

AA 2022/2023

Machine Learning for Modelling: *Supervised Learning*

Simone Bianco

1

AA 2022/2023

Basic concepts

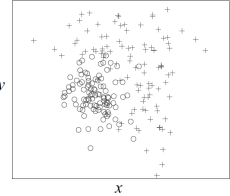
2

Basic concepts

- One major task of machine learning, pattern recognition and data mining is to construct good **models** from **data sets**
- A data set generally consists of **feature vectors**, each of them being a description of an object(entity?) by using a set of **features**

3

Basic concepts



Three-gaussians synthetic dataset

- Each object is a data point described by the features (x-coord., y-coord., shape) and a feature vector looks like (0.5, 0.2, circle), (0.1, 0.9, cross)
- The number of features of a data set is called **dimension** or **dimensionality**
- Features are also called **attributes**, a feature vector is also called an **instance**
- A model is usually a predictive model or a model of the structure of the data that we want to construct or discover from the data set, e.g., decision tree, NN, SVM, etc.

4

Basic concepts

- The process of generating models from data is called **learning** or **training**, which is accomplished by a **learning algorithm**
- The learned model can be called a **hypothesis**, or a **learner**
- There are different learning settings, among which the most common ones are **supervised learning** and **unsupervised learning**
- In supervised learning, the goal is to predict the value of a target feature on unseen instances, and the learned model is also called a **predictor**
 - For example, if we want to predict the shape of the three-Gaussians data points, we call "cross" and "circle" labels
 - the predictor should be able to predict the label of an instance for which the label information is unknown, e.g., $(2, .3)$.
 - if the label is categorical, such as shape, the task is also called **classification** and the learner is also called **classifier**;
 - if the label is numerical, the task is also called **regression** and the learner is also called **fitted regression model**.
- the training process is conducted on data sets containing label information, and an instance with known label is also called an **example**.

5

Basic concepts

- Unsupervised learning instead does not rely on label information. Its goal is to discover some inherent distribution information in the data (e.g., clustering)
- Whether the model is "good" depends on whether it can meet the requirements of the user or not
- It is common to evaluate and estimate the performance of the models, and then let the user to decide whether a model is acceptable, or choose the best available model from a set of candidates
- Since the fundamental goal of learning is **generalization**, i.e., being capable of generalizing the "knowledge" learned from training data to unseen instances, a good learner should generalize well, i.e., have a small **generalization error**, also called the **prediction error**.
- It is infeasible to estimate the generalization error directly, since that requires knowing the **ground-truth** label information which is unknown for unseen instances.
- A typical empirical process is to let the predictor make predictions on test data of which the ground-truth labels are known and take the **test error** as an estimate of the generalization error.

6

Basic concepts

- The process of applying a learned model to unseen data is called **testing**.
- Before testing, a learned model often needs to be configured, e.g., tuning the parameters/hyper-parameters, and this process also involves the use of data with known ground-truth labels to evaluate the learning performance; this is called **validation** and the data is **validation data**.
- Generally, the test data should not overlap with the training and validation data; otherwise, the estimated performance can be over-optimistic.

7

Formal formulation of the learning process

- Let \mathcal{X} be the instance space, \mathcal{D} a distribution over \mathcal{X} , and $f(\cdot)$ the ground-truth target function.
- Given a training data set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ where the instances x_i are drawn i.i.d. (independently and identically distributed) from \mathcal{D} , and $y_i = f(x_i)$

- Taking classification as an example

- The goal is to construct a learner $h(\cdot)$ which minimizes the generalization error:

$$err(h) = \mathbb{E}_{x \sim \mathcal{D}}[\mathbb{I}(h(x) \neq f(x))]$$

- Where $\mathbb{I}(\cdot)$ is the indicator function which takes 1 if its argument is true, and 0 otherwise (i.e., it counts the number of samples for which the predicted label is different from the ground-truth label)

8

Popular Learning Algorithms

9

Linear Discriminant Analysis

- A linear classifier consists of a weight vector w and a bias b .
 - Given an instance x , the predicted class label y is obtained according to

$$y = \text{sign}(w^T x + b)$$
 - The classification process is accomplished by two steps: First, the instance space is mapped onto a one-dimensional space through the weight vector w ; Then, a point on the line is identified to separate the positive instances from the negative ones
 - To find the best w and b , a classical linear learning algorithm is Fisher Linear Discriminant Analysis (LDA)
- The idea of LDA is to push instances of different classes far away, and instances within the same class to be close; it can be accomplished by making the distance between centers of different classes large while keeping the variance within each class small

10

Linear Discriminant Analysis

Given a 2-class training set, we consider all the positive instances and obtain the mean μ_+ and the covariance matrix Σ_+ .

