Supervised learning

Profesor: Juan R. Rico 

Técnicas de Aprendizaje Automático

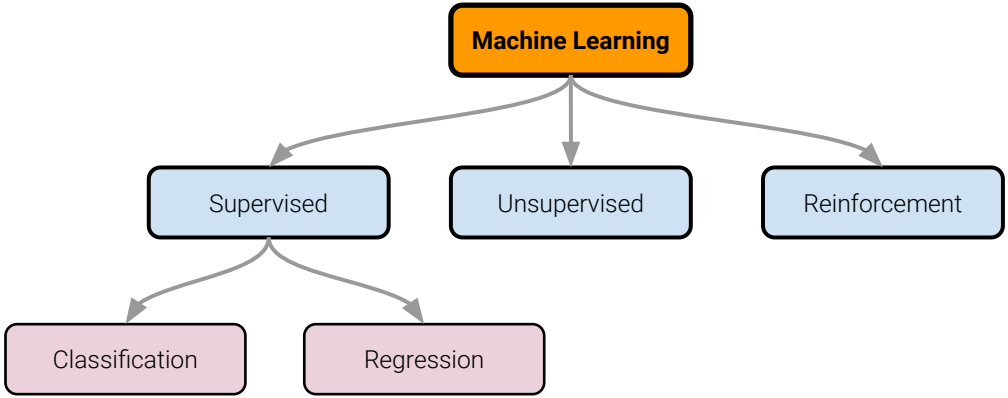
Dpto. Lenguajes y Sistemas Informáticos. Universidad de Alicante

Index

1. Introduction

Table of contents

Let's remind the general scheme



Target variable

Based on the target variable, we have problems of:

Classification when this variable describes **categories** or **classes**.

E.g., being sick or not, investing/holding/selling certain securities in the stock market, identifying a car license plate with digits from 0 to 9 and letters from A to Z.

Regression when this variable describes a **numerical value**.

E.g. temperature forecast, housing value in the real estate market, or profits/losses of a business action.

Supervised Learning: classification and regression

- Although these problems may seem similar, **learning algorithms** act **differently**.
- As we will explain in the next slides, most algorithms are capable of addressing both approaches.
- However, there are some specific to each problem.
  E.g. **linear regression** is typical for **regression**, **logistic regression** and the **Naive Bayes** algorithms are typical for **classifiers**.

Main learning styles

According to the learning strategy used

Style	Structured Types	Inherently Explainable
Neighborhood (distance). kNN (k-nearest neighbors)	Yes	Yes
Decision Trees (DT)	No	Yes (1) / No (N)
Bayesian Models	No	Yes
Support Vector Machine (SVM)	No	No
Artificial Neural Networks (ANN)	No	No

Confusion matrix

Confusion matrix (binary classification)

True value	1	TP (True Positive)	FN (False Negative)
	0	FP (False Positive)	TN (True Negative)
		1	0
		Prediction	

where N is the total number of samples.

Most common metrics for classification

Clasification I

- Precision (precisión):

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

- Recall (exhaustividad):

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

Clasification II

- Accuracy (exactitud):

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

- F1 (unbalanced classes):

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Metrics in Python

Example:

```
from sklearn import metrics
y_true = [1,1,1,0,0,0]
y_pred = [1,1,0,0,0,1]
```

Classification

- (Binary) tn, fp, fn, tp = `metrics.confusion_matrix(y_true, y_pred).ravel()`
- Precision: `metrics.precision_score(y_true, y_pred)`
- Recall: `metrics.recall_score(y_true, y_pred)`
- Accuracy: `metrics.accuracy_score(y_true, y_pred)`
- F1 (imbalanced classes): `metrics.f1_score(y_true, y_pred)`

Most common metrics for regression

Regression I

- Mean Absolute Error (MAE):

$$MAE = \frac{\sum_{i \in N} |y_i - \hat{y}_i|}{N}$$

- Mean Squared Error (MSE):

$$MSE = \frac{\sum_{i \in N} (y_i - \hat{y}_i)^2}{N}$$

Regression II

- Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i \in N} (y_i - \hat{y}_i)^2}{N}}$$

- R^2: $MSE(baseline) = \frac{\sum_{i \in N} (y_i - \bar{y}_i)^2}{N}$

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)}$$

Metrics in Python

Example:

```
from sklearn import metrics
y_true = [1.1,2.1,3.5,4.0]
y_pred = [1.2,2.0,3.6,3.8]
```

Regression

- MAE: `metrics.mean_absolute_error(y_true, y_pred)`
- MSE: `metrics.mean_squared_error(y_true, y_pred)`
- RMSE: `metrics.mean_squared_error(y_true, y_pred, squared=False)`
- R^2: `metrics.metrics.r2_score(y_true, y_pred)`

2. Neighborhood-based algorithms

Table of contents

Table of contents

Comparing samples

The **distance** function returns a value that indicates the **difference** between the features of samples.

Examples of distances between two samples I

Vectors

Applied to vectors of real numbers.

- **Manhattan or L1**: Distance in rectilinear blocks, defined as $d_{L1}(\overrightarrow{x}, \overrightarrow{y}) = \sum_i \|x_i - y_i\|$ (Craw, 2010).
- **Euclidean or L2**: Most commonly used, defined as $d_{L2}(\overrightarrow{x}, \overrightarrow{y}) = \sqrt{\sum_i (x_i - y_i)^2}$.

Strings

Applied to sequences of symbols.

- **String edit or Levenshtein**: Defined as the minimum number of operations to convert one string into another, efficiently implemented by (Wagner and Fischer, 1974).

Examples of distances between two samples II

Other structures

- **Tree edit**: Distance between trees with labeled and ordered nodes and edges (Zhang and Shasha, 1989).
- **Graph edit**: The distance between two graphs is an NP-complete problem (Kubicka et al., 1990), so we can only find approximations to the solution in polynomial time.

DTW (Dynamic Time Warping)

It is applied in the analysis of **time series** when there is variation in time or speed (Gupta et al., 1996).

