Pattern Recognition and Machine Learning

Slide Set 5: Ensemble Methods

Heikki Huttunen
heikki.huttunen@tut.fi

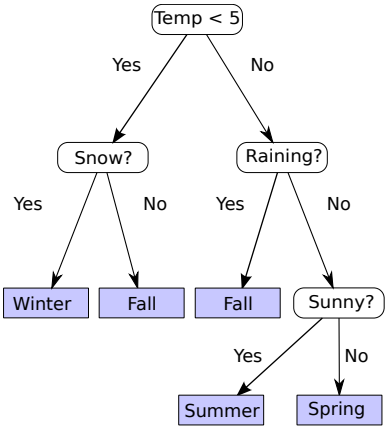
Department of Signal Processing
Tampere University of Technology

January 2017

RANDOM FOREST
(and other ensemble methods)

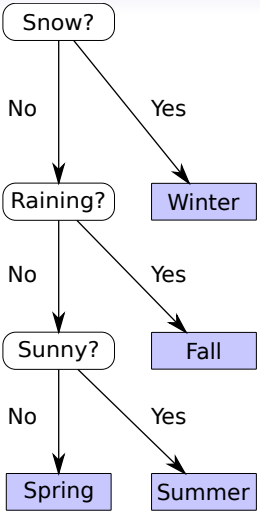
Random Forest

- Random forest (RF) is popular tree based classifier, proposed by Breiman in 1993.
- The RF is based on a collection of *decision trees*.
- A decision tree is a straightforward if-then-else-like diagram that combines individual attributes into a decision.
- Decision trees are easily trained to learn the data.
- For example, the tree on the right predicts the time of year from the following measured attributes: $\{\text{Temperature}, \text{Snow}, \text{Rain}, \text{Sunny}\}$.



Random Forest

- The problem of decision trees is that they *overlearn* the data, *i.e.*, the training data is memorized with a poor ability to generalize.
- With no restrictions, the DT will have one root-leaf-path per sample, which is essentially same as nearest neighbor.
- RF avoids this by training many "imperfect" trees.
- Each tree is trained with a subset of samples and a subset of features, *i.e.*, some of the attributes are hidden from the training.



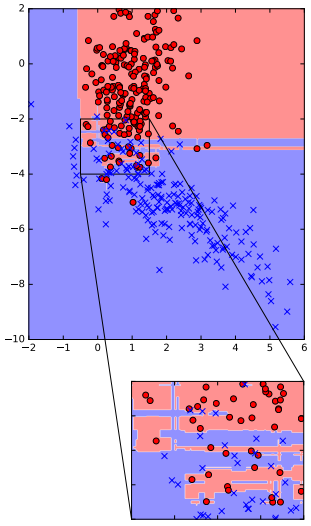
Decision Tree without the *Temperature* attribute.

Random Forest

- The RF extracts values at randomly selected coordinates.
- For example, if we have a data matrix $\mathbf{X} \in \mathbb{R}^{10\times 4}$ (*i.e.,* 10 samples with 4 features each), we might train trees with the following subsets of the data:
  - **Tree 1:** Train using rows {6,4,7,2,6,3} and columns {1,2,4}
  - **Tree 2:** Train using rows {1,10,2,1,7,9} and columns {1,3}
  - **Tree 3:** Train using rows {7,2,7,3,9,3} and columns {2}
  - ...
- Note that rows (samples) are sampled *with replacement*, *i.e.,* rows may appear more than once.
- Columns are sampled *without replacement* (it does not make sense to repeat the same data).

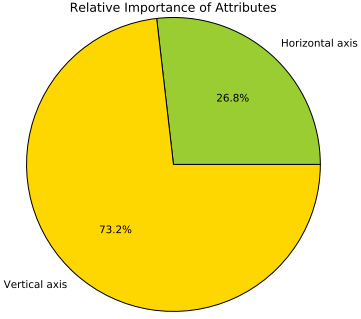
Random Forest

- After training many trees each with different samples and features, the RF predicts the class by taking the majority vote: what is the most frequent label.
- The number of trees varies from a few dozen to few thousand—default in Python is 10.
- The attached picture is produced by a 10-tree random forest.

