A(E)3M33UI—Exercise F: Receiver operating characteristic (ROC). Learning curve and its use in "model diagnostics".

Petr Pošík

March 24, 2015

The goal of this exercise is

- to practice the tree models,
- to introduce the ROC curves,
- to reiterate the difference between training and testing error, the model validation, and
- to demonstrate the value of the so-called learning curves.

1 Classification and regression trees

The goal of this section is to get some intuition, what the tree models for regression and classification look like.

1.1 Regression tree

As in the previous exercises, we shall work with the `auto-mpg.csv` dataset. In the first part, we will model the relationship of `disp` and `hp` with a regression tree.

Task 1: In `exF-1.py`, implement a piece of code which would fit a regression tree with maximal depths 1, 2, 3, and 5, and will draw their predictions in the same figure over the data for easy comparison.

Hints:

- Find inspiration in the documentation.
- You shall use `sklearn.tree.DecisionTreeRegressor` class as the model with suitably set parameters.
- To plot the data, use function `plot_1D_regr_model()` from `plotting.py`.

1.2 Classification tree

Task 2: In `exF-1.py`, fill in the necessary code to plot the decision boundary induced by a classification tree with certain maximal depths. Try various values of maximal depths, starting from 1.

Hints:

- Use `sklearn.tree.DecisionTreeClassifier` class as the model.
- To plot the decision boundary, use function `plot_2D_class_model()` from `plotting.py`.

2 Receiver operating characteristic (ROC)

So far, when solving a classification task, we have evaluated classifiers based on their missclassification rate (error) or equavalently based on their accuracy. However, more detailed information about the classifier performance can be found in the so called *missclassification or confusion matrix*.

For the binary classification task, the confusion matrix is just a 2x2 matrix. For each classifier, its values depend on the threshold used to decide if the example is positive or negative.

Receiver operating characteristic (ROC) is one of the graphical methods to compare binary classifiers. It shows how true positive rate (TPR) and false positive rate (FPR) change according to the change of the threshold which is used to decide to which class an example belongs. An ideal classifier corresponds to a point (0,1), i.e. TPR = 1, FPR = 0.

Task 3: Study the example for ROC and the documentation for `sklearn.metrics.roc_curve()`.

Task 4: In `exF-2.py`, experiment with the `max_depth` parameter of the decision tree model.

Task 5: In `exF-2.py`, experiment with the number of features loaded from the mpg dataset.

Hint: The function `load_mpg_for_classification()` accepts a list of features that shall be loaded from the mpg dataset. If you pass no argument to this function, it will use all the relevant features.

Task 6: In exF-2.py, try to setup several models so that you can compare them using ROC curve.

Hints:

- Study the function `plot_roc()` in `plotting.py`.
- You can compare several models of different kind, e.g. logistic regression, decision tree, SVM, etc., or you can compare the same type of model with different settings, e.g. decision tree with `max_depth` equal to 1, 2, 3, ...

3 Learning curves

A learning curve in machine learning is used to show how the training and testing error (or accuracy) changes with an increasing number of training examples. It may be used to detect if we face a high bias or high variance problem, and gives some indication if obtaining more training examples would help to reach better testing error.

Task 7: The script `exF-3.py` shall display a learning curve for a particular model, but it calls a function which is not implemented yet. In `model_evaluation.py`, implement the function `compute_learning_curve()` which shall take as its arguments

- the model,
- the training dataset sizes for which we want to compute the training and testing errors,
- the training data set, and
- the testing data set,

and shall produce on its output

- an array of the number of examples used for training,
- an array of training errors corresponding to the number of training examples,
- an array of testing errors corresponding to the number of training examples.

Hints:

- You get the training and testing data sets as input to the function. To simulate the increasing training dataset size, the function shall iteratively use increasingly large part of the training data.
- To compute the training error, you should use only the smaller part of the training data set, i.e. the part that was actually used for training.
- To compute the error, you can use the `score()` method of the classifiers. Remember, however, that the method returns accuracy, not the error of the model (but you can use error = 1-accuracy).

Task 8: Try to display the learning curve several times, it is a stochastic process. Try to display learning curves for various classifiers. Discuss what you see in the graph. Is there anything you can do to decrease the error of the classifier?

4 Homework

As usual, finish the exercise as a homework, ask questions on the forum, and upload the solution via Upload system!

5 Have fun!