Data Science

Marc Tommasi

November 16, 2023

Outline

Outline

Main measures

- Taux d'erreurs (accuracy) : the ratio of the number of wrongly classified examples and the total number of examples.
- Confusion Matrix : number of examples according to prediction and true class.

	Positifs	Negatifs
Prediction Positive	TP	FP
Prediction Negative	FN	TN

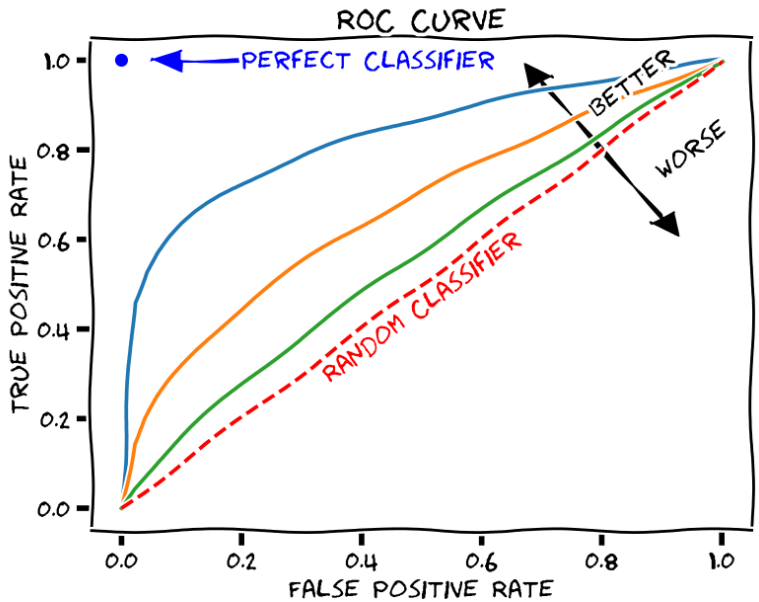
- Precision ($\frac{\mathrm{TP}}{\mathrm{TP+FP}}$) and Recall ($\frac{\mathrm{TP}}{\mathrm{TP+FN}}$)
- The harmonic mean between precision and recall is the F1 score and gives a unique score. ($2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision+Recall}}$)
- Other important measures are the True Positive Rate (TPR) and the False Positive Rate (FPR) ($\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ et $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$)
- Read the Wikipedia page !!!

Decision function

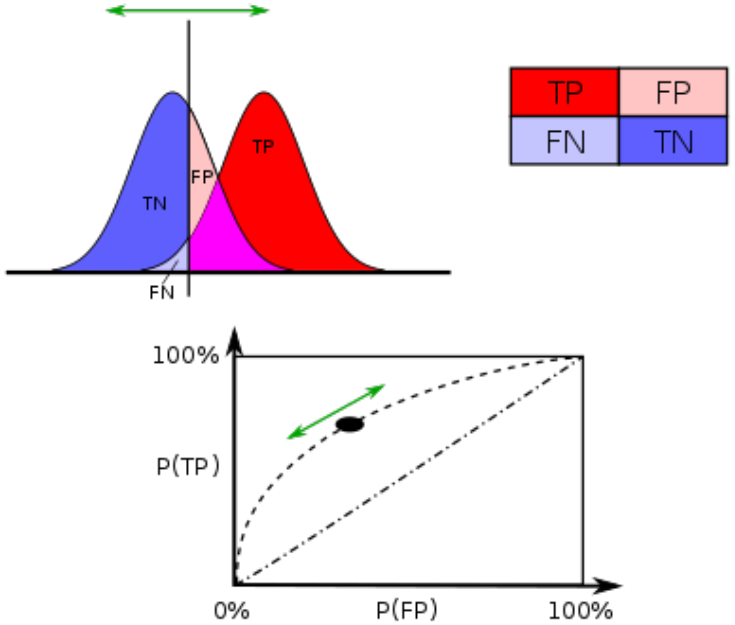
- To obtain predictions, a score is calculated and then the score is compared to a threshold.
- The threshold can be changed and seen as a hyperparameter

ROC curves

- La courbe ROC (Receiver operating characteristic) trace FPR et TPR dans un diagramme en fonction du seuil.



- When the output of the classifier is considered as a random variable



- The area under the ROC curve can be interpreted as the probability of the classifier giving a higher score to a positive example than to a negative example.

In the case of multiclass classification

- The measures also work in a multi-class context, with certain restrictions.
- FPR, TPR, etc are calculated per class
- Averages are calculated across classes
- The average is not a good indicator
- In real problems, macro or micro average techniques can be used to correct the bias introduced by the mean
- For example
  - Class A: 1 TP and 1 FP; Class B: 10 TP and 90 FP; Class C: 1 TP and 1 FP; Class D: 1 TP and 1 FP.
  - macro-average: $(0.5 + 0.1 + 0.5 + 0.5)/4 = 0.4$
  - micro-average: $(1 + 10 + 1 + 1)/(2 + 100 + 2 + 2) = 0.123$
- We can also weight the averages by the proportions in each class.

