Machine Learning Project 1: Overfitting, Underfitting and Metaparameters

1 Goal of the Project

In the first part of the course, you learned about basic models, over/underfitting and metaparameters. In this project, you will use decision trees to solve a classification task and select the complexity of these trees with respect to two metaparameters. You are expected to handle out a report of max. 4 pages, including plots. **Additional pages will not be read**. If the produced decision trees are too large, you can submit them in separate files (png, jpg, etc.). On the implementation side, you will use Python 3 with the `scikit-learn` framework (`http://scikit-learn.org`), which provides many tools for machine learning and has a detailed documentation. In order to complete this project, you will need the following Python packages:

- `sklearn` (`http://scikit-learn.org`)
- `numpy` (`http://www.numpy.org`)
- `matplotlib` (`http://matplotlib.org`)
- `pydot` (`https://code.google.com/p/pydot`)

Notice that the exact package names depend on your OS distribution, but they are usually available on repositories or through the `pip3` system. You will also need the `Graphviz` library (`http://www.graphviz.org`) to visualize trees. If you need more information on Graphviz and other packages installation, see the wiki of the course on Webcampus (called wikiML).

2 Training and Test Datasets

The Adult dataset is widely known and used in machine learning. It contains census information from 1994 about American citizens for predicting if their income is greater or lower than 50 000$/year. To learn more about this classification dataset, see the following link: `http://archive.ics.uci.edu/ml/datasets/Adult`. The dataset has been split into two parts: the training set (file named `Adult_train.csv`) and the test set (file named `Adult_test.csv`). The former set will be used to train your decision trees and the latter set for evaluating them.

3 Training Decision Trees

Once you have loaded the training and test sets (be careful, the first line contains the feature names and the last column contains the target to predict), you can train decision trees (DTs). `scikit-learn` provides a DT implementation which can deal with continuous features:

- `http://scikit-learn.org/stable/modules/tree.html#classification`
- `http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

Make sure to always set `criterion = 'entropy'` when you create your decision tree to use entropy as a splitting criterion.

Your first task is to train a DT with a maximal depth of 3. Show the DT in your report and comment on the result. What can you say about the entropy values and the class distributions at leaf nodes? Is the maximum depth of the decision tree large enough? Do you see over/underfitting cases by looking at the leaves? What is the training and test errors achieved by the DT?

Now that you have built your first DT, it is time to explore new metaparameters. Based on your conclusions from the first task, choose a new metaparameter.

Your second task is to train a DT with another metaparameter than the maximal depth. Choose one metaparameter from the scikit-learn documentation, replace the maximal depth metaparameter and comment on the result. How is this DT different with respect to the one from the first task? Compare the two DTs with respect to the over/underfitting at the leaves, the interpretability of the models (is one of the DTs easier/harder to understand?) and the training and test errors.

Tip: DTs can be drawn with the `DT_to_PNG` function of the `utils.py` file.

4 Comparing Models of Increasing Complexity

While the training set allows you to train a DT, the test set can be used to compare several DT models of increasing complexity.

Your third task is to train DTs with a maximal depth of 1 to a maximal depth of 100 (meaning that 100 models are to be trained). Use the training set to train each DT and compute the training error (mean percentage of misclassification on training data). Use the test set to compute the test error (mean percentage of misclassification on test data). Show a plot of the training and test errors for each depth in your report and comment on the result. Does the plot correspond to the theory? Can you spot over/underfitting cases in the plot?

As for the Section 3, it is interesting to compare the effect of the metaparameter choice.

Your fourth task is to train DT with an increasing value for the second metaparameter you chose in Section 3. Use the training set to train each DT and compute the training error (mean percentage of misclassification on training data). Use the test set to compute the test error (mean percentage of misclassification on test data). Show a plot of the training and test errors in your report and comment on the result. Does the plot correspond to the theory? Can you spot over/underfitting cases in the plot? What is the difference with the maximal depth plot?

Tip: plots can be made with the `plot` function from `matplotlib.pyplot`. Here is an example of use:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4])
y1 = x
y2 = x**2

plt.plot(x, y1, 'r')
plt.plot(x, y2, 'b')

plt.xlabel('x')
plt.ylabel('y')

plt.show()
```

5 Choose your Model

Using the results of Section 4, you will now have to select your final DTs.

Finally, choose the best maximal depth value (according to you) based on the corresponding plot of the third task. **Justify your choice** theoretically. Comment the tree very shortly (in terms of over/underfitting, interpretability and training/test errors). Repeat the same procedure (choosing the best value based on the plot and analyzing the resulting tree) for the other metaparameter you chose in Section 3. Compare these two new DTs. How are they different with respect to the ones of Section 3?