CBU5201:Machine Learning Revision Note

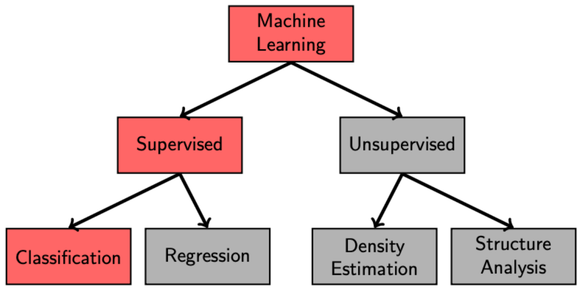
Chenyang Li
International School, BUPT

February 27, 2025

This note is adapted from the CBU5201 lecture slides. This note can help you review the big picture of the module, but it doesn't cover all the points, especially the very detailed ones.

1 Introduction

1.1 Taxonomy



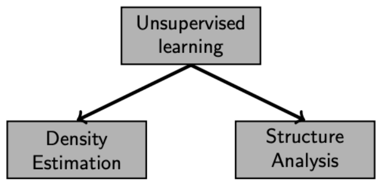
1.2 Methods

In **supervised learning**, we designate one attribute as a label and treat the rest as predictors. We set out to build a model that estimates the value of the label based on the value of the predictors. **Unsupervised learning** does not elevate any attribute to the category oflabel: all the attributes are treated equally.The essence of unsupervised learning is encapsulated in the simple question where is my data?

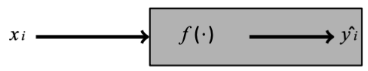
2 Regression

Regression is a supervised problem: Our goal is to predict the value of one attribute (label) using the remaining attributes (predictors). The label is a continuous variable.

- **Density estimation**: Creates models that allow us to quantify the probability of finding a sample within a region of the attribute space (**probability density**).
- **Structure analysis**: Creates models that identify regions within the attribute space (**cluster analysis**) or directions (**component analysis**) with a high density of samples.



2.1 Mathematical notation



Population:
- x is the **predictor** attribute
- y is the **label** attribute

Dataset:
- N is the number of samples, i identifies each sample
- x_i is the predictor of sample i
- y_i is the actual label of sample i
- (x_i, y_i) is sample i, $\{(x_i, y_i) : 1 \le i \le N\}$ is the entire dataset

Model:
- $f(\cdot)$ denotes the model
- $\hat{y_i} = f(x_i)$ is the **predicted label** for sample i
- $y_i - \hat{y_i}$ is the **prediction error** for sample i

In simple linear regression, models are defined by the mathematical expression:

$$f(x) = w_0 + w_1 x \tag{1}$$

Hence, the predicted label $\hat{yi}$ can be expressed as

$$f(x_i) = w_0 + w_1 x_i$$

2.2 Error

You **MUST** know how to calculate a given type of error.Basically, three types: Mean squared error, sum of squared error and mean abstract error.

$$e_i = y_i - \hat{y}_i \tag{2}$$

$$SSE = \sum_{i=1}^{n} e_i^2$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} e_i^2$$

$$MAE = \sum_{i=1}^{n} |e_i|$$

In addition to the MSE, we can consider other quality metrics:

- **Root mean squared error**. Measures the sample standard deviation of the prediction error.

$$E_{RMSE} = \sqrt{\frac{1}{N}\sum e_i^2}$$

- **Mean absolute error**. Measures the average of the absolute prediction error.

$$E_{MAE} = \frac{1}{N}\sum |e_i|$$

- **R-squared**. Measures the proportion of the variance in the response that is predictable from the predictors.

$$E_R = 1 - \frac{\sum e_i^2}{\sum (y_i - \bar{y})^2}, \text{ where } \bar{y} = \frac{1}{N}\sum y_i$$

EXERCISE ♯1. Consider the following simple dataset, where ID denotes the sample identifier and x and y are two attributes:

ID	x	y
1	2	1
2	3	2
3	1	2
4	1	1
5	0	1
6	5	3
7	4	3
8	6	7
9	5	6
10	3	5

In this exercise we will use this dataset to test the following five linear models:

- $f_1(x) = 2x + 1$

2.3 Vector Notation

You **MUST** know how to write the vector notation or mathematical expression of a given problem.

