Overfitting and performance evaluation with PYTHON

Objective: The objective of this exercise is to understand how cross-validation can be used to avoid overfitting as well as evaluate and compare model performance.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* C9 as well as the files in the exercise 6 folder available from Campusnet.

Preparation: Exercises 1-5

Part 1: Group discussion (max 15 min)

For the group discussion, each group should have selected a *discussion leader* at the previous exercise session. The purpose of the discussion leader is to ensure all team members understands the answers to the following two questions:

Multiple-Choice question: Solve and discuss **problem 9.1** from chapter 9 of the lecture notes. Ensure all group members understand the reason why one of the options is true and why the other options can be ruled out. (After today's exercises make sure to complete the remaining multiple-choice problems listed as part of the preparation for week 6 on the course homepage).

Discussion question: Discuss the following question in the group

- Explain what we wish to accomplish by using cross-validation and two-layer cross-validation. In particular, why do we ever need to perform two-layer cross-validation? Go over the pseudo-code for one-layer cross-validation on slide 27 and explain what the various steps do and what they involve.

Part 2: Programming exercises

Piazza discussion forum: You can get help by asking questions on Piazza: https://piazza.com/dtu.dk/fall2017/02450

Software installation: Extract the Python toolbox from Campusnet. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/`

Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

6.1 Decision tree pruning using cross-validation

In this exercise we will use cross-validation to prune a decision tree. When applying cross-validation the observed data is split into training and test sets, i.e., `X_train`, `y_train` and `X_test` and `y_test`. We train the model on the training data and evaluate the performance of the trained model on the test data.

6.1.1 Inspect and run the script `ex6_1_1.py`. The script load the `wine2.mat` file with wine data using the `loadmat()` function. In this version of the wine data, outliers have already been removed. Notice how the script divides the data into a training and a test data set. Now, we want to find optimally pruned decision tree, be modifying its maximum depth. For different values of parameter (depth from 2 to 20) explain how the script fits the decision tree, and compute the classification error on the training and test set (holdout cross-validation). Notice how the script plot the training and test classification error as a function of the pruning level. What does this plot tell you?

Script details:

- *Take a look at the module* `sklearn.cross_validation` *and see how it can be used to partition the data into a training and a test set (holdout validation,* `train_test_split()` *function). Note, that the package contains also functions to partition data for K-fold cross-validation. Some of the functions can ensure that both training and test sets have roughly the same class proportions.*
- *Fit and train the classification tree similarly like in the previous week exercises, modify regularizing parameter in every iteration (here:* `max_depth`*)*

What appears to be the optimal tree depth? Do you get the same result when you run your code again, generating a new random split between training and test data? What other parameters of the tree could you optimize in cross-validation?

6.1.2 Inspect the script `ex6_1_2.py`. The script repeat the exercise above, using 10-fold cross-validation. To do this, the data set is divided into 10 random training and test folds. For each fold, a decision tree is fitted on the training set and it's performance is evaluated on the test set. Finally, the average classification error is computed across the 10 cross-validation folds.

Script details:

- *This time* `KFold()` *function from module* `sklearn.cross_validation` *can be used to partition the data into the 10 training and test partitions. It returns* `CV` *object through which you can iterate to obtain train/test indices at each fold.*

What appears to be the optimal tree depth? Do you get the same result when you run your code again, generating a new random split between training and test data? How about 100-fold cross-validation or leave-one-out cross-validation?

6.2 Variable selection in linear regression

In this exercise we consider cross-validation for variable selection and model performance evaluation in linear regression. We will try to predict the body-weight of a person based on a number of body measurements using linear regression with feature subset selection. The data is a subset of the data available at `http://www.sci.usq.edu.au/courses/STA3301/resources/Data/` described in [1]. To measure how well we can predict the body-weight, we will use the squared error between the true and estimated body-weight.

In our estimation we will use two levels of cross-validation: 1) On the outer level, we use 5-fold cross-validation to estimate the performance of our model, i.e., we compute the squared error averaged over 5 test sets. 2) On the inner level, we use 10-fold cross-validation to perform sequential feature selection (see figure 1).

