Machine Learning and Optimisation

COMP24111 Course Notes

Jonathan Tang

Contents

II	Section II

Section I

1. Introduction to Machine Learning

1.1 Supervised learning

In **supervised learning** problems, there is an input, X, an target output, Y, provided by a *teacher* and the task is to learn the relationship between the input and the output. A training example in supervised learning is the pair (x, y) where x is the input and y is the target output. The labels provided with the training samples are know as the **ground truth**. We assume a model defined up to a set of parameters, $y = g(x \mid \theta)$ where $g(\cdot)$ is the model and θ are its parameters.

1.1.1 Classification

For a classification problem, the task of the classifier is to assign a class label to a given input. For example, a classification problem may be to decide whether an image contains a picture of a cat or a dog, or neither.

1.1.2 Regression

Suppose we want to predict the salary of a Computer Science graduate role. Inputs are the students attributes - that we believe affects a graduate's worth. The output is the predicted salary. Such problems where the output is a continuous number is known as a **regression** problem.

1.2 Unsupervised learning

Unlike in supervised learning, we do not have a supervisor and we only have input data. The task in **unsupervised learning** is to form a natural understanding of the hidden structure of unlabelled data. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not.

1.3 Reinforcement learning

In reinforcement learning, there is a *teacher* who provides feedback on the action of an agent, in terms of reward and punishment.

2. k-nearest Neighbour

The **k-nearest Neighbour** estimation is one of the simplest machine learning algorithms, it is a **non-parametric method** (no parameter needs to be optimised) and requires **no explicit training**.

2.1 k-nearest Neighbour Classification

The k-nearest neighbour classifier assigns an instance to the class most heavily represented among its k neighbours. It is based on the idea that the more similar the instances, the more likely it is that they belong to the same class. We measure how similar two data points are by using a reasonable similarity or distance measure.

2.1.1 k-NN classification rule

```
let  x_te  be  the  testing  point
for  each  training  data  point  x_tr
      measure  distance ( x_te ,  x_tr )
end
sort  distances
select  k  nearest  points
assign  most  common  class  to  x_te
```

In order to measure the distance $d(x_{te}, x_{tr})$ mentioned in the algorithm, we need to first decide on a distance measure. Euclidean distance and inner product are two measures we can used, that are defined below. An additional distance measure that we can use is cosine similarity, which is described in a later chapter.

> **Exercise 2.1** The hyper-parameter k should be odd and co-prime to the number of classes in the data set, why? ∎

> **Definition 2.1.1 — Euclidean Distance.** Given two d-dimensional data points, $\mathbf{p}$ and $\mathbf{q}$, where $\mathbf{p} = \begin{bmatrix} p_1, p_2, \cdots, p_d \end{bmatrix}$ and $\mathbf{q} = \begin{bmatrix} q_1, q_2, \cdots, q_d \end{bmatrix}$, we define the **Euclidean distance** $d(\mathbf{p}, \mathbf{q})$ as:
>
> $$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_d - q_d)^2} = \sqrt{\sum_{i=1}^{d}(p_i - q_i)^2}$$
>
> In this case, the k-nearest neighbours are the training points with the lowest Euclidean distances to the testing point.

> **Definition 2.1.2 — Inner Product.** Given two d-dimensional data points, $\mathbf{p}$ and $\mathbf{q}$, where $\mathbf{p} = \begin{bmatrix} p_1, p_2, \cdots, p_d \end{bmatrix}$ and $\mathbf{q} = \begin{bmatrix} q_1, q_2, \cdots, q_d \end{bmatrix}$, we define the **inner product** as:
>
> $$s_{inner}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{d} p_i q_i$$
>
> Since the inner product is a similarity measure, the k-nearest neighbours are the training points possessing the highest similarity values to the testing point.

2.2   k-Nearest Neighbour Regression

For regression, we take a sample's k nearest neighbours to estimate its target output, for example, through the weighted average of these neighbours target output. This is very similar to classification but you need to take into account the distance of this sample from its nearest neighbours.

Worked Example

Since the k-NN classification algorithm is one of the simplest, you should be able to follow it from the lecture slides - hence a worked example has been omitted. It is also trivial to derive the algorithm for k-NN regression.