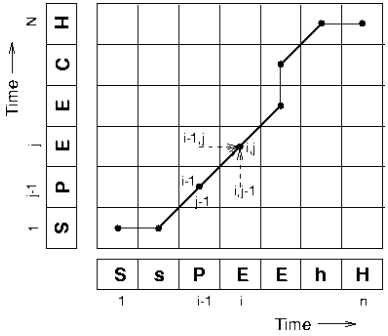
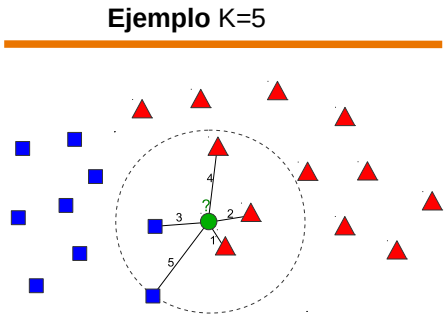


Table of contents

Definition

The **k-NN** method, k Nearest Neighbors (Silverman and Jones, 1989), is a **supervised** classification method that estimates the density function used to calculate the **posterior probability**, $\hat{P}(\omega_i|x) = k_i/k$, that an example x belongs to class ω_i (k_i is the number of neighbors from class i).



Highlights

- **Easy** to implement.
- The **error** of the prediction is **bounded** by twice the minimum Bayes error (Cover and Hart, 1967).
- In many cases, the **1-NN** technique is **sufficient** to achieve **good** classification rates.
- Let $\mathcal{D}_n = \{(x_1, \theta_1), (x_2, \theta_2), \ldots, (x_n, \theta_n)\}$ be a set of prototypes where $\theta_i = \{\omega_1, \ldots, \omega_c\}$ is the class of element x_i, and x is a new example. The k-NN technique can be formulated as:

$$\text{mode}\left(\theta_k\left(\arg\min_{x_i \in \mathcal{D}_n}{}_k\{d(x, x_i)\}\right)\right)$$

Neighbor search doesn't have to be linear

- In **principle**, to find the **nearest neighbor** of a prototype x in a set $\mathcal{D}_n$, one needs to calculate **all the distances**, n.
- There are several **generic algorithms** (based on branch and bound techniques) to **accelerate** these searches. Examples include the well-known AESA (Approximation-Elimination Search Algorithm (Vidal, 1986)) and its extensions (Micó et al., 1994, 1996).
- Moreover, if the data is of vector type and the distance satisfies the triangular inequality, there are algorithms like Ball-Tree (Omohundro, 1989) or k-d tree (Bentley, 1975) to accelerate the search.

How to detect problematic samples?

- There are techniques called Editing.
- They remove samples that may be mislabeled or belong to an overlapping region between classes.
- As an example, we have the simple algorithm by Wilson (1972).

Wilson's editing algorithm


```
def Editing(k:int, T:list) -> list: # T: Training set
    S = T.copy() # Edited set
    R = []       # Misclassified set

    for p in S:
        # If p is a misclassified example
        if p.label_class != p.get_classkNN(k, S-{p}):
            R.append(p)      # Remove example

    # Remove all misclassified examples
    S = [p for p in S if p not in R]

    return S
```

Which samples are the most important?

There are simple techniques for condensing samples like CNN (The Condensed Nearest Neighbor rule Hart (1968)) and derivatives.

```
def CNN(k:int, T:list) -> list: # T: Training set
    S = [] # CNN set
    updated = true

    random.shuffle(T)  # Shuffle array elements
    while len(S) < len(T) and updated:
        updated = False
        for p in T:
            # If p is a misclassified example
            if p.label_class != p.get_classkNN(k, S):
                S.append(p)      # Add example
                updated = True

    return S
```

Combining editing and condensing

- The **purpose** of Editing techniques is to remove **mislabelled** samples based on certain criteria, but we won't be sure if we eliminate a correctly labelled prototype. We depend on the training set.
- The **purpose** of Condensing techniques is to remove **redundant prototypes**, but we also cannot verify if these affect the final classification rate, which is what interests us the most.
- If we repeatedly apply the Editing technique until all samples are classified correctly, it is known as Multiedit.
- If we combine both techniques, it is known as Multiedit Condensing.

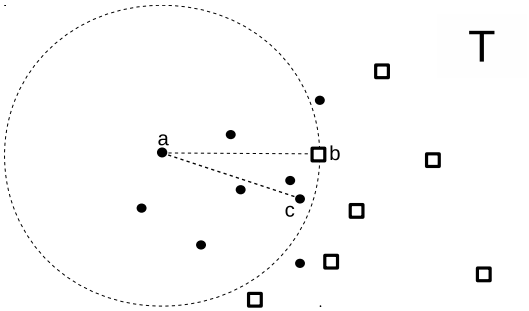
Assessing the importance of each sample

One approach is to measure how important the samples are for classification (Rico-Juan and Iñesta, 2012).

1. 1-FN (Fartest Neighbor): Voting system for the farthest neighbor of the same class before the nearest enemy of a different class.
2. 1-NE (Nearest Enemy): Voting system for the nearest friend that is close to the nearest enemy for correct classification.

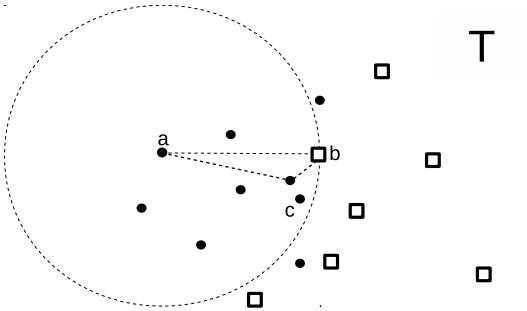
1-FN (Fartest Neighbor)

1. The entire training set would be run through. In this case, a is the prototype currently under examination.
2. b is the nearest enemy to prototype a.
3. c is the farthest prototype from a before b, and its presence allows for correct classification of prototype a.



1-NE (Nearest Enemy)

1. The entire training set would be run through. a is the prototype currently under examination.
2. b is the nearest enemy to prototype a.
3. c is the prototype of the same class as a and nearest to c. With its presence, prototype a would be correctly classified.

