"UniTs" - University of Trieste

Faculty of Data Science and Artificial Intelligence

Department of mathematics informatics and geosciences

Introduction to Machine Learning

Lecturer:
Prof. Eric Medvet

Author:
Christian Faccio

January 1, 2025

github.com/christianfaccio christianfaccio@outlook.it

Abstract

As a student of the “Data Science and Artificial Intelligence” master’s degree at the University of Trieste, I have created these notes to study the course “Introduction to Machine Learning” held by Prof. Eric Medvet. The course aims to provide an introduction to the field of machine learning, covering the fundamental concepts in theoretical terms. The topics are the followig:

- Supervised learning techniques
- Unsupervised learning techniques
- Dimensionality reduction (not covered in the lectures)
- Introduction to NLP

While these notes were primarily created for my personal study, they may serve as a valuable resource for fellow students and professionals interested in this field.

Contents

1 Basic Concepts

What is Machine Learning?

Definition: *Machine Learning*

Machine Learning is the science of getting computers to learn something without being explicitly programmed.

- What science?
- Who is doing that?
- To learn what?
- Who is not being explicitly programmed?

Let's formalize the decision making process:

$$y = f(x)$$

- x is the entity about which the decision has to be made
- y is the decision
- f is the procedure that, given an "x" results in a decision "y"

X is an **observation**, while Y is a **response**.

So, the function f is what has to be learned, and is also denoted as $f_{predict}$, since it predicts a response ($\hat{y}$) given an observation.

$$f : R^2 \to R$$

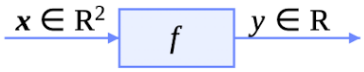

Figure 1.1: Abstract definition of the learning process

The learning process is not written by a human, so we can say that the machine learns autonomously.

- condsider the universe of functions $F_{X \to Y} = \{f, f : X \to Y\}$
- choose the function that does what expected

Definition: *New Definition*

Machine Learning is the science of getting computers to learn $f_{predict} : X \to Y$ **autonomously**.

There has to be some sort of supervision facilitating the search of a good f and when it is in the form of some **examples** and the function should process them correctly we are in the field of **Supervised Machine Learning**.

- **Supervised Learning**: the function is learned from a set of labeled examples
- **Unsupervised Learning**: finds structures in the data without any labels
- **Reinforcement Learning**: the function is learned from a set of examples and a reward signal

Examples are pairs of observations and responses (x, y), and the set of examples is called **training set** (or learning set). The dataset is a bag of pairs of observations and responses.

The learning set has to be consistent with the domain and codomain of the function $f_{predict}$ to be learned.

$$f_{predict} = f_{learn}(D_{learn})$$

- **learning phase**: the function f_{learn} is learned from the learning set(i.e. is applied to obtain $f_{predict}$ from D)
- **prediction phase**: the function $f_{predict}$ is used to predict the response (y) of new observations

Definition: *Supervised Learning Technique*

A **Supervised Learning Technique** is a way to learn an $f_{predict} \in F_{X \to Y}$ given a $D_{learn} \in P^*(X \times Y)$

Different SL techniques differ in applicability, efficiency and effectiveness.

- **Applicability**: with respect to X andor Y, e.g. some require $X = R^p$, some require $Y = R$
- **Efficiency**: the computational resources needed to learn the function wrt $|D_{learn}|$, e.g. some are fast and other slow
- **Effectiveness**: the quality of the learned functions $f_{predict}$

There is another view of it, and I like it more:

Definition: *SL Technique as Optimisation*

A **Supervised Learning Technique** is a way to solve the optimization problem:

$$f_{predict} = argmin_{f \in F_{X \to Y}} L(f, D_{learn})$$

where L is a loss function that measures the quality of the function f wrt the learning set D_{learn} and $F_{X \to Y}$ is the set of functions that can be learned.

The most practical solution is to reduce $F_{X \to Y}$ size by considering only the f of some nature.

Templating f means defining a relationship between data as we do in statistics, specifying it with some coefficients that we can evaluate.

We can then explicit the unknown parameters and specify the template as a reduced F'.

$$f_{predict} = f'_{predict}(x, m)$$

Where m is the model of how y depends on x.

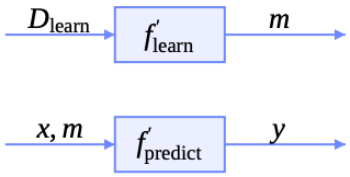

Figure 1.2: Learning a model with template function

Definition: *ML system*

An **information processing system** in which there is a SL technique and a pair of **pre-processing** and **post-processing** steps.

