Machine Learning 2016/2017, Assignment 2

Yevgeny Seldin, Christian Igel
Department of Computer Science, University of Copenhagen

The deadline for this assignment is **12:00 pm (noon, not midnight) 5/12/2016**. You must submit your individual solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your source code in this PDF file.
- Your solution source code (Matlab / R / Python scripts or C / C++ / Java code) with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF.
- Your code should be structured such that there is one main file that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Your code should also include a README text file describing how to compile and run your program, as well as list of all relevant libraries needed for compiling or using your code.

1 K-Nearest Neighbors

Please download the data sets `IrisTrainML.dt` and `IrisTestML.dt` from the course homepage. These data sets have been generated from the famous Iris flower data set [2], which has been used as an example for classification algorithms since the work of Ronald Fisher [4]. However, instead of the original four input features only two are considered. Furthermore, a feature has been rescaled and some examples have been removed.

The data set describes three different species of Iris, namely Iris setosa, Iris virginica and Iris versicolor. That is, we have a three class classification problem. Each line in the data files corresponds to one flower. The first two columns of

our version of the data are the lengths and the widths of the sepals. Sepals are modified leaves that are part of the calyx of a flower. In our modified version of the data set, the length is measured in millimeters and the width in centimeters. The last column encodes the species.

Figure 1: An example of an Iris versicolor taken from Wikipedia.

You may find some of the algorithms you are supposed to implement (nearest neighbor classification, cross-validation, data normalization) useful later on in the course. Thus, a reliable, general implementation is recommended.

1.1 Nearest neighbor

Nearest neighbor classification is the fundamental non-linear, non-parametric method for classification. Implement a k-nearest neighbor classifier (k-NN). Use the data `IrisTrainML.dt` as training data and report the zero-one loss on both the training data and the test data in `IrisTestML.dt` for $k = 1, 3, 5$.

Deliverables: Source code of your nearest neighbor classifier (in a separate `.zip` file); training and test results of your k-NN classifier for $k = 1, 3, 5$; short discussion of the results

1.2 Hyperparameter selection using cross-validation

The performance of the nearest neighbor classifier depends on the choice of the parameter k determining the number of neighbors. Such a parameter of the learning *algorithm* (i.e., not a parameter of a resulting model) is called a *hyperparameter*.

Cross-validation is useful to determine proper hyperparameters. You are supposed to find a good value for k from $\{1, 3, 5, \ldots, 25\}$. For every choice of k, estimate the performance of the k-NN classifier using 5-fold cross validation. Plot a graph of cross-validation loss (averaged over the 5 folds) as a function of k. Pick the k with the lowest average 0-1 loss (classification error), which we will call k_{best} in the following. Only use the training data `IrisTrainML.dt` in the cross validation process to generate the folds.

After you have determined k_{best}, you can estimate the performance of the classifier using the test data `IrisTestML.dt`.

> As a general rule, you must not use the test data in the model building process at all (neither for training, data normalization, nor hyperparameter selection), because otherwise you may get a biased estimate of the generalization performance of the model (see [1, Example 5.3]).

To estimate the generalization performance, build a k_{best}-NN classifier using the complete training data set `IrisTrainML.dt` and evaluate it on the independent test set `IrisTestML.dt`.

Deliverables: Source code (in a separate `.zip` file); a short description of how you proceeded (e.g., did the cross-validation); a plot of cross-validation error as a function of k; number of neighbors as suggested by hyperparameter selection; training and test error of k_{best}-NN. IMPORTANT: any plot in any question should come with axis labels, a legend, and a caption describing the plot!

1.3 Data normalization

Data normalization is an important preprocessing step. A basic normalization is to generate zero-mean, unit variance input data.

Consider the *training* data in `IrisTrainML.dt`. Compute the mean and the variance of every input feature (i.e., of every component of the input vector). Find the affine linear mapping $f_{\text{norm}} : \mathbb{R}^2 \to \mathbb{R}^2$ that transforms the training data such that the mean and the variance of every feature in the transformed data are 0 and 1, respectively (verify by computing these values).

Use the same function f_{norm} to also encode the test data. Compute the mean and the variance of every feature in the transformed test data.

The normalization is part of the model building process. Thus, you may only use the training data for determining f_{norm} (always remember that you are supposed to not know the test data).

Now repeat exercise 1.2 with the normalized data instead of the raw data. Perform cross-validation and report the training and test error using the value of k found by the cross-validation procedure. Can you explain the differences compared to the results achieved using the raw (not normalized) data?