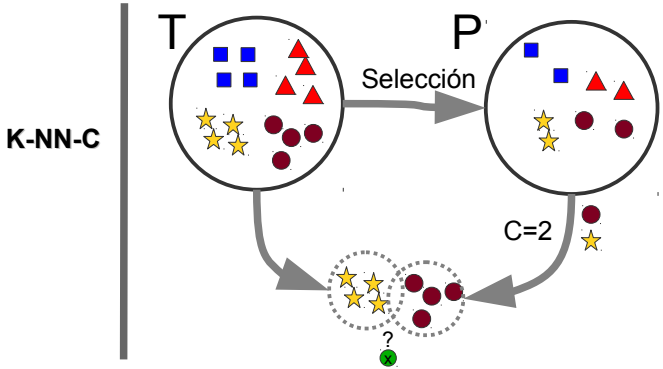


kNNc accelerates classification in two steps I

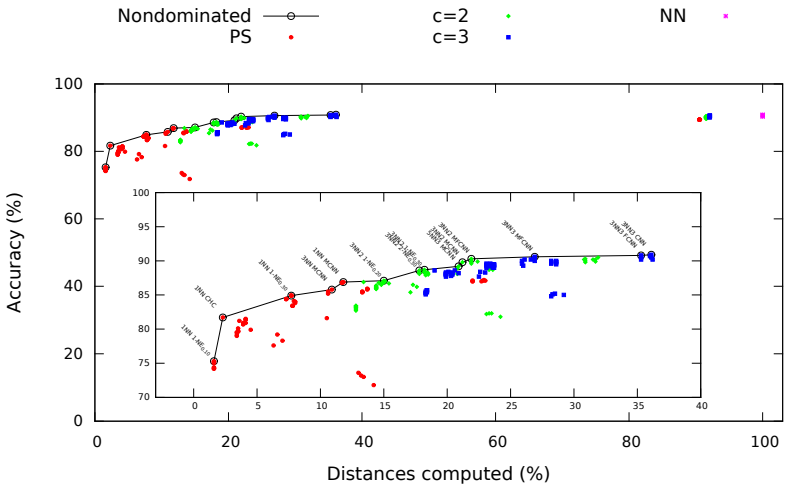
This technique **accelerates** classification and allows **discarding** classes that are very different from the unknown prototype x during classification. This technique is useful for problems with **multiple classes** (Calvo-Zaragoza et al., 2015).
Description:

1. Select a representative set, P, of prototypes from the initial set, T.
2. To classify a new unknown prototype x, request the c nearest classes in P.
3. Use k-NN to classify x only with the c nearest classes.

kNNc accelerates classification in two steps II



kNNc without noise



kNNc with 40% noise

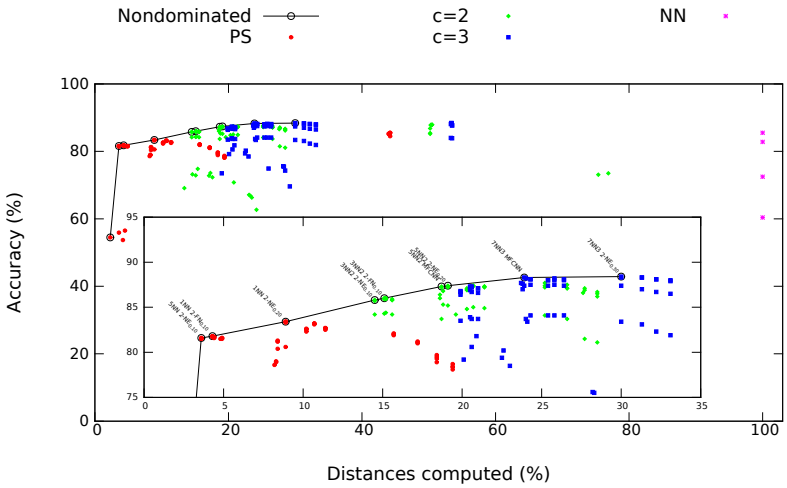


Table of contents

Advantages and disadvantages

Advantages

- It is a simple algorithm, explainable, and its error is bounded by double the Bayes minimum.
- It only needs to use a distance function.
- It can be applied to structured data such as vectors, trees, or graphs.
- It can accelerate neighbor searches.
- Noisy samples can be removed, and the best ones can be selected.
- They are robust to class imbalance.

Disadvantages

- High memory consumption if we use all training samples.
- If we seek the best accuracy or minimize error, there are other more complex algorithms that usually achieve better results.

3. Decision Trees (DT)

Table of contents

Decision Tree (DT)

A **decision tree** (Von Winterfeldt and Edwards, 1993) is a hierarchical diagram in which **nodes** represent states and their possible **decisions** are represented by branches.
In general, it is constructed from a set of **labeled** feature **vectors**. Each **node** in the tree generates **partitions** of itself, forming the next **level**, by following certain criteria (decisions) that minimize some **error estimate**. This process is applied recursively to each node until all elements are exhausted or the desired error is reached.

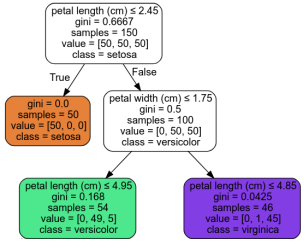


Table of contents

ID3 Algorithm

- **ID3** stands for (Iterative Dichotomiser 3) (Quinlan, 2014).
- For more details, you can check:
  - Wikipedia ID3.
  - How Decision Trees work (Video).
  - Step-by-step ID3 example (Video).

We will now start an example that we will complete in practical classes.

ID3 Application Example

Simplified dataset for COVID-19.

Id	Fever	Cough	Respiratory Problems	Infected
1	No	No	No	No
2	Yes	Yes	Yes	Yes
3	Yes	Yes	No	No
4	Yes	No	Yes	Yes
5	Yes	Yes	Yes	Yes
6	No	Yes	No	No
7	Yes	No	Yes	Yes
8	Yes	No	Yes	Yes
9	No	Yes	Yes	Yes
10	Yes	Yes	No	Yes
11	No	Yes	No	No
12	No	Yes	Yes	Yes
13	No	Yes	Yes	No
14	Yes	Yes	No	No

ID3 Metrics I

- The input variables are **Fever**, **Cough**, and **Respiratory Problems**, and the target variable is **Infected**.
- ID3 selects the best feature at each step while constructing a decision tree.
- Entropy is used to measure disorder:

$$Entropy(S) = -\sum_{i \in N} p_i * log_2(p_i)$$

  where p_i is the probability and S is the categories or classes (Yes/No).
- In our case, entropy is 0 if all values in the target column are homogeneous (similar) and 1 if the opposite occurs.

ID3 Metrics II

- To compute the Information Gain (IG) of each feature (attribute) at each step:

$$IG(S, A) = Entropy(S) - \sum(\frac{|S_v|}{|S|} \times Entropy(S_v))$$

where S_v are the elements of S for each value v belong to A; and $|S_v|$ or $|S|$ is the size of each set.