Using vector notation, predictors can be packed together into a vector:

$$\boldsymbol{x}_i = [1, x_{i,1}, x_{i,2}, \dots, x_{i,K}]^T,$$

where $x_{i,k}$ denotes the k-th predictor of the i-th sample and K is the number of predictors. The constant 1 is prepended for convenience.

Using vector notation, multiple regression can then be expressed as

$$\hat{y_i} = f(\boldsymbol{x}_i)$$

In multiple linear regression, the **training dataset** can be represented by the **design matrix X**:

$$\mathbf{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^T = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,K} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \dots & x_{N,K} \end{bmatrix}$$

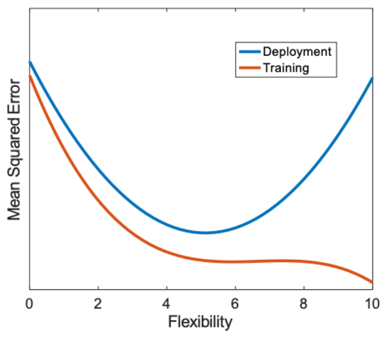
and the **label vector *y***:

$$\boldsymbol{y} = [y_1, \dots, y_N]^T = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

2.4 Flexibility and Generalization

More complex model leads to more flexibility, less complex model tends to be a rigid model. Generalisation is the ability of our model to successfully translate what we was learnt during the learning stage to deployment. **A model generalises well when it can deal successfully with samples that it hasn't been exposed to during training.**

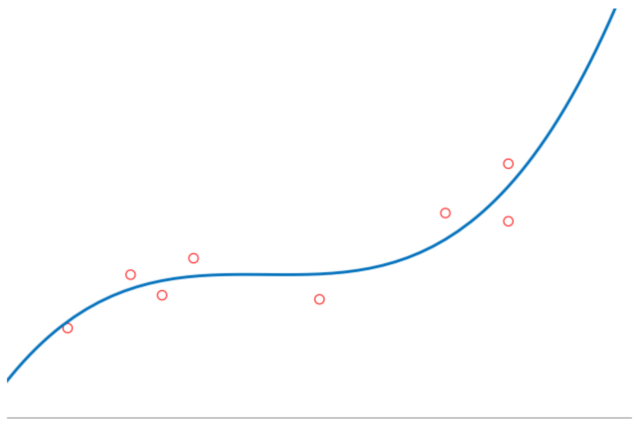
In this figure, the red curve represents the **training MSE** of different models of increasing complexity, whereas the blue curve represents the **deployment MSE** for the same models. What's happening?



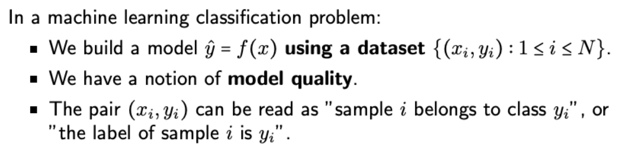
2.5 Overfitting and Underfitting

You **MUST** know identify whether it is underftting or Overfitting of a given model and explain the definition of these concepts.

- Underfitting: Large training and deployment errors are produced.The model is unable to capture the underlying pattern.Rigid models lead to underfitting.
- Overfitting: Small training errors, large errors during deployment. The model is memorising irrelevant details. Too complex models and not enough data lead to overfitting.
- Just right: Low training and deployment errors. The model is capable of reproducing the underlying pattern and ignores irrelevant details.



3 Classification



In a machine learning classification problem:

- We build a model $\hat{y} = f(x)$ **using a dataset** $\{(x_i, y_i) : 1 \le i \le N\}$.
- We have a notion of **model quality**.
- The pair (x_i, y_i) can be read as "sample i belongs to class y_i", or "the label of sample i is y_i".

