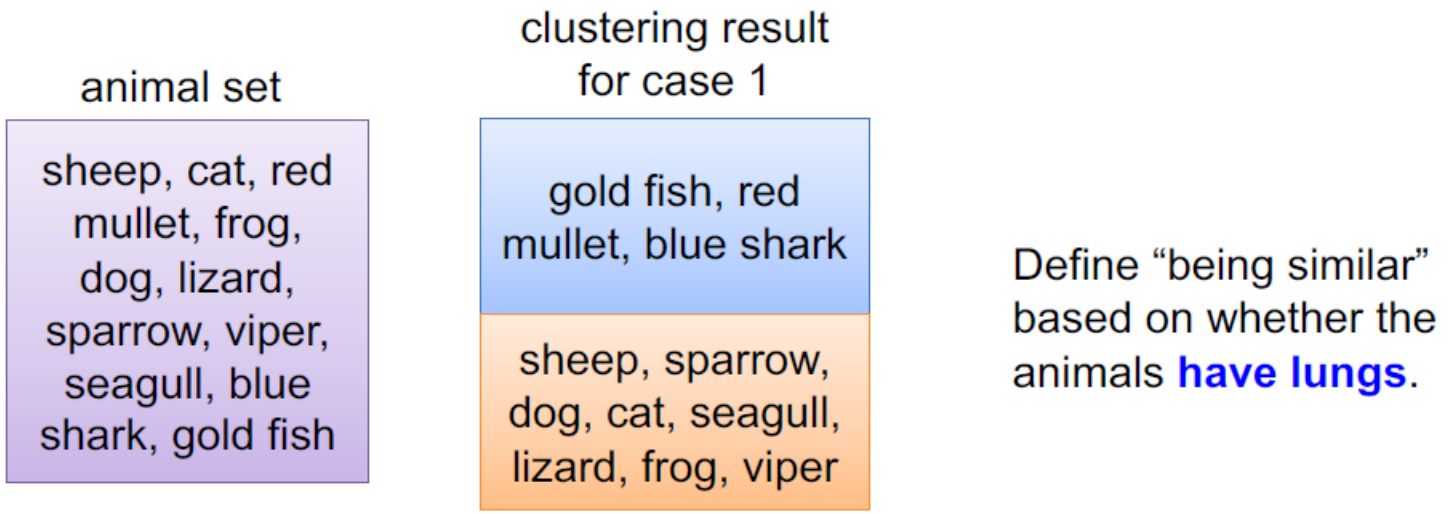
- Example: Clustering Animals
  - Given a set of animals, we would like to cluster these into two groups.



How do you define "being similar" in these two cases?

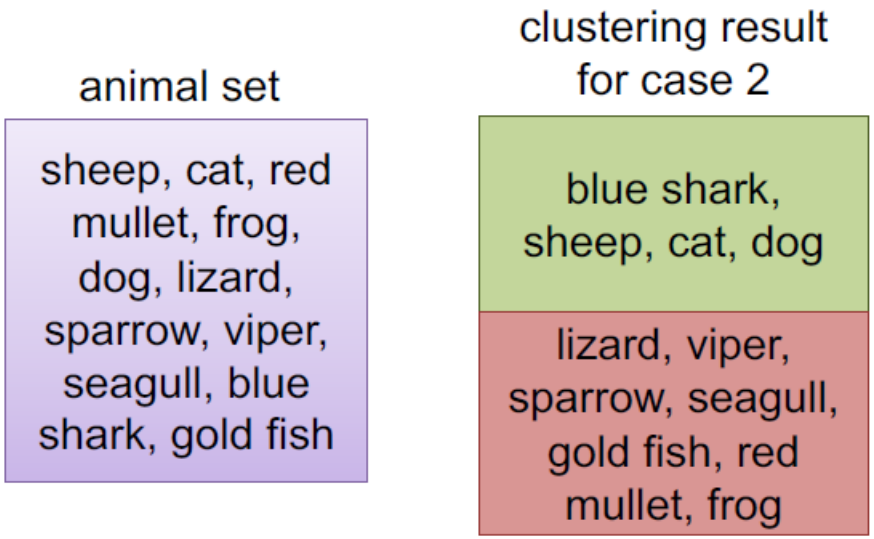
Cluster Analysis

- Example: Clustering Animals
  - Given a set of animals, we would like to cluster these into two groups.



Cluster Analysis

- Example: Clustering Animals
  - Given a set of animals, we would like to cluster these into two groups.



Define "being similar" based on how the animals **produce offspring**.

Cluster Analysis

- Example: Clustering Animals
  - Let's understand this clustering process from a feature/distance point of view.

Object ID	Object Instance	Feature: Whether there is a lung	Notation
1	sheep	1	$x_1 = 1$
2	cat	1	$x_2 = 1$
3	red mullet	0	$x_3 = 0$
4	frog	1	$x_4 = 1$
5	dog	1	$x_5 = 1$
6	lizard	1	$x_6 = 1$
7	sparrow	1	$x_7 = 1$
8	viper	1	$x_8 = 1$
9	seagull	1	$x_9 = 1$
10	blue shark	0	$x_{10} = 0$
11	gold fish	0	$x_{11} = 0$

1 for yes, 0 for no

Cluster Analysis

- Example: Clustering Animals
  - We calculate a distance matrix between the objects.

	sheep	cat	red mullet	frog	dog	lizard	sparrow	viper	seagull	blue shark	gold fish
sheep	0	0	1	0	0	0	0	0	0	1	1
cat	0	0	1	0	0	0	0	0	0	1	1
red mullet	1	1	0	1	1	1	1	1	1	0	0
frog	0	0	1	0	0	0	0	0	0	1	1
dog	0	0	1	0	0	0	0	0	0	1	1
lizard	0	0	1	0	0	0	0	0	0	1	1
sparrow	0	0	1	0	0	0	0	0	0	1	1
viper	0	0	1	0	0	0	0	0	0	1	1
seagull	0	0	1	0	0	0	0	0	0	1	1
blue shark	1	1	0	1	1	1	1	1	1	0	0
gold fish	1	1	0	1	1	1	1	1	1	0	0

For example:

Distance between “cat” and “sheep”:

$$\sqrt{(x_1 - x_2)^2} = \sqrt{(1-1)^2} = 0$$

Distance between “sheep” and “blue shark”:

$$\sqrt{(x_1 - x_3)^2} = \sqrt{(1-0)^2} = 1$$

Cluster Analysis

- Example: Clustering Animals
  - **Re-order** the distance matrix based on the clustering result 1.

	gold fish	red mullet	blue shark	sheep	sparrow	dog	cat	seagull	lizard	frog	viper
gold fish	0	0	0	1	1	1	1	1	1	1	1
red mullet	0	0	0	1	1	1	1	1	1	1	1
blue shark	0	0	0	1	1	1	1	1	1	1	1
sheep	1	1	1	0	0	0	0	0	0	0	0
sparrow	1	1	1	0	0	0	0	0	0	0	0
dog	1	1	1	0	0	0	0	0	0	0	0
cat	1	1	1	0	0	0	0	0	0	0	0
seagull	1	1	1	0	0	0	0	0	0	0	0
lizard	1	1	1	0	0	0	0	0	0	0	0
frog	1	1	1	0	0	0	0	0	0	0	0
viper	1	1	1	0	0	0	0	0	0	0	0

clustering result 1

gold fish, red mullet, blue shark

sheep, sparrow, dog, cat, seagull, lizard, frog, viper

- Similar within the same cluster.
- Dissimilar between different clusters.

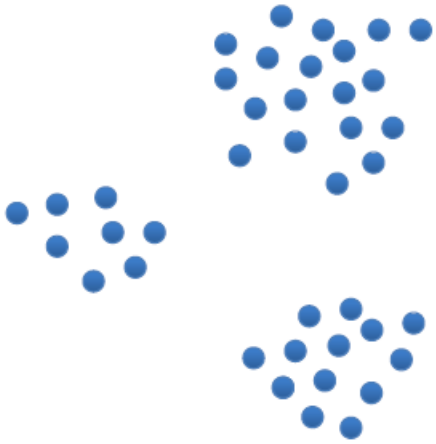
Cluster Analysis

- Key Tasks in Clustering
  - Define an appropriate distance (similarity) measure.
  - Identify the “natural” cluster number.
  - Find a way to group objects into “sensible” clusters. Different ways correspond to different clustering algorithms.
  - Assess whether the clustering output is a good one (evaluation).

Cluster Analysis

- Key Tasks in Clustering
  - Define an appropriate distance (similarity) measure.
  - Identify the “natural” cluster number.
  - Find a way to group objects into “sensible” clusters. Different ways correspond to different clustering algorithms.
  - Assess whether the clustering output is a good one (evaluation).

Example: Similarity between points is reflected by Euclidean distance

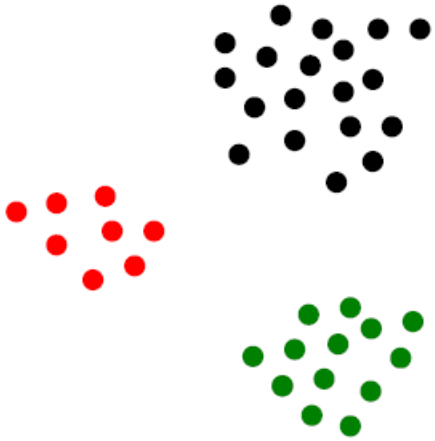


How many clusters here?

Cluster Analysis

- Key Tasks in Clustering
  - Define an appropriate distance (similarity) measure.
  - Identify the “natural” cluster number.
  - Find a way to group objects into “sensible” clusters. Different ways correspond to different clustering algorithms.
  - Assess whether the clustering output is a good one (evaluation).

Another Example: Similarity between points is reflected by Euclidean distance

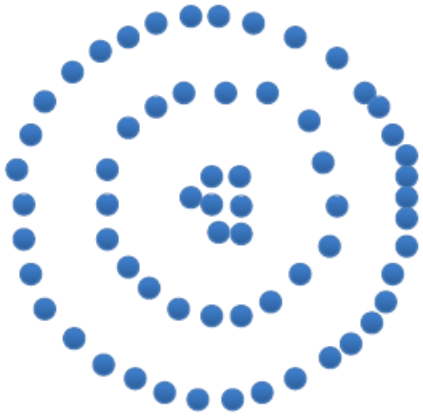


It is obvious in this case that there are 3 natural clusters.

Cluster Analysis

- Key Tasks in Clustering
  - Define an appropriate distance (similarity) measure.
  - Identify the “natural” cluster number.
  - Find a way to group objects into “sensible” clusters. Different ways correspond to different clustering algorithms.
  - Assess whether the clustering output is a good one (evaluation).

Another example:

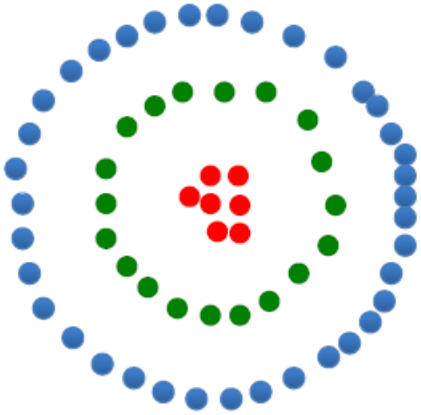


How many clusters here?

Cluster Analysis

- Key Tasks in Clustering
  - Define an appropriate distance (similarity) measure.
  - Identify the “natural” cluster number.
  - Find a way to group objects into “sensible” clusters. Different ways correspond to different clustering algorithms.
  - Assess whether the clustering output is a good one (evaluation).

Another example:



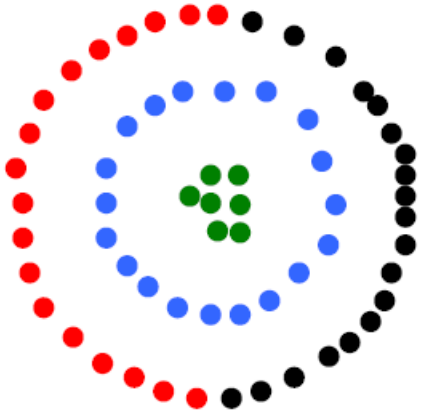
Different clusters are distinguished by colors.

Is this a reasonable clustering output?

Cluster Analysis

- Key Tasks in Clustering
  - Define an appropriate distance (similarity) measure.
  - Identify the “natural” cluster number.
  - Find a way to group objects into “sensible” clusters. Different ways correspond to different clustering algorithms.
  - Assess whether the clustering output is a good one (evaluation).

Another example:



Different clusters are distinguished by colors.

Is this a reasonable clustering output?

For data points with complex patterns, different algorithms are likely to produce different clustering results.

Cluster Analysis

- Comments on Clustering
  - Often which distance (similarity) measure to use and what cluster number to adopt is treated as a model selection process, where cluster number is a hyper-parameter.
  - There are some clustering algorithms that attempt to decide a cluster number for you, but they requires to set other type of hyper-parameters.
  - A good collection of implemented clustering algorithms is provided by Sikit-learn.

https://scikit-learn.org/stable/modules/clustering.html#clustering

Cluster Analysis

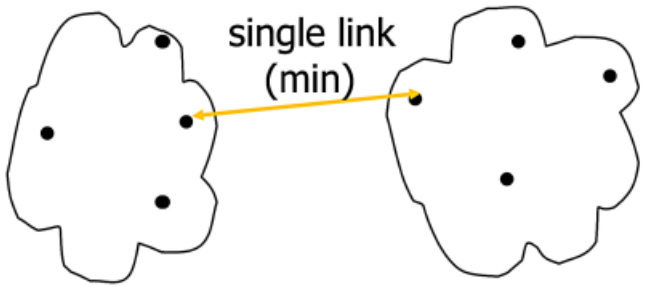
- Cluster Distance Measures

Cluster Analysis

- Cluster Distance Measures
  - **Question:** Knowing the distance (or similarity) measure between two data points, how to measure **the distance (or similarity) between two data clusters**?
  - Single-link measure: It is the smallest distance between a data point in one cluster and a data point in the other cluster.

$$d\left(\text{cluster}_i, \text{cluster}_j\right) = \min_{\substack{p \in \text{cluster}_i \\ q \in \text{cluster}_j}} d\left(\mathbf{x}_p, \mathbf{x}_q\right)$$

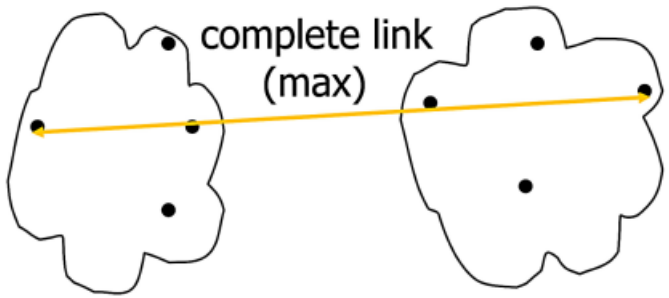
$$d\left(\text{cluster}_i, \text{cluster}_i\right) = 0$$



Cluster Analysis

- Cluster Distance Measures
  - **Question:** Knowing the distance (or similarity) measure between two data points, how to measure **the distance (or similarity) between two data clusters**?
  - Complete-link measure: It is the largest distance between a data point in one cluster and a data point in the other cluster.

$$d\left(\text{cluster}_i, \text{cluster}_j\right) = \max_{\substack{p \in \text{cluster}_i \\ q \in \text{cluster}_j}} d\left(\mathbf{x}_p, \mathbf{x}_q\right)$$

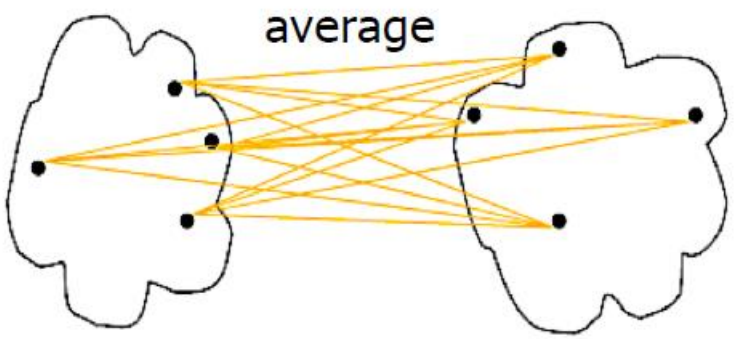


Cluster Analysis

- Cluster Distance Measures
  - **Question:** Knowing the distance (or similarity) measure between two data points, how to measure **the distance (or similarity) between two data clusters**?
  - Average-link measure: It is the averaged distance between data points in one cluster and data points in the other cluster.

$$d\left(\text{cluster}_i, \text{cluster}_j\right) = \frac{1}{n_i n_j} \sum_{\substack{p \in \text{cluster}_i \\ q \in \text{cluster}_j}} d\left(\mathbf{x}_p, \mathbf{x}_q\right)$$

n_i: number of data points in cluster i.



Cluster Analysis

- Example
  - There are five data points each has one single feature.

	a	b	c	d	e
Feature	1	2	4	5	6

They are grouped to two clusters:

- Cluster 1 (C1): {a, b}
- Cluster 2 (C2): {c, d, e}

Calculate the single-link, multi-link and average-link distance between the two clusters, by using Euclidean distance to calculate between-point distance.

Euclidean Distance

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

$$d_{single}(C_1,C_2) = \min\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \min\{3, 4, 5, 2, 3, 4\} = 2$$
$$d_{multi}(C_1,C_2) = \max\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \max\{3, 4, 5, 2, 3, 4\} = 5$$
$$d_{average}(C_1,C_2) = \frac{d(a,c)+d(a,d)+d(a,e)+d(b,c)+d(b,d)+d(b,e)}{6}$$
$$= \frac{3+4+5+2+3+4}{6} = \frac{21}{6} = 3.5$$

Unsupervised Learning: K-Means Clustering

- History: k-means Clustering
  - Simplest and most frequently used clustering algorithm.
  - The idea dates back to Hugo Steinhaus in 1956 (a Jewish-Polish mathematician and educator, 1887-1972).


Hugo Steinhaus (1968)

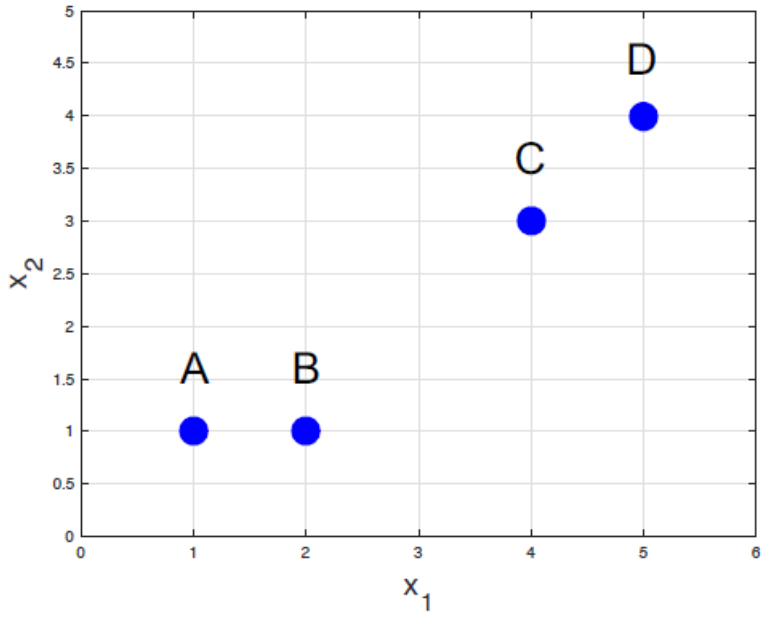
H. Steinhaus, Sur la division des corps matériels en parties. Bull. Acad. Polon. Sci. (in French). 4 (12): 801-804, 1956.

  - It was first used by James MacQueen in 1967.

J. B. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1.
University of California Press. pp. 281-297, 1967.

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.

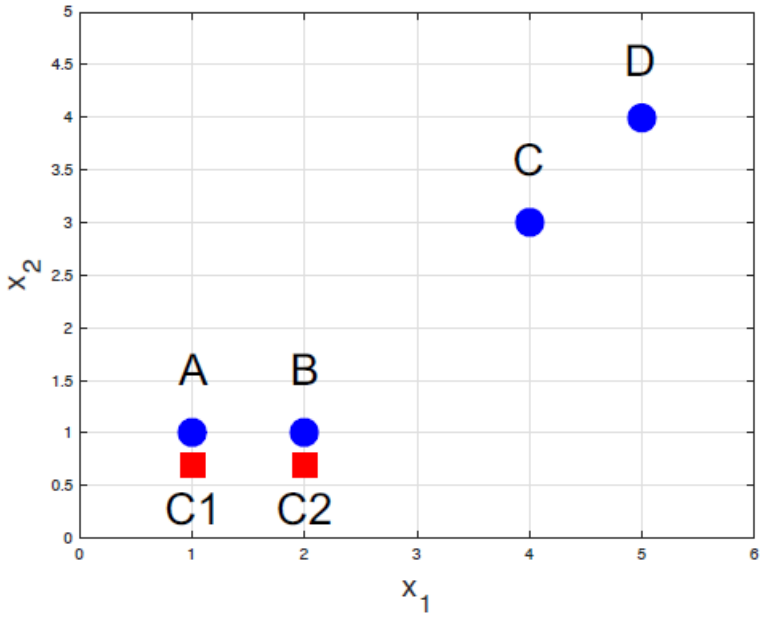


A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

Goal: Group a given set of points to a fixed number of clusters.

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



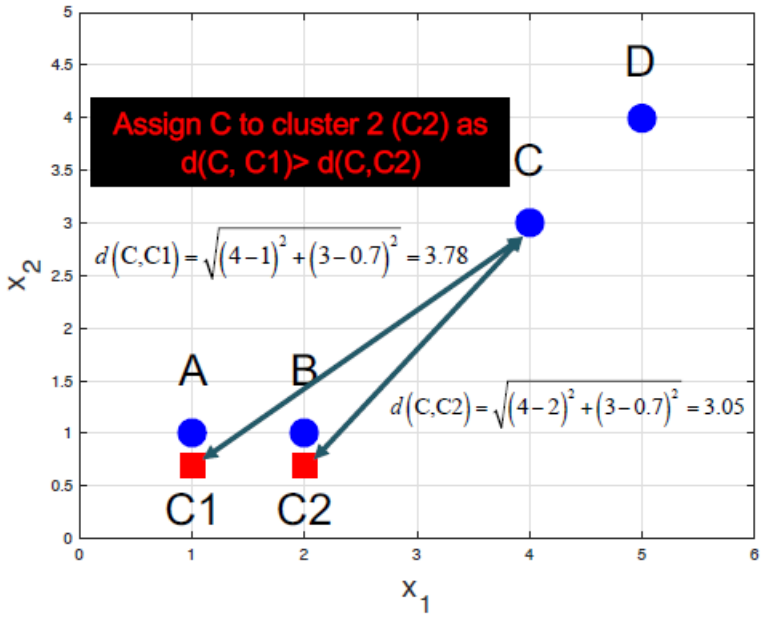
A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the dissimilarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

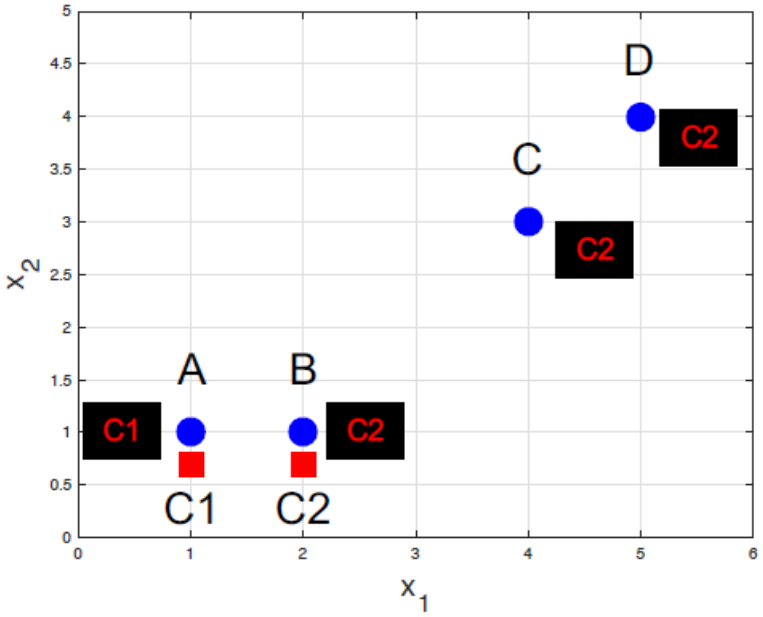
Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the (dis)similarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

- Step 1: Calculate distances (or similarities) between the data points and the cluster center points.
- Step 2: Find the nearest cluster center to each data point, and assign the data point to that cluster.