The designer of the ML system has to choose the learning technique, the pre/post processing steps and the relative parameters. The steps are:

- Decide if to use a ML system
- Supervised vs Unsupervised vs Reinforcement
- Define the problem (X, Y, way of assessing solutions)
- Design the ML system (choose the ML technique, pre/post processing steps)
- Implement the ML system (learning/prediction phases, obtaining the data)
- Assess the system

2 Supervised Learning

2.1 Assessment

The main axes of assessment are:

- Effectiveness: how well the model performs on the test set.
- Efficiency: how much time and resources are needed to train the model (achieving the goal).
- Interpretability: how well the model can be understood by humans.

There are two purposes for assessment:

- **Absolute Assessment:** does something meet the expectation with respect to a determined axis?
- **Comparison:** is one thing better than one other thing in terms of the determined axis?

If the output of the assessment is a quantity, than a simple check for < or > is enough, so we want a number as output.

A **ML system** can be seen as a composite learning technique. It has two running modes: one in which it tunes itself and the other in which it makes decisions. The goals are:

- **Training:** tune the model to minimize the error on the training set (tuning properly).
- **Testing:** evaluate the model on the test set (making good decisions).

The SLT has then this two goals, while a **model** has the only goal to make good decision when used in an $f'_{predict}$.

To measure the **effectiveness** of a model and to have a number as output, we need to compare our system with the real system that generated the data (assuming there is one).

Figure 2.1: Model vs Real System

Definition: *Model vs Real System*

- Collect examples of **s** behavior
- Feed **m** with examples
- Compare responses of **s** and **m**

Effectiveness: to which degree the comparison step measures if m behaves like s.

Warning: $f_{collect}$

The data collection is really important, since:

- small n → **poor** effectiveness, **great** efficiency
- large n → **good** effectiveness, **poor** efficiency

(effectiveness = accuracy, efficiency = resources)
and also:

- poor coverage → **poor** effectiveness
- good coverage → **good** effectiveness

where coverage = how well the data represents the real system.

Figure 2.2: Comparing the responses

where $(y^{(i)}, \hat{y^{(i)}})_i \in P^*(Y^2)$ is a multiset of pairs of y.

Example: ***Performance Indexes***

Classification:

- all types → error, accuracy
- binary → EER, AUC, FPR, FNR and variants
- multi-class → weighted accuracy

Regression:

- MSE, MAE, R2

2.1.1 Classification

In Classification Y is a finite set with no ordering.

The **Classification Error** is then

$$f_{err}(y^{(i)}, \hat{y^{(i)}}) = \frac{1}{n} \sum_{i=1}^{i=n} \mathbf{1}(y^{(i)} \neq \hat{y^{(i)}})$$

where **1(b)** is the indicator function and takes value 1 when (in this case) the prediction is wrong.

The **Classification Accuracy** is instead:

$$f_{acc}(y^{(i)}, \hat{y^{(i)}}) = \frac{1}{n} \sum_{i=1}^{i=n} \mathbf{1}(y^{(i)} = \hat{y^{(i)}})$$

Reminder: $f_{acc} = 1 - f_{err}$

Extreme cases for accuracy (and errors) are:

- m is s: **perfect model**
- m is random: does not model any dependence

Example: *Random Classifier - Lower Bound*

$f_{random}(x) = y_i$ with $i\ U(1,\ldots,|Y|)$ just pick random uniformly

Example: *Dummy Classifier - Better Lower Bound*

$f_{dummy}(x) = argmax\frac{1}{n}\sum_{i=1}^{i=n}\mathbf{1}(y = y^{(i)})$ just pick the category with more example always

Example: *Perfect Classifier - Upper Bound*

$f_{perfect}(x) = s(x)$

But the world is not deterministic, so a Bayes (stochastic) classifier may be better:

Example: *Bayes Classifier - Better Upper Bound*

$f_{Bayes}(x) = argmaxPr(s(x) = y|x)$

2.1.2 Binary Classification

The main problem here are the skewed datasets, meaning the datasets with more example of a category than the other (we are considering only two categories here).

Definition: *FPR - False Positive Rate*

It is the rate of **negatives** that are wrongly classified as positives
$FPR = \frac{FP}{N}$

Definition: *FNR - False Negative Rate*

It is the rate of positives that are wrongly classified as negatives. $FNR = \frac{FN}{P}$

Where

- FP = False Positives
- FN = False Negatives
- P = Total positives
- N = Total Negatives

		Test Result	
		0	1
Ground Truth	0	TN (True Negative)	FP (False Positive)
	1	FN (False Negative)	TP (True Positive)

Figure 2.3: Confusion Matrix

TPR: True Positive Rate = 1 - FNR

FPR: False Positive Rate = 1 - TNR

Err: Error Rate = $\frac{FP+FN}{P+N}$

Acc: Accuracy = $\frac{TP+TN}{P+N}$

Definition: ***Balanced Data***

In classification, a dataset is **balanced** with respect to the response variable y, if the frequency of each value of y is roughly the same.