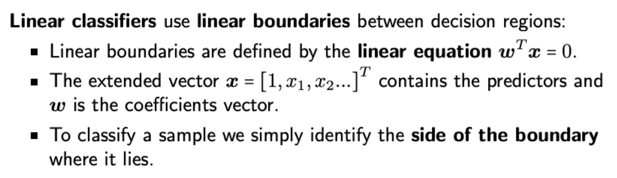
3.1 Linear Classification

The simplest boundary is:

- A single point (known as threshold) in 1D predictor spaces.
- A straight line in 2D predictor spaces.
- A plane surface in 3D predictor spaces.

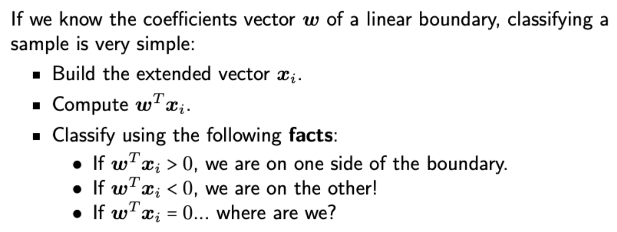
3.1.1 Linear Classifier

You **MUST** know how to classify a given sample according to a given classifier.



Linear classifiers use **linear boundaries** between decision regions:

- Linear boundaries are defined by the **linear equation** $\boldsymbol{w}^T\boldsymbol{x} = 0$.
- The extended vector $\boldsymbol{x} = [1, x_1, x_2 ...]^T$ contains the predictors and $\boldsymbol{w}$ is the coefficients vector.
- To classify a sample we simply identify the **side of the boundary** where it lies.



If we know the coefficients vector $\boldsymbol{w}$ of a linear boundary, classifying a sample is very simple:

- Build the extended vector $\boldsymbol{x}_i$.
- Compute $\boldsymbol{w}^T\boldsymbol{x}_i$.
- Classify using the following **facts**:
  - If $\boldsymbol{w}^T\boldsymbol{x}_i > 0$, we are on one side of the boundary.
  - If $\boldsymbol{w}^T\boldsymbol{x}_i < 0$, we are on the other!
  - If $\boldsymbol{w}^T\boldsymbol{x}_i = 0$... where are we?

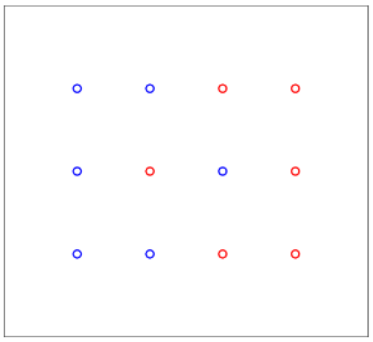
3.1.2 Quality Metrics

Accuracy and the error (or misclassification) rate E.

$$A = \frac{\#\text{correctly classif. samples}}{\#\text{samples}} \quad , \quad E = \frac{\#\text{incorrectly classif. samples}}{\#\text{samples}}$$

3.2 Nearest neighbours

In nearest neighbours (NN), new samples are assigned the label of the closest (most similar) training sample. It does not involve training.As K increases, the boundary becomes less complex. We move away from overfitting (small K) to underfitting (large K) classifiers.I n binary problems, the value of K is usually an odd number. The idea is to prevent situations where half of the nearest neighbours of a sample belong to each class.

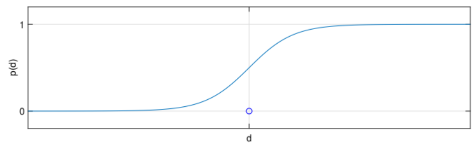


3.3 Logistic model

The logistic model

The logistic function $p(d)$ is defined as

$$p(d) = \frac{e^d}{1+e^d} = \frac{1}{1+e^{-d}}$$



Note that

- $p(0) = 0.5$.
- As $d \to \infty$, $p(d) \to 1$.
- As $d \to -\infty$, $p(d) \to 0$.

Given a linear boundary $\boldsymbol{w}$ and a predictor vector $\boldsymbol{x}_i$, the quantity $\boldsymbol{w}^T\boldsymbol{x}_i$ can be interpreted as the **distance** from the sample to the boundary.