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

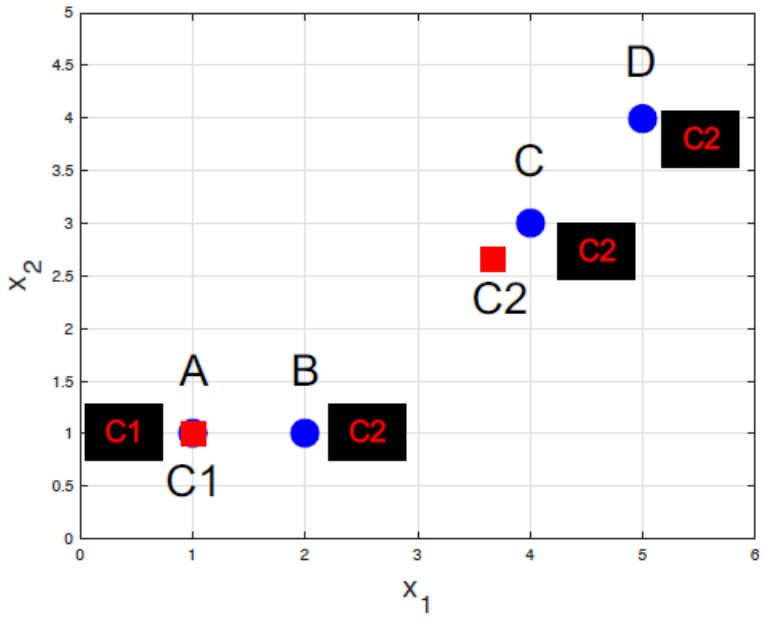
Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the (dis)similarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

- Step 1: Calculate distances (or similarities) between the data points and the cluster center points.
- Step 2: Find the nearest cluster center to each data point, and assign the data point to that cluster.

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the (dis)similarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

- Step 3: Calculate the new cluster center for each cluster, by **averaging** its member points.

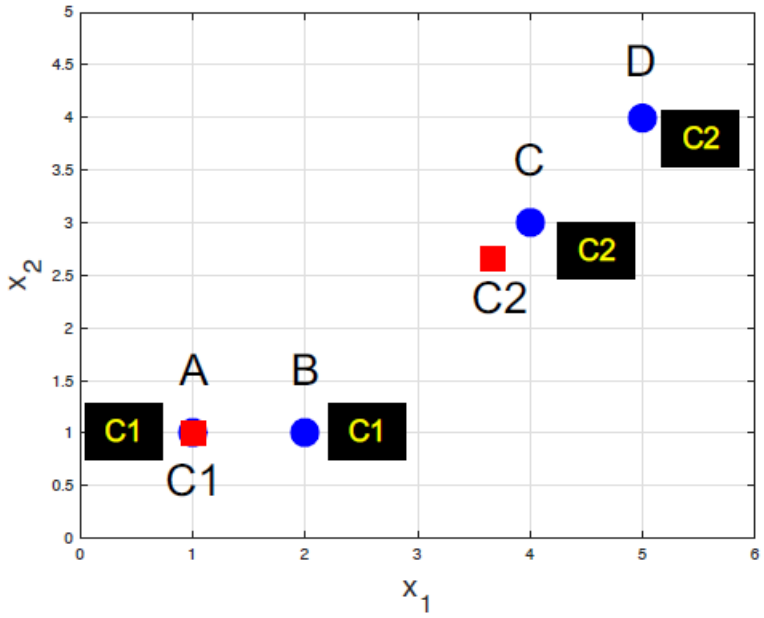
A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

$$C1 = A = (1,1)$$

$$C2 = \frac{B+C+D}{3} = \frac{(2,1)+(4,3)+(5,4)}{3} = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = (3.67, 2.67)$$

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

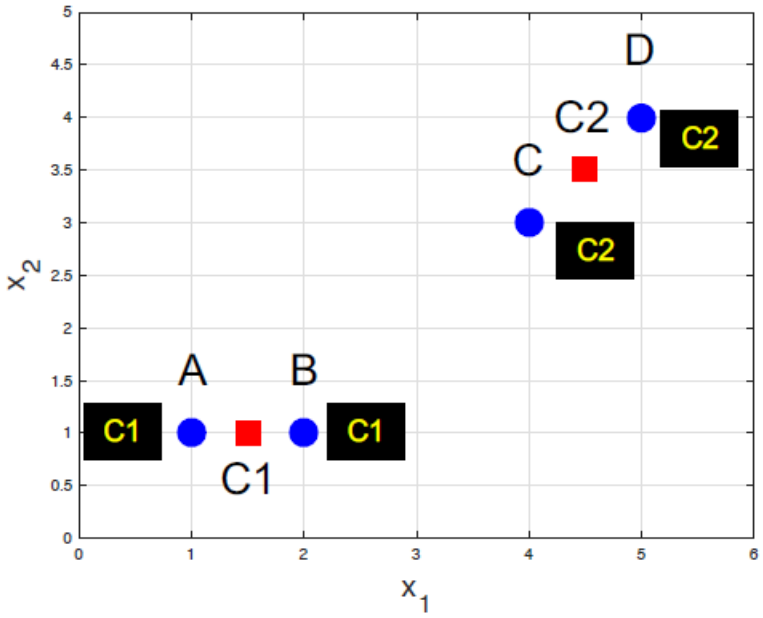
Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the (dis)similarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

- Repeat Steps1-2 to update cluster membership.

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the (dis)similarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

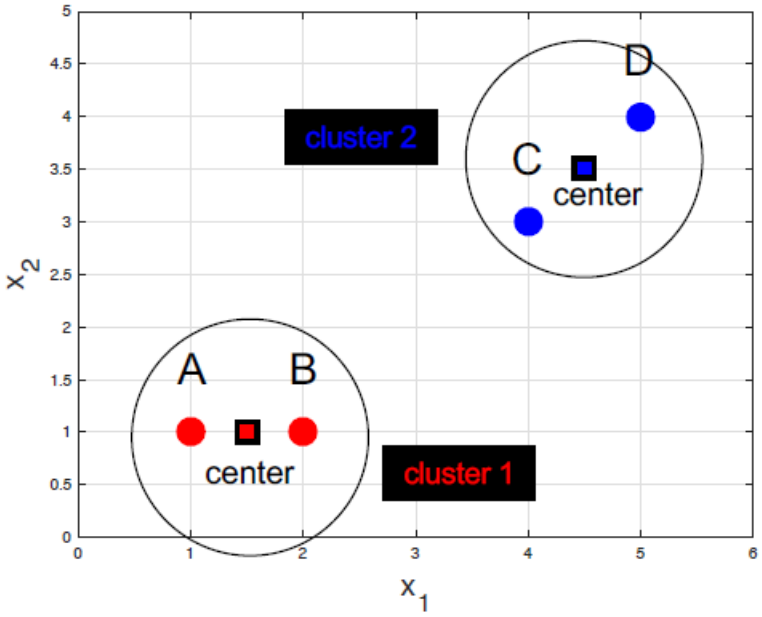
- Repeat Step 3 to update cluster center.

A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

$$C1 = \frac{A+B}{2} = \frac{(1,1)+(2,1)}{2} = (1,1.5)$$
$$C2 = \frac{C+D}{2} = \frac{(4,3)+(5,4)}{2} = (4.5,3.5)$$

K-Means Clustering

- How does K-means work?
  - Example: Group the 4 data points below.



A: (1, 1)
B: (2, 1)
C: (4, 3)
D: (5, 4)

Determine in advance:

- Group to K=2 clusters.
- Use Euclidean distance to measure the (dis)similarity between data points.
- Set initial cluster centers. For instance,
  C1: (1, 0.7)
  C2: (2, 0.7)

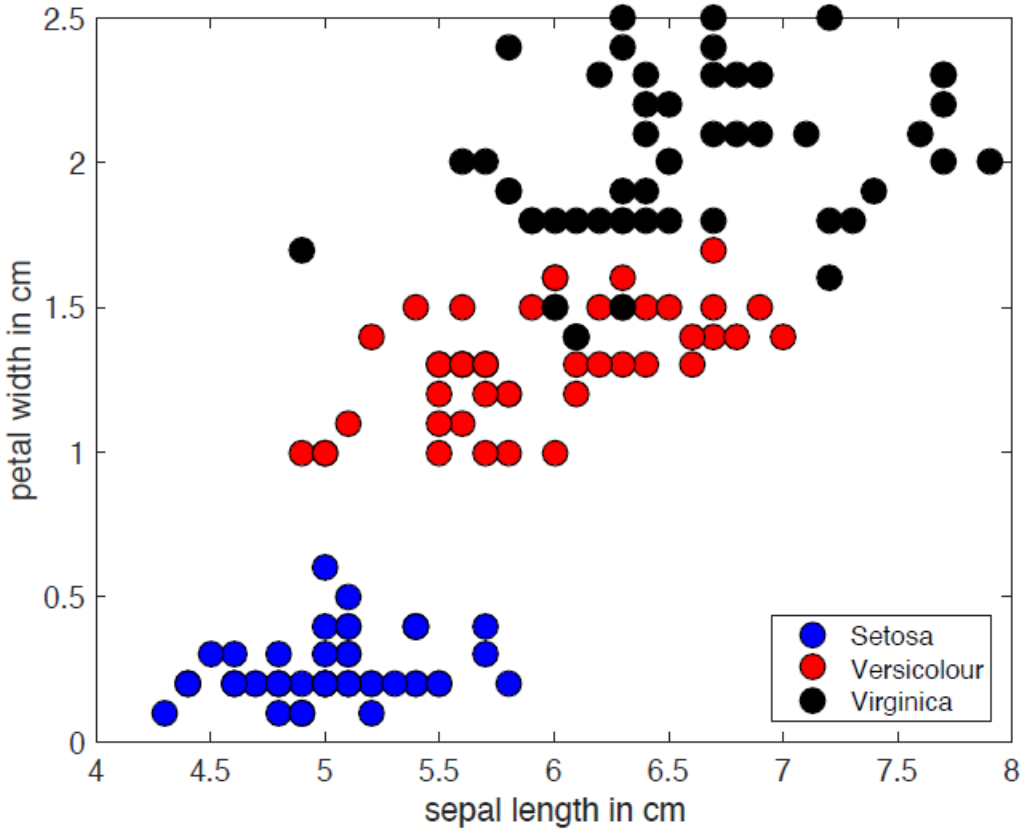
- Stop repeating when there is no change in the membership of each cluster.

K-Means Clustering

- Summary
- To group a set of data points to *k* clusters by K-means.
  - Pre-determine the cluster number *k*.
  - Choose your distance (or similarity) measure.
  - Initialise by random selecting k cluster center points.
  - Iterate:
    - Step 1: **Calculate distances (or similarities)** between the data points and the cluster center points.
    - Step 2: Find the **nearest** cluster center to each data point, and assign the data point to that cluster.
    - Step 3: Calculate the new cluster center for each cluster, by **averaging** its member points.
    - Step 4: Go back to Step 1), stop when there is no change in the membership of each cluster.

K-Means Clustering

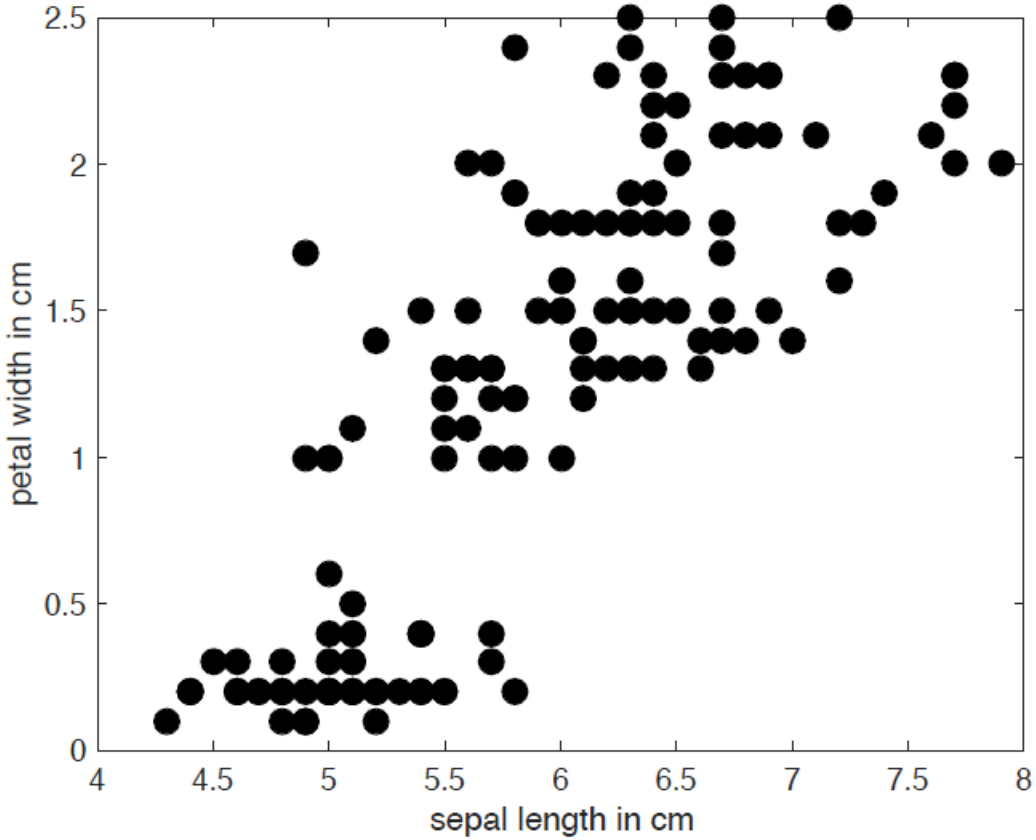
- Example: Clustering Iris Data



You have data points that are actually from three classes.

K-Means Clustering

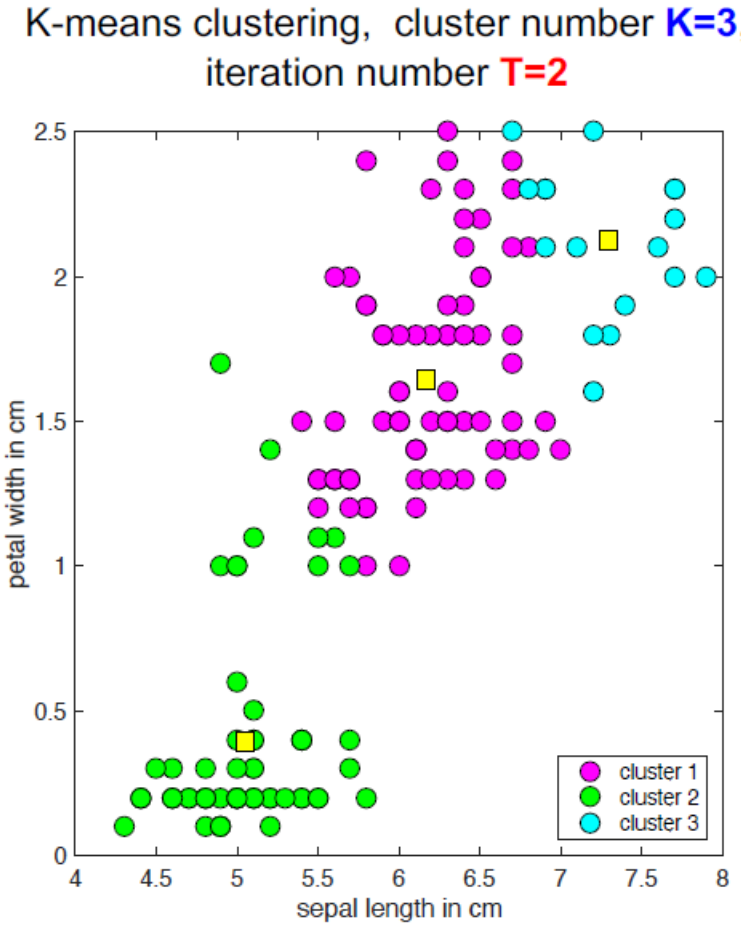
- Example: Clustering Iris Data

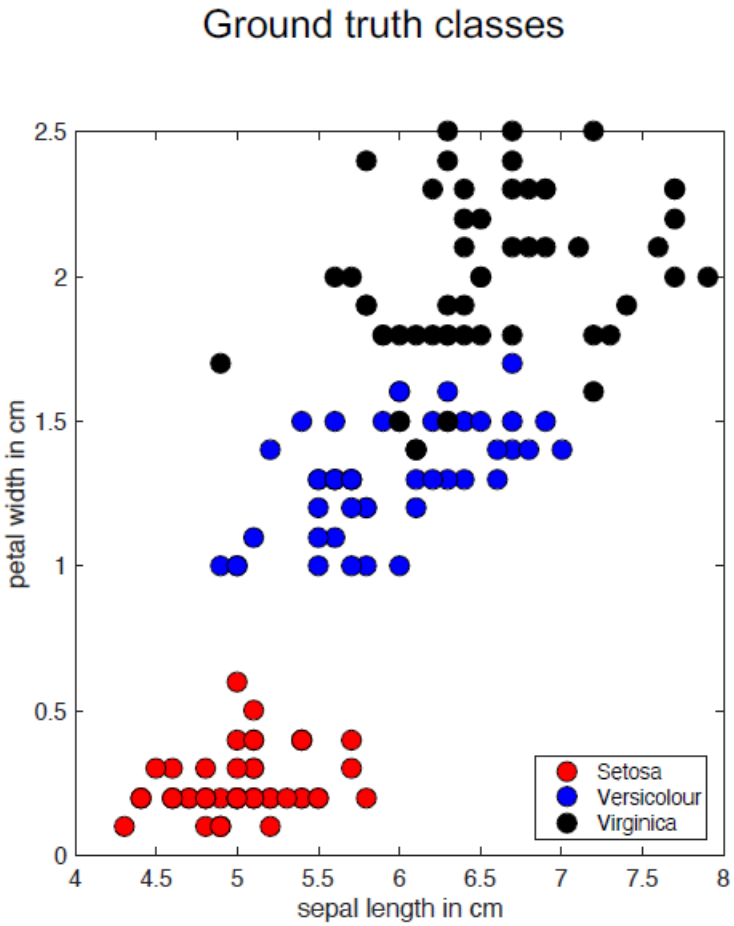


Remove the class information and analyse the data samples based on only their features.

