UNSW AUSTRALIA

Classification (I)

Never Stand Still

COMP9417 Machine Learning & Data Mining

Term 1, 2020

Adapted from slides by Dr Michael Bain

Aims

This lecture will introduce you to machine learning approaches to the problem of classification. Following it you should be able to reproduce theoretical results, outline algorithmic techniques and describe practical applications for the topics:

- outline a framework for solving machine learning problems
- outline the general problem of induction
- describe issues of generalisation and evaluation for classification
- outline the use of a linear model as a 2-class classifier
- describe distance measures and how they are used in classification
  - outline the basic k-nearest neighbour classification method

Introduction

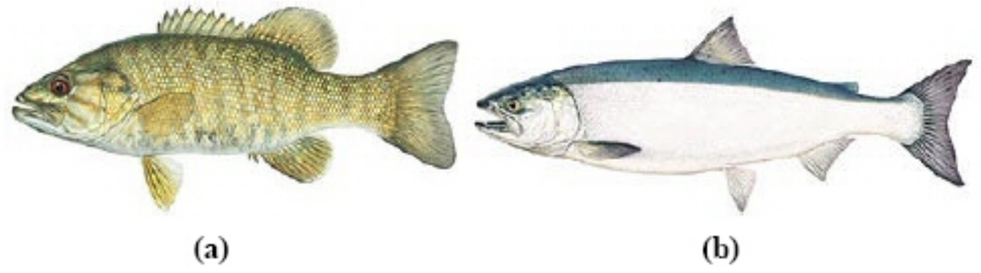
Classification (sometimes called *concept learning*) methods dominate machine learning . . .

. . . however, they often don't have convenient mathematical properties like regression, so are more complicated to analyse. The idea is to learn a *classifier*, which is usually a function mapping from an input data point to one of a set of discrete outputs, i.e., the classes.

We will mostly focus on their advantages and disadvantages as learning methods first, and point to unifying ideas and approaches where applicable.

Classification

Example: Imagine that we want to automate the process of sorting incoming fish in a fish-packing plant. And as a pilot, we start by separating sea bass from salmon using some information collected through sensing.