Example step 1

- In the initial set, S, we have 14 rows.
- In the target variable, **Infected**, we have 8 **Yes** values and 6 **No** values.
- Thus, the initial entropy of S is:

$$Entropy(S) = -(8/14) * log_2(8/14) - (6/14) * log_2(6/14) = 0.99$$

Example step 2

We need to calculate the gain for all possible input variables.

Gain of **Fever**

- In the **Fever** column, we have 8 rows with **Yes** and 6 rows with **No**.
- Among them, the target variable **Infected** is distributed as follows:
  - `Fever = Yes`: 6 `Infected = Yes` and 2 `Infected = No`
  - `Fever = No`: 2 `Infected = Yes` and 4 `Infected = No`

Example step 2 (calculations)

Entropy calculations:

$$|S| = 14$$

$$\text{For } v = \text{Yes}, |S_v| = 8$$

$$Entropy(S_{\text{Yes}}) = -(6/8) * log_2(6/8) - (2/8) * log_2(2/8) = 0.81$$

$$\text{For } v = \text{No}, |S_v| = 6$$

$$Entropy(S_{\text{No}}) = -(2/6) * log_2(2/6) - (4/6) * log_2(4/6) = 0.91$$

Gain calculation:

$$IG(S, \text{Fever}) = Entropy(S) - (|S_{\text{Yes}}|/|S|) * Entropy(S_{\text{Yes}}) - (|S_{\text{No}}|/|S|) * Entropy(S_{\text{No}})$$

$$IG(S, \text{Fever}) = 0.99 - (8/14) * 0.81 - (6/14) * 0.91 = 0.13$$

Example step 2 (calculations)

Gains for the first level of the tree:

$$IG(S, \text{Fever}) = 0.99 - (8/14) * 0.81 - (6/14) * 0.91 = 0.13$$

$$IG(S, \text{Cough}) = 0.04$$

$$IG(S, \text{Respiratory Problems}) = 0.40$$

The highest gain is in $\text{Respiratory Problems}$ with a value of 0.40.

The detailed calculation of $IG(S, \text{Cough})$ and $IG(S, \text{Respiratory Problems})$ is left as an exercise.

Table of contents

Evolution of decision trees I

- The algorithm C4.5, presented by Quinlan (Quinlan, 2014), improved upon ID3 by refining some details in attribute management and using pruning for better generalization.
- C5.0 was the last version released by Quinlan with proprietary license. It used less memory and created smaller rule sets compared to C4.5 while being more accurate.
- CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees recursively using a feature and threshold that produce the highest information gain at each node.
- However, decision trees have a high dependence on data, meaning that making slight changes in the original data can lead to significantly different resulting trees.

Evolution of decision trees II

- One solution is to generate many trees and compute an average solution, as done by Random Forest (Breiman, 2001) or Rotation Forest Rodriguez et al. (2006).
- There are also recent algorithms such as XGB (eXtreme Gradient Boosting) or LightGBM (Light Gradient Boosting Machine) that combine different techniques, such as gradient descent (similar to neural networks), random selection of samples and attributes (rows and columns), along with successive refinements to generate a forest of decision trees.

Table of contents

Advantages and disadvantages

Advantages

- Decisions are easy to understand and interpret; they can also be visualized.
- Requires little data preparation (numeric or categorical).
- The model is fast and efficient in predictions.
- Moreover, it can handle multi-output problems.

Disadvantages

- Decisions become difficult to understand and interpret when dealing with multiple trees (ensemble).
- Tends to overfit.
- Small variations in data lead to very different trees.
- Predictions are in blocks (not continuous), making extrapolation less effective.
- Sensitive to class imbalance.

4. Bayesian Algorithms

Table of contents

Bayesian algorithms

This type of classification algorithms is based on Bayes' theorem (Duda et al., 2001), considering each value as independent of any other. This allows predicting a class or category based on a given set of features using probability. There are also other algorithms based on linear regression, which utilize Bayesian techniques for weight adjustments.

Bayes' theorem

Theorem

The probability of an event A given another event B can be calculated as:

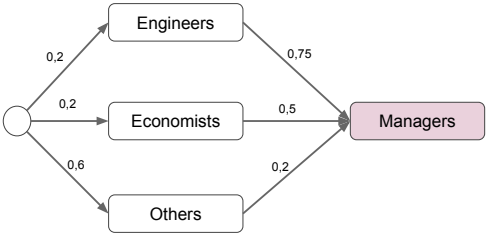
$$P(A|B) = \frac{P(A,B)}{P(B)}$$

where $P(A,B)$ is the probability of both A and B occurring simultaneously, and $P(B)$ denotes the probability of B.

Table of contents

Simple example

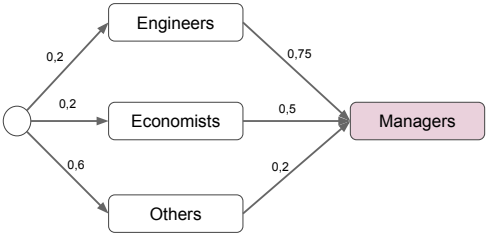
20% of the employees in a company are engineers, and another 20% are economists. Among the engineers, 75% hold manager position, while 50% of the economists also do. The rest of the employees (non-engineers, non-economists) have only a 20% chance of holding a manager position.



What is the probability that a randomly chosen manager employee is an engineer?

Simple example

20% of the employees in a company are engineers, and another 20% are economists. Among the engineers, 75% hold manager position, while 50% of the economists also do. The rest of the employees (non-engineers, non-economists) have only a 20% chance of holding a manager position.



What is the probability that a randomly chosen manager employee is an engineer?

Solution: $P(\text{engineer}|\text{manager}) = \frac{0.2 \cdot 0.75}{0.2 \cdot 0.75 + 0.2 \cdot 0.5 + 0.6 \cdot 0.2} = 0.405$

Table of contents

Naive Bayes for classification

- We assume independence between variables and replace the previous denominator, $P(B)$, with the sum of all the numerators of the classes.
- The formula becomes:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{\sum_j P(y_j)\prod_{i=1}^{n} P(x_i|y_j)}$$

where y is a class of the target variable, $P(y)$ is its probability, and $P(x_i|y)$ is the conditional probability of x_i given y. The solution is the class that achieves the highest probability:

$$y = \arg\max_{y_j} \; P(y_j|x_1, x_2, \ldots, x_n)$$

Classification example with Bayes (I)

To simplify, let's consider the previous dataset on COVID-19 with only one input variable, Fever, and the target variable, Infected.