Similarly for the negative instances we obtain μ_- and the covariance matrix Σ_- .

The distance between the projected class centers is measured as:

$$S_B(w) = (w^T \mu_+ - w^T \mu_-)^2$$

and the variance within classes is measured as:

$$S_W(w) = w^T \Sigma_+ w - w^T \Sigma_- w$$

LDA combines these two measures by maximizing:

$$J(w) = S_B(w) / S_W(w)$$

which has an optimal closed-form solution:

$$w^* = (\Sigma_+ + \Sigma_-)^{-1}(\mu_+ - \mu_-)$$

11

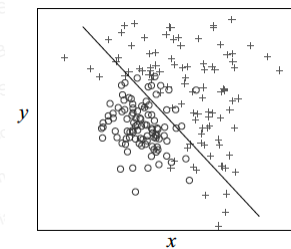
Linear Discriminant Analysis

After obtaining w^* it is easy to calculate the bias b .

The simplest approach is to set it as the middle point between the projected centers:

$$b^* = w^{*T}(\mu_+ + \mu_-)/2$$

Which is optimal if the two classes have normal distribution with the same variance.



Decision boundary of LDA on the three-gaussians data set

12

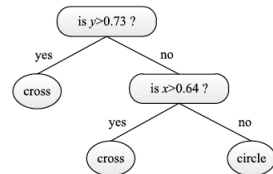
Decision Trees

A decision tree consists of a set of tree-structured decision tests working in a divide-and-conquer way.

Each non-leaf node is associated with a feature test, also called split: data falling into the node will be split into different subsets according to their different values on the feature test.

Each leaf node is associated with a label, which will be assigned to instances falling into this node.

In prediction, a series of feature tests is conducted starting from the root node, and the result is obtained when a leaf node is reached.



13

Decision Trees

Decision tree learning algorithms are generally recursive processes.

In each step, a data set is given, and a split is selected; then this split is used to divide the data set into subsets, and each subset is considered as the given data set for the next step.

The key element of a decision tree algorithm is how to select the splits.

In the **ID3** algorithm the information gain criterion is employed for split selection: give a training set D , the entropy of D is defined as

$$Ent(D) = \sum_{y \in \mathcal{Y}} P(y|D) \log(P(y|D))$$

If the dataset is divided into subsets D_1, \dots, D_k the entropy may be reduced, and the reduction is the information gain, i.e.:

$$G(D; D_1, \dots, D_k) = Ent(D) - \sum_{i=1}^k \frac{|D_k|}{|D|} Ent(D_k)$$

Thus, the feature-value pair which will cause the largest information gain is selected for the split

14

Decision Trees

One problem with the information gain criterion is that features with a lot of possible values will be favored, disregarding their relevance for the final classification.

This deficiency is addressed in **C4.5**, the most famous decision tree algorithm, that employs the **golden ratio**:

$$P(D; D_1, \dots, D_k) = G(D; D_1, \dots, D_k) \cdot \left(- \sum_{i=1}^k \frac{|D_k|}{|D|} \log \frac{|D_k|}{|D|} \right)^{-1}$$

which is a variant of the information gain criterion, taking **normalization** on the number of feature values: the feature with the highest gain ratio, among features with better-than-average information gains, is selected as the split.

15

Decision Trees

CART is another famous decision tree algorithm, which uses Gini index for selecting the split maximizing the Gini index:

$$G_{gini}(D; D_1, \dots, D_k) = I(D) - \sum_{i=1}^k \frac{|D_k|}{|D|} I(D_k)$$

where

$$I(D) = 1 - \sum_{y \in \mathcal{Y}} P(y|D)^2$$

16

Decision Trees

CART example.

We want to answer the question "what kind of person would play ground-game"?

Age	Gender	Sportive
22	f	yes
24	m	yes
30	f	yes
31	f	no
27	f	no
32	m	no
25	f	yes
30	m	no
24	f	yes
21	f	yes
29	m	yes
26	m	no
21	m	no

17

Decision Trees

CART example.