2.3   Neighbour number k

The neighbour number k is called the **hyper-parameter** and the processing of determining the best value of k to use is called **hyper-parameter selection** or model selection. If our choice of k is too small, we may model noise and if our choice of k is too large, neighbours will include too many samples from other classes.

The number of training samples also plays a role, a small number of training samples will lead to insufficient information about the data set. However, a large number of training samples comes with a tradeoff in time and memory cost for calculating and sorting distances.

3. Linear Classification and Regression

4. Logistic Regression

5. Support Vector Machine

5.1 Model Validation

5.1.1 Holdout method

With the **holdout method**, the whole dataset is split into two groups, a training set and a testing set. The model is trained using the training set and asked to predict output values for the data in the testing set, which it has never seen before. This evaluation can have a high variance as we may get an unfortunate split as it may depend heavily on which data points end up in the training set and which end up in the testing set. Another drawback is that we may only have a small dataset, so we may not afford to set aside a portion of the dataset for testing.

5.1.2 Random subsampling

In each split of **random subsampling**, we select a fixed number of samples for testing and the remaining samples are used for training. For each data split, the classifier is training from scratch and its error rate is estimated with the testing examples.

5.1.3 k-fold cross validation

k-fold cross validation is one way to improve over the holdout method, the dataset is divided into k partitions and the holdout method is performed k times. Each time, one of the k subsets is used as the testing set and the remaining $k-1$ subsets are combined and used as the training set. We evaluate the error estimate as the average error across the k trials. The advantage of this method is that it matters less how the dataset gets divided and all the examples in the dataset are eventually used for both training and testing, therefore the variance of the resulting estimate is reduced as k is increased. However, the training has to be done k times, which means this method gets expensive to compute as k is increased.

5.1.4 Leave-one-out cross validation

Leave-one-out cross validation is the degenerate case for k-fold cross validation, with $k = N$, the number of data points in the dataset. As before, the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross validation error is good, but it is very expensive to compute compared to other methods.

5.1.5 Bootstrap

With **bootstrap**, on each iteration, you randomly select, with replacement, a fixed number of examples to use for training. The remaining examples that were not selected are used for testing - this means that the number of testing samples can change over repeats, as training samples are selected with replacement.

Model Selection (hyper-parameter selection)

To select the best model, we run the algorithm with different hyper-parameters, which yields different models and use one of the above validation schemes to calculate an error estimate. It follows that the best model is the one that yields the lowest error estimate.

5.2 Confusion matrix

A confusion matrix is a table with two rows and two columns which allows further analysis rather than just the proportion of correct classifications. For example, if we had a dataset containing 98 examples from Class 1 and 2 examples from Class 2, there may be a classifier which classifies all the observations as Class 1, which yields a overall accuracy of 98%. However, accuracy is not a reliable metric for this classifier as the classifier would have 100% recognition rate for Class 1 but 0% recognition rate for Class 2.

> **Definition 5.2.1 — True Positives (TP).** Both the predicted class and actual class is 'Yes'.

> **Definition 5.2.2 — True Negatives (TN).** Both the predicted class and actual class is 'No'.

> **Definition 5.2.3 — False Positives (FP).** The predicted class is 'Yes' and actual class is 'No'.

> **Definition 5.2.4 — False Negatives (FN).** The predicted class is 'No' and actual class is 'Yes'.

Worked Example

Suppose that we predicted the presence of a disease that a patient and obtain the results below.

n = 165		Predicted Class	
		No	Yes
Actual	No	50	10
Class	Yes	5	100

■ **Example 5.1** For the above example, **TP** = 100, **TN** = 50, **FP** = 10 and **FN** = 5.   ■

Some questions we may want to ask about our predictions are:

1. When the patient actually has the disease, how often did we predict this?
2. When the patient does not have the disease, how often did we predict this?
3. When we predict that a patient has a disease, how often are we correct?

First, we define some terms which directly relate to the questions above.