Id	Fever	Infected
1	No	No
2	Yes	Yes
3	Yes	No
4	Yes	Yes
5	Yes	Yes
6	No	No
7	Yes	Yes
8	Yes	Yes
9	No	Yes
10	Yes	Yes
11	No	No
12	No	Yes
13	No	No
14	Yes	No

Classification example with Bayes (II)

$P(\text{Infected})$

- $P(\text{Infected} = \text{Yes}) = 8/14 = 0.57$
- $P(\text{Infected} = \text{No}) = 6/14 = 0.43$

$P(\textit{Fever}|\text{Infected})$

- $\text{Infected} = \text{Yes}$
  - $P(\text{Fever} = \text{Yes}|\text{Infected} = \text{Yes}) = 6/8 = 0.75$
  - $P(\text{Fever} = \text{No}|\text{Infected} = \text{Yes}) = 2/8 = 0.25$
- $\text{Infected} = \text{No}$
  - $P(\text{Fever} = \text{Yes}|\text{Infected} = \text{No}) = 2/6 = 0.33$
  - $P(\text{Fever} = \text{No}|\text{Infected} = \text{No}) = 4/6 = 0.67$

Classification example with Bayes (III)

Input $X = (\text{Fever} = \text{Yes})$

- $P(\text{Infected} = \text{Yes}|\text{Fever} = \text{Yes}) = \frac{0.57 \cdot 0.75}{0.57 \cdot 0.75 + 0.43 \cdot 0.33} = \frac{0.43}{0.57} = 0.75$
- $P(\text{Infected} = \text{No}|\text{Fever} = \text{Yes}) = \frac{0.43 \cdot 0.33}{0.57 \cdot 0.75 + 0.43 \cdot 0.33} = \frac{0.14}{0.57} = 0.25$

Note: An example with all the variables of the COVID-19 dataset will be studied in the face-to-face sessions.

Table of contents

Advantages and disadvantages

Advantages

- Provides a natural way to combine information through probability distributions (Bayes' theorem).
- Gives conditional probabilities without relying on asymptotic approximations.
- Follows the principle of likelihood (that the samples come from a model).
- Provides interpretable answers with their probability and confidence.
- Widely used in language models (word distributions).

Disadvantages

- Need to design how to estimate the prior and conditional probabilities.
- Can produce posterior distributions heavily influenced by the priors.
- Depending on the number of samples and parameters, it can be computationally expensive.

5. Support Vector Machines

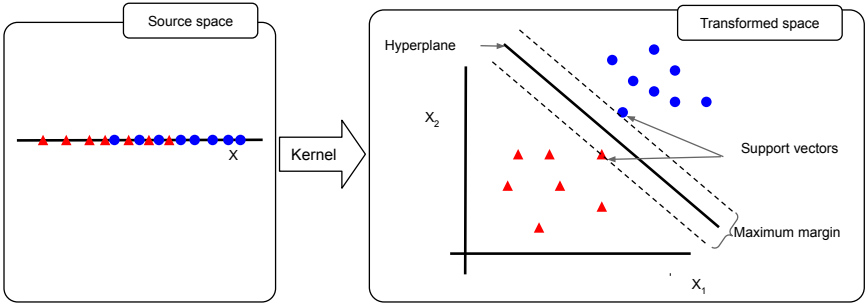
Table of contents

Support Vector Machines

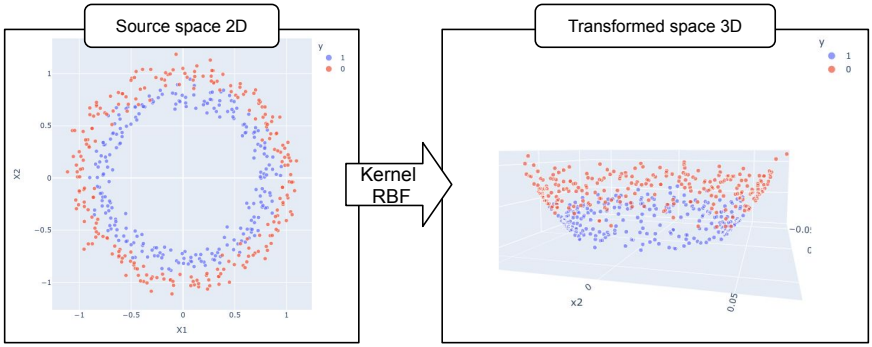
Support Vector Machine (SVM) is a technique initially designed for binary classification. It consists of two steps: the first step called kernel trick involves **transforming the original data** (n-dimensional) where they are not linearly separable into another dimension, usually higher, where they become separable (these functions are known as kernels); the second step calculates a **hyperplane** that is as far as possible from both classes. This technique was initially developed by Vladimir Vapnik (Cortes and Vapnik, 1995).

Table of contents

Generic example



Example with Radial Basis Function (RBF) kernel



Kernel functions and hyperplane search

As we mentioned, SVM has two phases:

Phase 1 - Feature transformation (kernel trick): Most commonly used kernel functions are:

- Polynomial: $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$
- Radial Basis Function (RBF): $k(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$ where $\gamma > 0$

Phase 2 - Search for the class-separating hyperplane. Loss functions:

- Hard-margin: Minimize $\|w\|$ such that $y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$ for $i = 1, \ldots, n$
- Soft-margin: $[\frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))] + \lambda \frac{1}{2} \|\vec{w}\|^2$

Table of contents

Advantages and disadvantages

Advantages

- Effective in high-dimensional spaces.
- Remains effective when the number of dimensions exceeds the number of samples.
- Uses a subset of training points in the decision function (support vectors), making it memory-efficient.

Disadvantages

- If the number of features is much greater than the number of samples, the choice of kernel is crucial.
- SVM does not directly provide probability estimates, and its estimation is computationally expensive.

6. Artificial Neural Networks (ANN)

Table of contents

Artificial Neural Networks (ANN)

Artificial Neural Networks known by their English acronym ANN, they are inspired by the **neural networks** of living organisms.
They are generally used in interconnected systems of neurons (layers). The connections have numeric weights that are adjusted with training data.
The first networks were created in the 1940s, and since then, the learning mechanisms and typologies have evolved to become one of the most well-known learning styles.