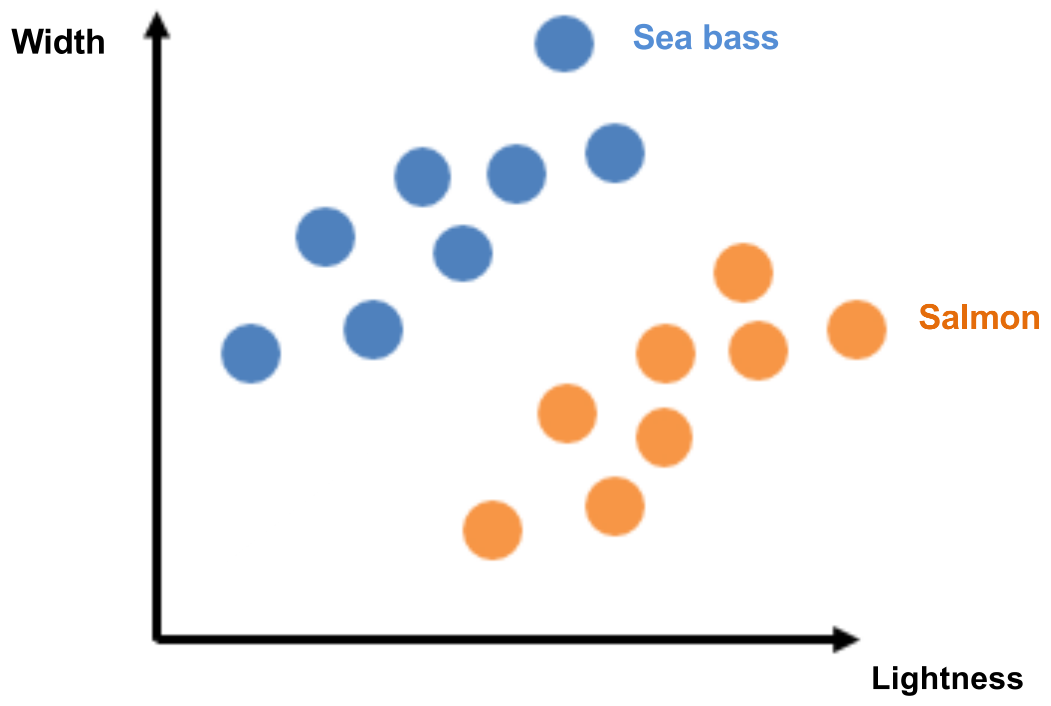


Classification

Example: classifying sea bass vs. salmon

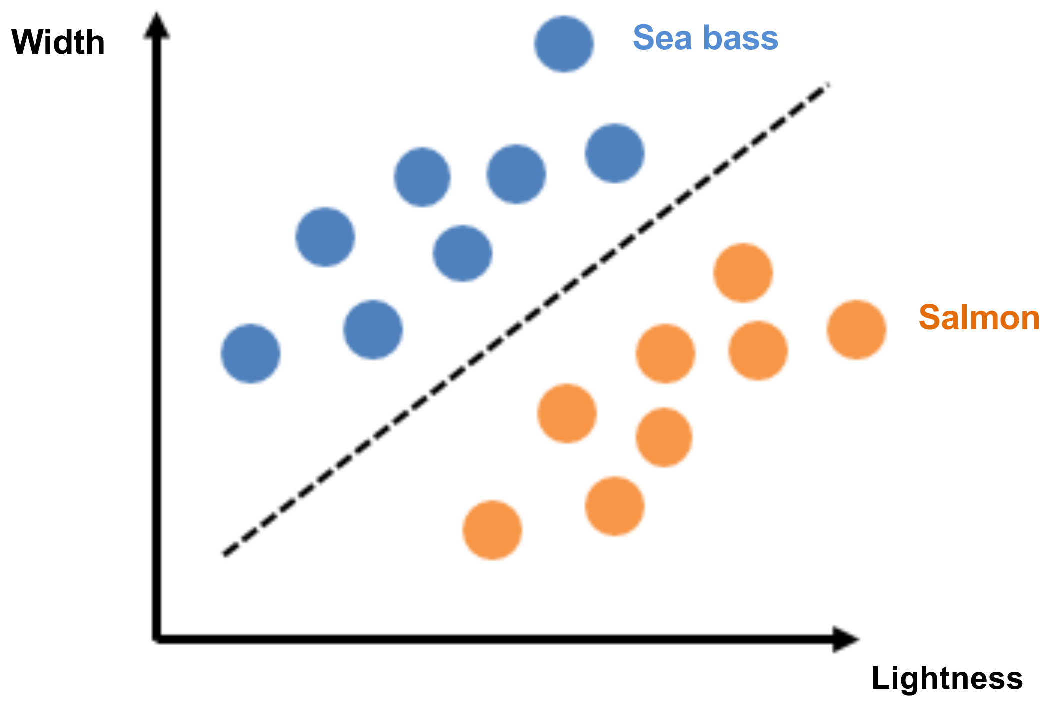
Features that can be used: width, length, weight, lightness, fins, eyes/mouth position, etc.

Question: how to separate these two classes?



Classification

Example: Maybe we can find a line that separates the two classes.



Classification

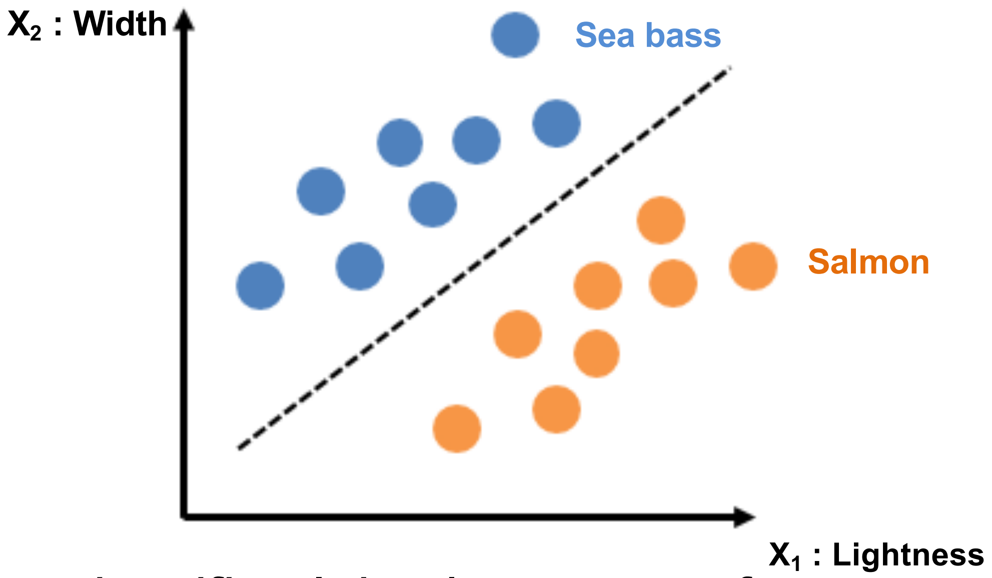
Example: If we find the line that separated the two classes, then how our algorithm makes prediction?

The line equation will look like:
$ax_1 + bx_2 + c = 0$

We can define a, b & c such that:

for any point above the line:
$ax_1 + bx_2 + c > 0$

and for any point below the line:
$ax_1 + bx_2 + c < 0$

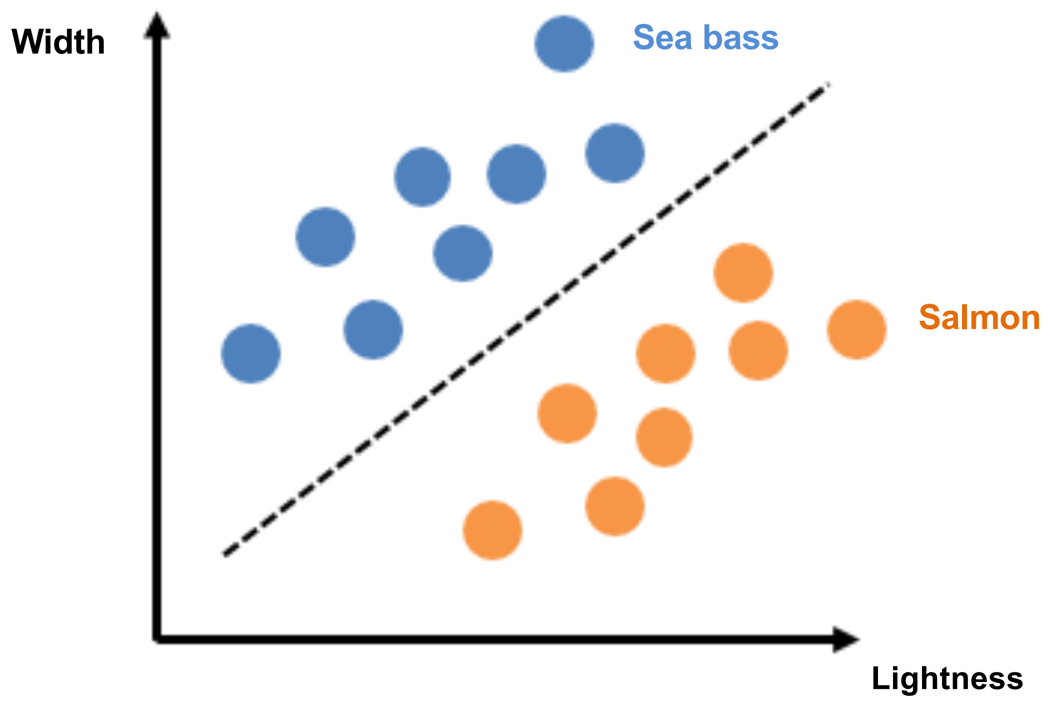


This type of classifier is called *linear classifier*. It is also a type of *discriminative learning* algorithm.

Classification

Example:

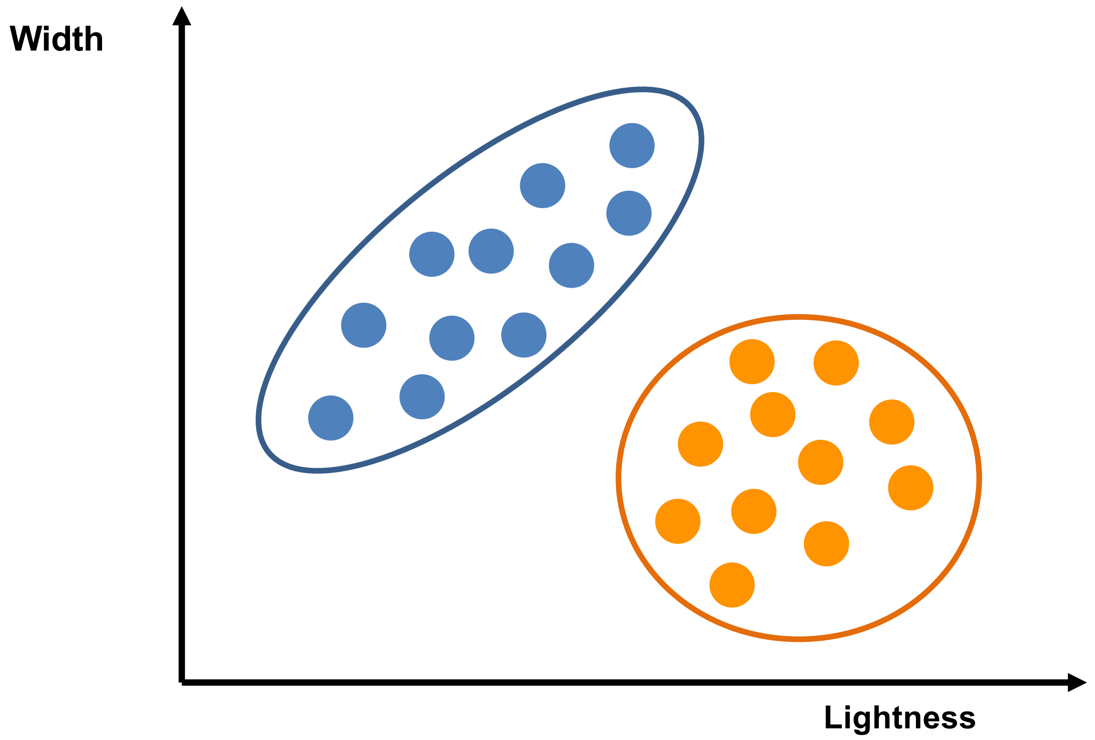
Can we do something different than finding the discriminative line (or some boundary) to be able to separate the two groups?



Classification

Example:

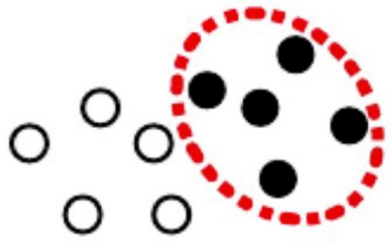
Instead of finding a discriminative line, maybe we can focus on one class at a time and build a model that describes how that class looks like; and then do the same for the other class. This type of models are called *generative learning algorithm*.



Classification

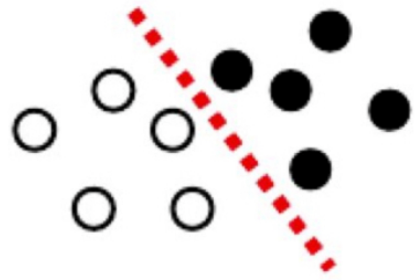
Generative algorithm: builds some models for each of the classes and then makes classification predictions based on looking at the test example and see it is more similar to which of the models.

