Overfitting, cross-validation and Nearest Neighbor with PYTHON

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.

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

Discussion forum: You can get help on our online discussion forum: https://piazza.com/dtu.dk/spring2023/02450

Software installation: Extract the Python toolbox from DTU Inside. 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, $\mathtt{y}_n \in \{0, 1, \ldots, C-1\}$, 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.
	$\star$`_train`	—	—	Training data.
	$\star$`_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).

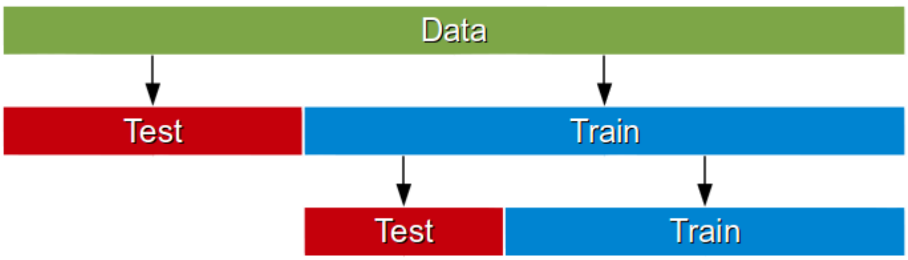

Figure 1: Multi-level cross validation

6.2.1 You can load the body data into Python with the command `loadmat('..\\Data\\body` 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 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 Python using the `loadmat` function. The data is stored in files `Data/synth1`, . . . , `Data/synth4`.

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

Script details:

- *The Python class* `KNeighborsClassifier` *from* `sklearn\neighbors` *module can be used to perform k-nearest neighbors classification.*
- *To generate a confusion matrix, you can use the function* `confusion_matrix()` *function from module* `sklearn.metrics` *in the course toolbox. You can use* `imshow()` *function to plot the confusion matrix.*

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.py`. The script loads the Iris data into Python. 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* `LeaveOneOut` *crossvalidation from module* `sklearn.cross_validation`*.*
- *As before, use the* `KNeighborsClassifier` *class 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.

1 Homework problems for this week

Problems

Question 1. Fall 2014 question 27: Alice is considering a linear regression model for a dataset comprised of $N = 1000$ observations. She wishes to both select the optimal regularization strength as well as estimate the generalization error of the model at the optimal regularization strength. To simplify the problem, she only considers the following 6 possible values of the regularization strength λ:

$$\lambda = 10^{-2},\ 10^{-1},\ 10^{0},\ 10^{1},\ 10^{2},\ 10^{3}.$$

Alice opts for a two-level strategy in which she uses the hold-out method to estimate the generalization error and cross-validation is used to select the optimal regularization strength, i.e. the dataset is first divided into a validation set $D_{\text{validation}}$, comprised of 20% of the full dataset, and the remainder D_{CV} is used for cross-validation. Alice uses standard $K = 10$ fold cross-validation to select the optimal regularization strength on D_{CV} and, having estimated the optimal regularization strength, uses the hold-out method on D_{CV} and $D_{\text{validation}}$ to estimate the generalization error.

Suppose for any fixed value of the regularization strength, the time taken to *train* the weights of the linear regression model on a dataset of size N_{train} is N^2_{train} units of time and the time taken to *test* a trained model on a dataset of size N_{test} is $\frac{1}{2}N^2_{\text{test}}$ units of time. Suppose the duration of all other tasks is neglible, what is the total time taken for the entire procedure?

A $12.78 \cdot 10^6$ units of time.

B $15.98 \cdot 10^6$ units of time.

C $31.30 \cdot 10^6$ units of time.

D $31.96 \cdot 10^6$ units of time.

E Don't know.

Question 2. Spring 2013 question 13: We would like to fit an artificial neural network to the PM10 dataset shown in Table 2. It is decided that DAY should not be included in the model as this cannot be influenced by decision makers. We therefore only consider x_1, x_2, x_3 and x_4 corresponding to logCAR, TEMP, WIND and TEMPDIF respectively. An artificial neural network is applied to the data with these four attributes. The neural network has three hidden units and is trained using different combinations of the four attributes x_1, x_2, x_3 and x_4. Table 1 gives the training and test performance of the artificial neural network for different combinations of the four attributes. Which one of the following statements is *correct*?