We want to answer the question "what kind of person would play ground-game"?

We divide data in binary, like f vs m and age ≤ 25 vs age > 25

$$\#(f \ \& \ yes) = 5; \#(f \ \& \ no) = 2$$

Age	Gender	Sportive
22	f	yes
24	m	yes
30	f	yes
31	f	no
27	f	no
32	m	no
25	f	yes
30	m	no
24	f	yes
21	f	yes
29	m	yes
26	m	no
21	m	no

18

Decision Trees

CART example.

We want to answer the question "what kind of person would play ground-game"?

We divide data in binary, like f vs m and age ≤ 25 vs age > 25

$$\#(f \ \& \ yes) = 5; \#(f \ \& \ no) = 2$$

$$\#(m \ \& \ yes) = 2; \#(m \ \& \ no) = 4$$

Age	Gender	Sportive
22	f	yes
24	m	yes
30	f	yes
31	f	no
27	f	no
32	m	no
25	f	yes
30	m	no
24	f	yes
21	f	yes
29	m	yes
26	m	no
21	m	no

19

Decision Trees

CART example.

We want to answer the question "what kind of person would play ground-game"?

We divide data in binary, like f vs m and age ≤ 25 vs age > 25

$$\#(f \ \& \ yes) = 5; \#(f \ \& \ no) = 2$$

$$\#(m \ \& \ yes) = 2; \#(m \ \& \ no) = 4$$

$$\#(low \ \& \ yes) = 5; \#(low \ \& \ no) = 1$$

$$\#(more \ \& \ yes) = 2; \#(more \ \& \ no) = 5$$

Age	Gender	Sportive
22	f	yes
24	m	yes
30	f	yes
31	f	no
27	f	no
32	m	no
25	f	yes
30	m	no
24	f	yes
21	f	yes
29	m	yes
26	m	no
21	m	no

20

Decision Trees

CART example.

We want to answer the question "what kind of person would play ground-game"?

We divide data in binary, like f vs m and age ≤ 25 vs age > 25

- #(f & yes) = 5; #(f & no) = 2
- #(m & yes) = 2; #(m & no) = 4
- #(low & yes) = 5; #(low & no) = 1
- #(more & yes) = 2; #(more & no) = 5
- Total = 13
- Total of f = 7; Total of m = 6
- Total of low = 6; Total of more = 7

Age	Gender	Sportive
22	f	yes
24	m	yes
30	f	yes
31	f	no
27	f	no
32	m	no
25	f	yes
30	m	no
24	f	yes
21	f	yes
29	m	yes
26	m	no
21	m	no

21

Decision Trees

CART example.

We want to answer the question "what kind of person would play ground-game"?

We divide data in binary, like f vs m and age ≤ 25 vs age > 25

- #(f & yes) = 5; #(f & no) = 2
- #(m & yes) = 2; #(m & no) = 4
- #(low & yes) = 5; #(low & no) = 1
- #(more & yes) = 2; #(more & no) = 5
- Total = 13
- Total of f = 7; Total of m = 6
- Total of low = 6; Total of more = 7

$$\text{Gini of f} = 1 - (f\&yes)^2 - (f\&no)^2 = 1 - \left(\frac{5}{7}\right)^2 - \left(\frac{2}{7}\right)^2 \approx 0.408$$

$$\text{Gini of m} = 1 - (m\&yes)^2 - (m\&no)^2 = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 \approx 0.445$$

$$\text{Gini of low} = 1 - (low\&yes)^2 - (low\&no)^2 = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2 \approx 0.028$$

$$\text{Gini of more} = 1 - (more\&yes)^2 - (more\&no)^2 = 1 - \left(\frac{2}{7}\right)^2 - \left(\frac{5}{7}\right)^2 \approx 0.408$$

22

Decision Trees

- Total = 13
- Total of f = 7; Total of m = 6
- Total of low = 6; Total of more = 7

Gini of f ≈ 0.408

Gini of m ≈ 0.445

Gini of low ≈ 0.028

Gini of more ≈ 0.408

Compute Gini Index

- Gender = $\frac{\text{Total of f}}{\text{Total}} \cdot \text{Gini of f} + \frac{\text{Total of m}}{\text{Total}} \cdot \text{Gini of m} = \frac{7}{13} \cdot 0.408 + \frac{6}{13} \cdot 0.445 \approx 0.425$
- Age = $\frac{\text{Total of low}}{\text{Total}} \cdot \text{Gini of low} + \frac{\text{Total of more}}{\text{Total}} \cdot \text{Gini of more} = \frac{6}{13} \cdot 0.028 + \frac{7}{13} \cdot 0.408 \approx 0.233$