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

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

F1: $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

Sensitivity: $\frac{TP}{TP+FN} = TPR$

Specificity: $\frac{TN}{TN+FP} = TNR$

Type I error: False Positive Rate = FPR

Type II error: False Negative Rate = FNR

Warning: ***Cost of the error***

Once $f_{predict}$ outputs a y, some action is taken, and if the action is wrong, there is some cost to be paid wrt the correct action.
$c = c_{FP} \times FPR \times N + c_{FN} \times FNR \times P$
If one knows the costs and N,P, it is possible to compute the overall cost and find a good trade-off for FPR and FNR.

Given a model, we can ”tune” it to prefer avoiding FPs rather than FNs, just by modifying the threshold that is used to determine the choice to be made wrt the category. Many learning techniques compute a probability distribution over Y before returning one y, and in the case of Binary Classification the threshold is 0.5.

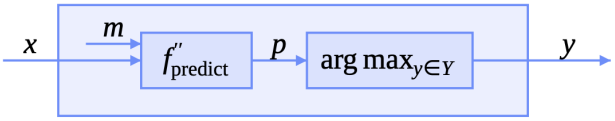

Figure 2.4: Threshold

So, by varying the threshold we can change FPR and FNR:

- the greater the threshold, the lower FPR and the greater FNR
- the lower the threshold, the greater FPR and the lower FNR

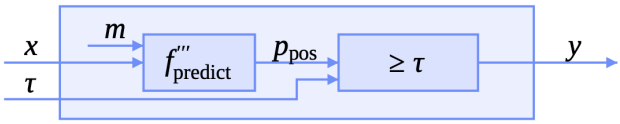

Figure 2.5: Threshold

Definition: *ROC Curve (Receiver Operating Characteristic)*

The ROC curve is a graphical representation of the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) for every possible threshold.

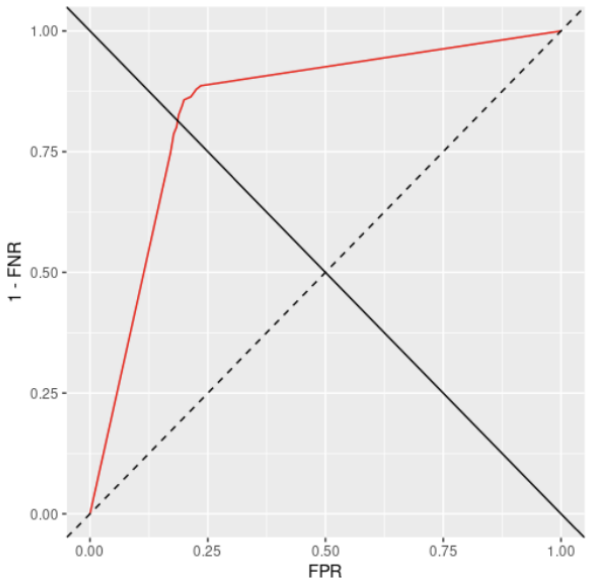

Figure 2.6: ROC Curve

The dotted line represents the random classifier, while the closer the curve is to the top-left corner, the better the classifier is.

How to choose the threshold? Take different values and compute the ROC curve, then choose the one that is closer to the top-left corner. To do this, the best way is to take the **midpoints** of the sorted probabilities.

2.1.3 Multiclass Classification and Regression

Here we use the concept of **weighted accuracy**, which is

$f_{acc} = \frac{1}{|Y|} \sum_{y \in Y} Acc_y$

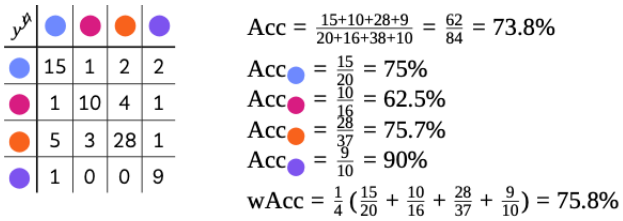

Figure 2.7: Multiclass Confusion Matrix

Moreover, the error there is different from the classification one: here we measure the distance from the real value.

- MAE: Mean Absolute Error = $\frac{1}{n} \sum_{i=1}^{i=n} |y^{(i)} - \hat{y^{(i)}}|$
- MSE: Mean Squared Error = $\frac{1}{n} \sum_{i=1}^{i=n} (y^{(i)} - \hat{y^{(i)}})^2$
- RMSE: Root Mean Squared Error = $\sqrt{MSE}$
- MAPE: Mean Absolute Percentage Error = $\frac{1}{n} \sum_{i=1}^{i=n} \frac{|y^{(i)} - \hat{y^{(i)}}|}{y^{(i)}}$

2.1.4 Assessing Learning Techniques

- an **effective** learning technique is a pair of $f'_{learn}, f'_{predict}$ that learns a good model m
- a good **model** is one that has the same behavior of the real system s

We, then, want to measure the effectiveness of the learning techniques (learning and prediction). To do so, we need to divide the dataset, since we want to see if the $f'_{predict}$ generalises well.

