Introduction to machine learning
Ensemble methods
Stacking, Bagging, Boosting

Mathilde Mougeot

ENSIIE

2019

Ensemble methods

- Ensemble learning helps improve machine learning results by combining several models.
- This approach allows the production of better predictive performance models compared to a single model.

$\rightarrow$ These methods are commonly used in machine learning competitions, such as the Netflix Competition, KDD 2009, and Kaggle.

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking)

Several groups of ensemble methods

Parallel vs Sequential.

- **parallel ensemble methods** where the base learners are generated in parallel (e.g. Random Forest). The basic motivation of parallel methods is to exploit independence between the base learners since the error can be reduced dramatically by averaging.
- **sequential ensemble methods** where the base learners are generated sequentially (e.g. AdaBoost). The basic motivation of sequential methods is to exploit the dependence between the base learners. The overall performance can be boosted by weighting previously mislabeled examples with higher weight.

Homogeneous vs Heterogeneous.

- **homogeneous base learners**, i.e. learners of the same type, leading to homogeneous ensembles.
- **heterogeneous ensembles** i.e. heterogeneous learners, i.e. learners of different types. In order for ensemble methods to be more accurate than any of its individual members, the base learners have to be as accurate as possible and as diverse as possible.

Ensemble methods

Definition :

$$F(x) = c_0 + \sum_{m=1}^{M} c_m T_m(x)$$

Avec

- with $T_m(x)$, $1 \leq m \leq M$ "simple learners" **(base learners)**
- regression (or classification)
- Exemple : T_m : one decision tree
- c_m aggregation parameters

Notations : $T_m(x) = T(x; p_m)$, with for example

- T_m : Neural networks, p_m : weights of the neural network
- T_m : Decision tree, p_m : split parameters (variables, thresholds)

Ensemble methods

Learning a Tree model with ensemble methods :

The goal is to find and compute the split parameters $p_m \in P$ and the constants $c_m \in R$ such as :

$$\{\hat{c}_m, \hat{p}_m\} = min^M_{c_m, p_m} \sum_{i=1}^{N} L(y_i, c_0 + \sum_{m=1}^{M} c_m T(x; p_m))$$

- Because it is a NP-Hard optimization problem
- A two steps heuristic is used :
  1. Find the best parameters p_m of each of the models ("split" of the trees)
  2. Find the best c_m aggregation parameters.
- Key ingredients
  - observation sampling, Monte Carlo method (Importance Sampling)
  - choice of the model parameters (Parameter importance measure)
  - ISLE : Importance Sampling Learning Ensembles, Friedman and Popescu, 2003

Bagging & Random Forest

- **Bagging** Bagging stands for bootstrap aggregation. One way to reduce the variance of an estimate is to average together multiple estimates. Example : bagging for decision trees. Bagging pour Bootstrap Aggegating (Breiman, 1996).

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x)$$

  For aggregating the outputs of base learners, bagging uses voting for classification and averaging for regression

- **Random Forest** In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e. a bootstrap sample) from the training set. In addition, instead of using all the features, a random subset of features is selected, further randomizing the tree. (Breiman, 2001)
  As a result, the bias of the forest increases slightly, but due to the averaging of less correlated trees, its variance decreases, resulting in an overall better model.

Boosting

- Boosting refers to a family of algorithms that are able to convert weak learners to strong learners.
- The predictions are then combined through a weighted majority vote (classification) or a weighted sum (regression) to produce the final prediction.
- The principal difference between boosting vs bagging, is that base learners are trained in sequence on a weighted version of the data.

Boosting. Adaboost

Adaboost :Adaptive Boosting, Freund and Shapire (1997)

1. "Ability to boost the performance of weak base classifier" T_m
2. Computation of the aggregation parameters c_m

AdaBoost (Original Algorithm), weights : $w_i^0 = 1/n$

For $m = 1$ to M $\{$

1. Fit a classifier $T_m(x)$ based on the training data set using the weights w_i^m
2. Compute $Err_m = \frac{\sum_{i=1}^n w_i^m I(y_i \neq T_m(x_i))}{\sum_{i=1}^n w_i^m}$
3. Compute $\alpha_m = \log((1 - Err_m)/Err_m)$
4. Set $w_i^{m+1} \leftarrow w_i^m \exp[\alpha_m I(y_i \neq T_m(x_i))]$

$\}$

Output sign $\{\sum_{m=1}^M \alpha_m T_m(x)\}$

Boosting. Adaboost

- We see that the first base classifier y1(x) is trained using weighting coefficients that are all equal. In subsequent boosting rounds, the weighting coefficients are increased for data points that are misclassified and decreased for data points that are correctly classified.
- The quantity epsilon represents a weighted error rate of each of the base classifiers. Therefore, the weighting coefficients alpha give greater weight to the more accurate classifiers.

Boosting. Gradient Tree Boosting

Gradient Tree Boosting is a generalization of boosting to arbitrary differentiable loss functions.
It can be used for both regression and classification problems.
Gradient Boosting builds the model in a sequential way.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

At each stage, the decision tree $h_m(x)$ is chosen to minimize a loss function given the current model $F_{m-1}(x)$.

$$F_m(x) = F_{m-1}(x) + \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x_i))$$

The algorithms for regression and classification differ in the type of loss function used.

Stacking

Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.

Algorithm

Input: training data $\mathcal{D}_n = \{x_i, y_i\}_{i=1}^m$
Output: ensemble classifier $\mathcal{H}$
Step 1 : learn base-level classifiers
for t=1 to T **do**
learn h_t based on $\mathcal{D}$
end **for**
Step 2 : construct **new** data set **for** predictions
for i=1 to n **do**
$\mathcal{D}_h = \{x_i', y_i\}$, where $x_i' = \{h_1(x_1), \ldots, h_T(x_i)\}$
end **for**
Step 3 : learn a meta-classifier
learn H based on $\mathcal{D}_h$
return H

Stacking ensembles are often heterogeneous (different learning algorithms).

Conclusion

- Random Forest : Extremely used for many applications
- RF provides often the best solution (similar to SVM)
- Explanation of the theoretical performances are still currently under study
- Ensemble methods are very useful (Boosting also)