K-Means Clustering

- Example: Clustering Iris Data

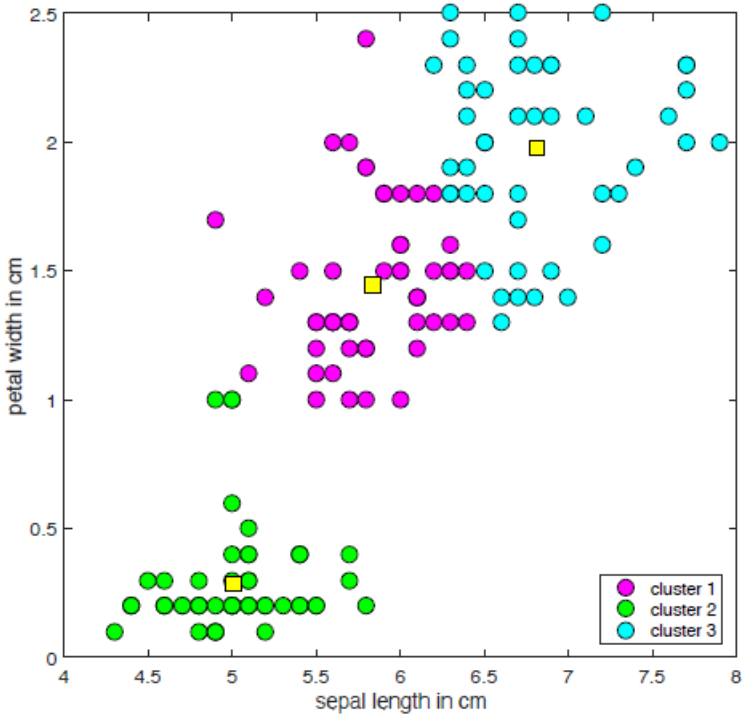




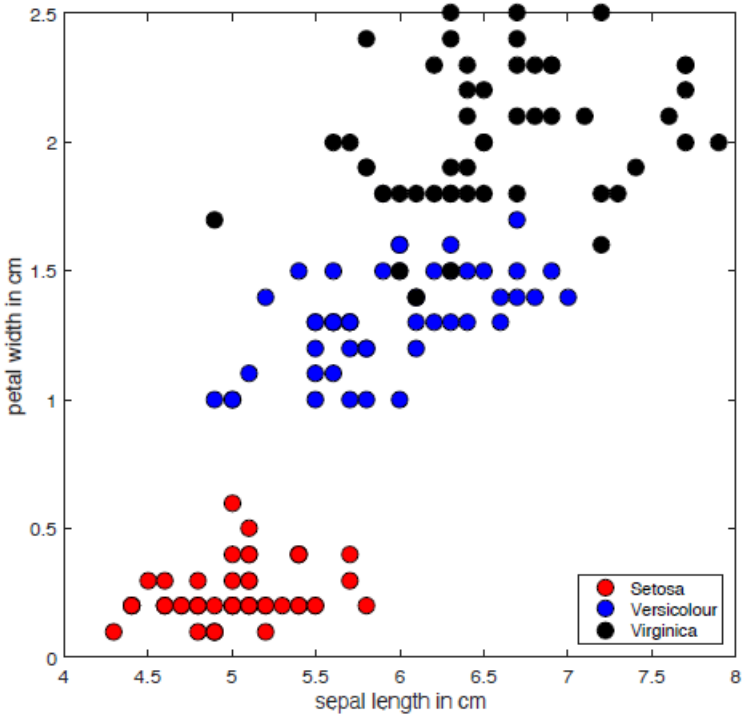
K-Means Clustering

- Example: Clustering Iris Data

K-means clustering, cluster number **K=3**, iteration number **T=500**



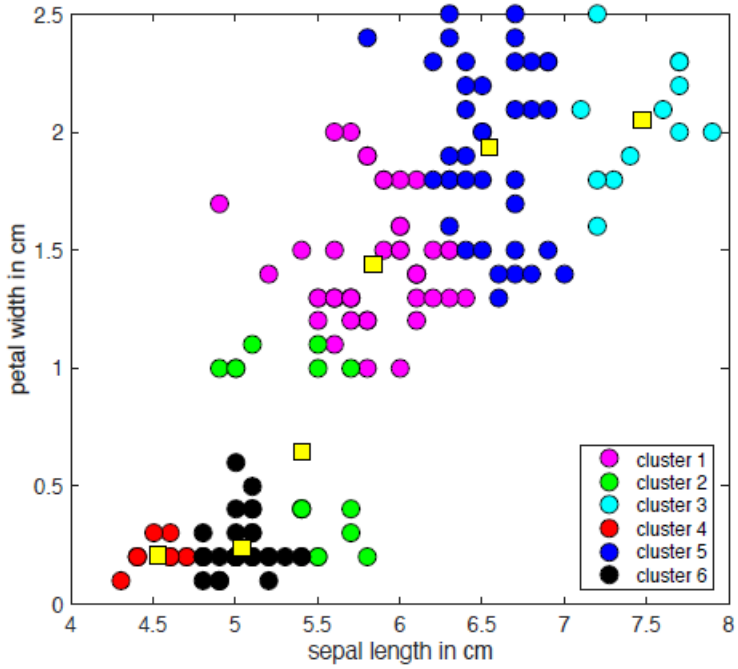
Ground truth classes



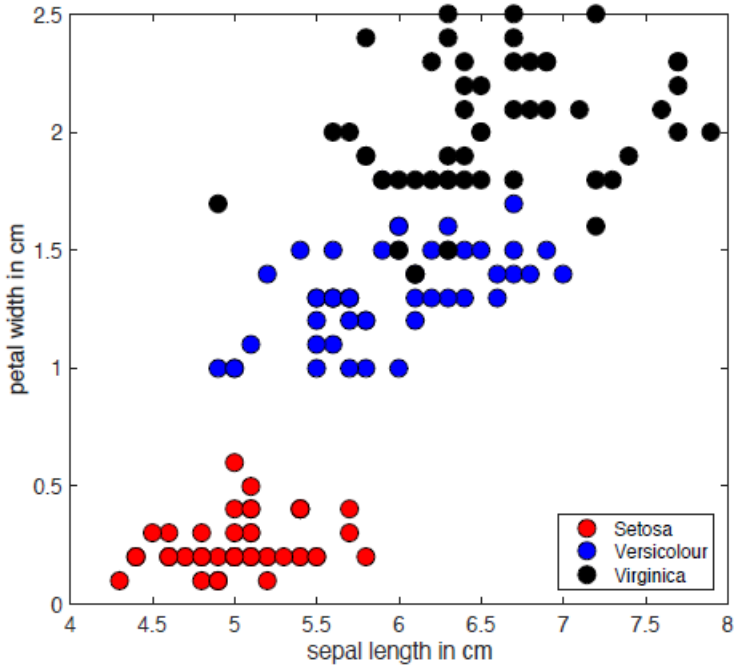
K-Means Clustering

- Example: Clustering Iris Data

K-means clustering, cluster number **K=6**, iteration number **T=1000**



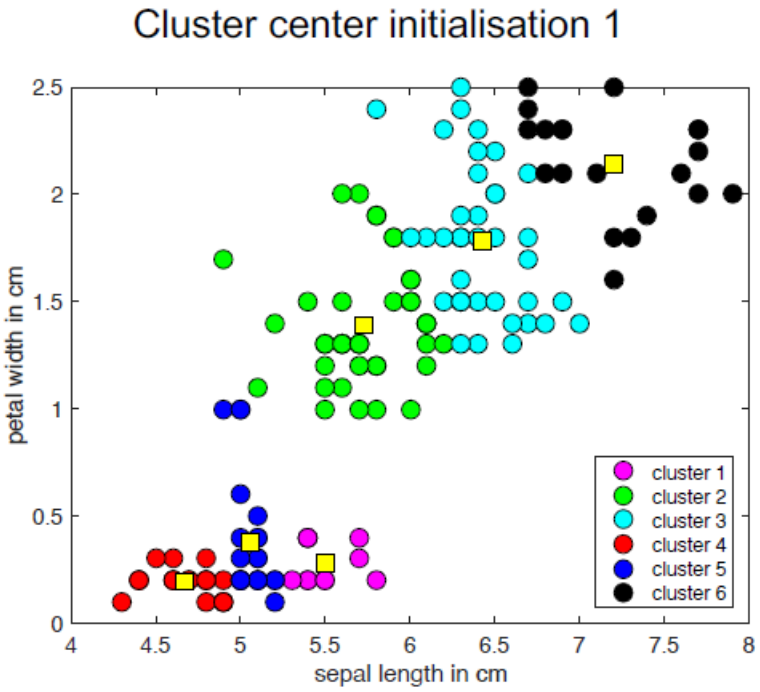
Ground truth classes

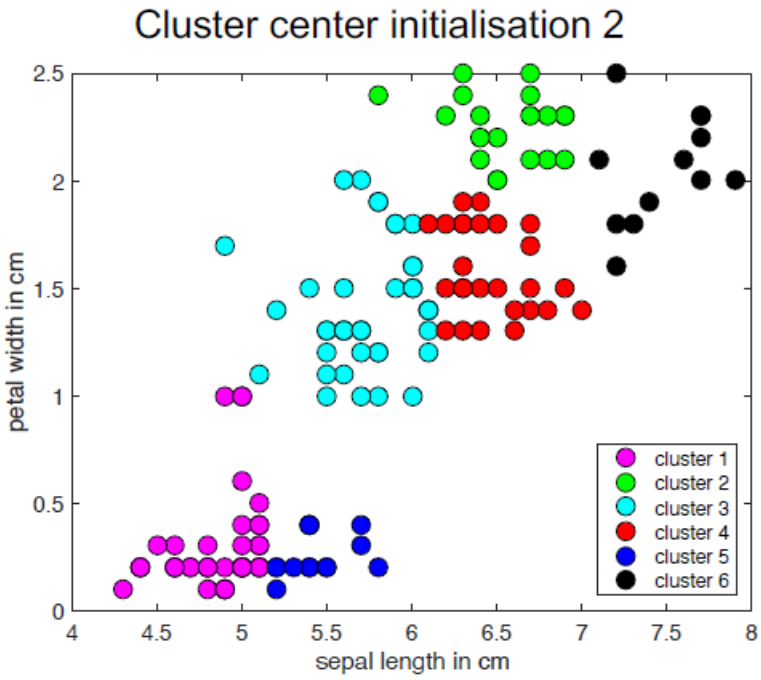


K-Means Clustering

- Example: Clustering Iris Data

K-means clustering, cluster number K=6, iteration number T=1000





K-Means Clustering

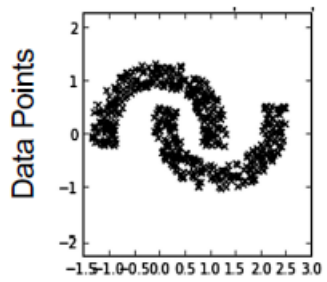
- Comments on K-means
  - This iterative process minimises the within-cluster sum of squares cost:

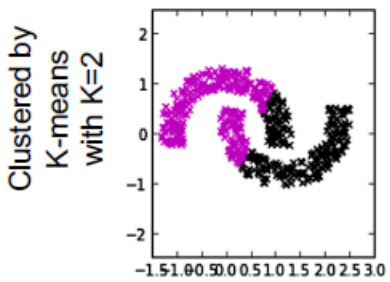
$$\min_{\substack{\text{cluster} \\ \text{membership}}} \sum_{i=1}^{K} \sum_{p \in \text{cluster } i} d^2\left(\mathbf{x}_p, \mathbf{c}_i\right), \text{ where } \mathbf{c}_i \text{ is the center vector of cluster } i$$

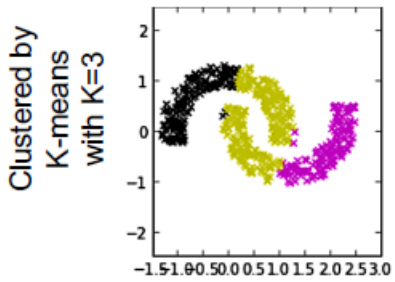
  - Complexity: Require *TKN* operations O(*TKN*), where *N* denotes the number of data points, *K* the cluster number, and *T* the iteration number. Normally, $K,T<<N$.
  - It always converges, but could converge to an unwanted solution.

K-Means Clustering

- Comments on K-means
  - The algorithm is sensitive to initial cluster center points.
  - Other issues:
    - Need to specify the number of clusters in advance.
    - Unable to handle noisy data and outliers.
    - Not suitable to discover clusters for complex data patterns.

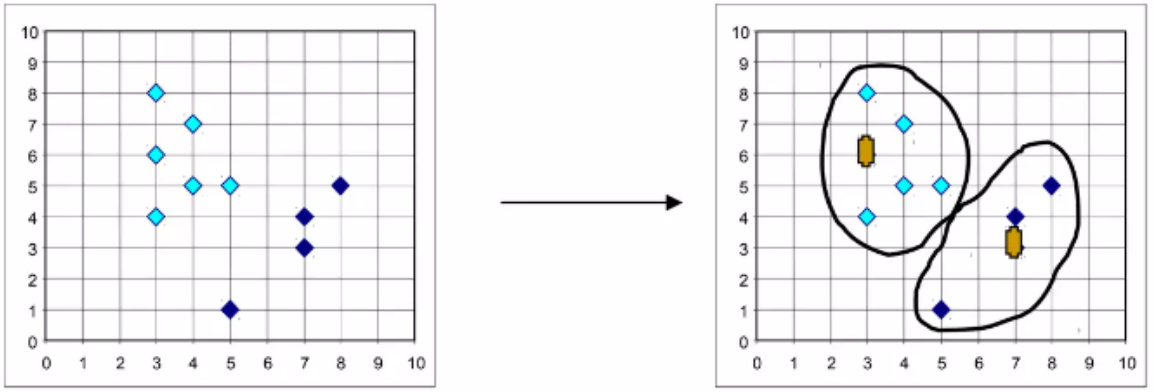






Unsupervised Learning: K-Medoids

- Problem of the K-Means Method
  - The K-Means algorithm is sensitive to outliers. Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids
  - Instead of taking the *mean* value of the object in a cluster as a reference point, *medoids* can be used, which is the *most centrally located* object in a cluster.



K-Medoids

- PAM (Partitioning Around Medoids)
  - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering.
  - All pairs are analyzed for replacement.
  - PAM works effectively for small data sets, but does not scale well for large data sets.

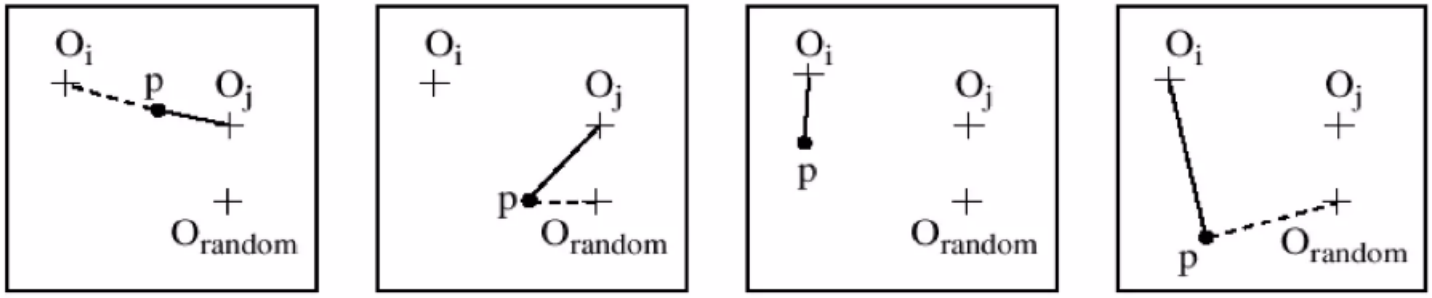
K-Medoids

- Input: k, database of n objects
- Output: A set of k clusters
- Methods
  - Arbitrarily choose k objects as initial medoids
  - Repeat
    - Assign each remaining object to cluster with nearest medoid
    - Randomly select a non-medoid o_{random}
    - Compute cost S of swapping o_j with o_{random}
    - If S<0 swap to form new set of k medoids
  - Until no change
- Working Principle
  - Minimize sum of the dissimilarities between each object and its corresponding reference point. That is, an absolute-error criterion is used

$$E = \Sigma_{j=1}^{k} \Sigma_{p \in Cj} | p - o_j |$$

K-Medoids

- **Case 1:** p currently belongs to medoid o_j. If o_j is replaced by o_{random} as a medoid and p is closest to one of o_i where i < > j then **p is reassigned to o_i.**
- **Case 2:** p currently belongs to medoid o_j. If o_j is replaced by o_{random} as a medoid and p is closest to o_{random} then **p is reassigned to o_{random}.**



1. Reassigned to O_i
2. Reassigned to O_{random}
3. No change
4. Reassigned to O_{random}

- **Case 3:** p currently belongs to medoid o_i(i< >j) If o_j is replaced by o_{random} as a medoid and p is still closest to **o_i assignment does not change**
- **Case 4:** p currently belongs to medoid o_i (i < > j). If o_j is replaced by o_{random} as a medoid and p is closest to o_{random} then **p is reassigned to o_{random}.**

K-Medoids

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of *k* representative objects;
(7) **until** no change;

Content

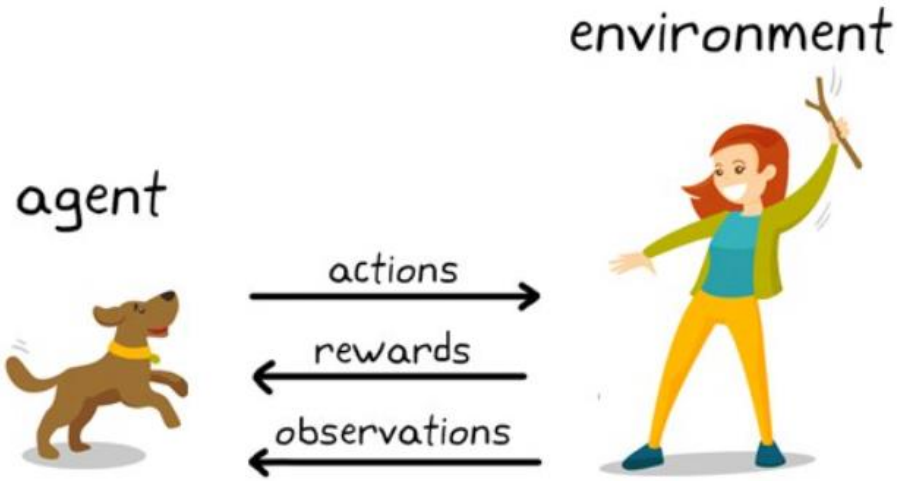
- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

Reinforcement Learning

- What is reinforcement learning?

Reinforcement Learning

- A family of algorithms and techniques used for Control (e.g. Robotics, Autonomous driving, etc.) and Decision making.
- Solve problems that need to be expressed as a Markov Decision Process (MDP)



Reinforcement Learning Example – KDNuggets

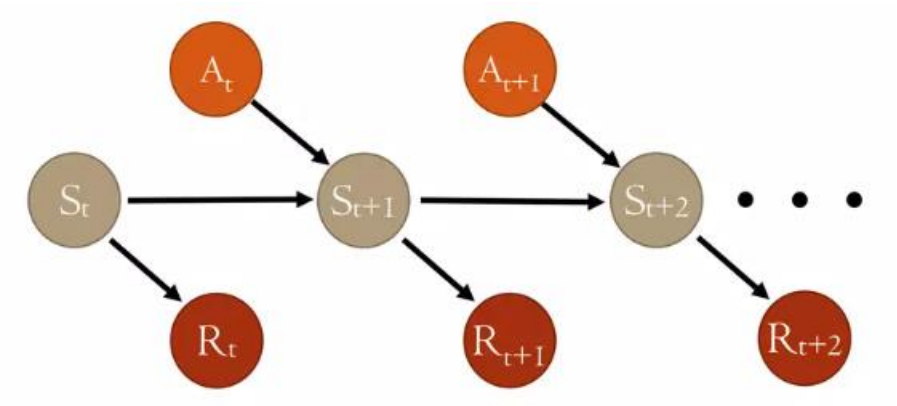
Reinforcement Learning

- Markov Decision Process (MDP)

  A sequential decision problem for a *fully observable, stochastic* environment with a markovian transition model and additive rewards.

  The *four components*:

  - *S:* A set of states (with an initial state S_0)
  - *A*: A set of actions (s) in each state
  - *T:* A transition model p(s'|s,a)
  - *R:* A reward function R(s)→ R(s,a,s')



Reinforcement Learning

- Markov Decision Process (MDP)

  Assumptions:

  - First-Order Markovian Dynamics (history independence)

$$P(S_{t+1}|A_t,S_t,A_{t-1},S_{t-1},\ldots,S_o) = P(S_{t+1}|A_t,S_t)$$

  - First-Order Markovian Reward Process

$$P(R_{t+1}|A_t,S_t,A_{t-1},S_{t-1},\ldots,S_o) = P(R_t|A_t,S_t)$$

  - Stationary Dynamics and Reward

$$P(S_{t+1}|A_t,S_t) = P(S_{k+1}|A_k,S_k) \text{ for all } t, k$$

The world dynamics do not depend on the absolute time

  - Full Observability

Reinforcement Learning

- Markov Decision Process (MDP)

  Utilities of Sequences:

  - Additive rewards

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$$

  - Discounted rewards

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

Discount Factor is a number between 0 and 1

  - With discounted rewards, the utility of an infinite sequence is finite. ($\gamma < 1$)

$$U_h([s_0, s_1, s_2, \ldots]) = \sum \gamma^t R(s_t) \leq \sum \gamma^t R_{max} = R_{max}/(1-\gamma)$$

Reinforcement Learning

- Markov Decision Process (MDP)
  - The Bellman Equation for Utilities

The utility of a state is the *immediate reward* for that state plus the *expected discounted utility* of the next state, assuming that the agent chooses the *optimal action*.

$$U'(s) = R(s) + \gamma \max_{a \in A(s)} [\sum P(s' \mid s, a) U(s')]$$

  - The Value Iteration Algorithm

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum change in the utility of any state in an iteration

repeat
  $U \leftarrow U'$; $\delta \leftarrow 0$
  for each state s **in** S **do**
    $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
    if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1 - \gamma)/\gamma$
return U

Reinforcement Learning

- Markov Decision Process (MDP)
  - Policy Iteration Algorithm

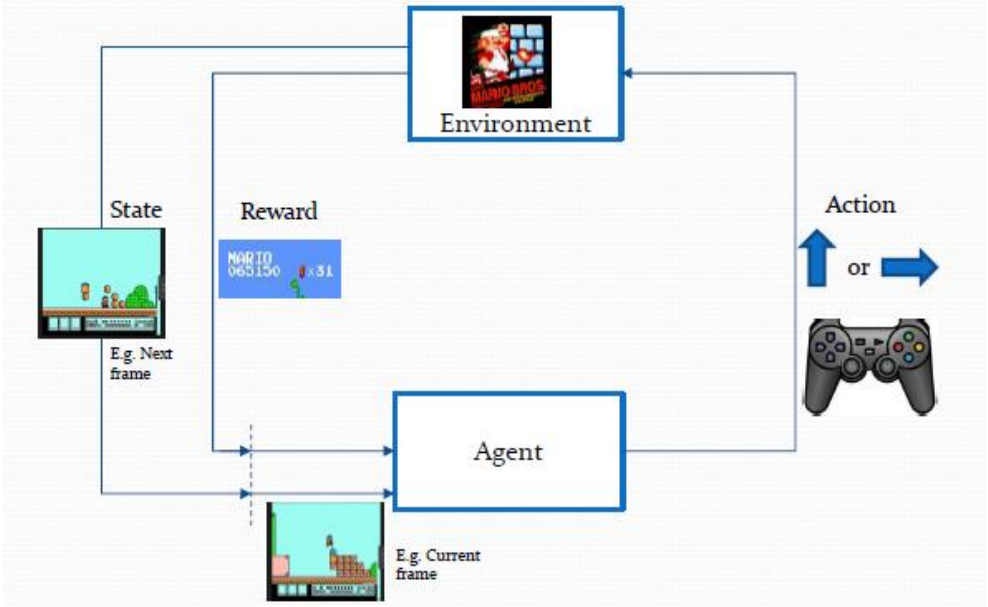
The policy iteration algorithm alternates the following *two steps*, beginning from some *initial policy* π_0.