Biological cell vs artificial neuron

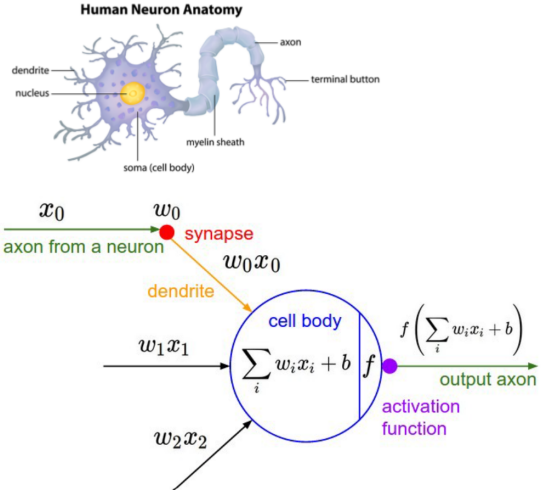
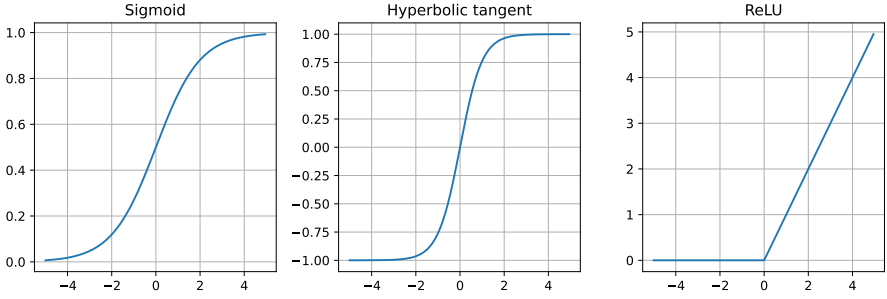


Table of contents

Activation Functions I

Types

- Sigmoid or Logistic: $f(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic tangent: $f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- Rectified Linear Unit (ReLU): $f(x) = max\{0, x\}$



Activation Functions II

What makes these functions special?

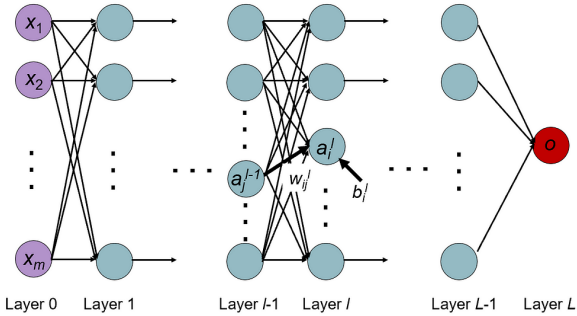
And what role do they play?

Activation Functions III

- These functions are used to normalize the values calculated between layers and neurons and introduce **non-linearity** into the model.
- In this way, they allow the model to **learn complex relationships and patterns** in the data.

Table of contents

Multilayer Perceptron (MLP)



- a_i^l is the activation (output) of neuron i in layer l;
- w_{ij}^l is the weight of the connection from neuron j in layer $l-1$ to neuron i in layer l;
- b_i^l is the bias term of neuron i in layer l.

Example extracted from Multi-Layer Perceptrons Explained and Illustrated.

Forward propagation

- Input data (features) are introduced into the network, and operations are applied layer by layer to generate the output.
- For example, the activation of neuron i in layer l is calculated as:

$$a_i^l = f(\sum_j w_{ij}^l a_j^{l-1} + b_i^l)$$

- In the initial layer with m features:

$$a_i^0 = x_i : 1 \leq i \leq m$$

Backpropagation

- During training, given **pairs** (input, output) $= (X, y)$, we estimate $P(X) = \hat{y}$ for several examples (forward propagation).
- A **loss function** is applied to the output to measure the difference between the true values and the estimated values, $loss(y, \hat{y})$.
- The **gradients** are calculated with respect to the losses, activation functions (**differentiable**), and weights of the layers. These gradients inform us whether to *increase or decrease* the individual weights of each neuron to **minimize the final loss** of the network.
- The weight adjustments are **propagated from the output to the input** (backpropagation), adjusting the weights in each layer to minimize the loss in each case.
- You can see an example in the video What is backpropagation and what does it actually do?

Considerations

- **Networks like MLP** are applied to problems where **temporal components are not involved**. Examples:
  - **OCR** (classification of numbers and letters) where numbers and letters do not change over time;
  - **determining the presence of a disease** based on certain patient analysis inputs.
- If the input has **temporal information** like:
  - **stock market investments** analyzing past day's quotations;
  - **sentiment analysis** in a text, analyzing words in order;
  - **scene description** in a video, analyzing sequences of images in an ordered manner.

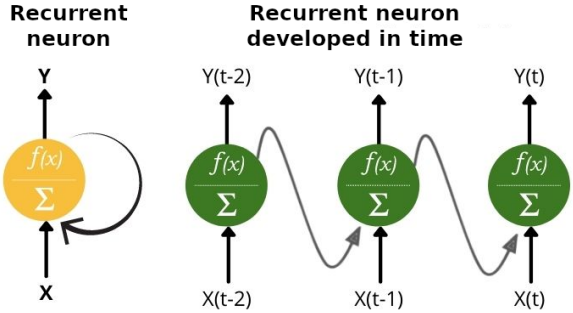
  in these latter cases, we need a special type of neural network.

Table of contents

Recurrent neural networks

Recurrence

- Useful for handling **temporal sequences**.
- The output of one or more neurons at time t is used as input for others at time $t + 1$.



Network parameters

- The recurrent network has two parameters: the input sequence ($X = (X_1, \ldots, X_T)$) and the output from the previous time step (Y_{t-1}). It can be represented as:

$$Y_t = f(W\ x_t + U\ Y_{t-1} + b)$$

where W represents the weights of the matrix at the current time step, U represents the weights of the state from the previous time step, and b is the bias term.

Short-term memory problem

- The memory of these types of networks is limited, and transmitting information between distant sequences is a problem.
- This phenomenon is known as *gradient vanishing*.
- One solution is to incorporate internal memory into each neuron of the network.

Long-term memory

The two most important types of networks are *Long-Short Term Memory* (LSTM) and *Gated Recurrent Units* (GRU).