- Learns $p(x|y)$ (and also $p(y)$, called class prior)
- So we can get $p(x,y) = p(x|y)p(y)$
- It learns the mechanism by which the data has been generated



Discriminative algorithm: Do not build models for different classes, but rather focuses on finding a decision boundary that separates classes

- Learns $p(y|x)$



Classification

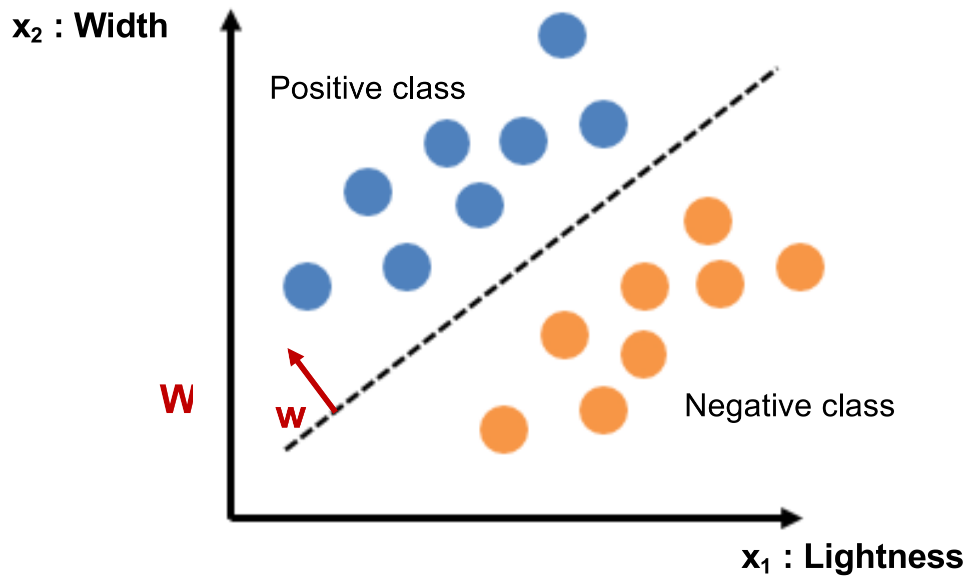
- To predict the output for sample x, in generative algorithm, we have to estimate $p(y|x)$:

$$p(y=0|x) = \frac{p(x|y=0)p(y=0)}{p(x)}$$
$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)}$$

If $p(y=0|x) > p(y=1|x)$, then X belongs to class $y=0$ and otherwise to class $y=1$.

- For discriminate algorithm, we can directly have $p(y=0|x)$ and $p(y=1|x)$ and similar to above, if $p(y=0|x) > p(y=1|x)$, then x belongs to class $y=0$ and otherwise to class $y=1$.

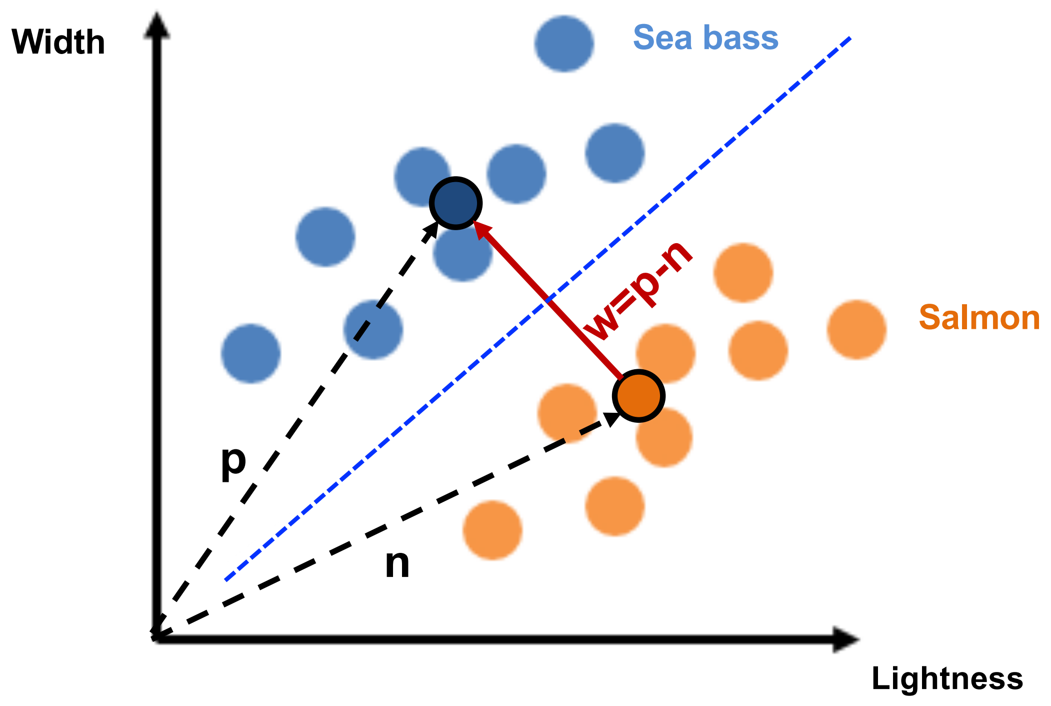
Linear classification in two dimensions



- We find the line that separates the two class: $ax_1 + bx_2 + c = 0$
- We define a weight vector $w^T = [a, b]$, $x^T = [x_1, x_2]$
- So the line can be defined by $x^T w = -c = t$
- w is perpendicular to decision boundary (in direction of positive class)
- t is the decision threshold (if $x^T w > \mathrm{t}$ then x belongs to positive class and if $x^T w < \mathrm{t}$ then x belongs to negative class)

Basic Linear Classifier

The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive and negative centres of mass.