- **Policy evaluation**: given a policy π_i, calculate $U_i = U^{\pi i}$, the utility of each state if π_i were to be executed.
- **Policy improvement**: Calculate a new MEU policy π_{i+1}, using one-step look-ahead based on U_i.

$$\pi^*(s) = \text{argmax}_{a \in A(s)} \sum P(s' \mid s, a) U(s')$$

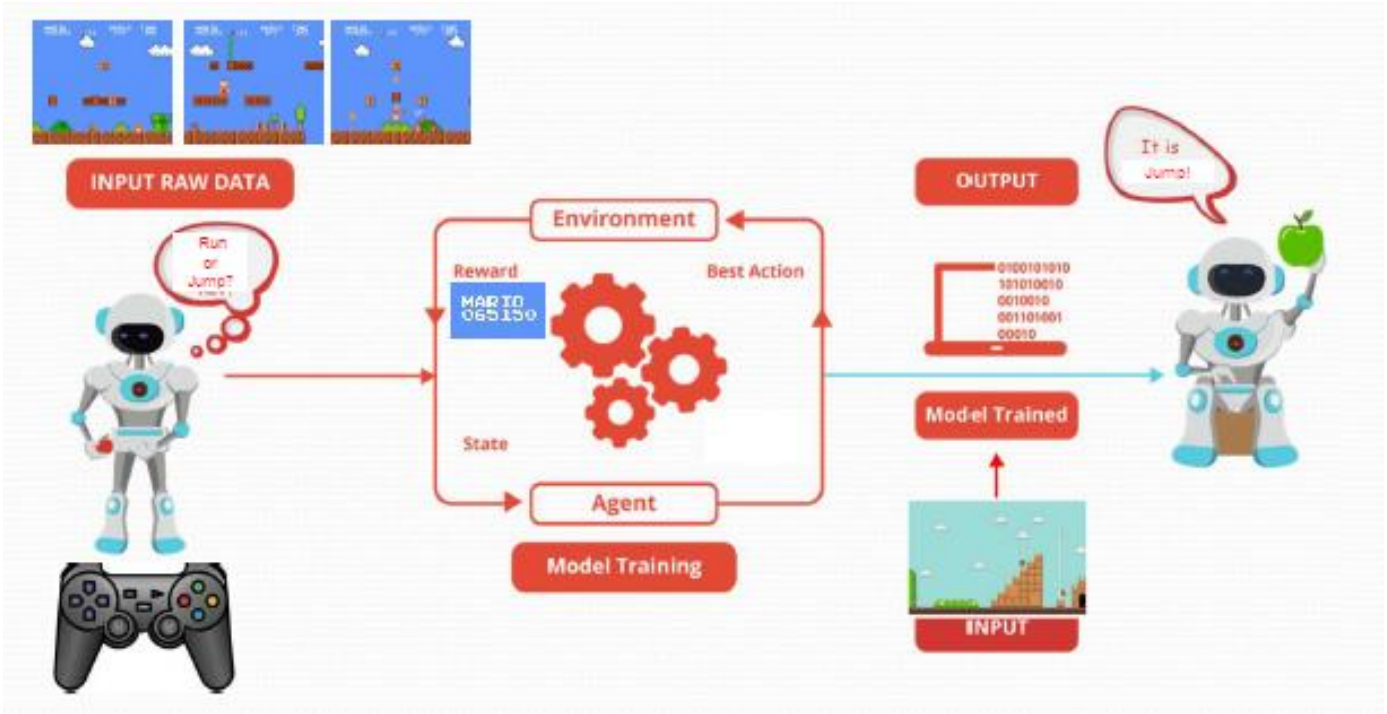
Reinforcement Learning

- Basics of reinforcement learning
  - **Agent:** the sole decision maker and learn
  - **Environment:** a physical world where an agent learns and decides the actions to be performed
  - **Action:** a list of action which an agent can perform
  - **State:** the current situation of the agent in the environment
  - **Reward:** For each selected action by agent, the environment gives a reward. It's usually a scalar value and nothing but feedback from the environment



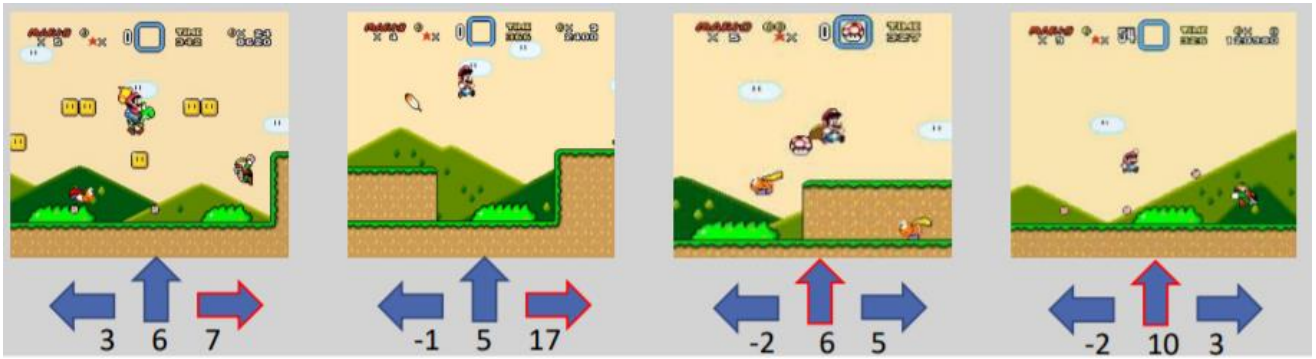
Reinforcement Learning

- Workflow



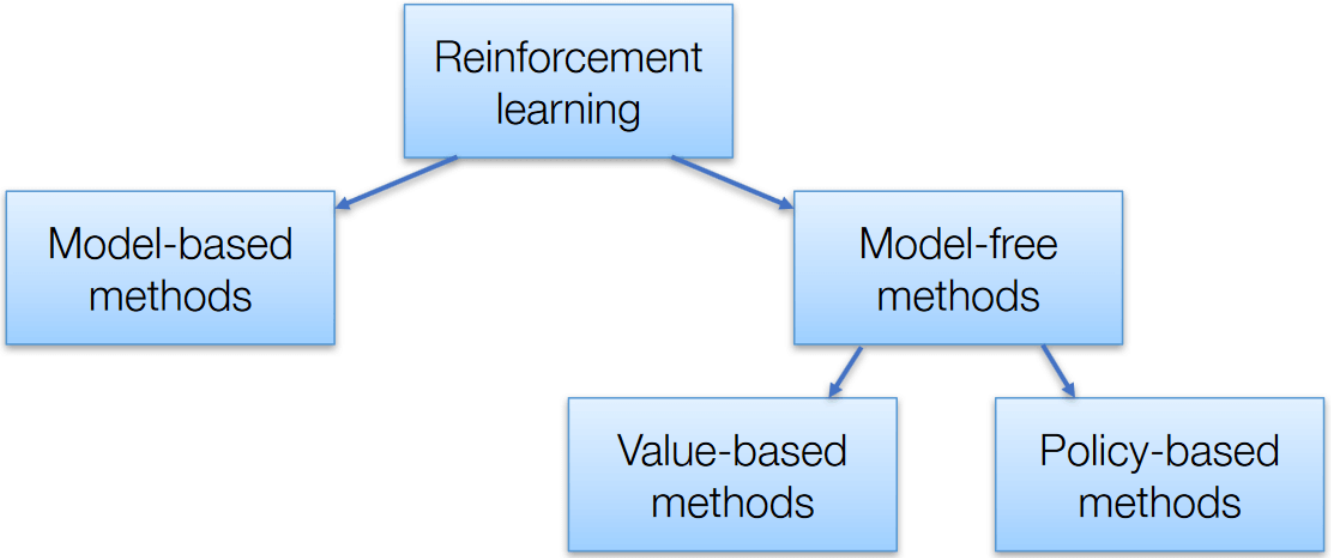
Reinforcement Learning

- Basics
  - An agent learns to take the action at a given state by maintaining a Action-Value function
    - This is known as the Q-function: denoted as $Q(s,a)$
    - The intuition is that for each state (s) an agent may be in, it knows the relative goodness of each action (a)
    - Then the policy could be to simply take the action with the highest value Q at state s (known as the greedy policy)



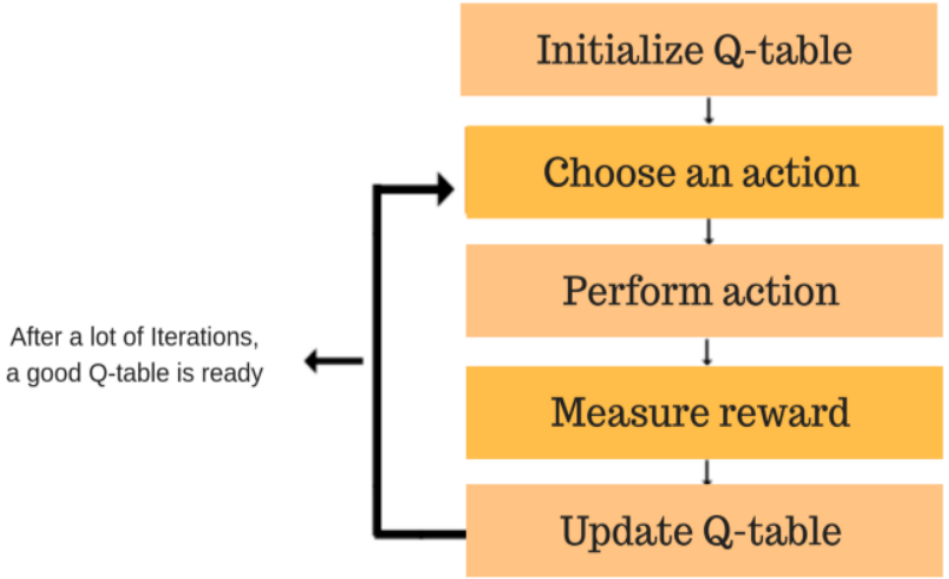
Reinforcement Learning

- Algorithms



Reinforcement Learning

- Q-learning algorithm
  - A value-based model-free approach.
  - The agent learns to take the action at a given state by maintaining a action-value function, Q-function Q(s,a).
  - It builds up a Q-table to store the Q-values, A Q-table is a matrix that correlates the state of the agent with the possible actions that the agent can adopt.



Reinforcement Learning

- Q-learning update
  - Q-learning function

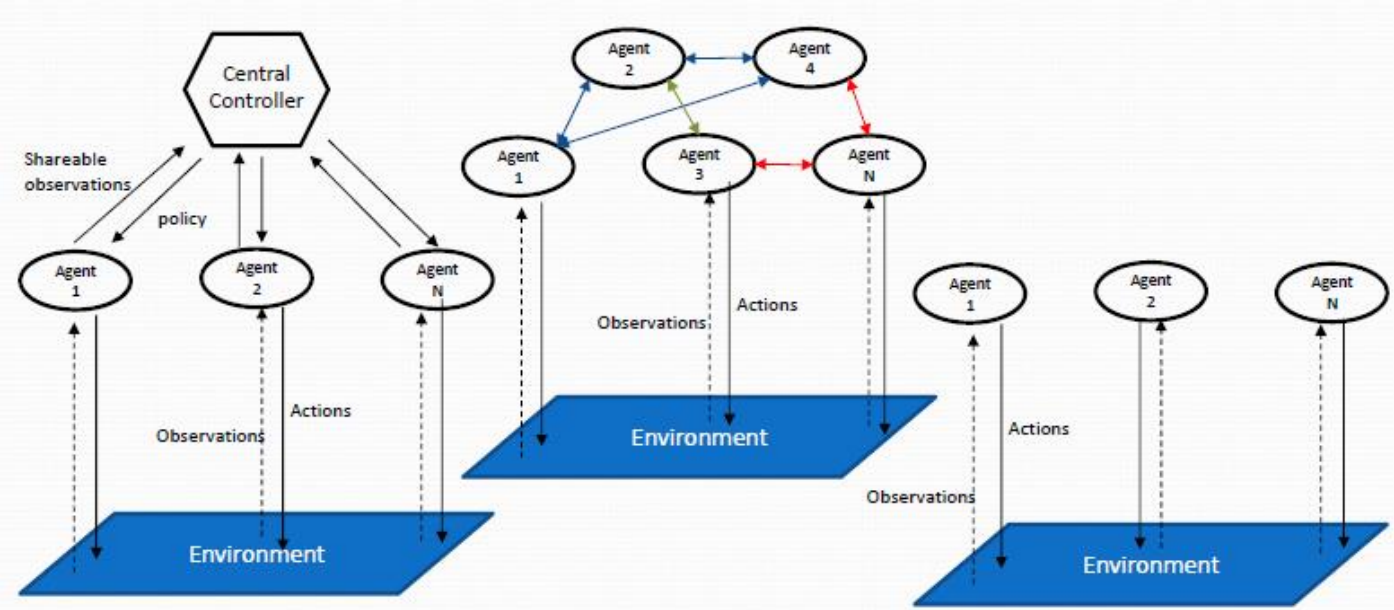
$$Q(s,a) = r + \gamma max'_a Q(s',a'))$$

  - Q-function update

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

Reinforcement Learning

- Multi-agent reinforcement learning



Zhang et al., Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms

Reinforcement Learning

- The applications of recent successes of MARL
  - Cooperative settings
    - A teams of UAVs accomplish a cooperation task in a decentralized fashion
    - Each UAV is communicating with other teammates
    - Decentralized paradigm with networked agents
    - High-mobility UAVs, time-varying communication links

UAV swarm

Reinforcement Learning

- The applications of recent successes of MARL
  - Competitive settings
    - Agents are solely focused on maximizing their own utility.
    - The Game of Go: two player zero-sum Markov game with deterministic state transitions
    - Texas holdem: multiplayer extensive-form game with incomplete information

Texas holdem

The game of go

Reinforcement Learning

- The applications of recent successes of MARL
  - Mixed settings
    - Usually formulated as a multiagent stochastic game with partial information
    - The agents are divided into two opposing teams that play zero-sum games
    - Both cooperation among teammates and competition against the opposing team need to be taken into consideration

Team battle video game

Summary on Supervised, Unsupervised and Reinforcement Learning

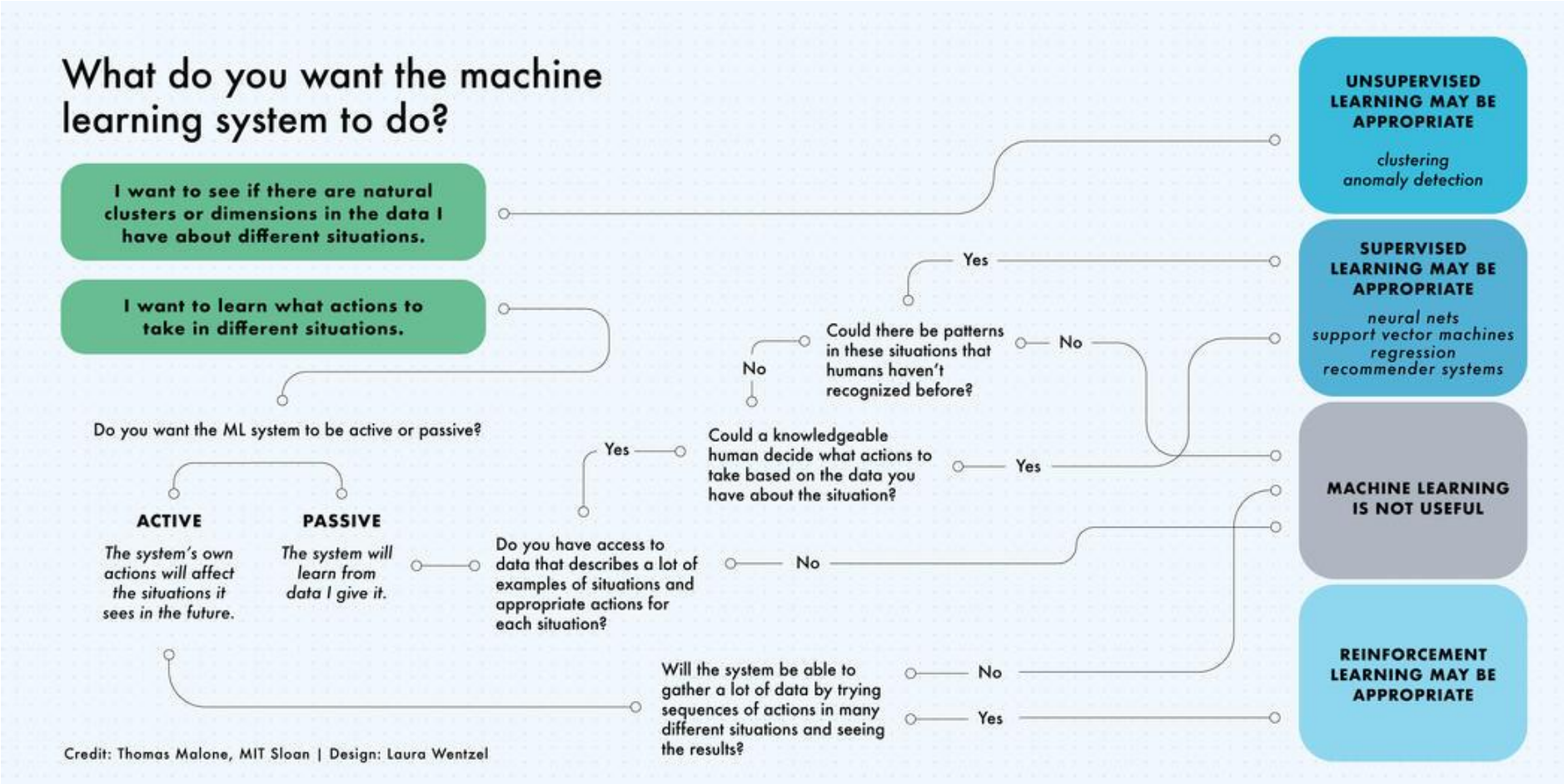
- Summary
  - Machine Learning
    - Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior.
    - The function of a machine learning system can be descriptive, meaning that the system uses the data to explain what happened; predictive, meaning the system uses the data to predict what will happen; or prescriptive, meaning the system will use the data to make suggestions about what action to take.
    - Some examples of ML applications are Autonomous vehicles, Netflix and Amazon recommendation systems, fraud detection, email spam filtering, games, sentimental analysis, images recognition, and chatbots.
    - There are three subcategories of machine learning: Supervised, Unsupervised and Reinforcement Learning.

Summary on Supervised, Unsupervised and Reinforcement Learning

- Summary
  - Supervised Learning
    - Supervised machine learning models are trained with labeled data sets, which allow the models to learn and grow more accurate over time. For example, an algorithm would be trained with pictures of dogs and other things, all labeled by humans, and the machine would learn ways to identify pictures of dogs on its own. Supervised machine learning is the most common type used today.
  - Unsupervised Learning
    - In unsupervised machine learning, a program looks for patterns in unlabeled data. Unsupervised machine learning can find patterns or trends that people aren't explicitly looking for. For example, an unsupervised machine learning program could look through online sales data and identify different types of clients making purchases.
  - Reinforcement Learning
    - Reinforcement machine learning trains machines through trial and error to take the best action by establishing a reward system. Reinforcement learning can train models to play games or train autonomous vehicles to drive by telling the machine when it made the right decisions, which helps it learn over time what actions it should take.

Summary on Supervised, Unsupervised and Reinforcement Learning

- Summary



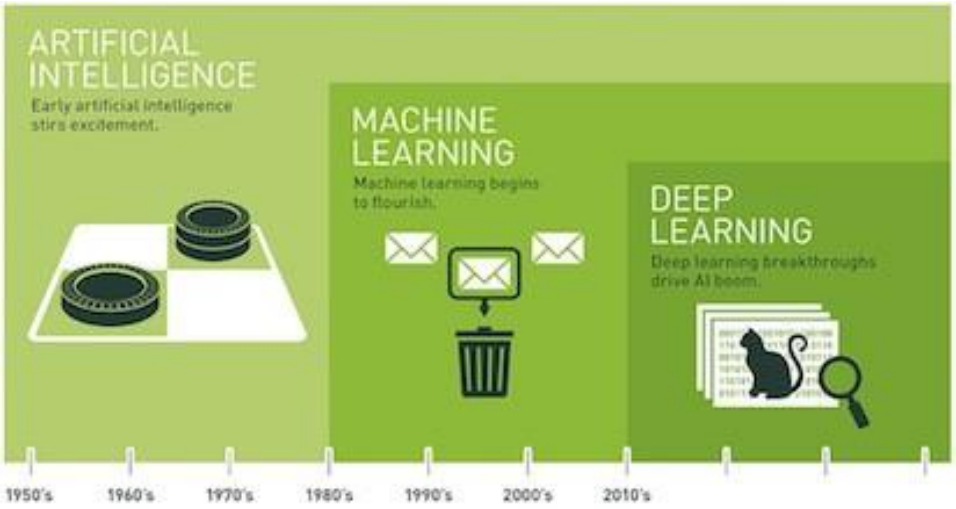
Source: Thomas Malone | MIT Sloan. See: https://bit.ly/3gvRho2, Figure 2

Content

- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

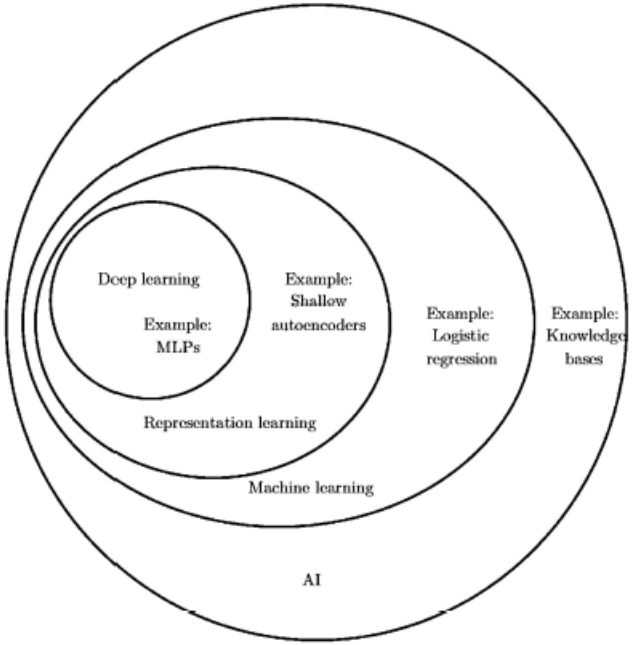
Deep Learning

- What is deep learning?