With sklearn

- Functions are listed in `sklearn.metrics`
  - `confusion_matrix`
  - `precision_score`, `recall_score` et `f1_score`,
  - `classification_report` : un tableau avec les principales mesures
- Most of the classifiers have a method `decision_function` or `predict_proba`
- You can draw curves with `precision_recall_curve` and `roc_curve` and the confusion matrix with `plot_confusion_matrix`, `plot_precision_recall_curve`, `plot_roc_curve`

Outline

Principle

- In ensemble methods, we design a classifier as a combination of (base) classifiers
- One kind of combination: a (weighted) vote of a set of classifiers. The majority vote wins the decision
  - theoretical results given by boosting.
  - Intuition: If the classifiers perform better than a random classifier and are sufficiently independent, then their vote forms a more accurate method than the classifiers.
- Numerous methods can be used to build independent classifiers and combine them : Bagging, Pasting, Boosting, Stacking, etc.

In details

Obtain diverse or independent classifiers

- Use different learning algorithms with the same data
- Modify the data
- Note: scaling is easier if you can distribute the calculations

Methods

- bagging (boostrap aggregating): uniform random selection of a subset of the data;
- pasting: same as above, but drawn without replacement.
- random subspace: similar to boostrap but drawn on attributes and not on examples.
- random patches: when you draw on both examples and attributes.

Combination

- Aggregation: by mode (highest frequency of predictions), by majority vote, by the average of decision functions.

Random Forests

- It's basically bagging with decision trees
- Introducing variety into the construction of a set of decision trees
- different decision trees are obtained by
  - drawing a sub-sample of examples, or
  - in the search for the best test: randomly mixed attributes, chosen from a subset of the attributes
- The readability of decision trees is lost
- However, the importance of attributes can be calculated as the average gain over all the trees in the forest, and this importance can be visualised.

Boosting principles

- We have a class of poor functions with a large approximation error: weak learners whose performance is just slightly better than random.
- How can we iteratively make it richer?
- Boost we aggregate weak learners to form stronger learners, whose performance can be as large as we want.
- Objectives:
  - a method that breaks down the complexity of learning a strong learner directly
  - to answer the theoretical question of how to transform a weak learner into a strong learner.
- Adaboost is an implementation that solves this theoretical problem.
  - Modify the distribution of examples by using a weight on each example that varies according to the rate of correct classification
  - The weight of poorly classified examples increases gradually
  - The decision is a weighted vote. The weight is a function of the accuracy of the classifier

Weak learner

Definition (Weak learner)

A is a γ-weak-learner for a class $\mathcal{H}$ if there exists a function $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ such that for any $\delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$ and for every labeling function $f : \mathcal{X} \to \{-1,1\}$, if the realization assumption holds with respect to $\mathcal{H}$, $\mathcal{D}$, f, then when running the learning algorithm A on $m \geq m_{\mathcal{H}}(\delta)$ iid examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that, with probability of at least $1-\delta$, $L_{\mathcal{D},f}(h) \leq 1/2 - \gamma$.

- ϵ (strong learner) is replaced by $1/2 - \gamma$.

Adaboost in details

- We are given a weak learner
- Adaboost is an iterative algorithm that modifies the distribution $\boldsymbol{D}^{(t)}$ over S at each round t
- The error of the hypothesis h_t built by the weak learner is $\epsilon_t = \sum_i^m D_i^{(t)} \mathbb{1}_{[h_t(\boldsymbol{x}_i) \neq y_i]}$. It is lower than $1/2 - \gamma$ with high probability.
- The weight associated with h_t is inversely proportional to the error: $w_t = \frac{1}{2} \log(\frac{1}{\epsilon} - 1)$: wrongly classified examples are given a higher weight.
- the output of Adaboost is a weighted vote of all hypotheses h_t

Algorithm

$\mathcal{D}^{(1)} = (1/m, \ldots, 1/m)$

for $t = 1, \ldots, T$

$h_t = WL(\mathcal{D}^{(t)}, S)$

$\epsilon_t = \sum_i^m D_i^{(t)} \mathbb{1}_{[h_t(\boldsymbol{x}_i) \neq y_i]}$

$w_t = \frac{1}{2} \log(\frac{1}{\epsilon_t} - 1)$

$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\boldsymbol{x}_i))}{\sum_j^m D_j^{(t)} \exp(-w_t y_j h_t(\boldsymbol{x}_j))}$

return $h_s(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^T w_t h_t(\boldsymbol{x})\right)$

- The hypothesis class of Adaboost, given a weak learner that builds hypothesis in B is $\{\text{sign}\left(\sum_{t=1}^T w_t h_t(x) \mid \boldsymbol{w} \in \mathbb{R}^T, h_t \in B\right\}$
- Theorem : The training error of Adaboost is bounded by $exp(-2\gamma^2 T)$

Gradient boosting

- Another way of focusing on errors: the next classifier in the loop attempts to correctly predict residual errors.
- Example with `DecisionTreeRegressor`.
  - An instance `dtr` of this classifier produces a residual `y2 =y - dtr.predict(X)`.
  - A `dtr2` is constructed to predict `y2`.
  - Combine the sum of the predictions from the 2 classifiers.
- The `GradientBoostingRegressor` class does this.

Stacking

- Bias reduction by classifier compositions
- we learn to combine the predictions of basic classifiers
- learning is divided into two parts:
  - part A trains n base classifiers
  - part B is used to create a set P of prediction vectors of dimension n
- the set P is used to train a combination (*blender*)

With sklearn

- a `VotingClassifier` class for simple weighted voting or no voting of classifiers
- the `AdaBoostClassifier` class
- the `BaggingClassifier` class
  - Bagging/pasting setting via the `boostrap` parameter
  - Setting of `max_features` and `max_samples`.
- the `RandomForestClassifier` class
- the `StackingClassifier` class
- In these implementations, the aggregation is the average of the decision functions.
- The same classes exist for regression.