Basic Linear Classifier

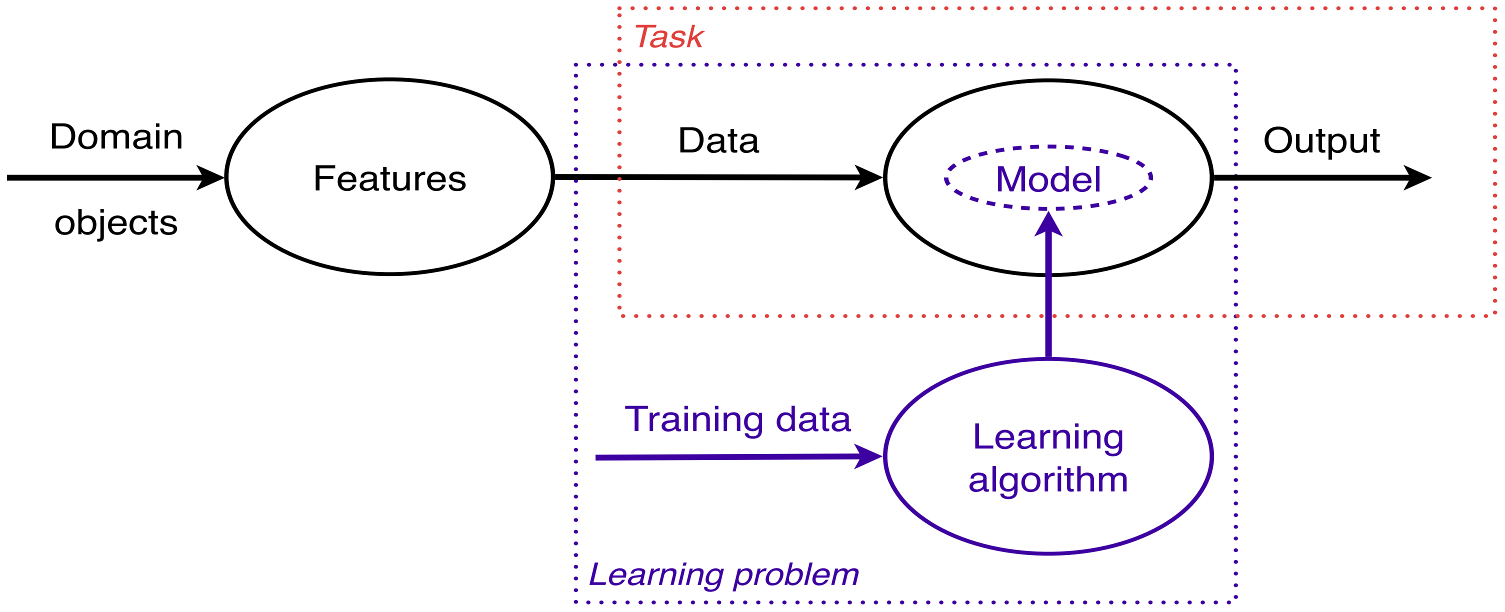
The basic linear classifier is described by the equation $x^T w = t$, and $w = p - n$

As we know, $\frac{p+n}{2}$ is on the decision boundary, so we have:

$$t = (\frac{p+n}{2})^T . (p - n) = \frac{||p||^2 - ||n||^2}{2}$$

Where $||x||$, denotes the length of vector x

How solve a task with machine learning



An overview of how machine learning is used to address a given task. A task (red box) requires an appropriate mapping – a model – from data described by features to outputs. Obtaining such a mapping from training data is what constitutes a learning problem (blue box).

Some terminology

Tasks are addressed by models, whereas learning problems are solved by learning algorithms that produce models.

Some terminology

Machine learning is concerned with using the right features to build the right models that achieve the right tasks.

Some terminology

Does the algorithm require all training data to be present before the start of learning ? If yes, then it is categorised as **batch learning** (a.k.a. **offline learning**) algorithm.

If however, it can continue to learn a new data arrives, it is an **online learning** algorithm.

Some terminology

If the model has a fixed number of parameters, it is categorised as **parametric**.

Otherwise, if the number of parameters grows with the amount of training data it is categorised as **non-parametric**. They do not make any strong assumption about the underlying model and so they are more flexible.

The philosophical problem

Deduction: derive specific consequences from general theories

Induction: derive general theories from specific observations

Deduction is well-founded (mathematical logic).

Induction is (philosophically) problematic – induction is useful since it often seems to work – an inductive argument !

Generalisation - the key objective of machine learning

What we are really interested in is generalising from the sample of data in our training set. This can be stated as:

The inductive learning hypothesis

> *Any hypothesis found to approximate the target (true) function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.*

A corollary of this is that it is necessary to make some assumptions about the type of target function in a task for an algorithm to go beyond the data, i.e., generalise or learn.