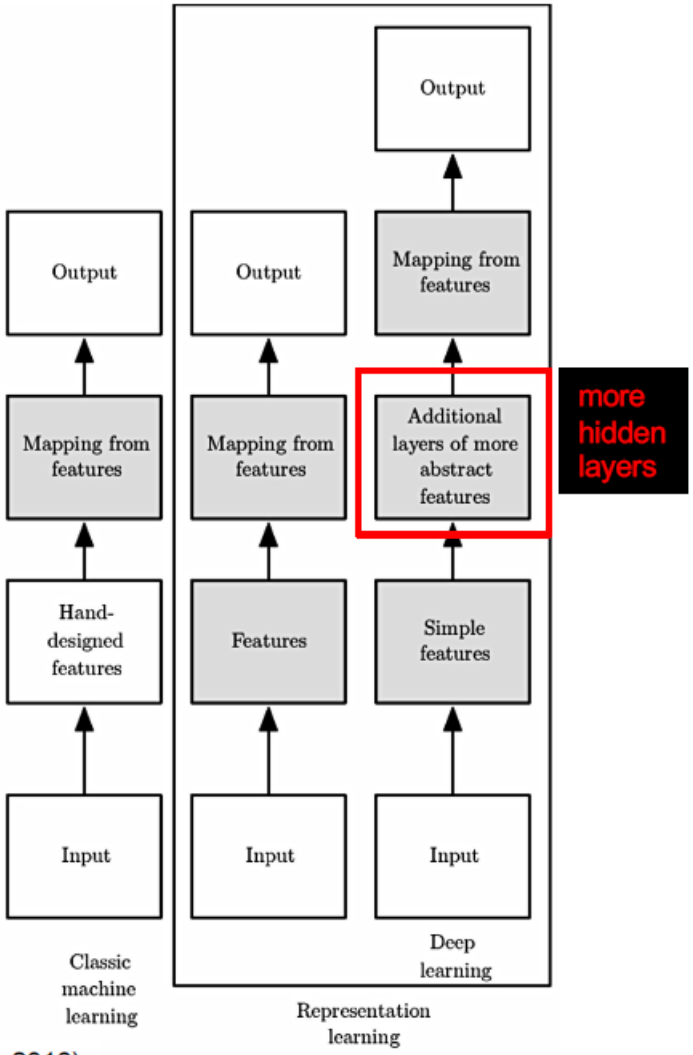


Deep Learning

- Deep learning refers to techniques for learning using neural networks.
- Deep learning is considered as a kind of representation (feature) learning techniques



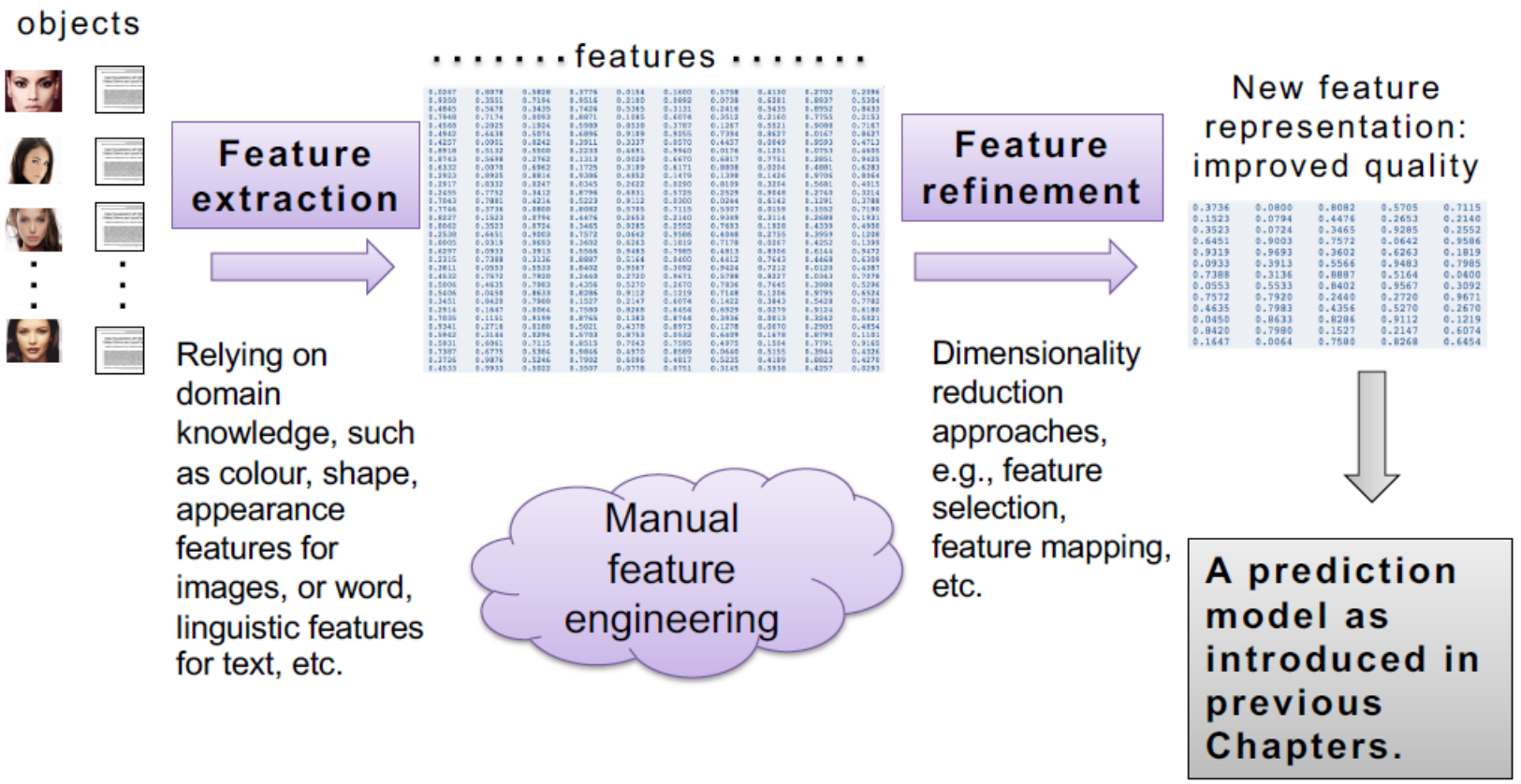
Example: AlexNet contains a total of 5 convolutional layers and 3 fully connected layers.



The two diagrams are from Figs. 1.5 and 1.4 of Deep Learning book (I. Goodfellow, et al. 2016).

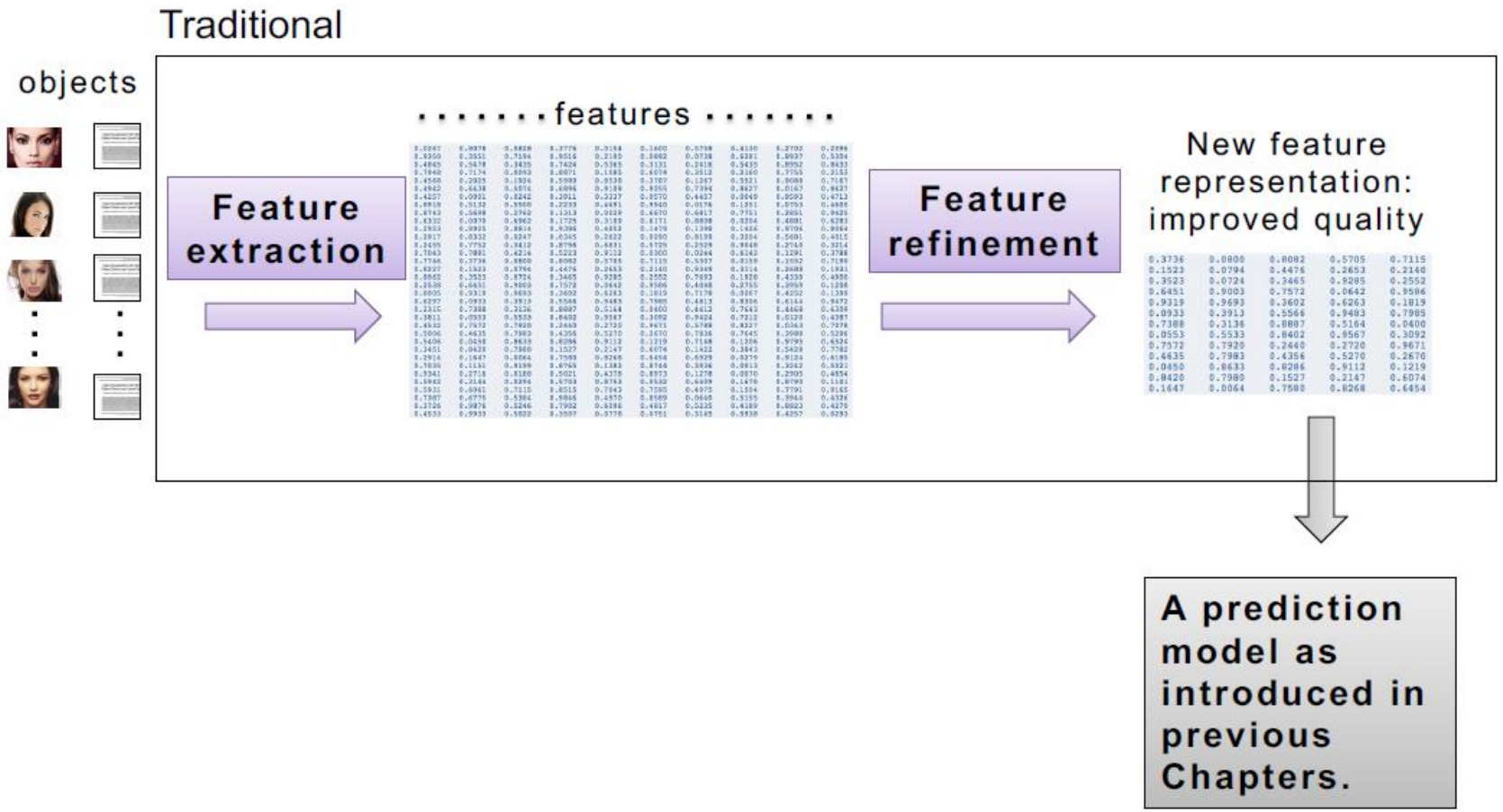
Deep Learning

- Traditional Machine Learning Strategy



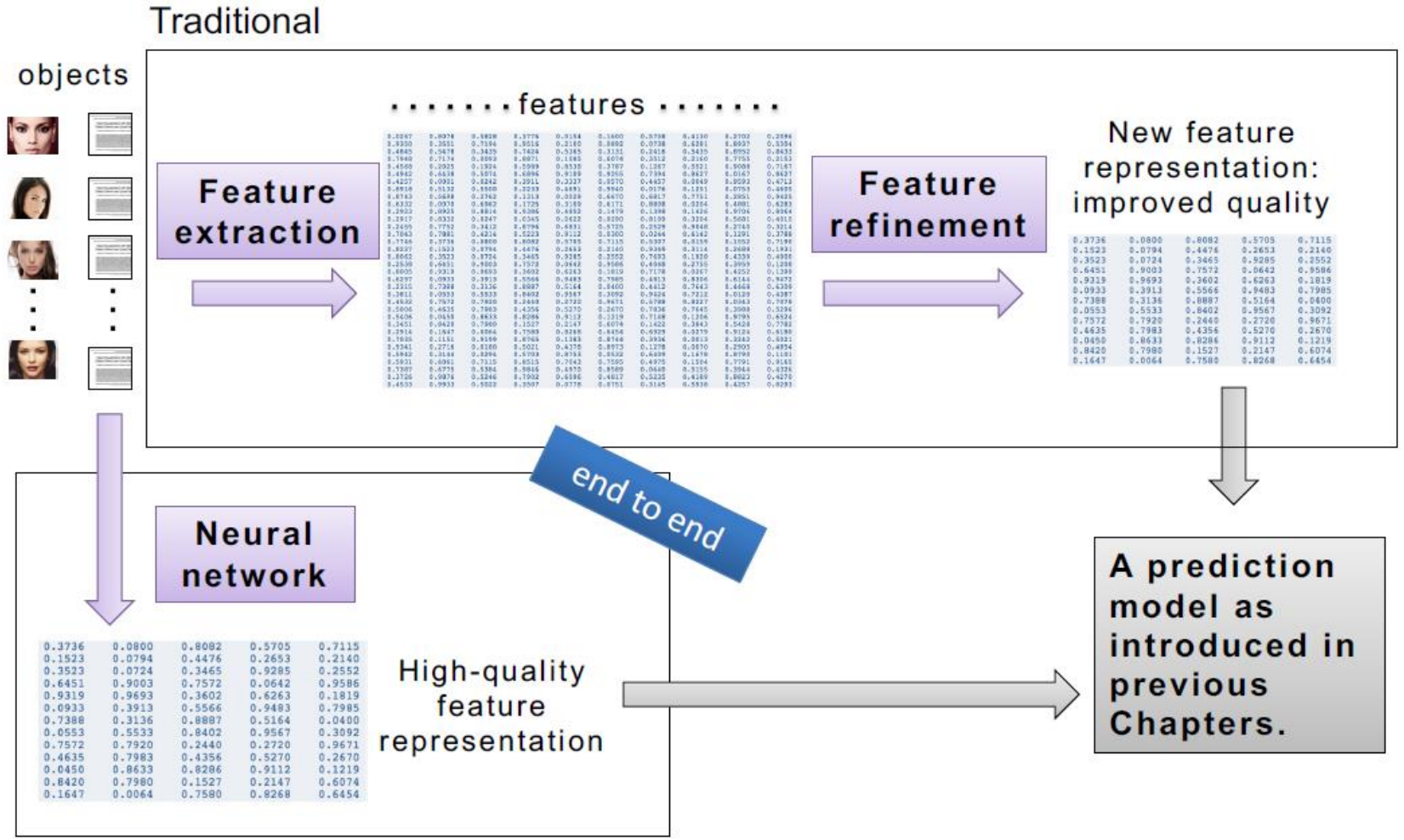
Deep Learning

- Deep Learning Strategy



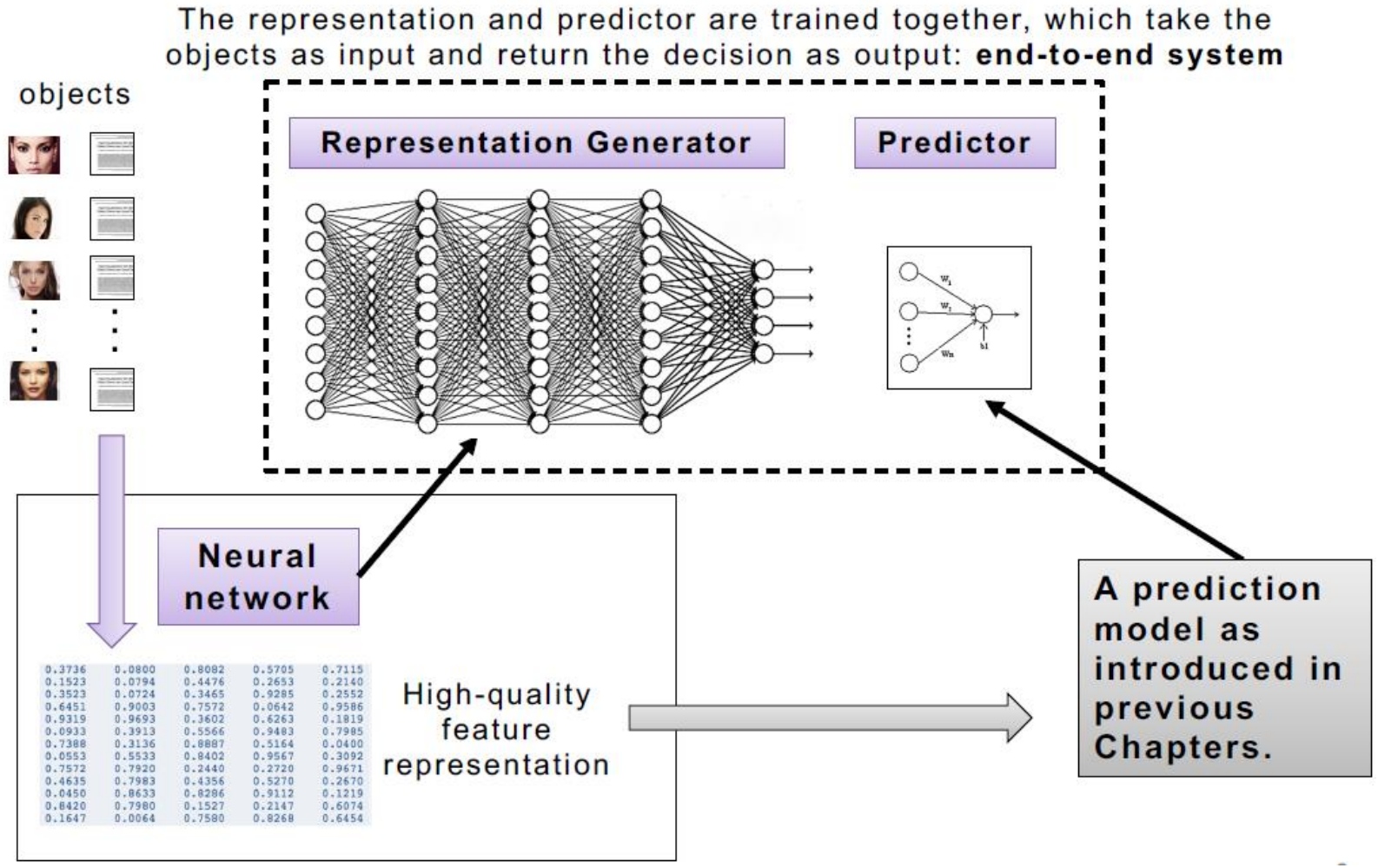
Deep Learning

- Deep Learning Strategy



Deep Learning

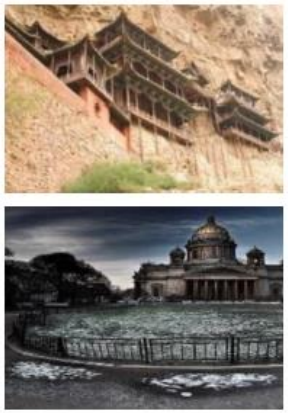
- Deep Learning Strategy



Deep Learning

- Examples of End-to-End Systems

input



output

Decision: These two images contain similar structures.

input

output

Decision: This video is about volleyball.

input

Question: how to push yourself to the limit during excercising?

Answer: try wearing a bandana and looking really cool and maybe you can "push yourself to the limit" in a top gun kind of way. listen to some bon jovy music.

output

Decision: This is a correct answer to the question.

input

Question: what is on the bed?

output

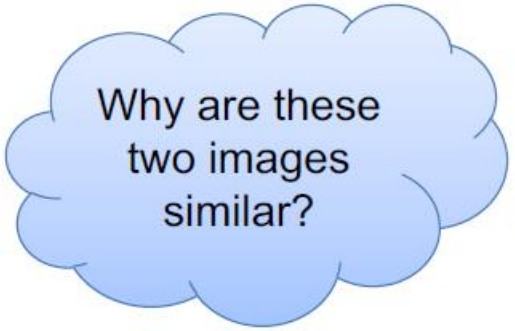
Answer: books

Deep Learning

- Advanced Neural Network Architecture
  - **Convolutional neural networks** (CNN) is used to automatically learn a good feature vector for an image from its pixels.
    - NeuralStyle, https://github.com/jcjohnson/neural-style
    - DeepDream, https://deepdreamgenerator.com
  - **Recurrent neural** network (RNN) is particularly useful for learning from sequential data. Each neuron can use its internal memory to maintain information about the previous input. This makes it suitable for processing natural languages, speech, music, etc.
    - Long short-term memory (LSTM) and gated recurrent unit (GRU) are two of the most popular types of RNN.

- Attention Mechanism
  - **Black box decision:** A decision supported by model parameters and mathematical operations that are “hard” to understand.





  - **Attention mechanism:** A type of effective mathematical diagrams based on weighted sum, which uses weight functions to locate salient information.
  - Embed an attention mechanism to a neural network would help identify the contributing parts to final decision making.

Deep Learning

- Training Tips of Deep Neural Networks
  - **Pre-train:** Consider to use a pre-trained neural network if your data shares some similar structure to existing training data that has been used to train some well-known neural network, e.g., ResNet, VGGNet trained on ImageNet data.
  - **Batch normalization:** remove the mean and scale by standard deviation for outputs of hidden layers.
  - **Regularisation**
  - **Skip connection:** A type of network connection proposed in ResNet.

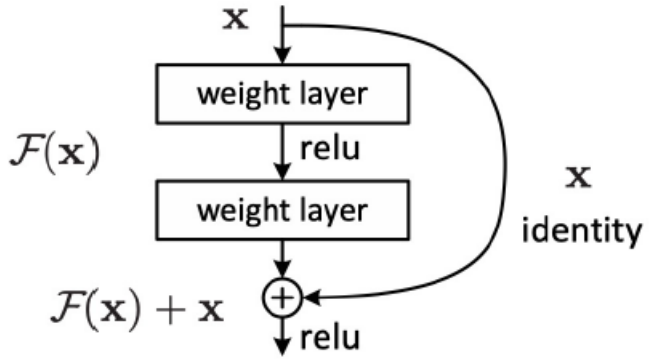


Figure 2 of He et al. CVPR 2016, https://arxiv.org/pdf/1512.03385.pdf

Deep Learning

- Regularisation Techniques
  - Given a neural network with powerful function approximating capability, it is important to employ techniques to **avoid over-fitting.**
  - Typical regularisation techniques:
    - Parameter norm penalties, remember $O(\mathbf{W}_{NN}) = loss(\mathbf{W}_{NN}) + \lambda \frac{1}{2} \|\mathbf{W}_{NN}\|_2^2$?
    - Dataset augmentation by creating fake data and adding to the training set.
    - Add small perturbation (noise) to network weights.
    - Early stopping by treating the number of training iterations as a hyper-parameter and finding the best one by using validation data.
    - Sparse representation by forcing hidden representation to have more zeros.
    - Ensemble methods by training multiple models separately at the same time, and letting them vote the final output.
    - Dropout by randomly removing a percentage of neurons during training.

More in Chapter II.2 Deep Learning book (I. Goodfellow, et al. 2016).

Deep Learning

- NN for Vision and Language
  - Neural networks have demonstrated great success in processing images, audios, natural language, etc.
    - NeuralTalk: Text caption generation from images.
      https://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/
    - TalkingMachines: Audio signal generation.
      https://deepmind.com/blog/wavenet-generative-model-raw-audio/
    - A system learns from images, sound, etc.
      https://teachablemachine.withgoogle.com
    - PoemGenerator: Text generation following special pattern.
      https://github.com/dvictor/lstm-poetry
    - Music generation: https://cdn.openai.com/papers/jukebox.pdf

Deep Learning

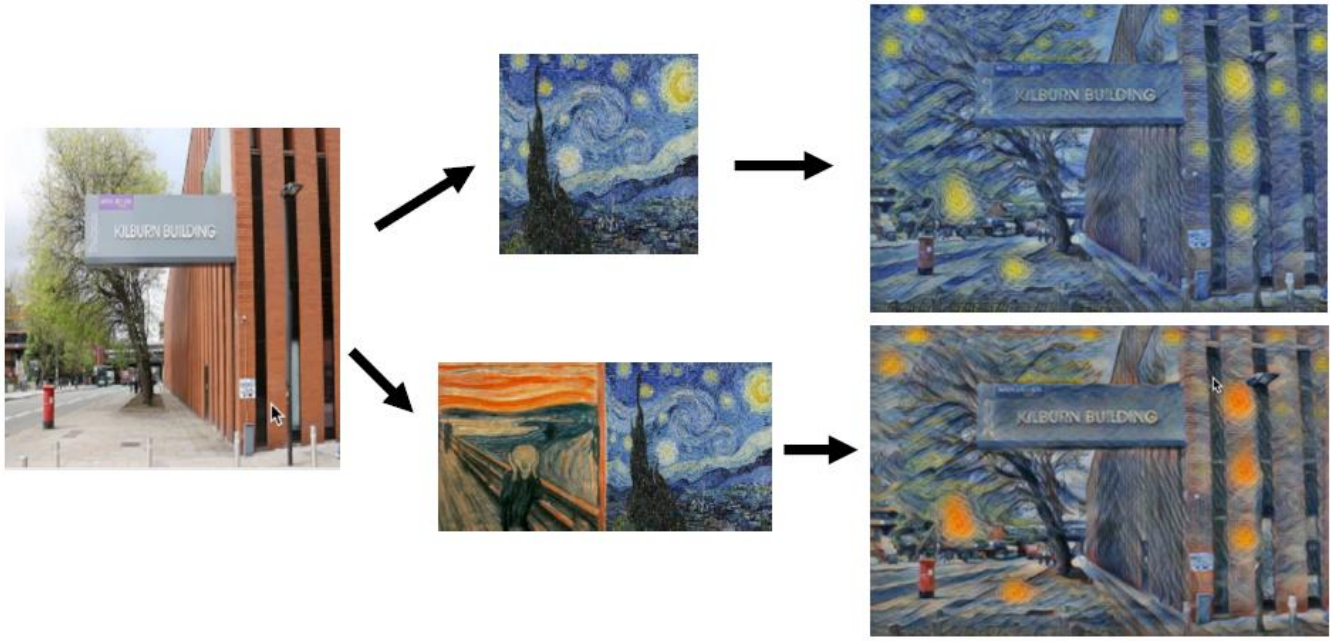
- Example 1: Face Generation
  - Task: To generate new faces that looks similar to some observed ones using GAN technique.