LSTM was introduced by Hochreiter and Schmidhuber (1997). It has three inputs: the **current input state**, the **short-term memory of the previous cell**, and the **long-term memory**. It also has **three gates**: **input gate**, **forget gate**, and **output gate**, which allow it to retain relevant information and discard irrelevant information.

GRU similar to LSTM, it has internal memory, but it **simplifies its operation structure**, making it consume **fewer resources** and be **faster**. Introduced by Cho et al. (2014), it uses **two operation mechanisms**: **update gate** and **reset gate**.

Comparison of schemes

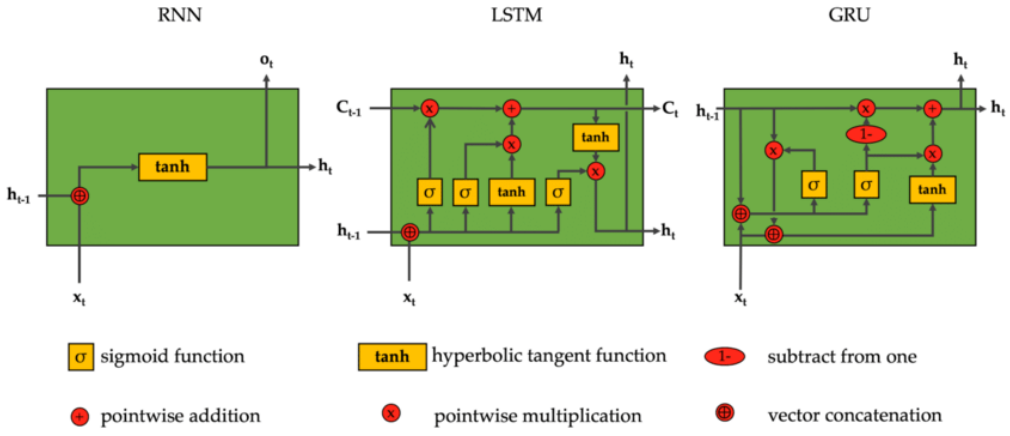


Table of contents

Advantages and disadvantages

Advantages

- Training and prediction operations are highly parallelizable.
- They are robust in handling noise in samples or errors in some neurons.
- Superior prediction capability when a large amount of data is available compared to other learning styles.
- Able to solve complex problems with unstructured data, such as images, audio, video, or text.

Advantages and disadvantages

Disadvantages

- High computational cost during the training phase (typically requires the use of GPUs).
- The optimal configuration for addressing a problem cannot be determined.
- The final network weights differ for each training (from the same start point) and with the same samples (pseudo-random mechanisms).
- Highly sensitive to preprocessing of input data.
- Predictions are difficult to understand and explain.
- Results similar to other algorithms when the data is tabular.

7. Ensembles

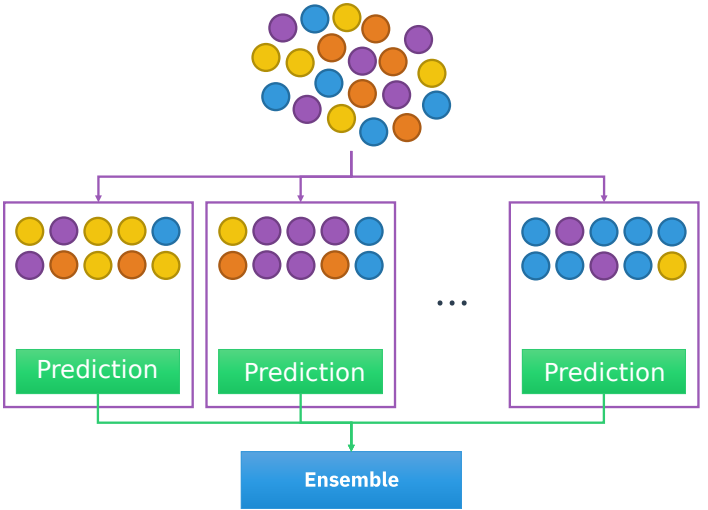
Table of contents

Table of contents

Bagging

- The term **Bagging** comes from **Bootstrap aggregating**.
- These methods use predictions from other (weak) models in **parallel**.
- These predictions are then subjected to a mechanism of simple or weighted **average** to obtain their **final prediction**.
- The intention is to reduce variance and improve the final prediction.
- Algorithms known as Random Forest (Breiman, 2001) use this technique to create a model.

Bagging scheme



Original Data

Bootstrapping

Aggregating

Bagging

Advantages and disadvantages

Advantages

- Predictions of the aggregated base models often outperform those of a model applied to the entire dataset.
- Reduces the variance that a base model with low bias may have.
- Can be processed in parallel.

Disadvantages

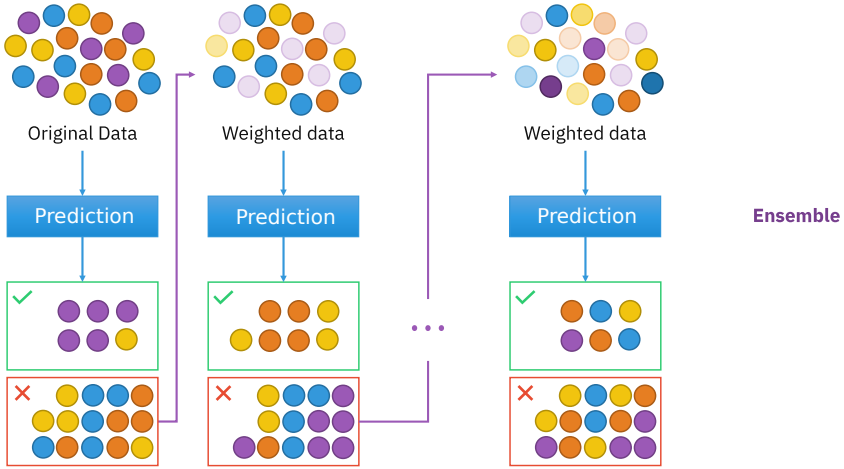
- If a base model has high bias, it will also be aggregated into the final prediction.
- There is a loss of interpretability of the final model.
- The predictions have a higher computational cost than those made with a single base model.

Table of contents

Boosting

- These methods use predictions in a **sequential** manner.
- These predictions are used to **adjust the model's parameters**.
- The intention is to start with an **initial (weak) model** with poor predictions and progressively make **adjustments** to end up with a **robust model** with better predictions.
- Well-known algorithms that use this technique are Adaboost (Freund and Schapire, 1997), XGBoost (Chen and Guestrin, 2016), or LightGBM (Ke et al., 2017).