Definition 5.2.5 — Sensitivity / Recall. The proportion of positives that are correctly classified.

$$\frac{\text{True Positives}}{\text{Number of Actual Positives}} = \frac{\text{TP}}{\text{TP+FN}}$$

Definition 5.2.6 — Specificity. The proportion of negatives that are correctly classified.

$$\frac{\text{True Negatives}}{\text{Number of Actual Negatives}} = \frac{\text{TN}}{\text{TN+FP}}$$

Definition 5.2.7 — Precision. The proportion of predicted positives which were positive.

$$\frac{\text{True Positives}}{\text{Number of Predicted Positives}} = \frac{\text{TP}}{\text{TP+FP}}$$

■ **Example 5.2** By definition, the answers to the above questions are sensitivity, specificity and precision respectively. ■

Exercise 5.1 Calculate the sensitivity, specificity and precision for the confusion matrix given above. ■

Definition 5.2.8 — F1-score. The harmonic mean of precision and recall.

$$2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

6. Deep Learning Models

Section II

7. Generative Models and Naive Bayes

8. Clustering Analysis Basics

Introduction

Clustering analysis is the process of finding similarities between data according to the character-istics underlying he data and grouping similar data objects into clusters. A **cluster** is a group of data points which are similar to one another within the same group and dissimilar to the points in other groups. Clustering analysis is **unsupervised** as there are no predefined classes for a training data set. The main approach of clustering analysis is to maximise the intra-cluster similarity and minimise the inter-cluster similarity. Typically, clustering analysis is either used as a stand-alone tool to gain an insight into data distribution or used as a preprocessing step of other algorithms in intelligent systems.

The two general tasks of clustering analysis are:
1. Identifying the "natural" number of clusters present in a data set
2. Properly grouping data points into "sensible" clusters

Real Life Applications

- **Marketing** - companies can discover distinct groups in their customer bases and then use this knowledge to develop targeted marketing programs
- **Social network mining** - discovering communities of similar interests in a large group of people
- **Image segmentation** - dividing an image into distinct regions for object recognition

8.1 Data Representation

A data matrix is a $n \times p$ matrix which represents n data points with p dimensions. The data matrix has *two* modes as rows and columns represent different entities.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

A distance / dissimilarity matrix is a $n \times n$ matrix which represents the distance between each data point. The distance matrix is a symmetric triangular matrix as $\mathbf{d}(x,y) = \mathbf{d}(y,x)$ where x,y are two data points. The distance matrix has *one* mode as the rows and column represent the distance for the same entity.

$$
\begin{bmatrix}
0 & & & & \\
d(x_2,x_1) & 0 & & & \\
d(x_3,x_1) & d(x_3,x_2) & 0 & & \\
\vdots & \vdots & \vdots & \ddots & \\
d(x_n,x_1) & d(x_n,x_2) & d(x_n,x_3) & \cdots & 0
\end{bmatrix}
$$

8.2 Distance Measures

The **Minkowski distance** is a metric in a normed vector space which can be considered as the generalisation of both the Euclidean distance and the Manhattan distance.

> **Definition 8.2.1 — Minkowski Distance.** The Minkowski distance of order p between two points $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ is defined as:
>
> $$d(\mathbf{p}, \mathbf{q}) = \left(|x_1 - y_1|^p + |x_2 - y_2|^p + \cdots + |x_n - y_n|^p \right)^{\frac{1}{p}}, p > 0$$

The value of p in the Minkowski Distance would be selected depending on the application. The **Manhattan** and **Euclidean** distances are simply Minkowski distances of order 1 and 2 respectively.

> **Definition 8.2.2 — Manhattan Distance.**
>
> $$d(\mathbf{p}, \mathbf{q}) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$
>
> This is also known as the city block distance, the reason becomes clear in the examples below.

> **Definition 8.2.3 — Euclidean Distance.**
>
> $$d(\mathbf{p}, \mathbf{q}) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \cdots + |x_n - y_n|^2}$$
>
> This is the distance that you are already familiar with, in n dimensions.