Definition: *Static Train/Test Division*

- **Effectiveness:** generalization is assessed, but there is no robustness wrt D division
- **Efficiency:** learning is executed only once

D_{test} is the unseen data and we treat is as said even if it has been collected at once with D_{train}.

Definition: *Repeated random train/test division*

- **Effectiveness:** generalization is assessed, and there is robustness wrt D division
- **Efficiency:** learning is executed multiple times $\propto$k

Definition: *Cross-Validation*

- **Effectiveness:** generalization is assessed, and there is robustness wrt D division
- **Efficiency:** learning is executed multiple times $\propto$k

Definition: *Leave-One-Out*

- **Effectiveness:** generalization is assessed, and there is robustness wrt D division
- **Efficiency:** learning is executed multiple times ∝n

This is the worst case for efficiency, since we repeat the learning phase n times.

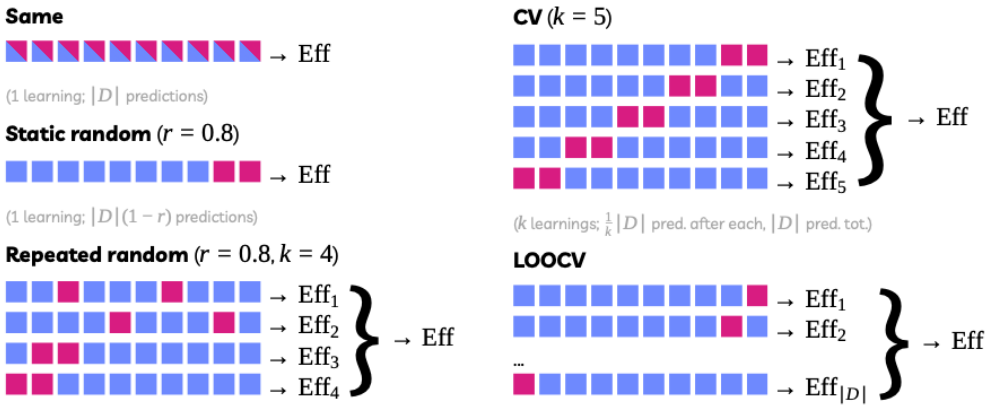

Figure 2.8: Train/Test Division Methods

For comparison btw learning techniques, we can use the mean and standard deviation of the performance indexes obtained with the previous division methods. **Boxplots** are useful in this case.

2.2 Tree-based Learning Techniques

2.2.1 Decision Trees

A **Decision Tree** is a tree where each node is a decision on the value of a feature, and each branch is a possible value of the feature. The leaves are the categories. It can be understood in simpler terms by considering a basic if-then-else nested structure.

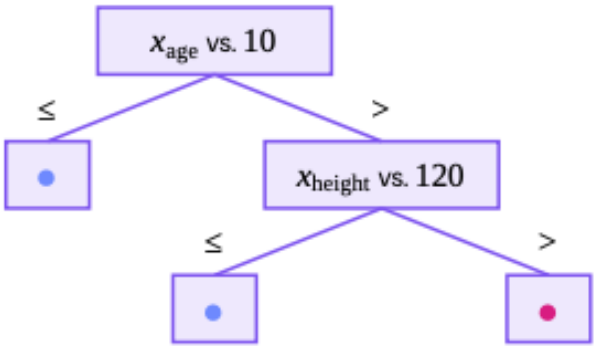

Figure 2.9: Decision Tree example

This is a binary tree, but it can be n-ary as well. The tree is built by recursively splitting the dataset into subsets, and the splitting is done by selecting the feature that best splits the dataset into the purest subsets. The purity is measured by the Gini Index or the Entropy, usually.

One can also represent the tree in a compact way:

$$t = [l, t', t'']$$

where $l \in L$ is the **label** and $t', t'' \in T_L \cup \{\emptyset\}$ are the left and right children.

Also, each **non-terminal** node is a pair (j, τ), where $j \in \{1, \ldots, p\}$ is the index of the independent variable and $\tau \in R$ is the threshold for comparison.

The above figure is then represented as:

$$t = [(1, 10); [\bullet]; [(2, 120); [\bullet]; [\bullet]]]$$

Figure 2.10: Compact representation of the Decision Tree