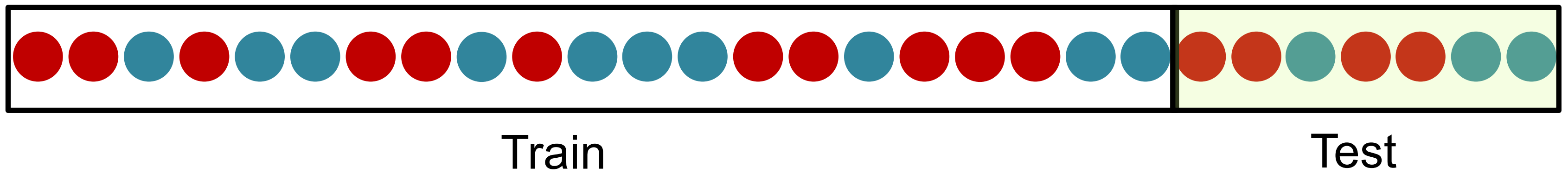
Cross-validation

There are certain parameters that need to be estimated during learning. We use the data, but NOT the training set, OR the test set. Instead, we use a separate *validation* or *development* set.

Cross-validation

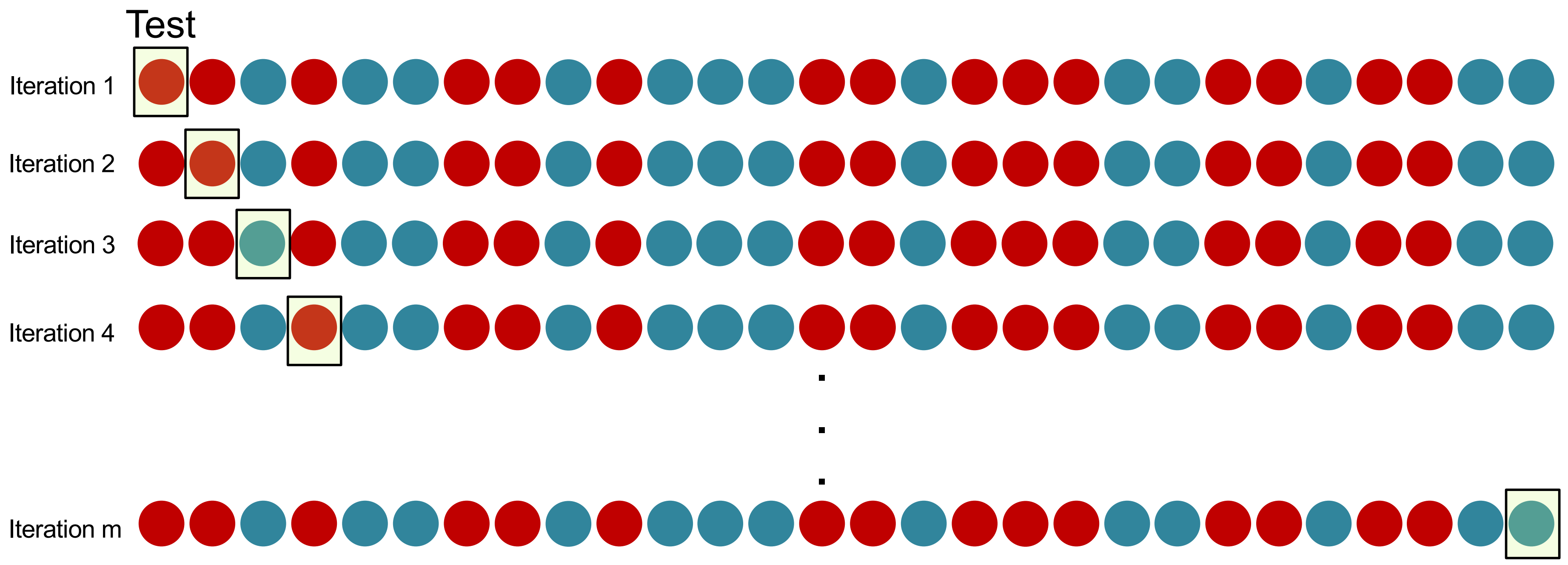
Also known as **out-of-sample testing** is validation technique to assess the results of a model to an independent data set