[Ioana Lazar, 3rd year project, 2020]

Deep Learning

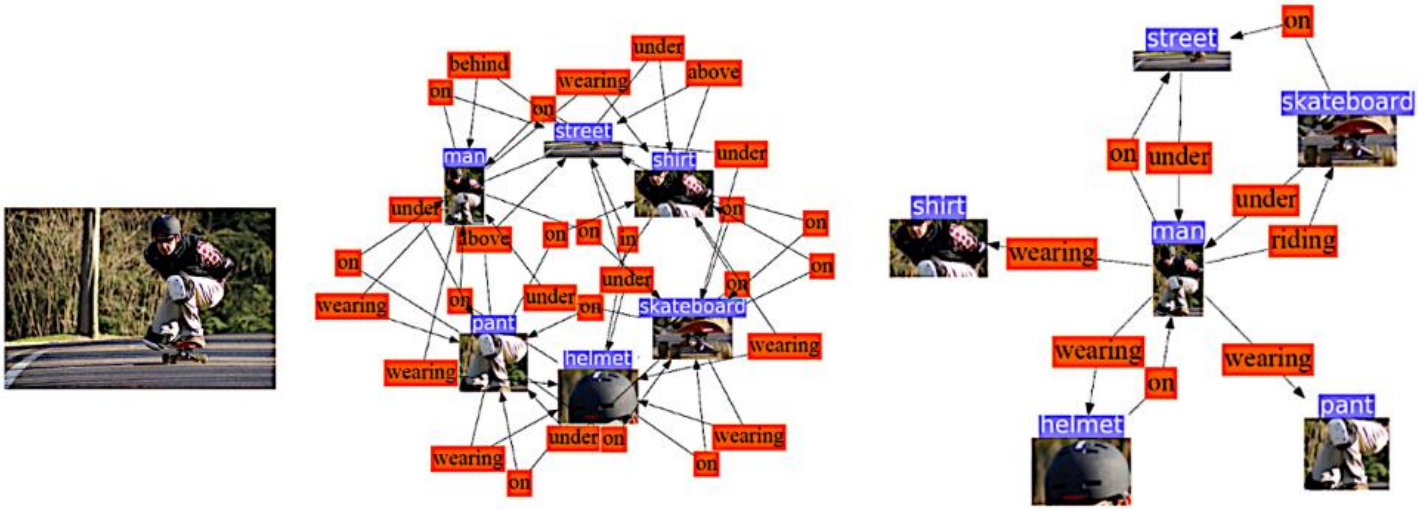
- Example 2: Artistic Style Transfer by CNN
  - Task: To transform an arbitrary photo to a painting in a desired artistic style.
    - Represent content image and style image by a ready CNN trained on a large object classification corpus.
    - Construct the transfer mapping function by a neural network.
    - Optimise the mapping function by minimising a content loss and a style loss.

    [Y. Li, 3rd year project, 2018]



- Example 3: Scene Graph Generation
  - Task: To explain visual scenes by constructing knowledge graphs.
    - Object recognition and bounding box proposal by a regional-CNN.
    - Object representation and link representation are learned by the neural motif architecture (a hybrid of RNN and CNN).
    - Loss designed to handle highly imbalanced and noisy relations.

[A. Sarullo and T. Mu, IJCNN, 2018]

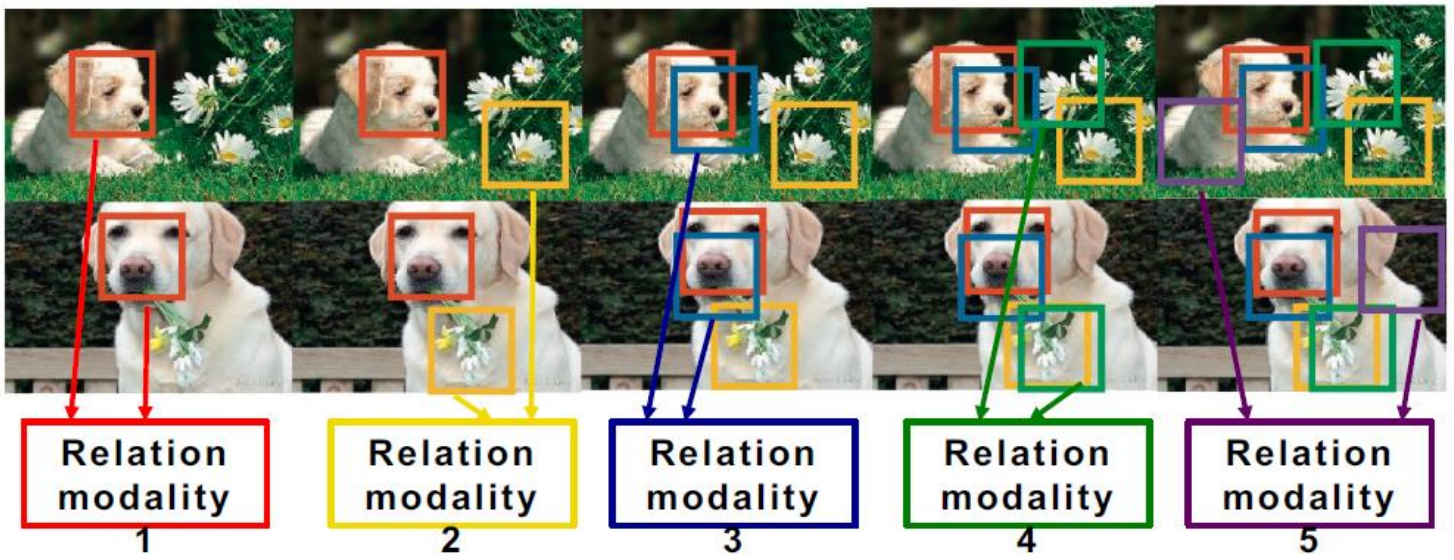


(a) Image (b) Scene graph by NM [12] (c) Scene graph by NM-CSL-NRF

Deep Learning

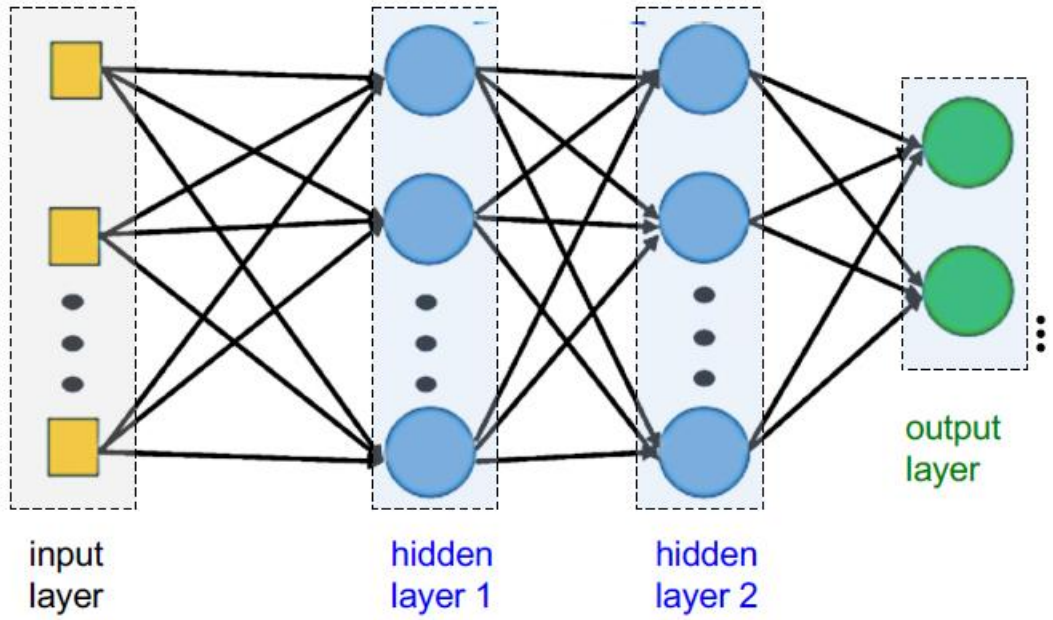
- Example 4: Image Retrieval
  - Task: To find an image in the database that is relevant to the query image.
    - Compute image similarity by CNN.
    - Attention scheme: automatically identification of salient pairs of image patches.

[X. Gao et al., IS, 2017]



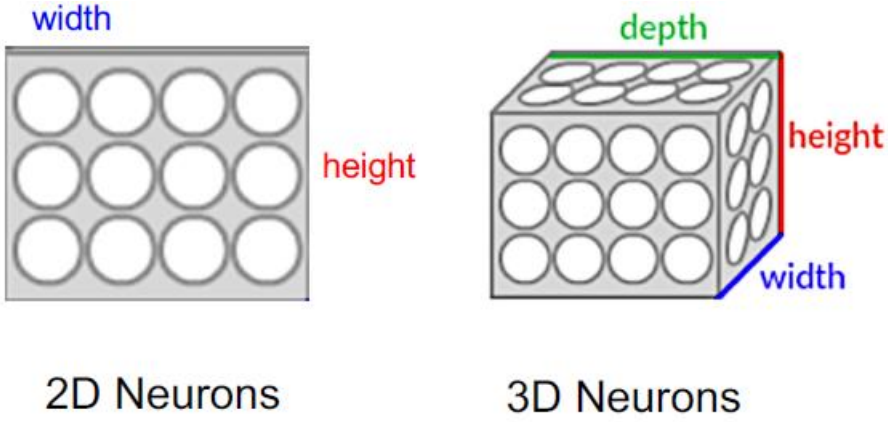
Deep Learning: CNN

- Key elements
  - MLP: 1-D neuron, fully connected



CNN

- The main recipe of a convolutional neural network (CNN) includes:
  - 2D/3D Neurons: The CNN supports layers that have neurons arranged in 2 dimensions (width and height) or 3 dimensions (width, height and depth).
  - Convolutional Layers
  - Pooling Layers
  - Fully connected Layers

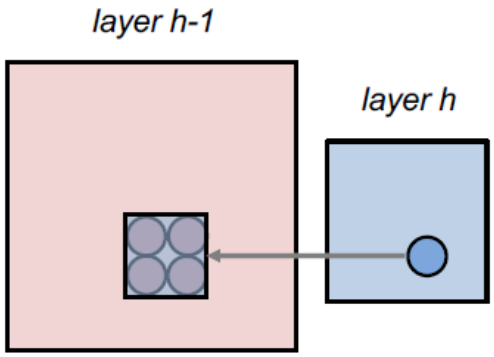


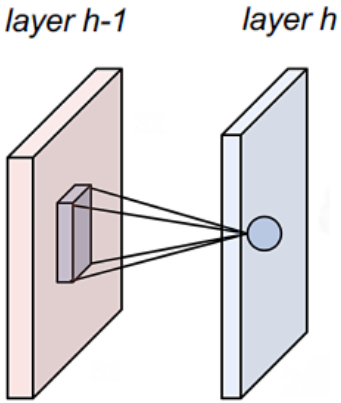
2D Neurons 3D Neurons

- Its training is based on backpropagation and stochastic gradient descent.

CNN

- Convolutional Layer
  - **Local connectivity**: Each neuron inside a layer is connected to only a small region of the previous layer, called a receptive field.

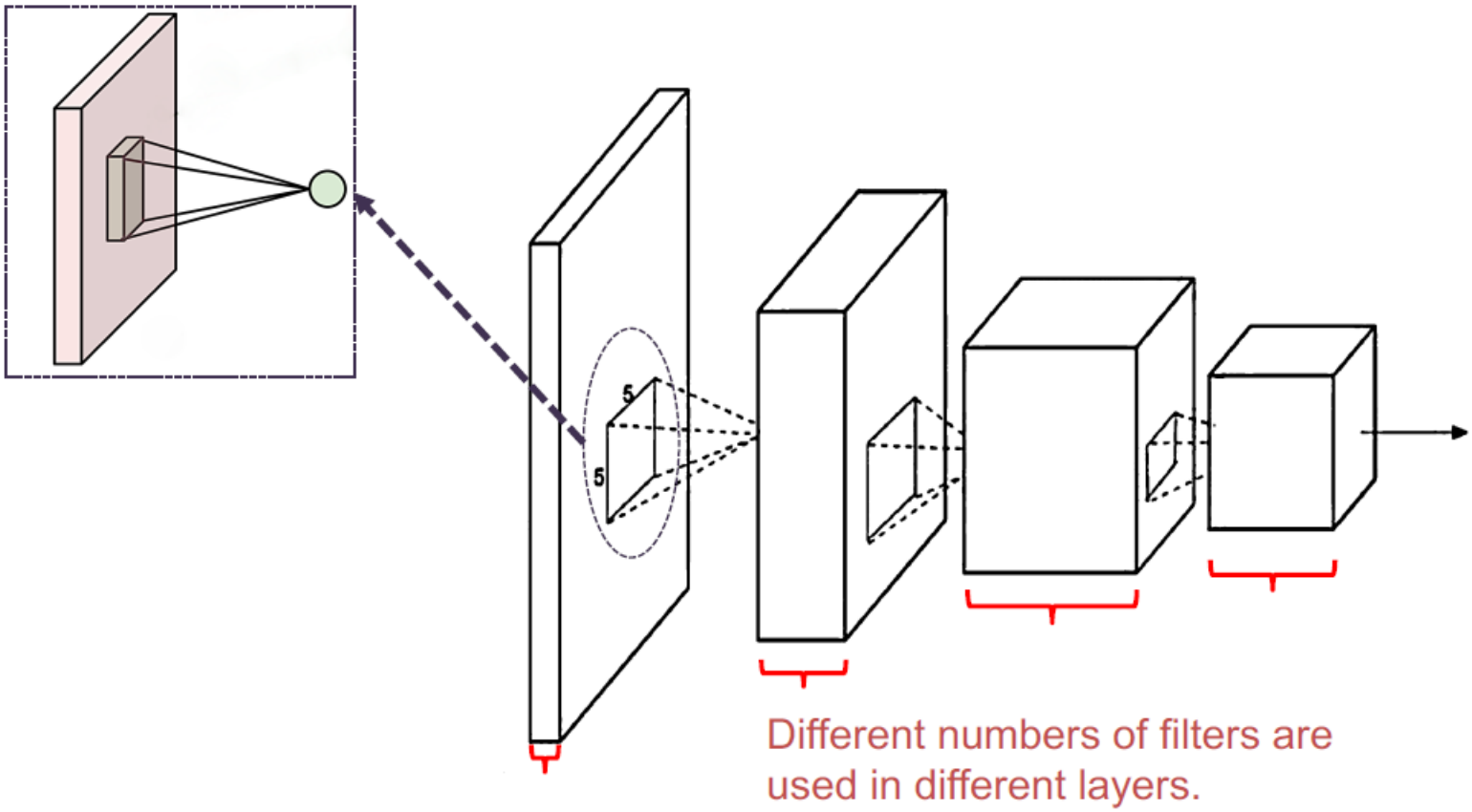




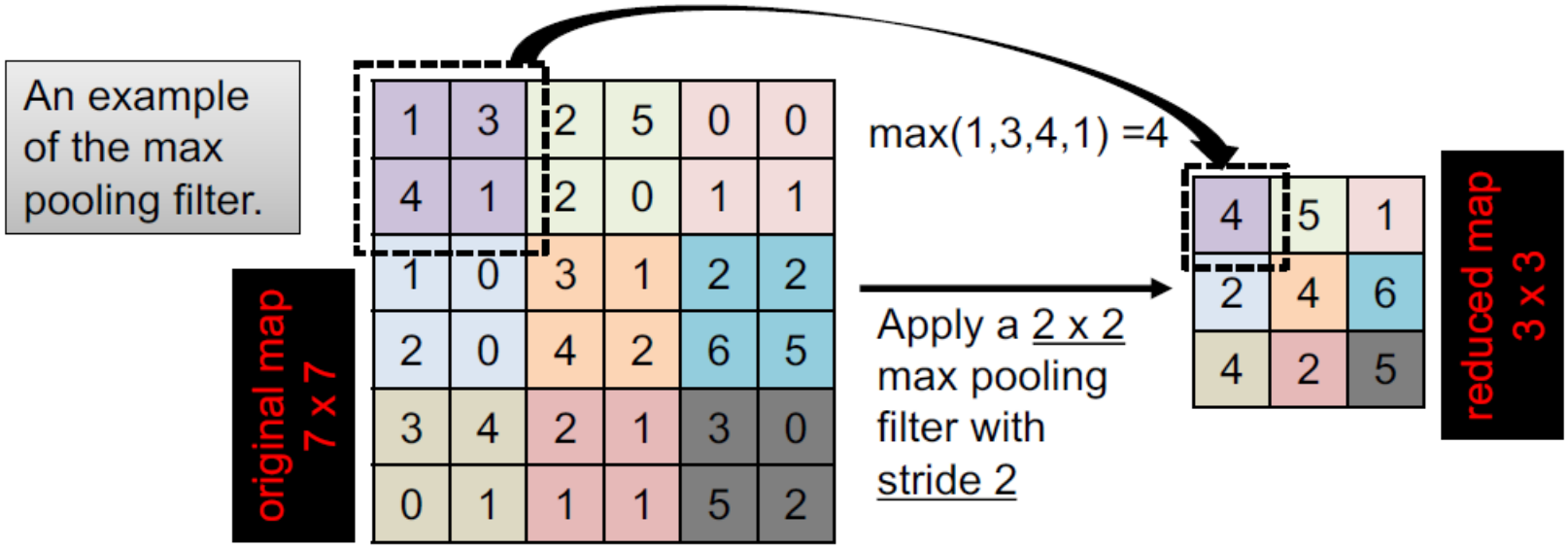
  - The output of a neuron is a number computed from the output of those neurons from the corresponding local region and the weights of the convolutional filter: $y = \text{Activation}(\boldsymbol{w}^T\boldsymbol{x}+b)$
  - Weight sharing: One same filter of the size of the local region slides over all spatial locations (in different words, slides over all the neurons in the layer).

CNN

- Convolutional Layer
  - Many connected convolutional layers:



- Pooling Layer
  - The pooling layer takes the activation maps returned by a convolutional layer as the input, and reduces the size of each activation map separately.
  - It can be viewed as an operation of down-sampling.
  - A pooling filter slides over all the spatial locations in an activation map in the same way as a 2D convolutional filter.

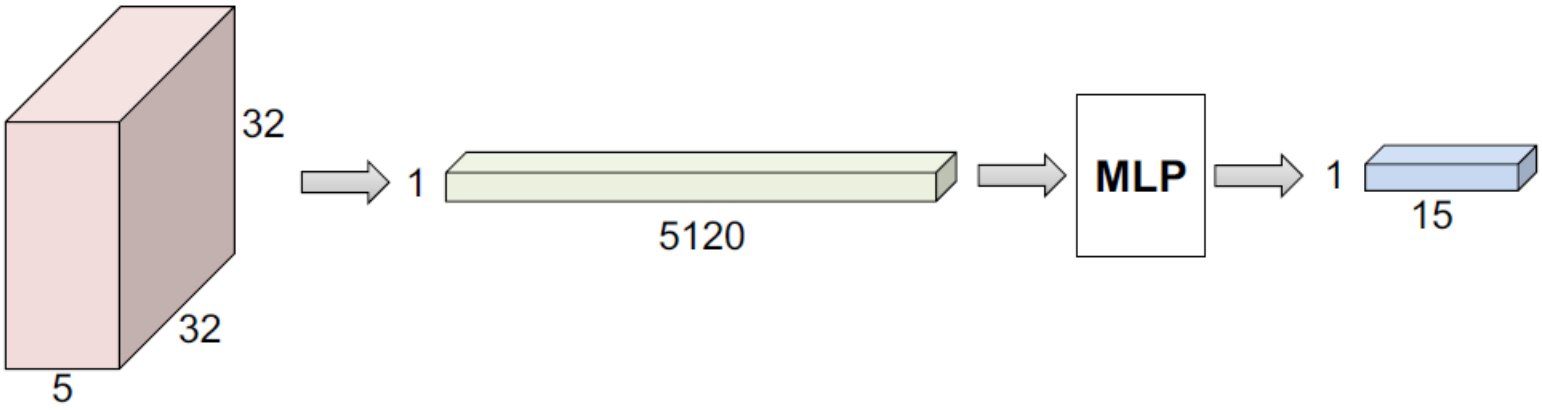


CNN

- Fully Connected Layer
  - The output of a convolutional (or pooling) layer is a set of activation maps, sored in a 3D array.
  - These values in the 3D array can be moved to a 1D array. For instance:

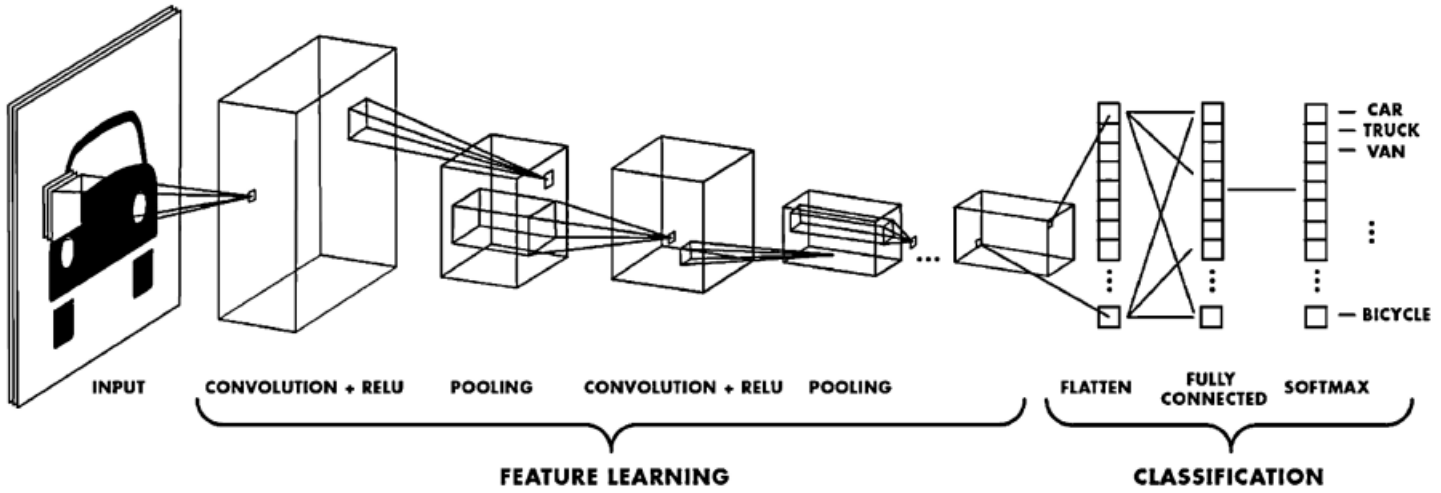
32 x 32 x 5 array ➔ 5120 x1 array

  - The fully connected layers are equivalent to a multilayer perceptron (MLP) taking the generated 1D array as the input.



CNN

- CNN Architecture
  - A CNN is a sequence of convolutional (and pooling) layers, followed by fully connected layers in the end:



Used image is from MathWorks: https://ww2.mathworks.cn/solutions/deep-learning/convolutional-neural-network.html

  - Its unique architecture (2,3D neurons, local connectivity, etc.) makes it suitable for processing 2,3D data with typical local patterns, particularly images (n x n x 3 input where 3 corresponds to the R, G and B channels).

Deep Learning

- Summary

  - Deep learning basics
  - Traditional machine learning vs deep learning
  - Key deep learning techniques: training tips, regularization
  - Application examples of deep learning
  - CNN architecture

Content

- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

Machine Learning in IoT

- General Ideas

  - IoT and machine learning deliver insights otherwise hidden in data for rapid, automated responses and improved decision making.
  - Machine learning for IoT can be used to project future trends, detect anomalies, and augment intelligence by ingesting image, video and audio.