If we set $d = \boldsymbol{w}^T\boldsymbol{x}_i$ in the logistic function, we get:

$$p(\boldsymbol{w}^T\boldsymbol{x}_i) = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}_i}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}$$

For a fixed $\boldsymbol{w}$, we will simply denote it as $p(\boldsymbol{x}_i)$ to simplify the notation:

- *When $\boldsymbol{w}^T\boldsymbol{x} \to \infty$, the logistic function $p(\boldsymbol{x}_i) \to 1$*
- *When $\boldsymbol{w}^T\boldsymbol{x} \to -\infty$, the logistic function $p(\boldsymbol{x}_i) \to 0$*

We will use the logistic function to quantify the notion of **certainty** in classifiers. This certainty is a quantity between 0 and 1.

The logistic model

Consider a linear classifier $\boldsymbol{w}$ that labels samples such that $\boldsymbol{w}^T\boldsymbol{x}_i > 0$ as o and samples such that $\boldsymbol{w}^T\boldsymbol{x}_i < 0$ as o.

Notice that:

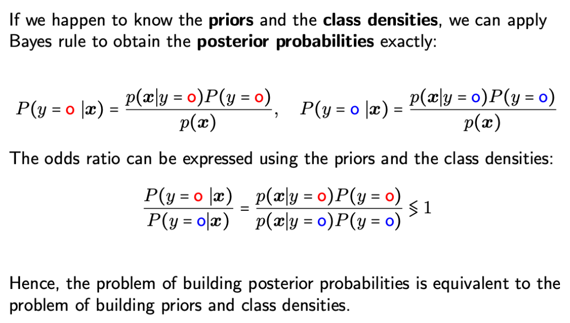
- If $\boldsymbol{w}^T\boldsymbol{x}_i = 0$ ($\boldsymbol{x}_i$ is on the boundary), $p(\boldsymbol{x}_i) = 0.5$.
- If $\boldsymbol{w}^T\boldsymbol{x}_i > 0$ ($\boldsymbol{x}_i$ is in the o region), $p(\boldsymbol{x}_i) \to 1$ as we move away from the boundary .
- If $\boldsymbol{w}^T\boldsymbol{x}_i < 0$ ($\boldsymbol{x}_i$ is in the o region), $p(\boldsymbol{x}_i) \to 0$ as we move away from the boundary.

Here is the crucial point, so use **all your neurons**:

- $p(\boldsymbol{x}_i)$ is the **classifier's certainty** that $y_i =$ o true.
- $1 - p(\boldsymbol{x}_i)$ is the **classifier's certainty** that $y_i =$ o true.

3.4 Bayesian Classifier

3.4.1 Bayes rule



If we happen to know the **priors** and the **class densities**, we can apply Bayes rule to obtain the **posterior probabilities** exactly:

$$P(y = o\,|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = o)P(y = o)}{p(\boldsymbol{x})}, \quad P(y = o\,|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = o)P(y = o)}{p(\boldsymbol{x})}$$

The odds ratio can be expressed using the priors and the class densities:

$$\frac{P(y = o\,|\boldsymbol{x})}{P(y = o|\boldsymbol{x})} = \frac{p(\boldsymbol{x}|y = o)P(y = o)}{p(\boldsymbol{x}|y = o)P(y = o)} \lessgtr 1$$

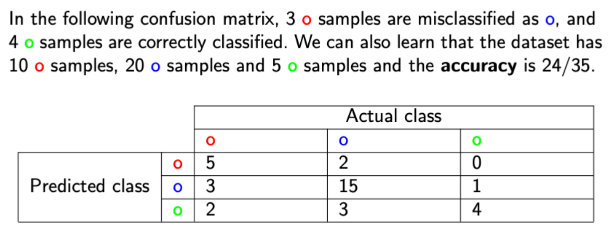
Hence, the problem of building posterior probabilities is equivalent to the problem of building priors and class densities.

Using a dataset to estimate the priors is very easy, we simply need to count the number of samples belonging to each class.

However, for the class densities $p(x|y = o)$ and $p(x|y = o)$, we often use Gaussian distribution to estimate it.