1. Holdout method:



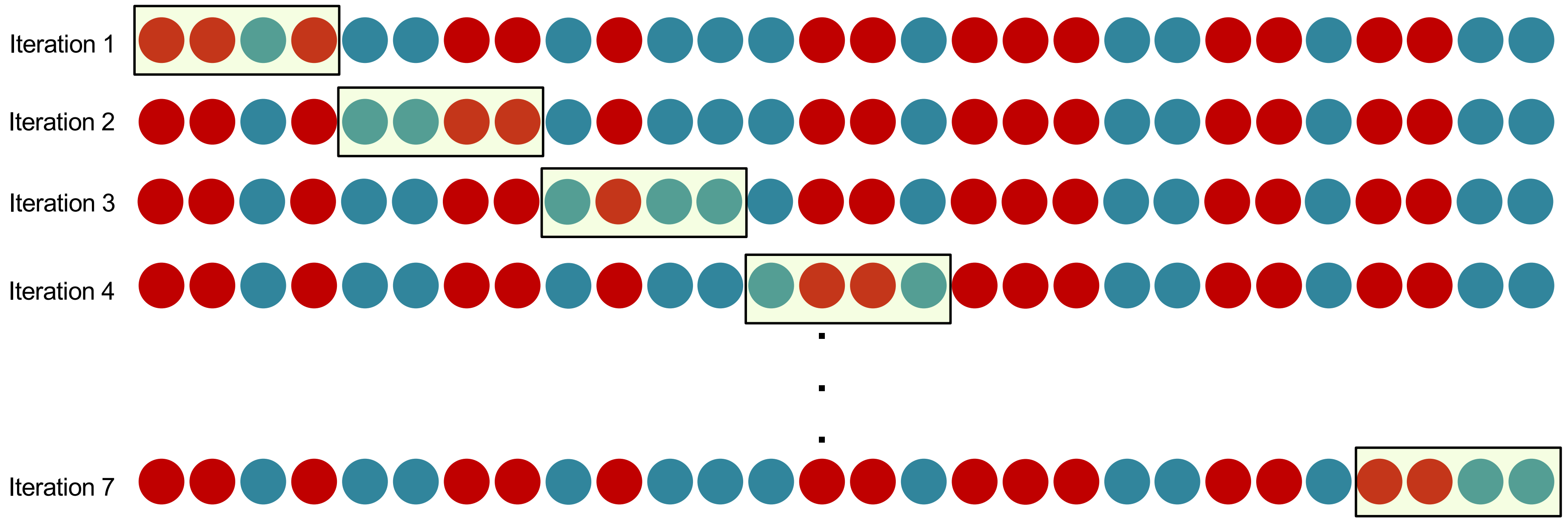
Cross-validation

2. Leave-One-Out Cross validation (LOOCV):



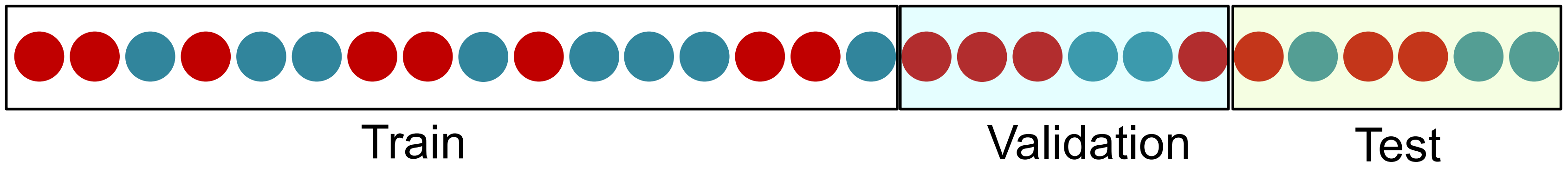
Cross-validation

3. K-fold Cross Validation



Cross-validation

Validation set: To make the hyperparameter tuning and model selection independent from the test set, we define another set within the train set



Data Types

In Machine Learning world, in general two types of data is defined:

- **Numerical**: Anything represented by numbers (e.g., integer , floating point)
- **Categorical**: everything that is not numerical (e.g. discrete labeled groups)

In general, for machine learning algorithms, data has to be represented in numeric form

Data Types

Another taxonomy of data types:

1. **Irrelevant**: it might be represented with strings or numbers but has no relationship with the outcome (e.g. participants name or code)
2. **Nominal**: discrete values with no numerical relationship between different categories (e.g. animal types, colors, nationality)
3. **Binary**: discrete data with only two possibilities (e.g cancerous vs. non-cancerous)

Data Types

4. **Ordinal**: discrete integers that can be ranked, but the relative distance between any two number can not be defined (e.g. students rank based on GPA)
5. **Count**: discrete whole numbers without any negatives
6. **Time**: a cyclical, repeating continuous form of data (e.g days, weeks)
7. **Interval**: data that the we can measure the distance between different values. (e.g temperature, income)

Binary Classification task

In a binary classification (or binomial classification) task, we always want to classify the data of a given set into two groups. We usually define one of the classes as positive and one as negative.

- Sometimes the classes are equally important (e.g. recognition of dog vs cat in image classification
- Sometimes misclassification in one of the classes is more costly than misclassification in the other class (e.g. predicting that someone has cancer while (s)he doesn't have vs predicting that someone doesn't have cancer while (s)he has) therefore we may prefer to have better classification in one class in the cost of more errors in the other class

Evaluation of error

If we have a binary classification, then we have two classes of $\mathrm{y} \in \{0,1\}$, where we call the class $y = 1$, *positive class* and $y = 0$, *negative class.*

Some evaluation metrics:

- **True positive:** number of instances from class one that have been predicted as one
- **True negative:** number of instances from class zero that have been predicted as zero
- **False positive:** number of instances from class zero that have been predicted as one
- **False negative:** number of instances from class one that have been predicted as zero

Contingency table

For two-class prediction case:

Actual Class	Predicted Class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

This is also called *confusion matrix*

Classification Accuracy

Classification Accuracy on a sample of labelled pairs $(X, c(X))$ given a learned classification model that predicts, for each instance X, a class value $\hat{c}(X)$:

$$acc = \frac{1}{|Test|} \sum_{x \in Test} I[\hat{c}(X) = c(X)]$$

where $Test$ is a test set and $I[]$ is the indicator function which is 1 iff its argument evaluates to true, and 0 otherwise.