Machine Learning in IoT

- General Ideas

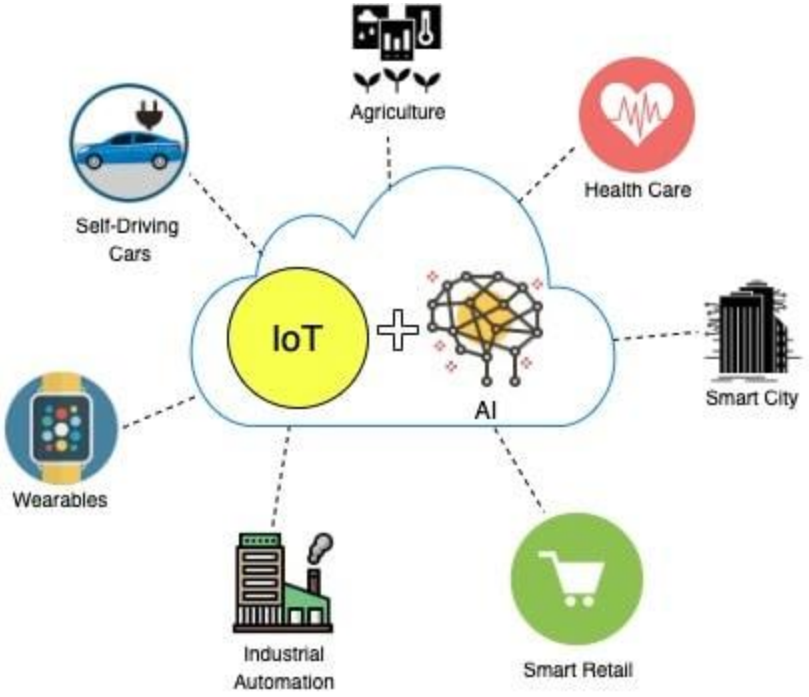
  - The IoT generates massive volumes of data from millions of devices. Machine learning is powered by data and generates insight from it.
  - Machine learning uses past behavior to identify patterns and builds models that help predict future behavior and events.
  - IoT and machine learning deliver insights otherwise hidden in data for rapid, automated responses and improved decision making.
  - Machine learning for IoT can be used to project future trends, detect anomalies, and augment intelligence by ingesting image, video and audio.

Machine Learning in IoT

- Why use machine learning for IoT?
  - Machine learning can help demystify the hidden patterns in IoT data by analyzing massive volumes of data using sophisticated algorithms.
  - Machine learning inference can supplement or replace manual processes with automated systems using statistically derived actions in critical processes.

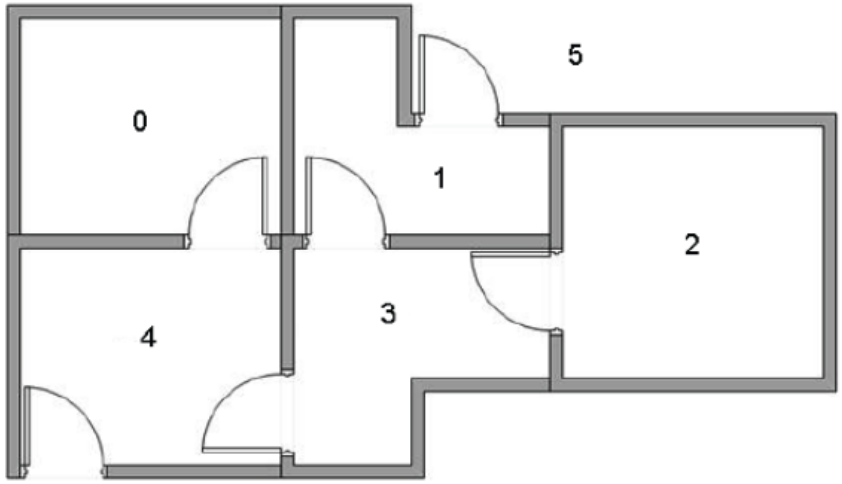
Machine Learning in IoT

- Applications
  - Industrial Automation
  - Agriculture
  - Self-driving cars
  - Wearables and Health-care
  - Smart retailing
  - …

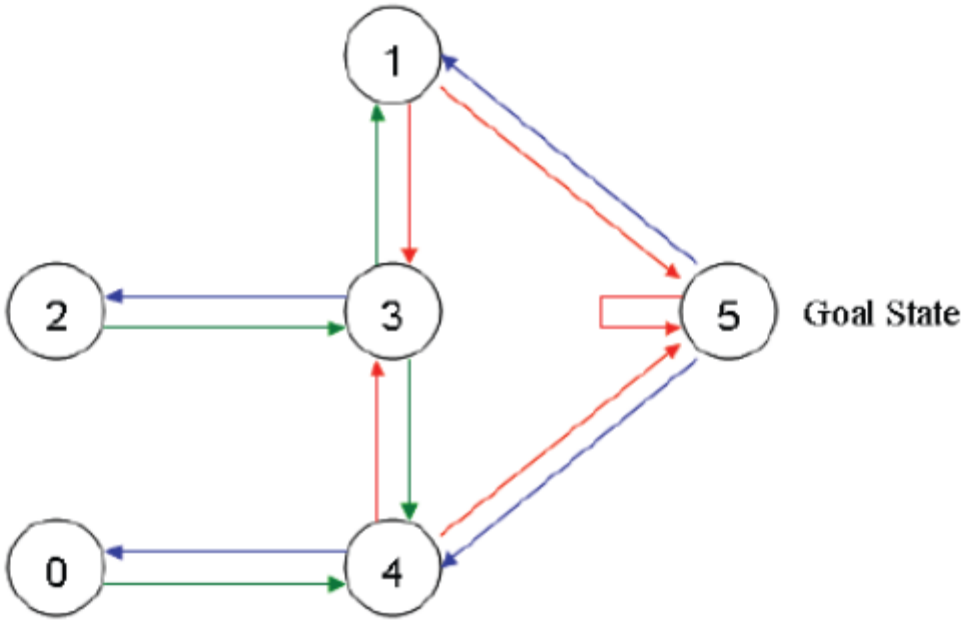


Machine Learning in IoT

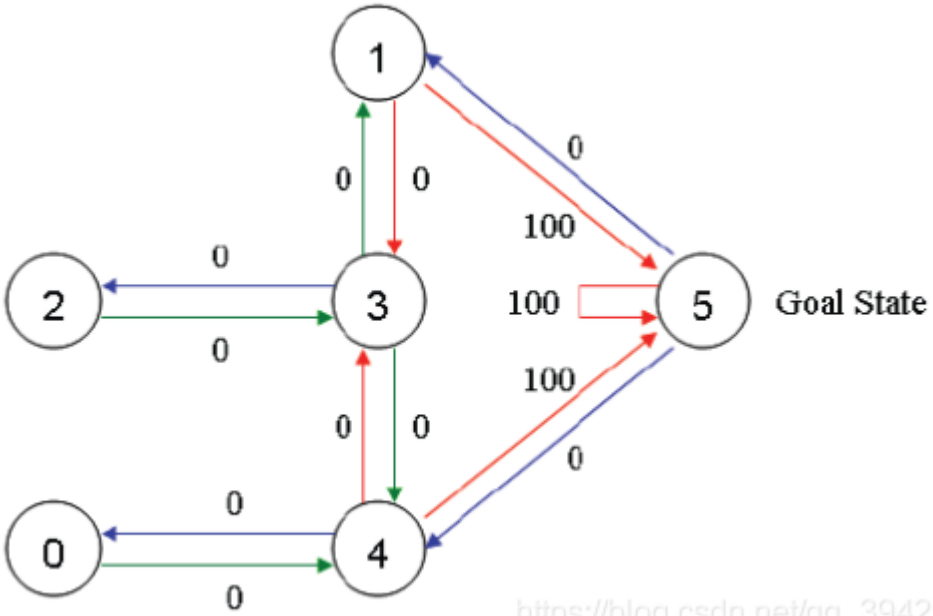
- Case Study I
  - 5 rooms in a building (0-4)
  - Outside space is 5
  - Rooms are connected through doors
- How a robot to get out?



- Case Study I
  - Represent into:

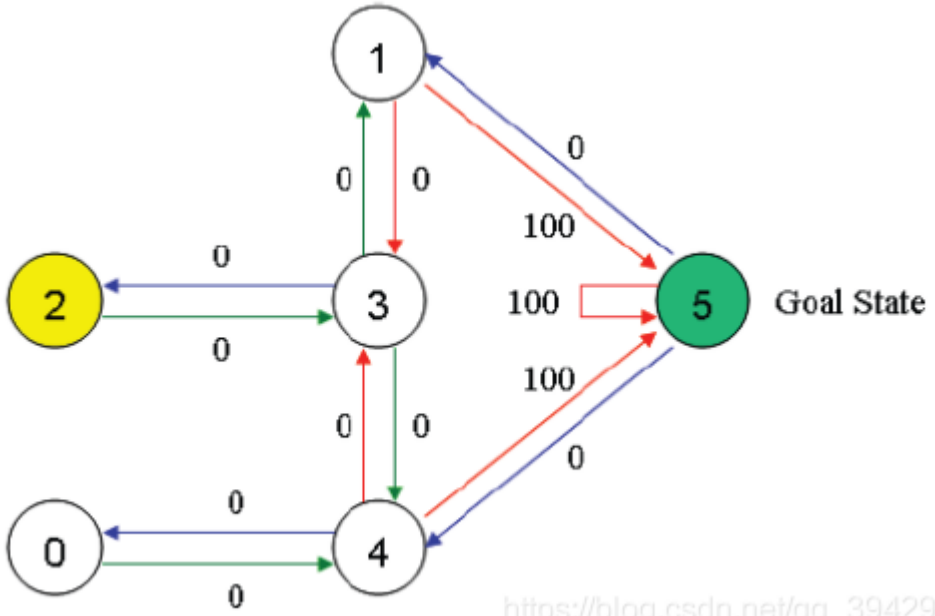


- Case Study I
  - Set rewards:



Machine Learning in IoT

- Case Study I
  - move to space 5
  - State: Room (0-5)
  - Action: Move to connected rooms (Arrows)



- Case Study I
  - Reward matrix

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

  - Q-learning

$$Q(s,a) = R(s,a) + \gamma \cdot \max_{\tilde{a}}\{Q(\tilde{s},\tilde{a})\},$$

Machine Learning in IoT

- Case Study I
  - Q-learning Algorithm:
  - Step 1: Give γ and Reward Matrix
  - Step 2: Set Q=0
  - Step 3 For each episode:
    - Randomly select an original state *s*
    - If the goal is not achieved:

    (1) Select an action

    (2) Get next state according to *a*

    (3) Compute Q(s,a)

    (4) Set state to the new one

    - Repeat until the state is the goal

- Case Study I
  - Set γ=0.8, original state is room 1
  - Original Q table

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

  - Select space 5 as the action

Q(1,5)= R(1,5)+0.8*max{Q(5,1), Q(5,4), Q(5.5)}
=100 +0.8*max(0, 0, 0)
=100

- Case Study I
  - Update Q table

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

  - As space 5 is goal, start next episode, select room 3 as original state
  - Randomly select room 1 as an action

Q(3,1)=R(3,1)+0.8*max{Q(1,3), Q(1,5)}
=0+0.8*max{0, 100}
=80

Machine Learning in IoT

- Case Study I
  - Update Q table

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

  - Start next episode, select room 3 as original state
  - Randomly select room 1 as an action
  - Make current state as room 1
  - Select space 5 as an action

$$Q(1,5)=R(1,5)+0.8*\max\{Q(5,1),\ Q(5,4),\ Q(5,5)\}$$
$$=100+0.8*\max\{0,\ 0,\ 0\}$$
$$=100$$

  - As space 5 is the goal, this episode is finished

Machine Learning in IoT

- Case Study I
  - Complete more episodes
  - Obtain the final converged Q table

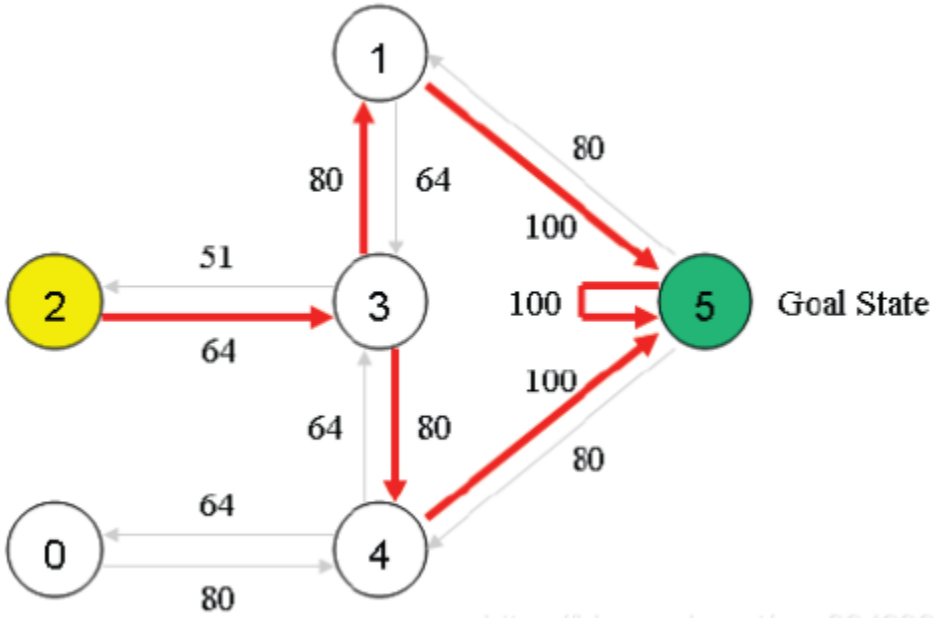
$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

  - Normalization

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

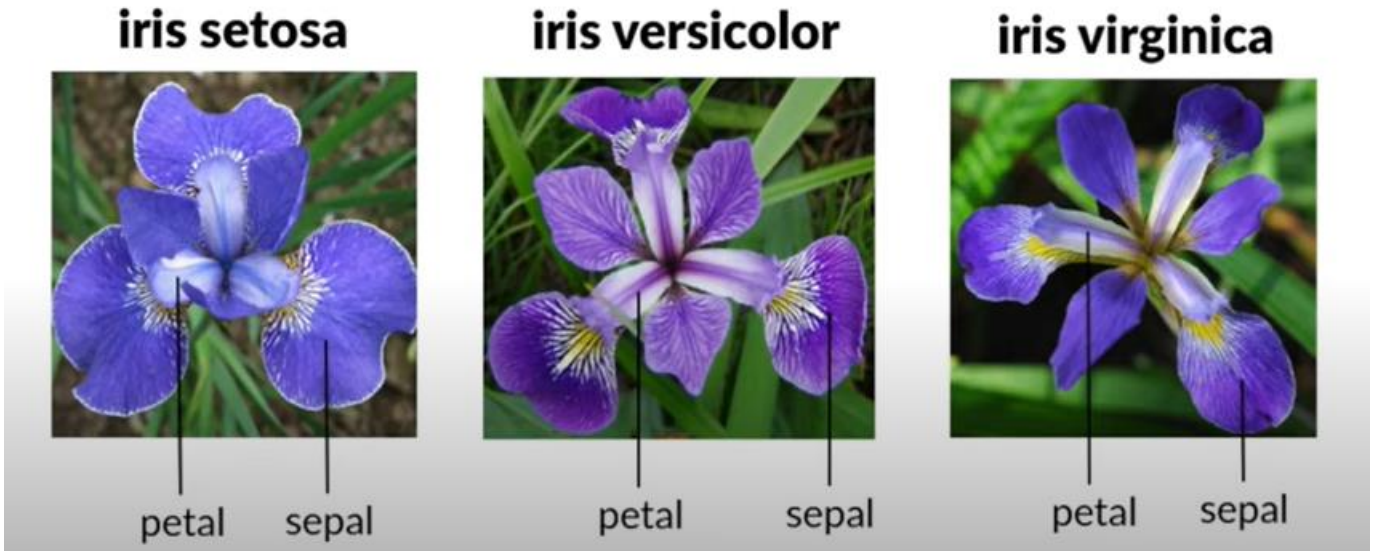
Machine Learning in IoT

- Case Study I
  - Obtain the best route
  - E.g. From room 2, the best route to space 5 is
  - 2-3-1-5 or 2-3-4-5



Machine Learning in IoT

- Case Study II
  - Recognize plant and flower species in smart agriculture
  - Iris classification
    - Create the dataset
    - Build the model
    - Train the model
    - Make predictions

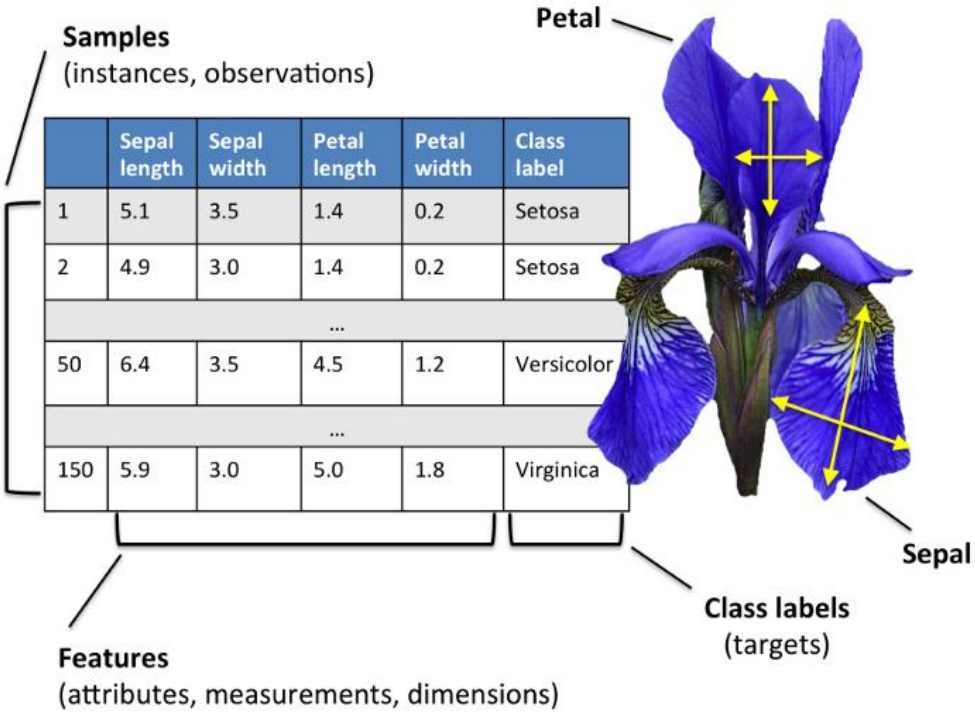


- Case Study II - Iris classification

  - Create the dataset
    - sklearn comes with the inbuilt datasets for the iris classification problem.
    - The dataset consists of

      - 150 samples
      - 3 labels: species of Iris (*Iris setosa, Iris virginica* and *Iris versicolor*)
      - 4 features: Sepal length,Sepal width,Petal length,Petal Width in cm

- Case Study II - Iris classification
  - Create the dataset



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

- Case Study II - Iris classification
  - Create the dataset
    - Load the Iris dataset.

```
from  sklearn import  datasets
iris=datasets.load_iris()
```

    - Assign the data and target to separate variables. x contains the features and y contains the labels.

```
x=iris.data
y=iris.target
```

    - Splitting the dataset. x_train contains the training features. x_test contains the testing features. y_train contains the training label. y_test contains the testing labels.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.5)
```

- Case Study II - Iris classification
  - Build the model
    - Use decision tree algorithm. Create the empty model.

```
from sklearn import tree
classifier=tree.DecisionTreeClassifier()
```

    - At this point, we have just made the model, but it cannot be able to predict whether the given flower belongs to which species of Iris. We have to train the model with the features and the labels.
  - Train the model
    - Use *fit* function.

```
classifier.fit(x_train,y_train)
```

    Now the model is ready to make predictions.

Machine Learning in IoT

- Case Study II - Iris classification
  - Make predictions
    - Use *predict* function.

```
predictions=classifier.predict(x_test)
```

These predictions can be matched with the expected output to measure the accuracy value.

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,predictions))
```

  - Full code

```
from  sklearn import  datasets
iris=datasets.load_iris()
x=iris.data
y=iris.target
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.5)
from sklearn import tree
classifier=tree.DecisionTreeClassifier()
classifier.fit(x_train,y_train)
predictions=classifier.predict(x_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,predictions))
```

https://medium.com/@jebaseelanravi96/machine-learning-iris-classification-33aa18a4a983

Machine Learning in IoT

- Summary
  - Machine learning and IoT are one of the topmost trending topics.
  - The machine learning field is growing rapidly, along with IoT.
  - Small cameras and other IoT components are now easily available on mobile devices, computers, traffic control systems, parking systems, and home appliances.
  - Machine learning and IoT help automate the daily task that occurs in businesses.
  - IoT sensors can provide the details of resources that are not useful for business, and here machine learning presents analyze data with the help of algorithms.
  - Machine learning can reduce human errors, allowing collected data to provide real-time insights.
  - More applications will appear in future

Content

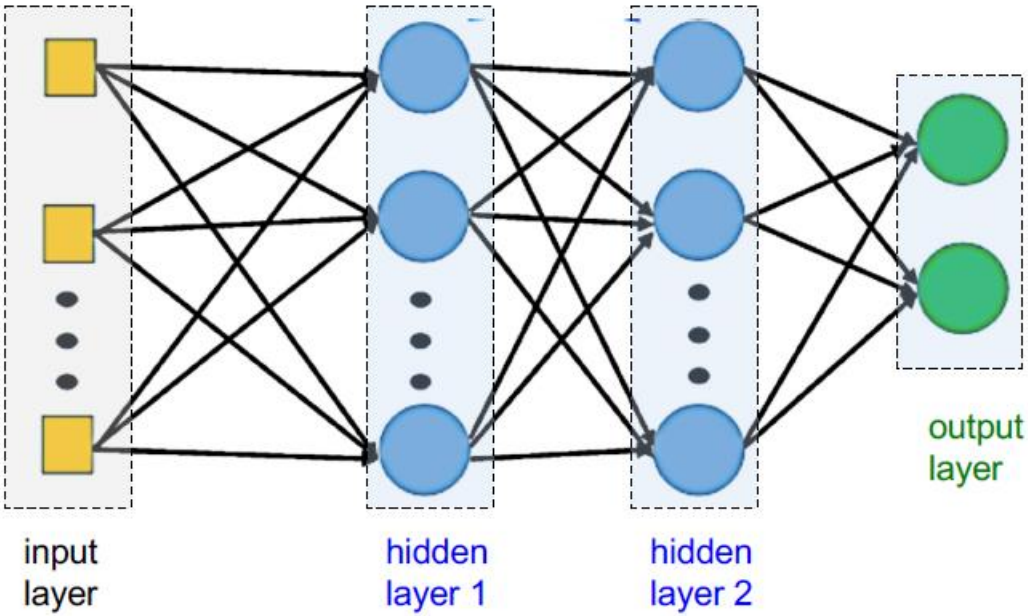
- Machine Learning Basics
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Deep Learning
- Machine Learning in IoT
- Deep Learning in IoT

Deep Learning in IoT

- Deep learning compared with conventional machine learning:
  - Alleviate the requirement for supervised feature sets to be utilized for training.
  - Features that might not be recognizable to a human can be extracted smoothly by deep learning models.
  - Generate more accurate prediction results.
  - Appropriate for modeling intricate behaviors of heterogeneous datasets.

