B(E)3M33UI — Exercise F: Decision trees and ensemble models.

Petr Pošík, Jiří Spilka

March 28, 2017

The goals of this exercise:

- learn about decision trees and ensemble models (adaboost, random forest)
- use cross-validation for hyper-parameters tuning and model evaluation
- reiterate validation curves

1 Decision trees and ensemble models

1.1 Classification

The goal of this section is to get some intuition about **Decision Trees** and to work with the state of the art models: **Adaboost** and **Random Forest**. You should already know how a tree is build and what criterion is used for splitting nodes. Note that we will use the `auto-mpg.csv` with two features (disp and hp) and a class label (origin: US vs rest). This dataset is relatively easy to classify and the benefits of ensemble models are not spectacular over the simple Decision Trees or Logistic Regression. Remember that we are using complex models for this dataset for educational purposes only. You should have an intuition about Occam's razor (parsimony) from lectures.

Task 1: In exF-1.py use Decision tree classifier and find optimal hyper-parameter/s using grid search. Also plot a validation curve and discuss areas of underfitting and overfitting.

Hints:

- Consult documentation of `sklearn.trees.DecisionTreeClassifier`, and parameter: `max_depth`.
- **Use only training data for the grid search** and do not touch the test data that are intended for performance estimation!

The decision tree is used as a base classifier (base learner) in ensemble models. A simple intuition behind the ensembles is that a **smart combination of different** classifiers would provide better generalization performance than individual classifiers. **Adaboost** targets to learn different weak learners using adaptive boosting. The AdaBoost focuses on training samples that are hard to classify correctly and lets the weak learners to subsequently learn from misclassified training samples to improve performance

of the ensemble and create a strong learner. In contrast, Random Forest uses bagging (bootstrap aggregation), where bootstrap samples (random samples with replacement) are drawn from the training data. These samples together with random feature subsets are used to fit individual trees, which are then aggregated.

Task 2: In exF-1.py use the Adaboost algorithm and find optimal hyper-parameter/s using grid search. Plot a validation curve and discuss differences to the Decision Tree Classifier.

Hints:

- Consult documentation of `sklearn.ensemble.AdaBoostClassifier`.
- Use a decision stump (tree with one node) as a base estimator (weak learner).
- Start by optimizing only `n_estimators` and make sure you understand the other hyper-parameters as well.

Task 3: In exF-1.py use the Random Forest algorithm and find optimal hyper-parameter/s using grid search. Plot a validation curve and discuss the results.

Hints:

- Consult documentation of `sklearn.ensemble.RandomForestClassifier` and parameters `max_depth` and `n_estimators`.
- Make sure you understand the other hyper-parameters as well.

Task 4: Compare classification performance of Decision tree, Adaboost, and Random Forest on the test set.

Task 5: Plot a decision boundary of each classifier and try to discuss differences between them.

Hints:

- Use functions `plot_xy_classified_data` and `plot_2D_class_model`.

1.1.1 Model ensemble via majority voting

In this section we explore model ensemble via majority voting. Lets suppose that we have already created several **independent models** of any kind (e.g. Decision Tree, SVM, Adaboost, Random Forest, etc.). The goal here is to combine these models using majority voting to further improve generalization ability. Classically this is the very last step to be performed in modeling. Again we note here that with our simple dataset we will not observe remarkable improvements but, in general, model ensembles works quite well.

Task 6: In exF-2.py set up three models and combine them using majority voting. Compare accuracy and decision boundaries.

Hints:

- For each model use optimal hyper-parameters from the previous tasks.

- Consult documentation of `sklearn.ensemble.VotingClassifier`.
- You can use different models such as Logistic Regression, SVM etc. Diversity counts!
- Remember, do not use the test data for model tuning or ensemble selection!
- If you attempt to combine prediction probabilities make sure that these probabilities are calibrated, cf. e.g. `sklearn.calibration`. Different models provide different probability distributions.
- Most probably, for this simple dataset, you will not get better results than using Random Forest alone.

1.2 Regression

Conceptually, the approach to regression is similar to the approach that we have already done for classification.

Task 7: Make sure you understand how the regression is performed using Decision Trees.

Task 8: You are welcome to perform similar study for regression as we did for the classification. This is optional.

2 Conclusion

In this section we have used very complex models for a simple classification task for educational purposes only. Remember that simple is usually better then complex (Occam's razor) and that you should always put effort in model understanding and interpretation (How a model behave under a change of hyper-parameters? Which features are used in the model? etc.).

3 Have fun!

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