Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Ensemble learning methods
Part I — Bagging

Outline

- Ensemble classifiers – a motivation exercise
- Combining classifiers into ensembles – general scheme
- Generating random samples by bootstrapping
- Bagging vs. boosting
- Bagging – example classifier
- Random Forests

Ensemble classifiers – a motivation exercise

Consider the following task – we have a binary classification problem and a number of predictors, each with error less than 0.5. Will the resulting majority voting ensemble have an error lower than the single classifers?

Depends on the *accuracy* and the *diversity* of the base learners!

Illustrative example

Particular settings – assume that you have

- 21 classifiers
- each with error $p = 0.3$
- their outputs are *statistically independent*

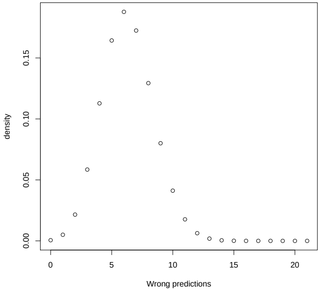
Compute the error of the ensemble under these conditions!

Solution of the exercise

How many classifiers will produce error output?

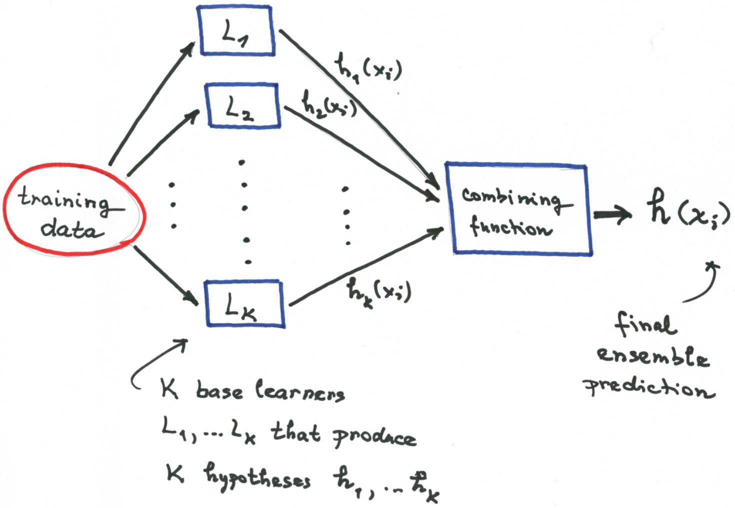
Key idea: The number of them will be binomially distributed! $\sim$ Bi(21, 0.3)

```
> plot(0:21, dbinom(0:21, 21, 0.3))
> dbinom(11, 21, 0.3)
[1] 0.01764978
> pbinom(10, 21, 0.3)
[1] 0.9736101
```



Conslusion: Accuracy of the ensemble will be more than 97.3 %!

General scheme of combining classifiers



Resampling approach

Resampling can be used as a way to produce diversity among base learners

- Distribute the training data into K portions
- Run the learning process to get K different models
- Collect the output of the K models use a combining function to get a final output value

Bootstrapping principle

- New data sets $Data_1, \ldots, Data_K$ are drawn from $Data$ with replacement, each of the same size as the original $Data$, i.e. n.
- In the i-th step of the iteration, $Data_i$ is used as a training set, while the examples $\{\mathbf{x} \mid \mathbf{x} \in Data \wedge \mathbf{x} \notin Data_i\}$ form the test set.
- The probability that we pick an instance is $1/n$, and the probability that we do not pick an instance is $1 - 1/n$. The probability that we do not pick it after n draws is $(1 - 1/n)^n \approx e^{-1} \doteq 0.368$.
- It means that for training the system will not use 36.8 % of the data, and the error estimate will be pessimistic. So the solution is to repeat the process many times.