Boosting Scheme



Advantages and disadvantages

Advantages

- It can significantly improve the predictions of a base model, meaning it can produce better results.
- In the case of imbalanced data, it allows for better adjustment of predictions for the minority class.
- When dealing with data with little noise, it attenuates its effect on the final predictions.

Disadvantages

- It is sensitive to overfitting, which can lead to generalization problems.
- It requires more time and resources since it involves training multiple models sequentially and combining them.
- When dealing with data with a lot of noise, the effect of successive adjustments can harm the quality of the predictions of the final model.
- The model is difficult to interpret.

8. Validation of model

Table of contents

Validation of model

Cross-validation is a technique commonly used to evaluate the results of a **statistical analysis** and **ensure** that they are **independent** of the chosen partition (training or testing) (Duda et al., 2001).

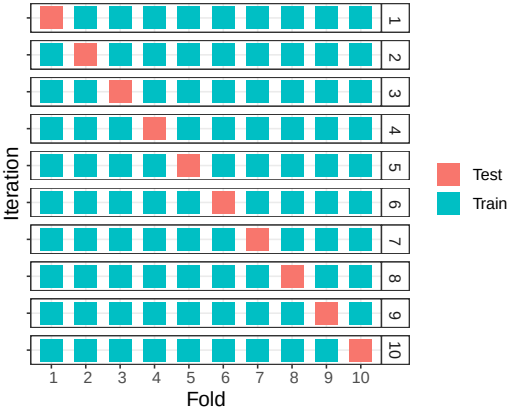
Characteristics

- It consists of calculating the **arithmetic mean** obtained from evaluation measures on different **partitions**.
- It is used in environments where the main objective is **prediction** and one wants to estimate a set of **metrics** for a **model** that will be used in practice.
- It is a technique **widely used** in artificial intelligence projects to validate **models**.

Example: 10-fold cross-validation

- Currently, the 10-fold CV scheme is the most widely used.
- In each iteration, each test set obtains an estimation, $\{e_1, e_2, \ldots, e_{10}\}$
- The final estimation would be their average:

$$e = \frac{\sum_i e_i}{10}$$



Special case: leave-one-out cross-validation

- Cross-validation, also known as **leave-one-out**, involves separating the data (N) such that we have 1 for testing and $N - 1$ for training.
- Up to N iterations can be performed.
- The **error** committed in evaluation is very **low**, at the expense of a **high computational cost**.

9. Comparing models

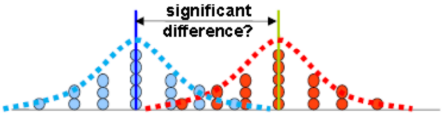
Table of contents

Comparing models

The **comparison** between **metrics** obtained from various **models** is necessary to ensure with a certain confidence that the results of one classifier are significantly better than those of another.

Null hypothesis

The null hypothesis is a statement of no effect or no difference. The experimenter expects the results to be rejected.



Example of two normal distributions

Confidence level α

- It is a confidence threshold that informs us whether the null hypothesis should be rejected or not.
- It must be determined by the experimenter, and common values are: 90% (0.1), 95% (0.05), which is the most commonly used, and 99% (0.01).
- If a confidence level of 0.05 (95% certainty) is chosen, we need a p value (provided by the test) less than 0.05 to reject the null hypothesis. In other words, there is a significant difference between the results.

Statistical tests

Parametric vs. Non-parametric: The former require the statistical distribution with its parameters as a basis, while the latter only contrast results.

Number of variables	**Parametric**	**Non-parametric**
2	t-test	**Wilcoxon test** Sign test
> 2	ANOVA	Friedman test, Iman-Davenport
	Turkey, Dunnet	**Holm**, Hochberg, Bonferroni-Dunn

More on statistical tests

- Slides with more detailed information about statistical analysis of results: Statistical Analysis of Experiments.
- One way to calculate statistical tests is to use Wilcoxon test (e.g., with Google Colab).

Referencias I

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Calvo-Zaragoza, J., Valero-Mas, J. J., and Rico-Juan, J. R. (2015). Improving knn multi-label classification in prototype selection scenarios using class proposals. *Pattern Recognition*, 48(5):1608–1622.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *CoRR*, abs/1603.02754.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, IT-13:21–27.

Referencias II

Craw, S. (2010). Manhattan Distance. In Sammut, C. and Webb, G. I., editors, *Encyclopedia of Machine Learning*, page 639. Springer.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification (2nd Edition)*. Wiley-Interscience Publication.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Gupta, L., Molfese, D., Tammana, R., and Simos, P. (1996). Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43(4):348 – 356.

Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3149–3157.

Referencias III

Kubicka, E., Kubicki, G., and Vakalis, I. (1990). Using graph distance in object recognition. In *Proceedings of the 1990 ACM annual conference on Cooperation*, CSC '90, pages 43–48, New York, NY, USA. ACM.

Micó, L., Oncina, J., and Carrasco, R. C. (1996). A fast branch & bound nearest neighbour classifier in metric spaces. *Pattern Recognition Letters*, 17(7):731–739.

Micó, L., Oncina, J., and Vidal, E. (1994). A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, 15(1):9–17.

Omohundro, S. M. (1989). *Five balltree construction algorithms*. International Computer Science Institute Berkeley.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Rico-Juan, J. R. and Iñesta, J. M. (2012). New rank methods for reducing the size of the training set using the nearest neighbor rule. *Pattern Recognition Letters*, 33(5):654–660.

Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630.

Referencias IV

Silverman, B. W. and Jones, M. C. (1989). E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951). *International Statistical Review / Revue Internationale de Statistique*, 57(3):233–238.

Vidal, E. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, 4:145–157.

Von Winterfeldt, D. and Edwards, W. (1993). Decision analysis and behavioral research.

Wagner, R. and Fischer, M. (1974). The String-to-String Correction Problem. *Journal of the ACM*, 21(1):168–173.

Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421.

Zhang, K. and Shasha, D. (1989). Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM J. Comput.*, 18(6):1245–1262.

Supervised learning

Profesor: Juan R. Rico

Técnicas de Aprendizaje Automático

Dpto. Lenguajes y Sistemas Informáticos. Universidad de Alicante