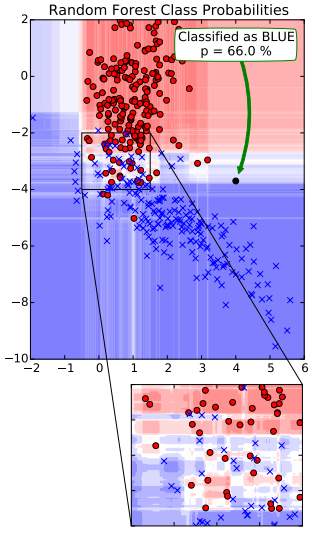


Random Forest

- The collection of trees gives a natural way of estimating class probabilities: Just use the proportion of trees voting for each class.
- RF's also includes a method for assessing feature importances: randomly shuffle each feature at a time and test how much the accuracy drops.



- Loosing an important feature drops the accuracy a lot.
- Shuffling a non-important feature will not change the result much.



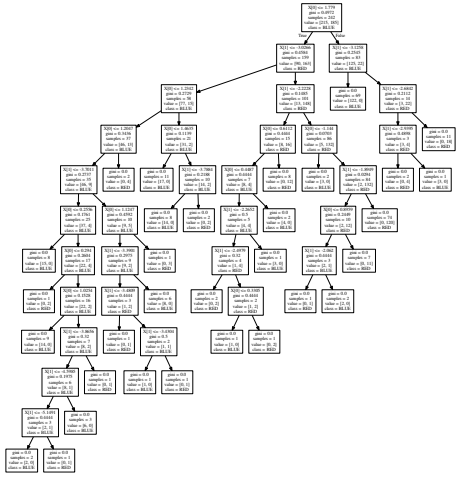
Random Forest in Scikit-Learn

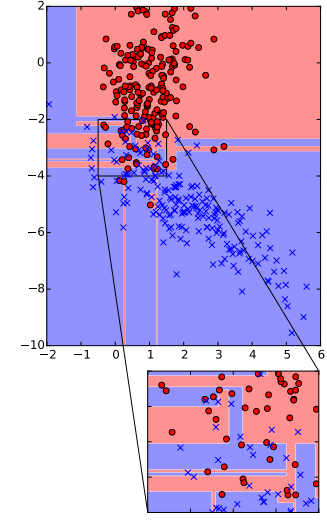
```
# Training code:
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X, y)
```

```
# Testing code:
>>>  clf.predict([0, -4])
array([ 0.])

>>>  clf.predict([0, -2])
array([ 1.])

>>> clf.predict_proba([[0, -4], [0, -2]])
array([[ 0.9,  0.1],
       [ 0.4,  0.6]])

>>> len(clf.estimators_)
10

>>> type(clf.estimators_[0])
sklearn.tree.tree.DecisionTreeClassifier
```

- The `RandomForestClassifier` class is used via the normal interface (`.fit()` and `.predict()`).
- Individual trees can be accessed, as well: `clf.estimators_` is a list of `DecisionTreeClassifier` objects.
- One of the 10 trained trees is visualized on the next slide (plot created using `sklearn.tree.export_graphviz`).

Decision Tree





Other Ensemble Classifiers

- Since the introduction of Random Forest by Breiman in 1993, several extensions have been proposed.
- As a group, these are called *ensemble methods*, because they all consist of an ensemble of weak classifiers,
- Most important ones are briefly summarized in the following slides.