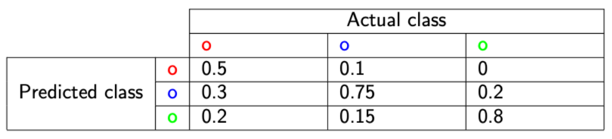
3.5 Confusion Matrix

NOTE: You **MUST** understand each part of the Confusion Matrix, because you will **definitely be asked** to **draw the confusion matrix** and **explain some relevant concepts** during the exam. In detection problems, the most commonly rates used are:



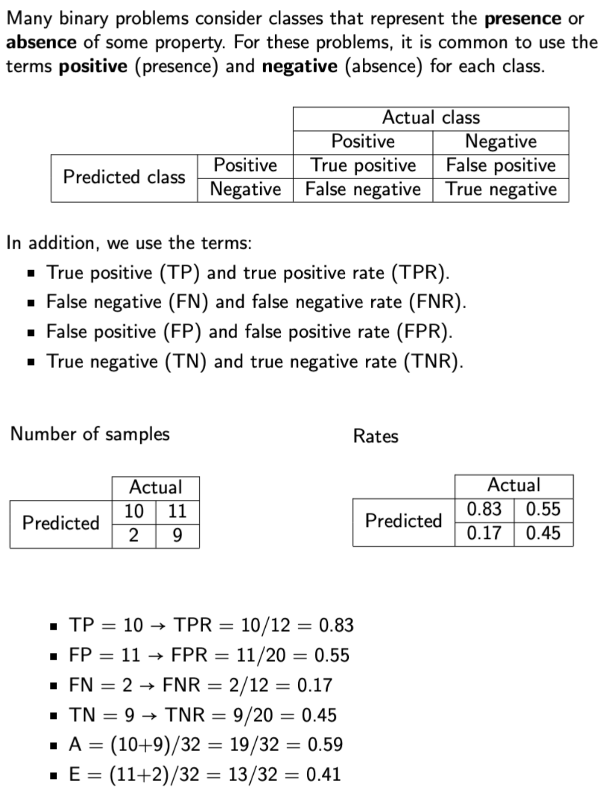
In the following confusion matrix, 3 o samples are misclassified as o, and 4 o samples are correctly classified. We can also learn that the dataset has 10 o samples, 20 o samples and 5 o samples and the **accuracy** is 24/35.

		Actual class		
		o	o	o
Predicted class	o	5	2	0
	o	3	15	1
	o	2	3	4



		Actual class		
		o	o	o
Predicted class	o	0.5	0.1	0
	o	0.3	0.75	0.2
	o	0.2	0.15	0.8

- Sensitivity (recall or true positive rate): TP/(TP+FN)
- Specificity (true negative rate): TN/(TN+FP)
- Precision (positive predictive value): TP/(TP+FP)



Many binary problems consider classes that represent the **presence** or **absence** of some property. For these problems, it is common to use the terms **positive** (presence) and **negative** (absence) for each class.

		Actual class	
		Positive	Negative
Predicted class	Positive	True positive	False positive
	Negative	False negative	True negative

In addition, we use the terms:

- True positive (TP) and true positive rate (TPR).
- False negative (FN) and false negative rate (FNR).
- False positive (FP) and false positive rate (FPR).
- True negative (TN) and true negative rate (TNR).

Number of samples

	Actual	
Predicted	10	11
	2	9

Rates

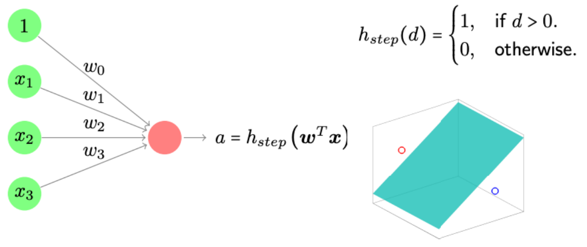
	Actual	
Predicted	0.83	0.55
	0.17	0.45

- TP = 10 → TPR = 10/12 = 0.83
- FP = 11 → FPR = 11/20 = 0.55
- FN = 2 → FNR = 2/12 = 0.17
- TN = 9 → TNR = 9/20 = 0.45
- A = (10+9)/32 = 19/32 = 0.59
- E = (11+2)/32 = 13/32 = 0.41

4 Neural Networks and Deep Learning

A neural network is a computing system loosely inspired by the human nervous system, commonly used as a family of Machine Learning models.