> **Exercise 8.1** When is the Manhattan distance between x and y equal to the Euclidean distance between x and y, where x and y are data points of two dimensions? ∎

> **Definition 8.2.4 — Cosine measure.** For $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)$
>
> $$\cos(\mathbf{x}, \mathbf{y}) = \frac{x_1 y_1 + \cdots + x_n y_n}{\sqrt{x_1^2 + \cdots + x_n^2} \sqrt{y_1^2 + \cdots + y_n^2}} \quad \text{and} \quad d(\mathbf{x}, \mathbf{y}) = 1 - \cos(\mathbf{x}, \mathbf{y})$$
>
> Note that $-1 \leq \cos(\mathbf{x}, \mathbf{y}) \leq 1$ and $0 \leq d(\mathbf{x}, \mathbf{y}) \leq 2$

Worked Example

Suppose we have some documents that we want to compare to see how similar texts are, without taking into account the order of the words. For this example, we will consider:

(a) Labeeba loves me more than Hani loves me
(b) Hani loves Labeeba more than Labeeba loves Hani
(c) Labeeba likes Hani more than Hani likes Labeeba

First, we make a list of all the words that appear across all texts.

$$[\text{Hani, Labeeba, likes, loves, me, more, than}]$$

Next, we construct a vector for each document, with the frequency of each word.

$$\mathbf{a} = [1, 1, 0, 2, 2, 1, 1]$$
$$\mathbf{b} = [2, 2, 0, 2, 0, 1, 1]$$
$$\mathbf{c} = [2, 2, 2, 0, 0, 1, 1]$$

Now we can compute the **cosine measures**: $\cos(\mathbf{a}, \mathbf{b})$, $\cos(\mathbf{a}, \mathbf{c})$ and $\cos(\mathbf{b}, \mathbf{c})$.

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{(1)(1) + (1)(2) + (0)(0) + (2)(2) + (2)(0) + (1)(1) + (1)(1)}{\sqrt{1^2 + 1^2 + 0^2 + 2^2 + 2^2 + 1^2 + 1^2} \sqrt{2^2 + 2^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2}} \approx 0.694$$

By similar calculations, we find that $\cos(\mathbf{a}, \mathbf{c}) \approx 0.463$ and $\cos(\mathbf{b}, \mathbf{c}) \approx 0.714$. This tells us that sentences **b** and **c** are the most similar pair out the above texts, as they have the highest cosine measure. This is also the reason $d(\mathbf{x}, \mathbf{y})$ is defined as $1 - \cos(\mathbf{x}, \mathbf{y})$ since a larger cosine measure would be converted into a smaller distance.

8.3 Distance for binary features

For binary features, their values can be converted to 1 or 0, then we calculate the contingency table.

		y	
		1	0
x	1	a	b
	0	c	d

Symmetric binary features

Binary features are **symmetric** if both of their states equally valuable and carry the same weight; i.e. no preference on which outcome should be coded as 1 or 0 , e.g. gender.

$$d(\mathbf{x}, \mathbf{y}) = \frac{b + c}{a + b + c + d}$$

Asymmetric binary features

Binary features are **asymmetric** if the outcomes of the states not equally important, e.g. the positive and negative outcomes of a disease test; the rarest one is set to 1 and the other is 0. Since the number of negative matches are considered unimportant, they are omitted from the calculations.

$$d(\mathbf{x}, \mathbf{y}) = \frac{b + c}{a + b + c}$$

8.4 Distance for nominal features

Nominal features are those that can take more than two states, for example *small, medium, large*. There are two methods to handle variables with nominal features; simple mis-matching and converting them into binary variables.

Simple mis-matching

$$d(\mathbf{x}, \mathbf{y}) = \frac{\text{number of mis-matching features between } \mathbf{x} \text{ and } \mathbf{y}}{\text{total number of features}}$$

Converting them into binary variables

We create new binary features for all of its nominal states, for example if our size feature takes three possible nominal states *small, medium, large* then this feature will be expanded into three binary features. *small, medium, large* = 100, 010, 001, then we can use the distance measures for binary features.

Worked Example

Consider the example below where we have two toys, T_1 and T_2, whose features are size, colour and price range.