The lowest one is Age → Our root node in the decision tree will be lowest Gini index node

23

Decision Trees

It is often observed that a decision tree, which is perfect on the training set, will have a worse generalization ability than a tree which is not-so-good on the training set; this is called **overfitting**

It may be caused by the fact that some peculiarities of the training data, such as those caused by noise in collecting training examples, are misleadingly recognized by the learner as the underlying truth.

To reduce the risk of overfitting, a general strategy is to employ **pruning** to cut off some tree branches caused by noise or peculiarities of the training set: **pre-pruning** tries to prune branches when the tree is being grown, while **post-pruning** re-examines fully grown trees to decide which branches should be removed.

When a validation set is available, the tree can be pruned according to the validation error: for pre-pruning, a branch will not be grown if the validation error will increase by growing the branch; for post-pruning, a branch will be removed if the removal will decrease the validation error.

24

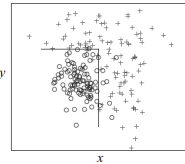
Decision Trees

Early decision tree algorithms, such as ID3, could only deal with categorical features.

Later ones, such as C4.5 and CART, are enabled to deal with numerical features.

The simplest way is to evaluate every possible split point on the numerical feature that divides the training set into two subsets, where one subset contains instances with the feature value smaller than the split point while the other subset contains the remaining instances.

When the height of a decision tree is limited to 1, i.e., it takes only one test to make every prediction, the tree is called a **decision stump**. While decision trees are nonlinear classifiers in general, decision stumps are a kind of linear classifiers.



Decision boundary of a typical decision stump on the three-Gaussian data set

25

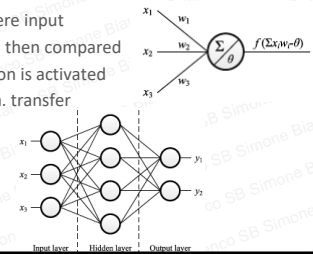
Neural Networks

Neural networks, also called **artificial neural networks**, originated from simulating biological neural networks.

The function of a neural network is determined by the model of neuron, the network structure, and the learning algorithm.

Neuron is also called **unit**, which is the basic computational component in neural networks.

The most popular neuron model is the McCulloch-Pitts model where input signals are multiplied with corresponding **connection weights** and then compared to a threshold (**bias**). If the result is higher than the bias, the neuron is activated and the output signal is generated by an **activation function** (a.k.a. transfer function or squashing function).



Neurons are linked by weighted connections to form a network. There are many possible NN structures, among which the most popular one is the multi-layer feed-forward network.

26

Neural Networks

The goal of training a NN is to determine the values of the connection weights and the biases of the neurons.

The most commonly applied idea for training a multi-layer feed-forward neural network is that, as long as the activation function is differentiable, the whole neural network can be regarded as a differentiable function which can be optimized by gradient descent method.

The most successful NN training algorithm is **Back-Propagation (BP)**:

- At first, the inputs are feedforwarded from the input layer via the hidden layer to the output layer, at which the error is calculated by comparing the network output with the ground-truth.
- Then, the error will be back propagated to the hidden layer and the input layer, during which the connection weights and biases are adjusted to reduce the error. The process is accomplished by tuning towards the direction with the gradient.
- Such a process will be repeated in many rounds, until the training error is minimized, or the training process is terminated to avoid overfitting.

27

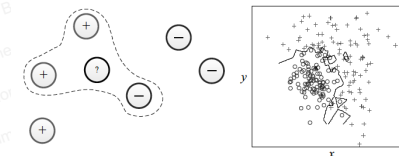
k-Nearest Neighbor

The k-nearest neighbor (kNN) algorithm relies on the principle that objects similar in the input space are also similar in the output space.

It is a **lazy learning** approach since it does not have an explicit training process, but simply stores the training set instead.

For a test instance, a k-NN learner identifies the k instances from the training set that are closest to the test instance. Then, for classification, the test instance will be classified to the majority class among the k instances; while for regression, the test instance will be assigned the average value of the k instances.