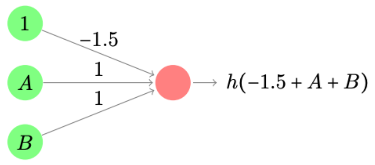
4.1 Perceptron


1
x_1
x_2
x_3
w_0
w_1
w_2
w_3
$a = h_{step}\left(\boldsymbol{w}^T\boldsymbol{x}\right)$
$h_{step}(d) = \begin{cases} 1, & \text{if } d > 0. \\ 0, & \text{otherwise.} \end{cases}$

A single perceptron can act as a basic logical function, such as **AND function, OR function**, but it **can not** perform XOR operation. Multiple-layer-perceptron can implement XOR function.

4.2 Convolutional Neural Network

You **MUST** know what CNN and pooling layer do and how they work.



A	B	$f(A,B)$
0	0	0
0	1	0
1	0	0
1	1	1

In short, CNN **filters** the input with kernel.

To calculate the size of the output feature map after applying a convolutional kernel (filter) to an input image or feature map, you can use the following formula:

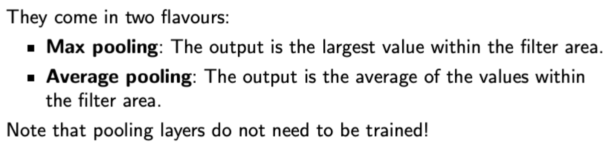
$$\text{Output Size} = \left\lfloor \frac{\text{Input Size} - \text{Kernel Size} + 2 \times \text{Padding}}{\text{Stride}} \right\rfloor + 1$$

Where:

- **Input Size**: The height or width of the input feature map .
- **Kernel Size**: The height or width of the convolutional kernel (filter).
- **Padding**: The number of pixels added to the border of the input feature map (typically 0 or 1).
- **Stride**: The step size with which the kernel moves across the input feature map.
- $\lfloor \cdot \rfloor$: The floor function, which rounds down to the nearest integer.

4.2.1 Pooling layer

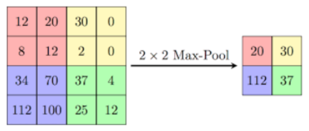
Pooling **reduces the size** of feature maps.



They come in two flavours:

- **Max pooling**: The output is the largest value within the filter area.
- **Average pooling**: The output is the average of the values within the filter area.

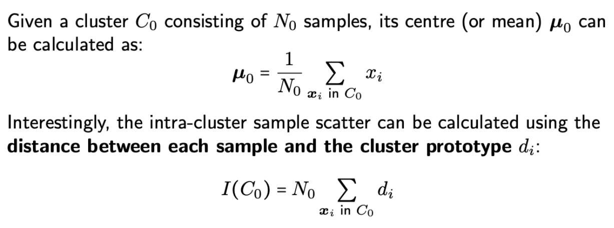
Note that pooling layers do not need to be trained!



5 Structure Analysis

You **NEED** to know how to explain structure analysis in your own word.

Structure analysis creates models that identify regions within the attribute space (cluster analysis) or directions (component analysis) with a high density of samples.



Given a cluster C_0 consisting of N_0 samples, its centre (or mean) $\boldsymbol{\mu}_0$ can be calculated as:

$$\boldsymbol{\mu}_0 = \frac{1}{N_0} \sum_{\boldsymbol{x}_i \text{ in } C_0} x_i$$

Interestingly, the intra-cluster sample scatter can be calculated using the **distance between each sample and the cluster prototype** d_i:

$$I(C_0) = N_0 \sum_{\boldsymbol{x}_i \text{ in } C_0} d_i$$

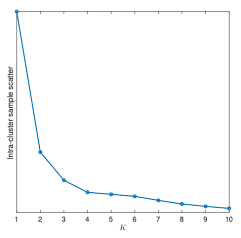
5.1 K-means clustering

You MUST know how K-means clustering works and how to implement it. The final solution of k-means clustering is a local optimum, not necessarily the global one. It depends on the initial choice of center.