	Size	Colour	Price Range
T_1	010	100	01
T_2	100	001	01

Simple mis-matching

Only the price range takes the same value for the features between T_1 and T_2 hence

$$d(\mathbf{T_1}, \mathbf{T_2}) = \frac{2}{3} \approx 0.66$$

Converting them into binary values

Size: [Small, Medium, Large] = [100, 010, 000]
Colour: [Green, Red, Yellow] = [100, 010, 000]
Price: [Cheap, Expensive] = [10, 01]

Now we do T_1 **XOR** T_2 = 01010001 **XOR** 10000101 = 11010100. Now from here we can get the distance by dividing the number of 1 bits in T_1 **XOR** T_2 by the total number of bits in T_1.

$$d(\mathbf{T_1}, \mathbf{T_2}) = \frac{4}{8} = 0.5$$

8.5 Clustering methodologies

- **Partitioning**: construct various partitions and then evaluate them by some criterion, e.g. minimising the sum of squares distance cost
- **Hierarchical**: create a hierarchical decomposition of the set of data using some criterion
- **Density-based**: based on connectivity and density functions
- Model-based: a generative model is hypothesised for each of the clusters and tries to find the best fir of that model to each other
- **Spectral clustering**: convert data set into a weighted graph then cut the graphs into subgraphs corresponding to clusters via spectral analysis
- **Clustering ensemble**: combine multiple clustering results

9. k-means Clustering

Introduction

The k-means algorithm is the simplest partitioning method for clustering analysis and is widely used in data mining algorithms. It is a **heuristic method** and is based on each clustering being represented by the centre of the cluster and the algorithm will converge to stable centroids of clusters. The goal of this algorithm is find a partition of k clusters to optimise the chosen partition criterion - the global optimum is achieved by exhaustively searching all partitions.

9.1 Partitioning clustering approach

A partitioning clustering approach is done by iteratively partitioning the training data set to learn a partition of the given data space to product several non-empty clusters, where the number of clusters is usually given in advance. In principle, an optimal partition is achieved by minimising the sum of the squared distance to its "representative object" in each cluster.

Exercise 9.1 Why do we have to used the squared distance? ∎

9.2 k-means algorithm

Before we can use the k-means algorithm, we must decide the cluster number k and an appropriate distance measure that will be used. We initialise the algorithm by selecting k distinct seed points, randomly.

1. assign each object to the cluster of the nearest point
2. compute new seed points as the centroids of the clusters of the current partition

The two steps are repeated until it converges (membership in each cluster no longer changes). At this point, we have k centroids which partition the whole data space into k mutually exclusive subspaces to form a partition.
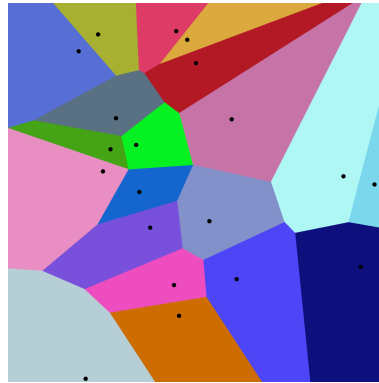
Figure 9.1: Voronoi Diagram

The figure above shows how a data space can be partitioned, where each black dot represents a centroid of a cluster. Notice that the distance used above is Euclidean distance.

9.3 Issues

- This algorithm has computational complexity of $O(tKn)$, where n is the number of data points, K is the number of clusters and t is the number of iterations. Normally, K, $t \ll n$.
- It is sensitive to initialisation, which may result to unwanted solutions
- Is unable to handle noisy data and outliers, which results in an inaccurate partition
- Requires prior knowledge of the cluster number
- Incapable of handling clusters of non-convex shape
- Inapplicable to categorical data, since the mean is not defined
- How do we evaluate the k-mean performance?

Example

The k-Means algorithm is very simple, so an example has be omitted. An example may be added in the future, if I have spare time - but for now, just check the lecture slides.

10. Hierarchical and Ensemble Clustering

10.1 Hierarchical clustering approach

Hierarchical clustering approach is done by sequentially partitioning the data set to construct nested partitions layer by layer, via grouping objects into a tree of clusters. We use a generalised distance matrix as the clustering criteria and we do not need to know the number of clusters in advance.