The 1-NN is also called the **nearest neighbor classifier**.



28

Support Vector Machines and Kernel Methods

SVMs, originally designed for binary classification, are **large margin classifiers** that try to separate instances of different classes with the maximum margin hyperplane.

The **margin** is defined as the minimum distance from instances of different classes to the classification hyperplane.

Considering a linear classifier $y = \text{sign}(w^T x + b)$ we can use the **hinge loss** to evaluate the fitness to the data: $\sum_{i=1}^m \max\{0, 1 - y_i(w^T x_i + b)\}$.

The Euclidean distance of an instance x_i to the hyperplane $w^T x + b$ is $\frac{|w^T x_i + b|}{\|w\|}$.

If we restrict $|w^T x_i + b| \geq 1$ for all instances, the minimum distance to the hyperplane is $\|w\|^{-1}$.

29

Support Vector Machines and Kernel Methods

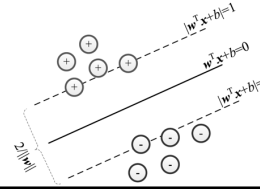
SVMs solve the optimization problem

$$(w^*, b^*) = \arg \min_{w, b, \xi_i} \frac{\|w\|^2}{2} + C \sum_{i=1}^m \xi_i$$

$$s. t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i = 1, \dots, m,$$

$$\xi_i \geq 0 \quad \forall i = 1, \dots, m$$

Where C is a parameter and ξ_i are slack variables introduced to enable the learner to deal with data that could not be perfectly separated. The above equation is called the *primal form* of the optimization.



30

Support Vector Machines and Kernel Methods

A limitation of the linear classifier is that, when the data is intrinsically nonlinear, linear classifiers cannot separate the classes well.

The general approach is to map the data points onto a higher-dimensional feature space where the data may become linearly separable. However, the learning process may become very slow and even intractable since the inner product (in the *dual form*) will be difficult to calculate in the high-dimensional space.

Fortunately, there is a class of functions, **kernel functions** (also called **kernels**), which can help address the problem. The feature space derived by kernel functions is called the **Reproducing Kernel Hilbert Space (RKHS)**.

An inner product in the RKHS equals kernel mapping of inner product of instances in the original lower-dimensional feature space:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

Where ϕ is a mapping function from the original feature space to a higher dimensional space and K is a kernel.

31

Support Vector Machines and Kernel Methods

According to Mercer's Theorem, every positive semi-definite symmetric function is a kernel.

Popular kernels include:

- Linear kernel
- Polynomial kernel
- Gaussian kernel (a.k.a. RBF kernel)

The **kernel trick**, i.e., mapping the data points with a kernel and then accomplishing the learning task in the RKHS, is a general strategy that can be incorporated into any learning algorithm that considers only inner products between the input feature vectors.

Once the kernel trick is used, the learning algorithms are called **kernel methods**. Indeed, SVMs are a special kind of kernel method, i.e., linear classifiers facilitated with kernel trick.

32

Evaluation and Comparison

33

Evaluation and comparison

Usually, we have multiple alternative learning algorithms to choose among, and a number of parameters to tune.

The task of choosing the best algorithm and the settings of its parameters is known as **model selection**, and for this purpose we need to estimate the performance of the learner. By empirical ways, this involves design of experiments and statistical hypothesis tests for comparing the models.

It is unwise to estimate the generalization error of a learner by its **training error**, i.e., the error that the learner makes on the training data, since training error prefers complex learners rather than learners that generalize well.

A proper process is to evaluate the performance on a validation set. Note that the labels in the training set and validation set are known *a priori* to the training process and should be used together to derive and tune the final learner once the model has been selected.

In most cases the training and validation sets are obtained by splitting a given data set into two parts.

34

Evaluation and comparison

While splitting, the properties of the original data set should be kept as much as possible. In classification, when the original data set is split randomly, the class percentage should be maintained for both training and validation sets; this is called **stratification**, or stratified sampling.

When there is not enough labeled data available to create a separate validation set, a commonly used validation method is **cross-validation**. In k-fold cross-validation the dataset is partitioned by stratified split into k equal-size disjoint subsets D_1, \dots, D_k , and then k runs of training-test are performed. In the i-th run, D_i is used as the validation set, while the union of all the other subsets $\bigcup_{j \neq i} D_j$ is used as training set.

The average results of the k runs are taken as the results of the cross-validation.