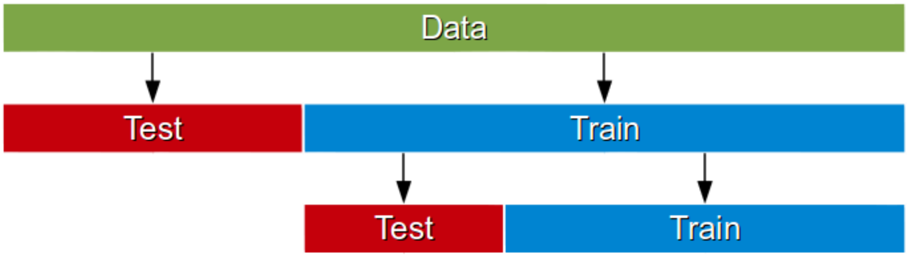

Figure 1: Multi-level cross validation

6.2.1 You can load the body data into Python with the command `loadmat('..\\Data\\body.mat')` The data set contains data for the 23 attributes in the matrix `X` and the body-weight in `y`.

Inspect and run the script `ex6_2_1.py`. The script applies 5-fold cross-validation to the problem of fitting a linear regression model to estimate the body-weight based on the attributes. Explain how the script, when fitting the models, compares two methods: 1) using all 23 attributes, and 2) using 10-fold cross-validation to perform sequential feature selection, thus choosing a subset of the 23 attributes.

Explain how the script computes the 5-fold cross-validated training and test error with and without sequential feature selection. Explain how it can be seen that without feature selection, the model overfits. Explain how it can be seen the feature selection tends to choose features such as height and waist girth, and disregard features such as the wrist diameter, which seems reasonable when predicting body-weight.

Script details:

- *Again, you may use `KFold()` function to set up the crossvalidation partitions needed.*
- *To fit a linear regression model, use the `sklearn.linear_model.LinearRegression` class (methods `fit()` and `predict()`), as you did in the previous exercises.*
- *To perform sequential features selection with linear regression model and k-fold cross-validation you can use the function `feature_selector_lr()` from the 02450 toolbox. Type `help(feature_selector_lr)` to read how it works, or give a closer look at its implementation in `toolbox_02450.py` file.*

Optional: Try modifying the solution to use backward feature subset selection. Does it give the same result? If you are interested in other methods for feature selection, have a look at module `sklearn.feature_selection`.

6.3 Comparing classifiers

In this part of the exercise we will compare a decision tree and logistic regression. The classifiers will be compared in terms of their difference in error rate. If this difference is significantly different from zero, one classifier is better than the other. To compare the classifiers we will use a credibility interval with $\alpha = 0.05$ as explained in section 9.3.3 of the lecture notes.

6.3.1 Inspect and run the script `ex6_3_1.py`. The wine data can be loaded using the `loadmat()` function. The script computes the 10-fold cross-validation classification error rate for 1) a logistic regression model and 2) an un-pruned decision tree with the same settings as in exercise 6.1.1.

Explain how it can be seen that for this classification problem, logistic regression outperforms the decision tree. Notice how the script runs the 10-fold cross-validation with random sampling several times, and in every test uses paired t-test to check the significance of difference between the two classifiers (p-value).

Script details:

- *Fit regression tree and logistic regression classifier as you did in previous exercises.*
- *To understand how the credibility interval is computed, compare the intermediate quantities with those computed in the book up to eq.9.15 in section 9.3.3. Which quantities in the code corresponds to which in figure 9.16? Does the computed values make sense intuitively?*

6.4 Task for the report

Find the optimal pruning level of a decision tree as well as the attributes useful for prediction in linear regression by selecting the model order using cross-validation. Use a two level cross-validation where you at the outer level evaluate the performance of the optimal model and on the inner level select for the optimal model (i.e. pruning level and features). Evaluate the significance of the difference in performance between your pruned decision trees and a logistic regression classifier.

References

[1] Grete Heinz, Louis J Peterson, Roger W Johnson, and Carter J Kerk. Exploring relationships in body dimensions. *Journal of Statistics Education*, 11(2), 2003.