Feature(s)	Training rmse	Test rmse
x_1	0.71	0.75
x_2	0.58	0.64
x_3	0.60	0.62
x_4	0.92	0.94
x_1 and x_2	0.60	0.69
x_1 and x_3	0.35	0.44
x_1 and x_4	0.52	0.66
x_2 and x_3	0.56	0.69
x_2 and x_4	0.45	0.52
x_3 and x_4	0.62	0.64
x_1 and x_2 and x_3	0.36	0.34
x_1 and x_2 and x_4	0.28	0.33
x_1 and x_3 and x_4	0.27	0.45
x_2 and x_3 and x_4	0.20	0.43
x_1 and x_2 and x_3 and x_4	0.10	0.35

Table 1: Root mean square error (rmse) for the training and test set when using an artificial neural network with three hidden units to predict the level of pollution (logPM10) based only on the first four attributes (x_1–x_4) using the hold-out method with 50 % of the observations hold-out for testing.

No.	Attribute description	Abbrev.
x_1	Logarithm of number of cars per hour	logCAR
x_2	Temperature 2 meter above ground (degree Celsius)	TEMP
x_3	Wind speed (meters/second)	WIND
x_4	Temperature difference between 25 and 2 meters (degree Celsius)	TEMPDIF
x_5	Wind direction (degrees between 0 and 360)	WINDDIR
x_6	Whole hour of the day	HOUR
x_7	Day number from October 1. 2001	DAY
y	Logarithm of PM10 concentration	logPM10

Table 2: The attributes of the PM10 data. The output is given by the hourly values of the logarithm of the concentration of PM10 particles (logPM10).

A Neither forward nor backward selection will identify the optimal feature combination for this problem.

B Backward selection will result in a better model being selected than using forward selection.

C Backward selection will use a model that include all the features x_1, x_2, x_3, and x_4.

D Forward selection will select the features x_1, x_2 and x_4.

E Don't know.

Question 3. Fall 2013 question 9: In order to predict if an observation corresponds to a relatively small or large island we will use a k-nearest neighbor (KNN) classifier based on the Euclidean distance between the eight observations given in Table 3. We will use leave-one-out cross-validation for the KNN in order to classify whether the eight considered observations constitute small islands (given in red, i.e. observation O1, O2, O3, O4) or large island (given in blue, i.e. observation O5, O6, O7, O8) using a three-nearest neighbor classifier, i.e. $K = 3$. The analysis will be based only on the data given in Table 3. Which one of the following statements is *correct*?

	O1	O2	O3	O4	O5	O6	O7	O8
O1	0	2.39	1.73	0.96	3.46	4.07	4.27	5.11
O2	2.39	0	1.15	1.76	2.66	5.36	3.54	4.79
O3	1.73	1.15	0	1.52	3.01	4.66	3.77	4.90
O4	0.96	1.76	1.52	0	2.84	4.25	3.80	4.74
O5	3.46	2.66	3.01	2.84	0	4.88	1.41	2.96
O6	4.07	5.36	4.66	4.25	4.88	0	5.47	5.16
O7	4.27	3.54	3.77	3.80	1.41	5.47	0	2.88
O8	5.11	4.79	4.90	4.74	2.96	5.16	2.88	0

Table 3: Pairwise Euclidean distance, i.e $d(Oa, Ob) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||_2 = \sqrt{\sum_m (x_{am} - x_{bm})^2}$, between eight observations of the Galápagos data. Red observations (i.e., O1, O2, O3, and O4) correspond to the four smallest islands whereas blue observations (i.e., O5, O6, O7, and O8) correspond to the four largest islands.

A The error rate of the classifier will be 1/8

B The error rate of the classifier will be 1/4

C The error rate of the classifier will be 3/8

D The error rate of the classifier will be 1/2

E Don't know.

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.