Same algorithm, different classifiers

Combining classifiers to improve the performance

Ensemble methods – key ideas

- combining the classification results from different classifiers to produce the final output
- using (un)weighted voting
- different training data – e.g. bootstrapping
- different features
- different values of the relevant paramaters
- performance: complementarity $\longrightarrow$ potential improvement

Two fundamental approaches

- **Bagging** works by taking a bootstrap sample from the training set
- **Boosting** works by changing the weights on the training set

Bagging and boosting — the difference

- **Bagging**: each predictor is trained independently
- **Boosting**: each predictor is built on the top of previous predictors trained

  – Like bagging, boosting is also a voting method. In contrast to bagging, boosting actively tries to generate complementary learners by training the next learner on the mistakes of the previous learners.

Are ensembles effective?

Combining multiple learners

- the more **complementary** the learners are, the more useful their combining is
- the simpliest way to combine multiple learners is **voting**
- in **weighted voting** the voters (= base-learners) can have different weights

Unstable learning

- learning algorithm is called unstable if small changes in the training set cause large differences in generated models
- typical unstable algorithm is the decision trees learning
- bagging or boosting techniques are a natural remedy for unstable algorithms

Bagging

- Bagging is a voting method that uses slightly different training sets (generated by bootstrap) to make different base-learners. Generating complementary base-learners is left to chance and to unstability of the learning method.
- Generally, bagging can be combined with any approach to learning.

Bagging – algorithm

Bootstrap **AGG**regat**ING**

1. for $i \leftarrow 1$ to K do
2. $Train_i \leftarrow \text{bootstrap}(Data)$
3. $h_i \leftarrow \text{TrainPredictor}(Train_i)$

Combining function

- **Classification:** $h_{final}(\mathbf{x}) = \text{MajorityVote}(\mathrm{h}_1(\mathbf{x}), \mathrm{h}_2(\mathbf{x}), \dots, \mathrm{h}_K(\mathbf{x}))$
- **Regression:** $h_{final}(\mathbf{x}) = \text{Mean}(\mathrm{h}_1(\mathbf{x}), \mathrm{h}_2(\mathbf{x}), \dots, \mathrm{h}_K(\mathbf{x}))$

Random Forests

- an ensemble method based on decision trees and bagging
- builds a number of random decision trees and then uses voting
- introduced by L. Breiman (2001), then developed by L. Breiman and A. Cutler
- very good (state-of-the-art) prediction performance
- a nice page with description
  `www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`
- important: Random Forests helps to
  - avoid overfitting (by random sampling the training data set)
  - select important/useful features (by random sampling the feature set)

Building Random Forests

The algorithm for building a tree in the ensemble

1. Having a training set of the size n, sample n cases at random – with replacement, and use the sample to build a decision tree.
2. If there are M input features, choose a less number $m \ll M$. When building the tree, at each node a random sample of m features is selected as split candidates from the full set of M available features. Then the best split on these m features is used to split the node. A fresh sample of m features is taken at each split.
   – m is fixed for the whole procedure
3. Each tree is grown to the largest extent possible. There is no pruning.

The more trees in the ensemble, the better.
There is no risk of overfitting!

Regularized Random Forests

- a recent extension of the original Random Forest
  - introduced by Houtao Deng and George Runger (2012)
- produces a compact feature subset
- provides an effective and efficient feature selection solution for many practical problems
- overcomes the weak spot of the ordinary RF: Random Forest importance score is biased toward the variables having more (categorical) values
- a useful page: `https://sites.google.com/site/houtaodeng/rrf`
  - a presentation
  - a sample code
  - links to papers
  - a brief explanation of the difference between RRF and guided RRF

R packages for Random Forests

- **randomForest**: Breiman and Cutler's random forests for classification and regression
  – Classification and regression based on a forest of trees using random inputs.

- **RRF**: Regularized Random Forest
  – Feature Selection with Regularized Random Forest. This package is based on the 'randomForest' package by Andy Liaw. The key difference is the RRF function that builds a regularized random forest.
  – `http://cran.r-project.org/web/packages/RRF/index.html`

- **party**: A Laboratory for Recursive Partytioning
  – a computational toolbox for recursive partitioning
  – `cforest()` provides an implementation of Breiman's random forests
  – extensible functionality for visualizing tree-structured regression models is available

Summary of examination requirements

- Ensembles, bagging, boosting – general principles
- Simple bagging algorithm
- Random Forests
- Practical use of `randomForest()` package in R