Approaches to hierarchical clustering

There are two main approaches to hierarchical clustering: **agglomerative clustering** (bottom-up) and **divisive clustering** (top-down). In agglomerative clustering, each data point is initially its own atomic cluster and we merge clusters into larger and larger clusters. Divisive clustering, as the name suggests, is where all data points belong to a single cluster initially, then the cluster is divided into smaller and smaller clusters.

10.2 Cluster distance measures

Definition 10.2.1 — Single link. The **smallest** distance between an element in one cluster and an element in another, i.e. $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$

Definition 10.2.2 — Complete link. The **largest** distance between an element in one cluster and an element in another, i.e. $d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$

Definition 10.2.3 — Average. The **average** distance between elements in one cluster and elements in the other, i.e. $d(C_i, C_j) = \mathrm{avg}\{d(x_{ip}, x_{jq})\}$

10.3 Agglomerative clustering

Algorithm

1. Select a cluster distance measure to use
2. Convert all object features into a distance matrix
3. Set each object as an atomic cluster (N objects means N clusters at start)
4. Repeat until number of clusters is one (or known # of clusters)
   - Merge two closest clusters
   - Update distance matrix

Example

Again, very simple algorithm - so an example has been omitted as the lecture example is sufficient.

> **Definition 10.3.1 — Lifetime.** The distance between that a cluster is created and that it disappears (merges with other clusters during clustering).

> **Definition 10.3.2 — k-cluster lifetime.** The distance from that K clusters emerge to that K clusters vanish (due to the reduction to $K-1$ clusters).

Relevant Issues

- How do we determine the number of clusters?
  - If the number of clusters is known, then we have a termination condition
  - We can use the **K-cluster lifetime** as a range of threshold values on the dendrogram that leads to the identification of K clusters
  - Heuristic method: cut the dendrogram tree with maximum life time to find a "proper" K
- Major weakness of agglomerative clustering methods
  - Can never undo what was done previously...
  - Sensitive to cluster distance measures, noise and outliers
  - Efficiency of $O(n^2 \log n)$, where n is the number of total objects

10.4 Ensemble Clustering

Ensemble clustering is concerned with the issues that may affect a single clustering algorithm, for example, sensitivity to initialisation, noise, outliers, distance metrics and it may be hard to choose a single algorithm which can handle all types of cluster shapes and sizes. The purpose of ensemble clustering is to utilise the results obtained by multiple clustering analyses for robustness.

Algorithm summary

1. Perform clustering analysis by using either different clustering algorithms or running a single clustering algorithm of different conditions, **leading to multiple partitions**
2. Convert clustering results on different partitions into **binary distance measure**
3. Evidence accumulation: form a **collective distance matrix** based on all the distance matrices
4. Apply a hierarchical clustering algorithm (with a proper cluster distance measure) to collective distance matrix and **use the maximum k-cluster lifetime to decide**

11. Cluster Validation

Introduction

Cluster validation refers to the procedures that evaluate the results of clustering in a *quantitive* and *objective* fashion. In cluster validation, we need to compare clustering algorithms, solve the problem of determining the number of clusters, avoid finding patterns in noise and finding the "best" clusters from the data.

We have two types of criteria, internal and external. An **internal criteria** (index) is used to validate without external information and can be applied to find the "proper" number of clusters in a data set. An **external criteria** (index) may be applied to evaluate the performance of a clustering algorithm against data sets where the ground truth is available, by comparing how similar the partition generated by the clustering algorithm is to the ground truth.

11.1 Internal Index

The internal index we will focus on is the **F-ratio index**, which is a variance based internal index.

