Decision trees; cours 4

29/01/2019

Reminder

* **Supervised learning**
  + Regression
  + Classification
  + **Decision trees**
* Unsupervised learning
  + Clustering
  + Dimension reduction
* Reinforcement learning

Used for classification, it is one of the algorithm.

For each feature it takes the possible values and maps it so to know at the end knows if person survived or not.

It creates a possible tree of classification.

Many trees can be created depending on the order of the features.

So for the same data (xi, yi) we have multiple decision trees. For this one we just have to find the one with the lowest classification error

Error = wrong predictions / total sample

## The “Greedy” algorithm allows us to find a good tree :

Step 1 : start with empty tree, get the output class splitting

Step 2 : select a feature and split on it, it gives intermediate nodes, decide the status by the majority in each node

How to know which feature to split on ?

For each possible feature split we calculate the classification error for each status decided by the majority class. We calculate the number of wrong predictions / total sample.

We choose the feature to split in which the classification error is the least.

Step 3 : Recursion & stopping

If we have a “pure” class, where there is no error. It will stop there because it doesn’t change anything to split it again. It is a leaf node now

The rest do a recursion : continue until it becomes a “pure” class or there is no more data to split on.

Effectiveness of split is either : classification error, gini index, entropy

## Features with real values

Don’t split on each value.

Gonna choose a threshold, how to find the best one ? need to consider midpoints

For each midpoints calculate the classification error and choose the midpoint for the one the error is the lowest

## Multiclass Classification

Just have more status at each node

## Overfitting

More depth = More complexity = Risk of overfitting  
→ Implement Early Stopping to prevent it

Can use sklearn.tree.DecisionTreeClassifier(max\_depth=X, min\_samples\_split=Y, min\_samples\_leaf=Z)

# Ensemble Methods

Combine predictions on several base estimators in order to improve over a single estimator

* **Bagging (Averaging methods) :** instead of having just 1 estimation of your class, you have several estimations & you do an average

Goal : build several estimators on different subsets of data. Prediction proceeds with majority vote.

Eg : Random Forest

* **Boosting :** are built sequentially and one tries to reduce error of previous one. Prediction proceeds with weighted vote

Tendance, with for example Adaboost

### Bagging

* Each tree in the ensemble is built from a sub-sample drawn with replacement (**bootstrap sample**) from the training set.
  + Usually s (bootstrap simple size/number of points drawn) = 60%
* **Averaging** = to predict a new observation, use the majority vote of the trees on it

### Random Forest

* Special case of bagging
  + The sub-sample size = original input sample size
  + When splitting, pick the best split among a random subset of the features. You don’t choose all the features, but only some of them.

### Boosting

Goal : turn a weak learning algorithm into a strong one.

Classic : Data → learn classifier → preditct y → evaluate

Boosting : Classic + learn where mistakes + on recursion put more weight on point where mistakes

* Adaboost
  + Start same weight for all point
  + For each :
    - Learn with data weight
    - Compute coefficient
    - Update data weight
    - Normalize data weight
  + Combine the classifiers
* EGBoost
* GradientBoost

4 : gini, classification error, anthropie.