Overfitting, cross-validation and Nearest Neighbor with MATLAB

Objective: The objective of this exercise is to understand how cross-validation can be used to avoid overfitting as well as the k-nearest neighbor method.

Piazza discussion forum: You can get help by asking questions on Piazza: piazza.com/dtu.dk/fall2019/october2019

Software installation: Extract the Matlab toolbox from the Dropbox folder . Start Matlab and go to the `<base-dir>/02450Toolbox_Matlab/` directory using the command `cd('<base-dir>/02450Toolbox_Matlab/')` and run `setup.m`. 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_Matlab/Scripts/`
Representation of data in Matlab:

	Matlab var.	Type	Size	Description
	`X`	Numeric	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	Cell array	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	Numeric	Scalar	Number of data objects.
	`M`	Numeric	Scalar	Number of attributes.
Regression	`y`	Numeric	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	Numeric	$N \times 1$	Class index: For each data object, `y` contains a class index, $y_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	Cell array	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	Numeric	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.m`. The script load the `wine2.mat` file with wine data by the command `load Data/wine2`. 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 function* `cvpartition` *and see how it can be used to partition the data into a training and a test set (holdout validation). Note, that this function can also be used to partition data for K-fold cross-validation. Note also how the function can ensure that both training and test sets have roughly the same class proportions.*
- *To classify the training set using the classification tree, you can use the* `eval` *function. Type* `help classregree/eval` *to get help. Note, that the third parameter,* `SUBTREES` *can be used to specify the pruning level used.*
- *To compare two cell arrays of strings, you can use the* `strcmp` *function.*

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.m`. 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:

- *As before,* `cvpartition` *can be used to partition the data into the 10 training and test partitions.*

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).

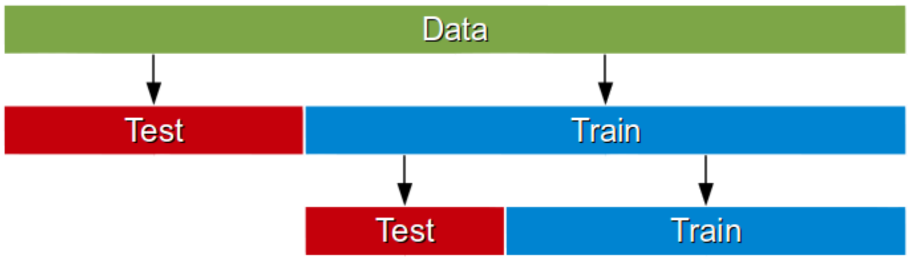

Figure 1: Multi-level cross validation

6.2.1 You can load the body data into Matlab with the command `load Data/body`. 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.m`. 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* `cvpartition` *to set up the crossvalidation partitions needed.*
- *To fit a linear regression model, use the functions* `glmfit` *and* `glmval` *as in the previous exercises.*
- *To perform sequential features selection in Matlab, you can use the function* `sequentialfs`*. One of the inputs required by this function is a so-called function-handle to a criterion function:*
  `CRITERION = FUN(XTRAIN,YTRAIN,XTEST,YTEST).`
- *By default,* `sequentialfs` *uses 10-fold crossvalidation for the feature selection, but you can use the argument* `CV` *if you want to try out other methods.*
- *One way to supply the criterion function is to write a function, e.g., called* `funLinreg`*, in a separate file* `funLinreg.m`*. This function must take the four arguments shown above, fit a linear model using the training data, evaluate the fitted model on the test data, and compute the squared test error as an output. Using this approach, the first argument to* `sequentialfs` *should be* `@funLinreg`*.*
- *Another way is to use a so-called anonymous function—that way you don't have to make a separate* `.m` *file for the criterion function; however, you must be able to write the whole criterion function on a single line, e.g., like this:*
  ```
  funLinreg = @(X_train, y_train, X_test, y_test) ...
   sum((y_test-glmval(glmfit(X_train, y_train), ...
   X_test, 'identity')).^2);
  ```
  When you use an anonymous function, you do not have to use a `@` *before the function name in the argument to* `sequentialfs`*.*
- *Type* `help function_handle` *to learn more about function handles and anonymous functions in Matlab.*

Optional: Try modifying the solution to use backward feature subset selection. Does it give the same result?

6.3 K-nearest neighbor classification

In this exercise we will use the k-nearest neighbors (KNN) method for classification. First, we will consider 4 different synthetic datasets, that can be loaded into Matlab using the commands `load Data/synth1`, . . . , `load Data/synth4`.

6.3.1 Consider the script `ex6_3_1.m`. For each of the four synthetic datasets, do the following. Load the dataset into Matlab and examine it by making a scatter plot. Classify the test data `X_test` using a k-nearest neighbor classifier. Choose a suitable distance measure (you should at least consider the following distance measures: `'euclidean'`, `'cityblock'`, `'correlation'`, and `'cosine'`.) Choose a suitable number of neighbors. Examine the accuracy and error rate.

Script details:

- *The Matlab function* `knnclassify` *can be used to perform k-nearest neighbors classification.*
- *To plot a confusion matrix, you can use the function* `confmatplot` *in the course toolbox. This function also displays the accuracy and error rate.*

Which distance measures worked best for the four problems? Can you explain why? How many neighbors were needed for the four problems? Can you give an example of when it would be good to use a large/small number of neighbors? Consider e.g. when clusters are well separated versus when they are overlapping.

In general we can use cross-validation to select the optimal distance metric and number of nearest neighbors k although this can be computationally expensive. We will return to the Iris data we have considered in previous exercises, and attempt to classify the Iris flowers using KNN.

6.3.2 Consider the script `ex6_3_2.m`. The script loads the Iris data into Matlab. Explain how the script uses leave-one-out crossvalidation to estimate the number of neighbors, k, for the k-nearest neighbors classifier and plots the crossvalidated average classification error as a function of k for $k = 1, \ldots, 40$.

Script details:

- *To load the Iris data, you can run your solution to exercise 4.1.1.*
- *Use* `cvpartition` *to set up the crossvalidation partitions needed.*
- *As before, use the* `knnclassify` *function for k-nearest neighbors classification.*

6.3.3 Discussion: What are the benefits and drawbacks of K-nearest neighbor classification and regression compared to logistic regression, decision trees and linear regression? (Hint: There are two important aspects of classification and regression methods, how well the methods can *predict* unlabeled data and how well the method *describe* what aspects in the data causes the data to be classified a certain way .)

6.4 Task for the report

The report will make use of cross-validation, but in conjunction with methods we have not seen yet. Please see report description for more information.

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.