AdaBoost Paradigm

- The **AdaBoost** paradigm was proposed by Freund and Schapire in 1995.
- Also in this case, the classifier consists of a collection of weak classifiers (most often decision trees).
- The difference to other ensemble methods is that trees are grown sequentially:
  1. Assign each sample a weight $w_1 \ldots, w_N$.
  2. Grow a tree minimizing the classification error weighted by w_n. That is, we emphasize the samples with large w_n.
  3. Append the new tree to our ensemble $\mathcal{E}$.
  4. Increase the weights for those samples that were incorrectly classified by $\mathcal{E}$.
  5. If not enough trees, then return to step 2.
- Implemented as `sklearn.ensemble.AdaBoostClassifier`.

Gradient Boosted Regression Trees

- **Gradient Boosted Regression Trees** were proposed by Friedman in 1999.
- Another boosting algorithm similar to AdaBoost.
- In this case, the training sequence is the following.
  1. Initialize the ensemble $\mathcal{E}$ by a single decision tree fit to the data.
  2. Train another tree for correcting the errors made by $\mathcal{E}$, *i.e.,* attempt to predict $\mathbf{y} - F_{\mathcal{E}}(\mathbf{X})$.
  3. Append the new tree to our ensemble $\mathcal{E}$.
  4. If not enough trees, then return to step 2.
- Implemented as `sklearn.ensemble.GradientBoostingClassifier`.

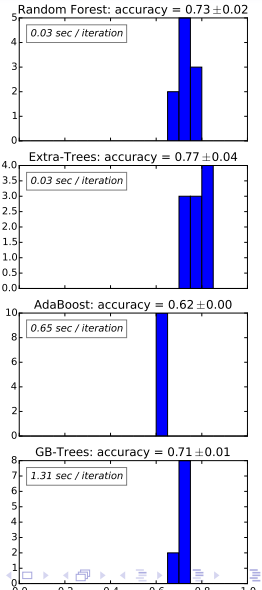
Extremely Randomized Trees

- **Extremely Randomized Trees** were proposed by Geurts *et al.* in 2006.
- The idea is to make the trees even weaker, but to compensate that with the large number of estimators in the ensemble.
- More specifically, not only the features and samples shown to the tree are randomized, but also the growing of individual trees is random.
- In particular the *split point i.e.,* the thresholds of comparisons like $X[1] <= -2.7025$ is randomized (see the graph of a decision tree at an earlier slide.
- Implemented as `sklearn.ensemble.ExtraTreesClassifier`.

Comparison of Methods

- The four ensemble methods were compared with the **arcene dataset**: https://archive.ics.uci.edu/ml/datasets/Arcene.
- For randomized algorithms, we iterate the experiment 100 times.

```
# Load Arcene data; 100+100 samples with dimension 10000:
# Mass spectrometer measurements from ovarian cancer patients and healthy
      controls.
X_train, y_train, X_test, y_test = load_arcene()

classifiers = [(RandomForestClassifier(), "Random Forest"),
               (ExtraTreesClassifier(), "Extra-Trees"),
               (AdaBoostClassifier(), "AdaBoost"),
               (GradientBoostingClassifier(), "GB-Trees")]

for clf, name in classifiers:
    clf.n_estimators = 100

    accuracies = []
    for iteration in range(100):
        clf.fit(X_train, y_train)
        y_hat = clf.predict(X_test)
        accuracy = accuracy_score(y_test, y_hat)
        accuracies.append(accuracy)
```



Comparison of Methods

- Let's add a bunch of other classifiers into our for loop.
  - *1-Nearest Neighbor*: **0.88**
  - *Logistic Regression*: 0.84
  - *Linear SVM*: 0.83
  - *5-Nearest Neighbor*: 0.82
  - *LDA*: 0.79
  - *Extra-Trees*: 0.77 ± 0.04
  - *9-Nearest Neighbor*: 0.73
  - *Random Forest*: 0.73 ± 0.02
  - *GB-Trees*: 0.71 ± 0.01
  - *AdaBoost*: 0.62
  - *SVM with RBF kernel*: 0.56
- It seems that linear models and NN excel with this data.