Variance-based methods

In **variance-based methods**, the goal is to minimise the intra-cluster variance and to maximise the inter-cluster variance. Assume an algorithm leads to a partition of K clusters where cluster i has n_i data points, c_i is its centroid and $d(\cdot, \cdot)$ is a chosen distance measure.

$$\text{Intra-cluster variance} = SSW(K) = \sum_{i=1}^{K} \sum_{j=1}^{n_i} d^2(\mathbf{x}_{ij}, \mathbf{c}_i)$$

$$\text{Inter-cluster variance} = SSB(K) = \sum_{i=1}^{K} n_i \cdot d^2(\mathbf{c}_i, \mathbf{c})$$

where $\mathbf{c}$ is the global centroid of the whole data set

11.1.1   F-ratio index

The **F-ratio index** is a measure of the inter-cluster variance against the intra-cluster variance. We calculate the F-ratio index for a partition of K clusters as follows.

$$F(K) = \frac{K \cdot SSW(K)}{SSB(K)} = \frac{K \cdot \sum_{i=1}^{K} \sum_{j=1}^{n_i} d^2(\mathbf{x}_{ij}, \mathbf{c}_i)}{\sum_{i=1}^{K} n_i \cdot d^2(\mathbf{c}_i, \mathbf{c})}$$

It is ideal to have a low F-ratio index, so we run clustering algorithms for a range of values of K (the number of clusters) and calculate a F-ratio index for each value of K and each algorithm, then select the algorithm and value of K which yields the lowest F-ratio index.

11.2   External Index

The external index we will focus on is the Rand Index, 1971.

11.2.1   Issues with external indexes

- **cluster-ID permutation**: the cluster IDs in a partition from clustering have been assigned arbitrarily due to unsupervised learning
- **inconsistence in cluster numbers**: the number off clusters from algorithm may be different from the number of classes (in ground truth)
- **point-pair correspondence**: how do we find all possible correspondences between the ground truth and a partition?

11.2.2   Rand Index

The idea behind **Rand Index** is to consider all pairs in the data set by looking into both *agreement* and *disagreement* against the ground truth.

X/Y	Pairs in the same class	Pairs in different classes
Pairs in the same cluster	a	b
Pairs in different clusters	c	d

Definition 11.2.1 — Rand Index. Let $\mathbf{X}$ be the partition from clustering and $\mathbf{Y}$ be the ground truth. The Rand Index is defined as

$$\text{RI}(\mathbf{X}, \mathbf{Y}) = \frac{a+d}{a+b+c+d}$$

To calculate a, b, c, d, we can use a contingency table as follows. k denotes the number of clusters in $\mathbf{X}$, l denotes the number of classes in $\mathbf{Y}$ and each n_{ij} denotes the number of points in both cluster i and class j.

$$
\begin{array}{cccc|c}
n_{11} & n_{12} & \cdots & n_{1l} & n_{1.} \\
n_{21} & n_{22} & \cdots & n_{2l} & n_{2.} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
n_{k1} & n_{k2} & \cdots & n_{kl} & n_{k.} \\
\hline
n_{.1} & n_{.2} & \cdots & n_{.l} & N
\end{array}
$$

$$a = \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{l} n_{ij}(n_{ij} - 1) \qquad\qquad c = \frac{1}{2} (\sum_{i=1}^{k} n_{i.} - \sum_{i=1}^{k} \sum_{j=1}^{l} n_{ij}^2)$$

$$b = \frac{1}{2} (\sum_{j=1}^{l} n_{.j} - \sum_{i=1}^{k} \sum_{j=1}^{l} n_{ij}^2) \qquad d = \frac{1}{2} (N^2 + \sum_{i=1}^{k} \sum_{j=1}^{l} n_{ij}^2 - (\sum_{i=1}^{k} n_{i.} + \sum_{j=1}^{l} n_{.j}))$$

11.3 Weighted Clustering Ensemble

Motivation

The problem with the **clustering ensemble** algorithm is that it does not distinguish between *non-trivial* and *trivial* partitions so all partitions in clustering ensemble are treated **equally important**. This algorithm is also sensitive to the cluster distance used in the hierarchical clustering for reaching a consensus.

Solution

With **weighted clustering ensemble**, multiple internal and external validity indexes reflecting various aspects are applied to different partitions obtained by initial clustering analysis. **Internal indexes** are used directly to measure the importance of a partition, quantitatively and objectively. **External indexes** are used indirectly to measure the importance of a partition via comparing with other partitions used in clustering ensemble for another round of *evidence accumulation*.

The values of these indexes are used as weights to highlight the non-trivial evidence and to diminish the trivial evidence simultaneously. Thus, the weighted clustering ensemble yield better yet robust performance in clustering analysis.

Index