Deliverables: Mean and variance of the training data; mean and variance of the transformed test data; plot of cross-validation error as a function of k; number of neighbors k_{best} as suggested by hyperparameter selection; training and test error of k_{best}-NN considering the normalized data; short discussion of the differences in performance when using normalized and raw data

2 Markov's inequality vs. Hoeffding's inequality vs. binomial bound

Let $X_1, \ldots, X_{10}$ be i.i.d. Bernoulli random variables with bias $\frac{1}{2}$.

1. Use Markov's inequality to bound the probability that $\sum_{i=1}^{10} X_i \geq 9$. (Hint: you have to define a new random variable $S = \sum_{i=1}^{10} X_i$ and apply Markov's inequality to S.)

2. Use Hoeffding's inequality to bound the probability of the same event. (Hint: now you do not need to group the variables together and you can exploit their independence.)

3. Calculate the exact probability of the above event. (Hint: S has binomial distribution.)

4. Compare the three results.

3 The effect of scale (range) and normalization of random variables in Hoeffding's inequality

Prove that Corollary 1.4 in Yevgeny's lecture notes (simplified Hoeffding's inequality for random variables in the $[0, 1]$ interval) follows from Theorem 1.2 (general Hoeffding's inequality).

4 Basic Statistics

1. An airline knows that any person making a reservation on a certain flight will not show up with a probability of 0.05 (5 percent). They introduce a policy to sell 100 tickets for a flight that can hold only 99 passengers. Bound the probability that the number of people that show up for a flight will be larger than the number of seats (assuming they show up independently).

2. An airline has collected an i.i.d. sample of 10000 flight reservations and figured out that in this sample 5 percent of passengers who made a reservation on a certain flight did not show up. They introduce a policy to sell 100 tickets for a flight that can hold only 99 passengers. Bound the probability of observing such sample and getting a flight overbooked. (Hint: there is a true probability p for showing up on a flight, but it is unknown. Introduce an auxiliary variable $\alpha \in (0.95, 1)$ and bound the probability of observing a sample of 10000 reservations with 95% of show-ups and a sample of 100 reservations with 100% show-ups when $p < \alpha$ and when $p > \alpha$. Play with α to get as good bound as you can.)

5 Linear regression

Consider the data in the file `DanWood.dt`, which contains measurements from an experiment originally described by Keeping [5] and used in several textbooks [3, 6]. The data is one of the statistical reference data sets of the US American *National Institute of Standards and Technology*.

The experiment studies a carbon filament lamp. For a given absolute temperature of the carbon filament the energy radiated from the filament was recorded. Temperature was measured in units of 1000 degrees Kelvin and energy radiation per cm^2 per second.

Each line in `DanWood.dt` is one training pattern. The first number is the input (x), the absolute temperature, and the second the corresponding output / target (y), the radiated energy.

Question 5.1.

1. Implement the linear regression algorithm as described in the lecture. For vector and matrix operations such as computing the inverse of a matrix you can use high-level (library) functions.

2. Build an *affine* linear model $h : \mathbb{R} \to \mathbb{R}$ of the data described above using linear regression. Report the two parameters of the model as well as the mean-squared-error of the model computed over the complete data set.

3. Plot the data and the regression line. The plot must have proper axes labels and a legend.

Deliverables: source code, plot of the data and the regression line, mean-squared error, parameters of the regression model

The next exercise trains working with the math behind linear regression. We use the notation introduced in the lecture, see also section 3.2.1 in the textbook [1]. If we apply the result of linear regression to the training data stored in the data matrix $\boldsymbol{X}$ (we drop the tilde for readability), we get the vector of predicted labels $\hat{\boldsymbol{y}}$ by

$$\hat{\boldsymbol{y}} = \boldsymbol{X} w^\star = \boldsymbol{X} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{y} \ .$$

The matrix $\boldsymbol{H} = \boldsymbol{X} \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}$ is called the *hat matrix* [1], because it turns $\boldsymbol{y}$ into $\hat{\boldsymbol{y}}$.

Question 5.2 ([1], exercise 3.3(a,b)).

1. Show that $\boldsymbol{H}$ is symmetric. Recall that $(\boldsymbol{AB})^{\mathrm{T}} = \boldsymbol{B}^{\mathrm{T}} \boldsymbol{A}^{\mathrm{T}}$
2. Show that $\boldsymbol{H}^k = \boldsymbol{H}$ for any positive integer power k.

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from Data*. AMLbook, 2012.

[2] E. Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936.

[3] C. Daniel and F. S. Wood. *Fitting Equations to Data*, pages 428–431. John Wiley and Sons, 1980.

[4] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[5] E. S. Keeping. *Introduction to Statistical Inference*, page 354. Van Nostrand Company, Princeton, NJ, 1962.

[6] A. B. Shiflet and G. W. Shiflet. *Introduction to Computational Science: Modeling and Simulation for the Sciences*. Princeton University, 2006.