To reduce the influence of randomness introduced by data split, the k-fold cross-validation can be repeated t times, which is called **t-times k-fold cross-validation** (Usual configurations: 10-times 10-fold, and 5-times 2-fold).

Extremely, when k equals the number of instances in the original data set, there is only one instance in each validation set; this is called **leave-one-out (LOO) validation**.

35

Evaluation and comparison

After obtaining the estimated errors, we can compare different learning algorithms.

A simple comparison on average errors, however, is not reliable since the winning algorithm may occasionally perform well due to the randomness in data split. **Hypothesis test** is usually employed for this purpose.

36

Evaluation and comparison

To compare learning algorithms that are efficient enough to run 10 times, the **5x2 cv paired t-test** is a good choice.

We run 5-time 2-fold cv. In each run the dataset D is split in D_1 and D_2 having equal size. Two algorithms a and b are trained on each set and tested on the other, resulting in four error estimates:

$err_a^{(1)}$ and $err_b^{(1)}$ (trained on D_1 and tested on D_2); $err_a^{(2)}$ and $err_b^{(2)}$ (trained on D_2 and tested on D_1).

We have the error differences $d^{(i)} = err_a^{(i)} - err_b^{(i)}$, $i = 1, 2$
with the mean:

$$\mu = \frac{d^{(1)} + d^{(2)}}{2}$$

and the variance:

$$s^2 = (d^{(1)} - \mu)^2 + (d^{(2)} - \mu)^2$$

37

Evaluation and comparison

Let s_i^2 denote the variance in the i -th time 2-fold cv, and $d_1^{(1)}$ the error difference in the first time. Under the null hypothesis the 5x2 cv \tilde{t} -statistic

$$\tilde{t} = \frac{d_1^{(1)}}{\sqrt{\frac{1}{5} \sum_{i=1}^5 s_i^2}} t_5$$

would be distributed according to the Student's t -distribution with 5 degrees of freedom.

We then choose a significance level α .

If \tilde{t} falls into the interval $[-t_5(\alpha/2), t_5(\alpha/2)]$, the null hypothesis is accepted, suggesting that there is no significant difference between the two algorithms. Usually, α is set to 0.05 or 0.1.

38

Evaluation and comparison

To compare learning algorithms that can be run only once, the **McNemar's test** can be used instead.

Let err_{01} denote the number of instances on which the first algorithm makes a wrong prediction while the second one is correct, and err_{10} the inverse.

If the two algorithms have the same performance, err_{01} and err_{10} will be close, and therefore the quantity

$$\frac{(|err_{01} - err_{10}| - 1)^2}{err_{01} + err_{10}} \sim \chi_1^2$$

would be distributed according to the χ^2 -distribution.

39

Evaluation and comparison

Sometimes, we evaluate multiple learning algorithms on multiple data sets. In this situation, we can conduct the **Friedman test**.

First, we sort the algorithms on each data set according to their average errors. On each data set, the best algorithm is assigned rank 1, the worse algorithms are assigned increased ranks, and average ranks are assigned in case of ties.

Then, we average the ranks of each algorithm over all data sets, and use the Nemenyi post-hoc test to calculate the *critical difference* value

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

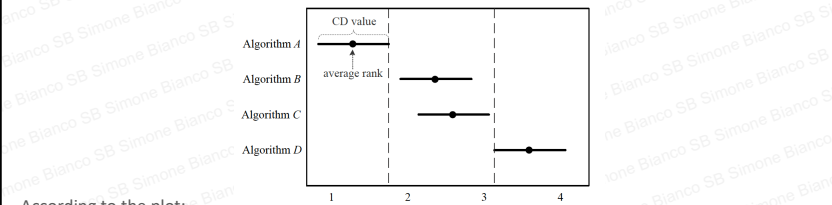
where k is the number of algorithms, N is the number of data sets and q_α is the critical value.

A pair of algorithms are believed to be significantly different if the difference of their average ranks is larger than the critical difference CD .

40

Evaluation and comparison

The Friedman test results can be visualized by plotting the **critical difference diagram**, where each algorithm corresponds to a bar centered at the average rank with the width of critical difference value.



According to the plot:

- algorithm A is significantly better than all the other algorithms
 - Algorithm D is significantly worse than all the other algorithms
 - Algorithms B and C are not significantly different
- according to the given significance level.