The elbow method

Assume the true number of clusters is K_T. For $K > K_T$, we should expect the increase in quality to be slower than for $K < K_T$, as we will be splitting true clusters.

The true number of clusters can be identified by observing the value of K beyond which the improvement slows down.



6 Density Estimation

Density estimation: Creates models that allow us to quantify the probability of finding a sample within a region of the attribute space(probability density).

7 Methodology

In machine learning we can identify the following tasks:

- Test: This is the most important task. It allows us to estimate the deployment performance of a model.
- Training: Used to find the best values for the parameters of a model, i.e. to tune a model.
- Validation: Necessary to compare different modelling options and select the best one, the one that will be trained.

7.1 Comparing Models

We use a subset of data, the **test dataset**, to compute the test deployment performance as an estimation of the true performance.

If you encounter a question asking you to choose a good model, you need to focus on the test/validation performance instead of training performance. That means, you may need to calculate the errors on the test/validation set, don't use training error for comparison.

7.2 Gradient Descent

Gradient descent is a numerical optimisation method where we update teratively our model using the gradient of the error surface.

The gradient provides the direction along which the error increases the most. Using the gradient, we can create the following update rule:

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w}_{\text{old}} - \epsilon \nabla E(\boldsymbol{w}_{old})$$

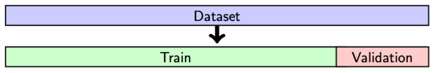
where ϵ is known as the **learning rate** or **step size**.

7.3 Validation

Validation methods allow us to use data for assessing and selecting different families of models. The same data used for validation can then be used to train a final model.

Validation set approach

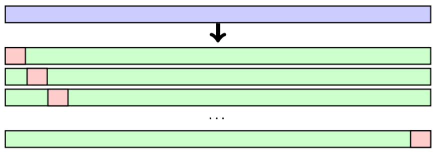
The validation set approach is the simplest method. It randomly splits the available dataset into a **training** and a **validation** (or hold-out) dataset.



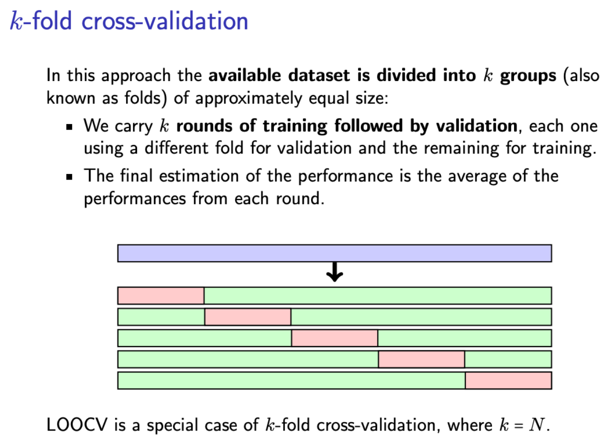
Models are then fitted with the training part and the validation part is used to estimate its performance.

Leave-one-out cross-validation (LOOCV)

This method also splits the available dataset into training and validation sets. However, the **validation set contains only one sample**.



Multiple splits are considered and the final performance is calculated as the average of the individual performances (for N samples, we produce N splits and obtain N different performances).



k-fold cross-validation

In this approach the **available dataset is divided into k groups** (also known as folds) of approximately equal size:

- We carry k **rounds of training followed by validation**, each one using a different fold for validation and the remaining for training.
- The final estimation of the performance is the average of the performances from each round.

LOOCV is a special case of k-fold cross-validation, where $k = N$.

7.3.1 Comparison

- The validation set approach involves one training round. Models are however trained with fewer samples and the final performance is highly variable due to random splitting.
- LOOCV requires as many training rounds as samples there are in the dataset, however in every round almost all the samples are usedfor training. It always provides the same performance estimation.
- k-fold is the most popular approach. It involves fewer training rounds than LOOVC. Compared to the validation set approach, the performance estimation is less variable and more samples are used for training.