$$Classification\ Error\ is\ =\ 1 - acc.$$

Other evaluation metrics

Precision/correctness

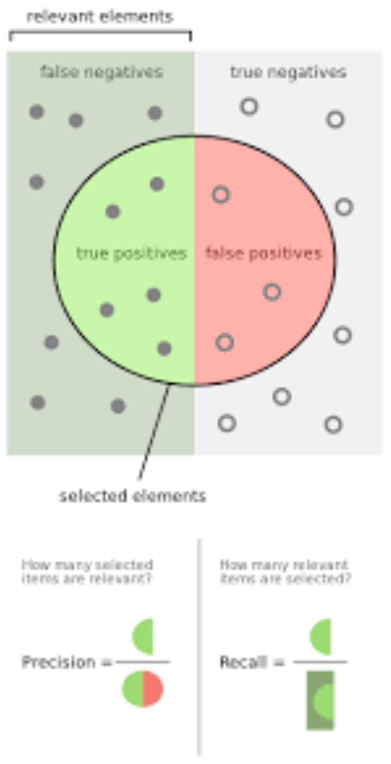
– is the number of relevant objects classified correctly divided by the total number of relevant objects classified

$$Precision = \frac{TP}{TP + FP}$$

Recall/sensitivity/completeness/true positive rate (TPR)

– is the number of relevant objects classified correctly divided by total number of relevant/correct objects

$$Recall = \frac{TP}{TP + FN}$$



Other evaluation metrics

F_1 score: a measure of accuracy, which is the harmonic mean of precision and recall and is defined as:

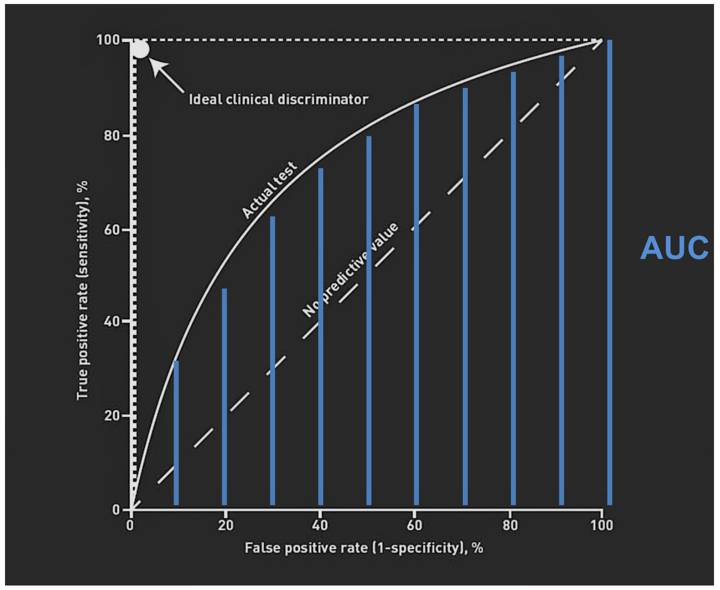
$$F_1 = 2.\frac{precision.recall}{precision + recall}$$

This measure gives equal importance to precision and recall which is sometime undesirable; so we have to decide which metric to use depending on the task and what's important for the task.

Other evaluation metrics

AUC-ROC curve: Area Under the Curve (AUC) – Receiver Operating Characteristics (ROC) curve is one of the most important evaluation metric for performance of classification models. This metric evaluates the model at different threshold settings and can inform us on the capability of the model in distinguishing between classes.

- $TPR = \frac{TP}{TP+FN}$
- $FPR = \frac{FP}{FP+TN}$
- A good model has AUC close to 1
- A very poor model has AUC close to 0
- $AUC = 0.5$ means no class separation



Missing Value: An issue to consider

Missing Values

- In practice it rarely happens that the data is complete and homogenous.
- Why data is incomplete:
    - Human errors
    - Sensor errors
    - Software bugs
    - Faulty preprocessing
    - …

Missing Values

How to handle missing values (common approaches):

- Deleting samples with missing values
- Replacing the missing value with some statistics from the data (mean, median, …)
- Assigning a unique category
- Predicting the missing values
- Using algorithms that support missing values

Missing Values

Deleting samples with missing values:

- Pros:
  - A robust and probably more accurate model
- Cons:
  - Loss of information and data
  - Works poorly if the percentage of missing values is high

Missing Values

Replacing the missing value with mean/median/mode:

- Pros:
  - When the data size is small, it is better than deleting
  - It can prevent data loss
- Cons:
  - Imputing the approximations add variance and bias
  - Works poorly compared to other methods

Missing Values

Assigning a unique category:

- Pros:
  - Less possibilities with one extra category, resulting in low variance after one hot encoding — since it is categorical
  - No loss of data
- Cons:
  - Adds less variance
  - Adds another feature to the model while encoding

Missing Values

Predicting the missing values:

- Pros:
  - Imputing the missing variable is an improvement as long as the bias from it is smaller than the omitted variable bias
  - Yields unbiased estimates of the model parameters
- Cons:
  - Bias also arises when an incomplete conditioning set is used for a categorical variable
  - Considered only as a proxy for the true values

Missing Values

Using algorithms that support missing values:

- Pros:
  - Does not require creation of a predictive model
  - Correlation of the data is neglected
- Cons:
  - Is a very time-consuming process and it can be critical in data mining where large databases are being extracted