Deep Learning in IoT

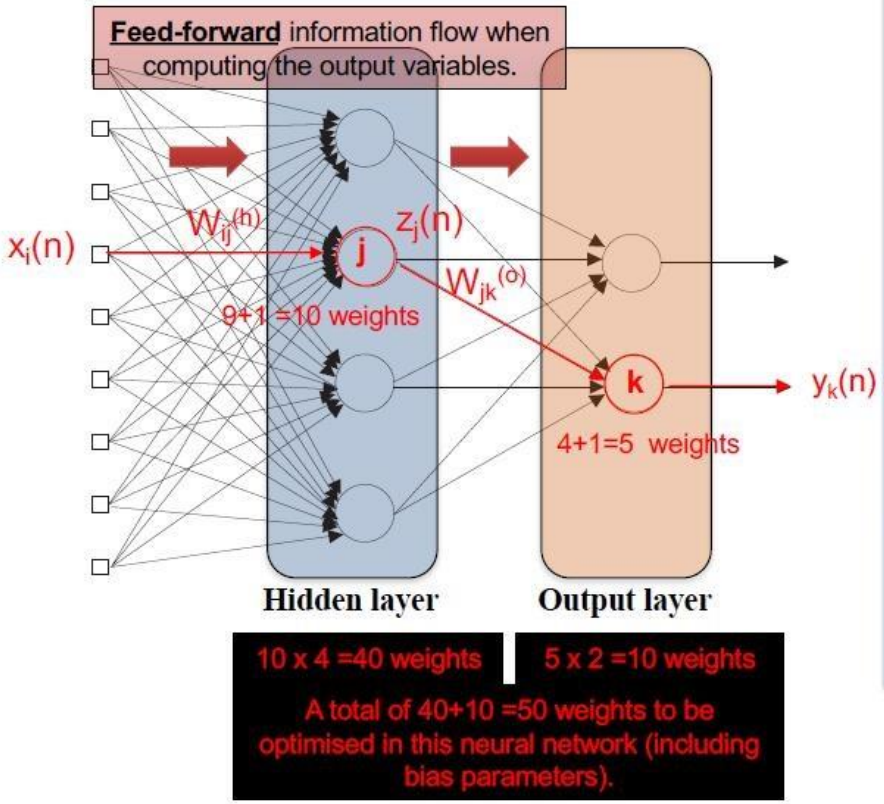
- Multilayer Perceptron
  - A multilayer perceptron (MLP), also called feedforward artificial neural network, consists of at least three layers of nodes: input, hidden (at least one) and output layers.



- Number of neurons in the input layer is equal to the number of input features.
- Number of hidden layers is a hyperparameter to be set.
- Numbers of neurons in hidden layers are hyperparameters to be set.
- Number of neurons in output layer depends on the task to be solved.

Deep Learning in IoT

- An MLP Example
  - With one hidden layer consisting of 4 hidden neurons. It takes 9 input features and returns 2 output variables (9 input neurons in input layer, 2 output neuron in output layer).



- Output of the j-th neuron in the hidden layer (j=1,2,3,4), for the n-th training sample:

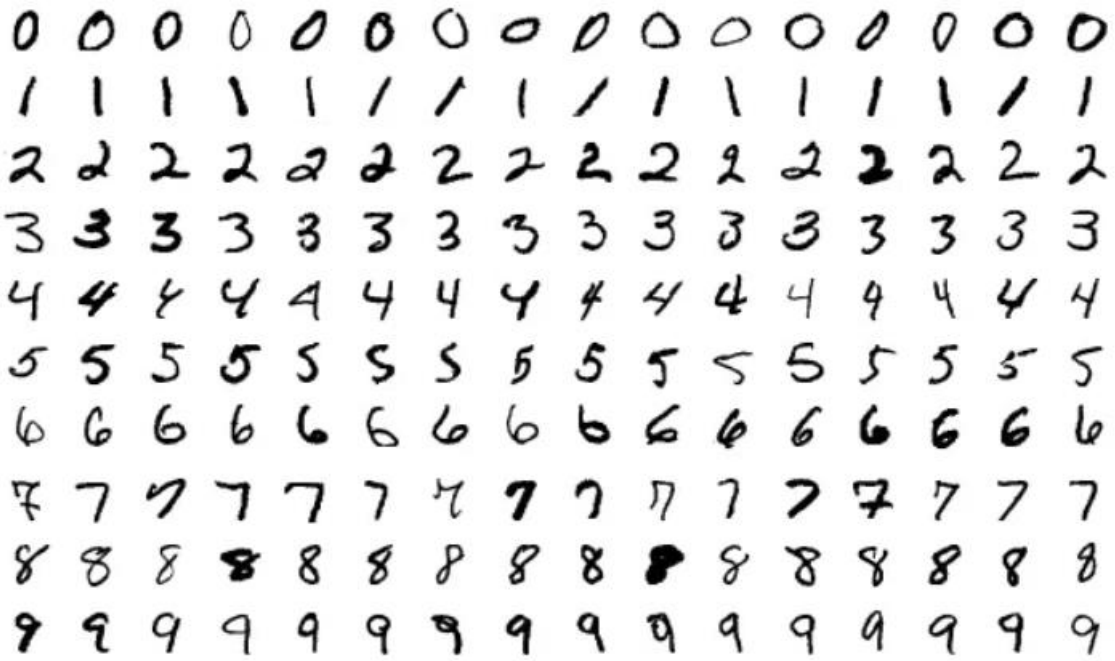
$$z_j(n) = \varphi\left(\sum_{i=1}^{9} w_{ij}^{(h)} x_i(n) + b_j^{(h)}\right)$$

- Output of the k-th neuron in the output layer (k=1,2), for the n-training sample:

$$y_k(n) = \varphi\left(\sum_{j=1}^{4} w_{jk}^{(o)} z_j(n) + b_k^{(o)}\right)$$

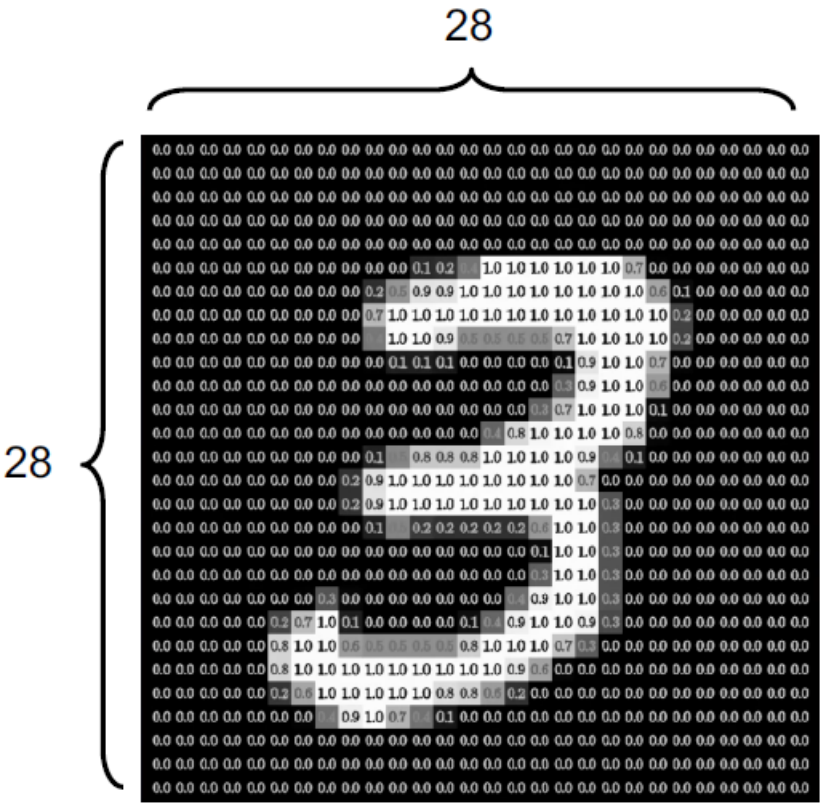
Deep Learning in IoT

- Case Study III
- Handwriting recognition
  - MNIST Dataset, which is famous database of handwritten digits. There are handwritten images of 10 different digits.

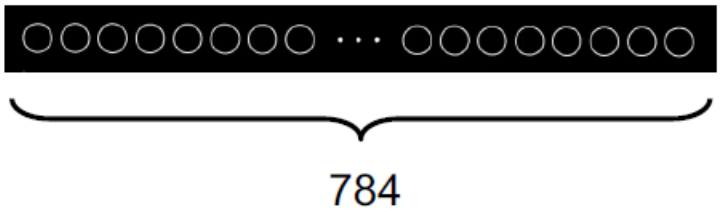


Deep Learning in IoT

- Handwriting recognition



Put these pixels in one long vector of dimension 28 x 28 =784.



Deep Learning in IoT

IoTrain
Master of Engineering in Internet of Things

- Handwriting Recognition
  - Data import and preparation (Python)

```
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier

# Load data
X, y = fetch_openml("mnist_784", version=1, return_X_y=True)

# Normalize intensity of images to make it in the range [0,1] since
255 is the max (white).
X = X / 255.0
```

  - Model training

3 hidden layers with 50, 20, and 10 neurons each, respectively. Set the max iterations to 100 and the learning rate to 0.1, which are hyperparameters.

```
# Split the data into train/test sets
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]

classifier = MLPClassifier(
    hidden_layer_sizes=(50,20,10),
    max_iter=100,
    alpha=1e-4,
    solver="sgd",
    verbose=10,
    random_state=1,
    learning_rate_init=0.1,
)

# fit the model on the training data
classifier.fit(X_train, y_train)
```

Deep Learning in IoT

- Handwriting Recognition
  - Model evaluation

    Estimate the mean accuracy on the training and test data and labels.

```
print("Training set score: %f" % classifier.score(X_train, y_train))
print("Test set score: %f" % classifier.score(X_test, y_test))
```

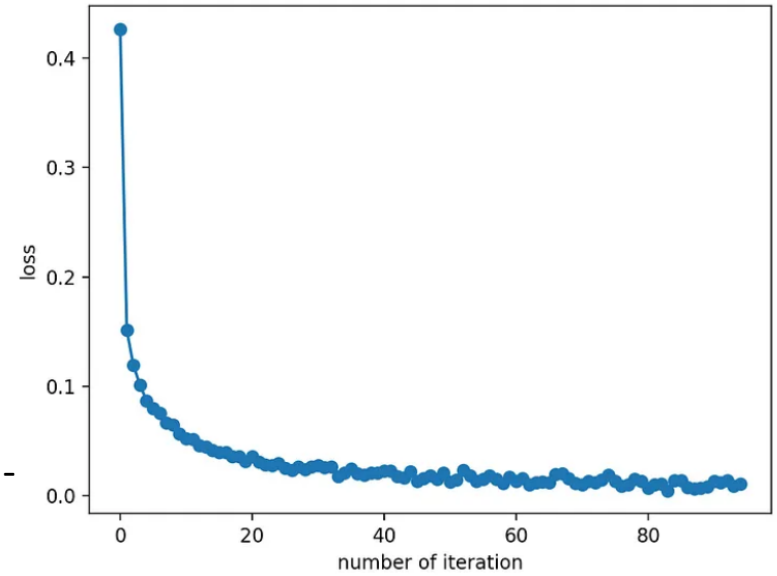
    Get

    Training set score: `0.998633`

    Test set score: `0.970300`

  - Visualizing the cost function evolution

    Loss decreases really fast during training and it saturates after the 40th iteration.



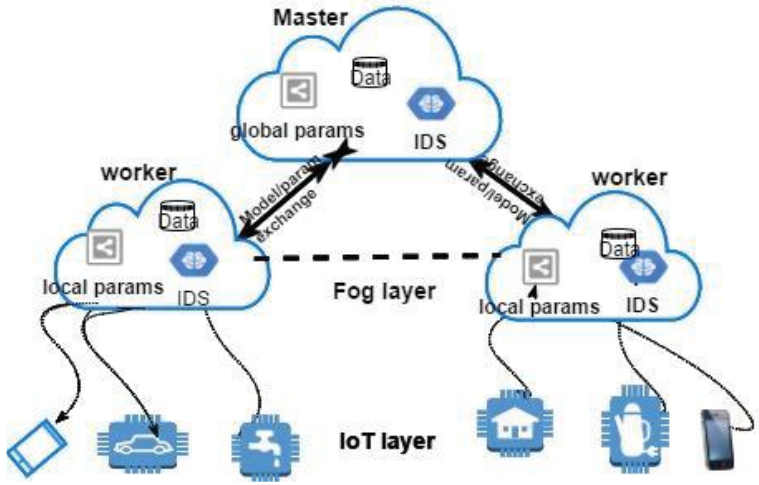
https://towardsdatascience.com/classifying-handwritten-digits-using-a-multilayer-perceptron-classifier-mlp-bc8453655880

Deep Learning in IoT

- Case Study IV
- Attack detection in Fog-to-Things Computing
  - Features of Fog-to-Things:
    - Solve the limitation of cloud computing
    - Distribute resources and services on distributed fog nodes
    - Pooling of idle resources along the cloud-to-thing continuum
    - Applications are closer to end users
    - Support delay-sensitive applications and enable data analytics at network edge
    - Save network resources and response time

Deep Learning in IoT

- Case Study IV
- Attack detection in Fog-to-Things Computing
  - Process:
    - Initialize deep learning parameters on the master fog node and send to worker nodes
    - Local training and hyper-parameter optimization
    - Use optimizers and gradients of DL to update parameters on each worker node
    - Aggregate parameter updates on the master fog node



  - The worker IDSs detect malicious events locally, while the master IDS can be responsible for updating the parameters using gradient descent

Deep Learning in IoT

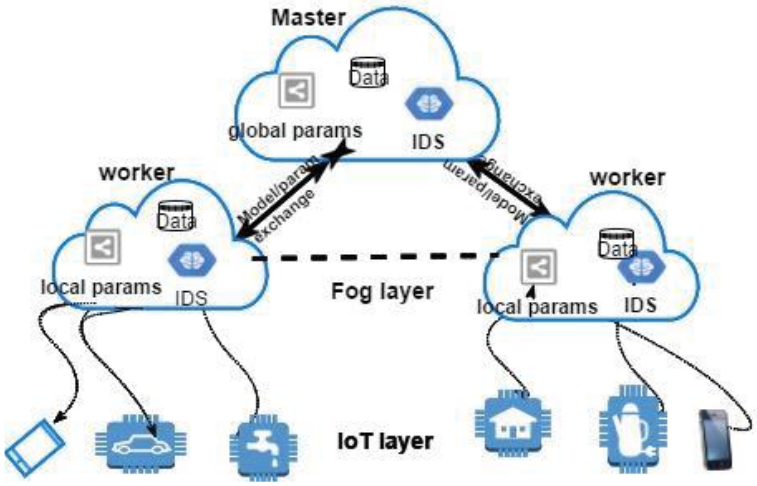
- Case Study IV
- Attack detection in Fog-to-Things Computing
  - Process:

- Initialize parameters on master node
- For threads n_c on node n, done in parallel:
  - Get training example $i \in \mathrm{DAT}_{\mathrm{Anac}}$
  - Update $w_{ji} \in W_n$, biases $b_{ji} \in b_n$

$$W_{ji} := W_{ji} - \alpha \frac{\partial L(W, b \mid j)}{\partial W_{ji}}$$

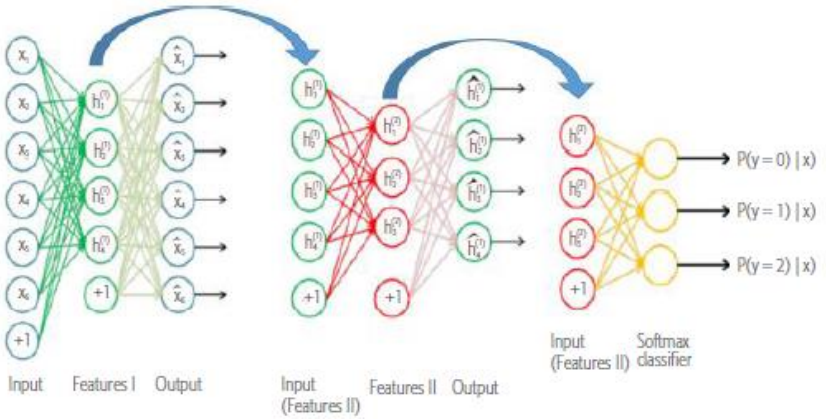
$$b_{ji} := W_{ji} - \alpha \frac{\partial L(W, b \mid j)}{\partial b_{ji}}$$

- Send updates of W_n, b_n to master node
- Apply the update parameters on the master node, and propagate the update



Deep Learning in IoT

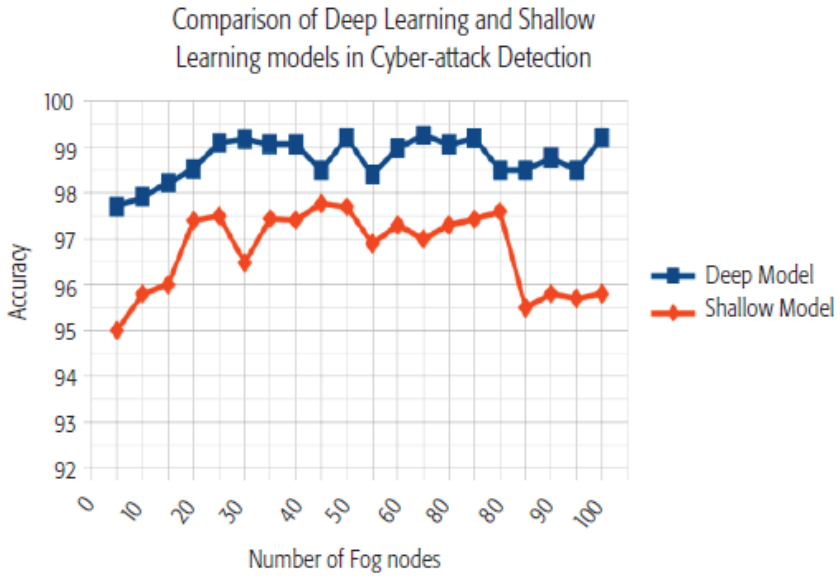
- Case Study IV
- Attack detection in Fog-to-Things Computing
  - Training Process:
    - The system is trained without labels using stacked autoencoders as a means of self-taught networks to extract hidden features (features I)
    - Obtained features I are applied to the test data to extract end features (features II) for softmax classification



  - 150 first layer neurons, 120 second layer neurons, 50 third layer neurons, and the last softmax layer with neurons equal to the number of classes

Deep Learning in IoT

- Case Study IV
- Attack detection in Fog-to-Things Computing
  - Publicly available NSL-KDD dataset



- Reference:

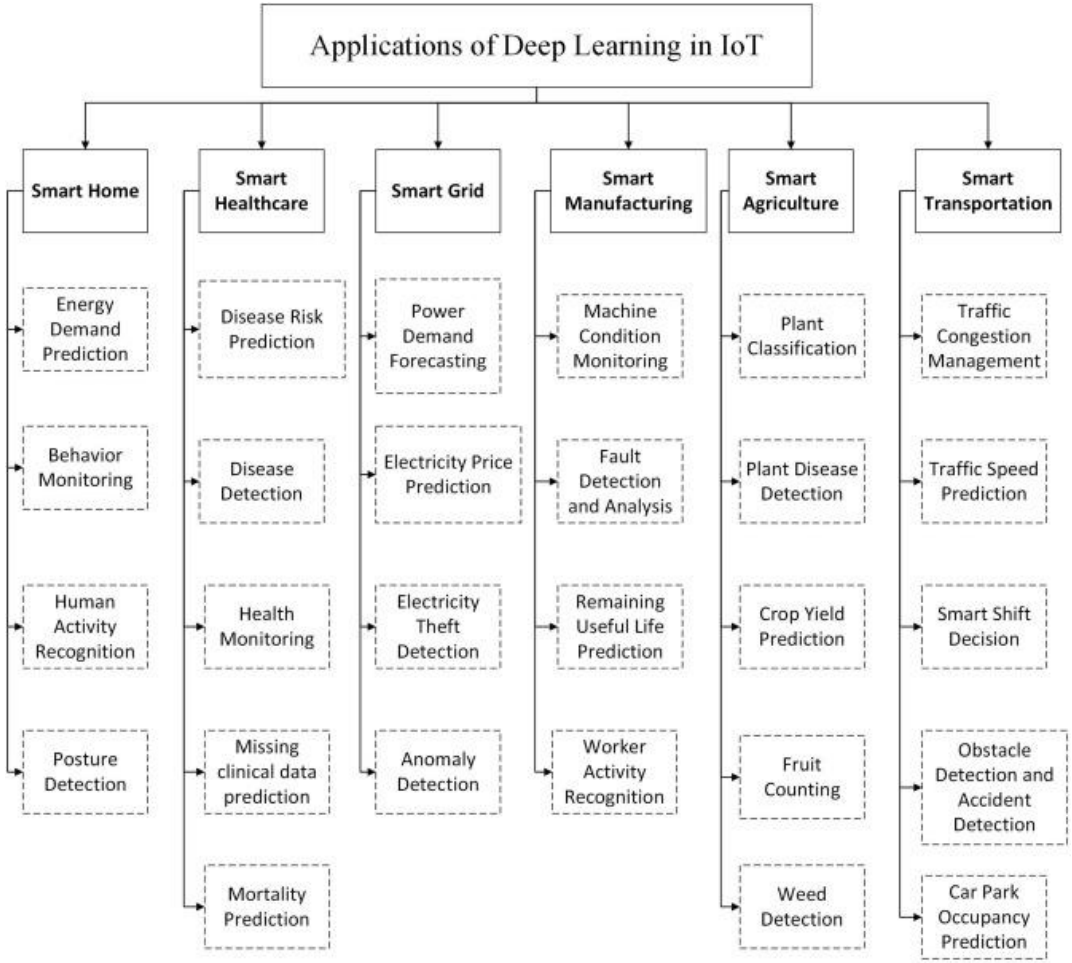
Abeshu, A., & Chilamkurti, N. (2018). Deep learning: The frontier for distributed attack detection in fog-to-things computing. IEEE Communications Magazine, 56(2), 169-175.

Deep Learning in IoT

- Potential benefits of deep learning for IoT
  - The data from the IoT has a timestamp for all data points, which is time series data. Deep learning models are very good at analyzing time-series data.
  - IoT data is affected by noise in the process of data acquisition and data transmission. Deep learning models have the ability to tolerate noise.
  - Data produced by IoT devices has weak semantics. Deep learning models have the capability to dig out high-level characteristics from the weak-semantics data.

Deep Learning in IoT

- Applications



Deep Learning in IoT

- Challenges and future directions
  - Massive scale of IoT data. The massive quantity of data creates a huge challenge for deep learning in terms of time and structure complexities.
  - Data pre-processing. The systems deals with data from diverse sources and may contain noisy, unlabeled and missing data.
  - High velocity. The rate of production of IoT data puts forth a challenge of fast processing and analytics of such data.
  - Heterogeneity. Managing the conflicts between diverse data formats is a challenge that requires a viable solution.
  - Deep learning for IoT devices. Deploying deep learning models on resource constrained IoT devices is a major challenge for IoT device designers.