Statistical Methods for Machine Learning 2015, Assignment I:

Foundations

Christian Igel, Kim Steenstrup Pedersen
Department of Computer Science, University of Copenhagen

The goal of this assignment is to get familiar with the mathematical foundations of machine learning and basic supervised learning methods.

You have to pass this and the following mandatory assignments in order to be eligible for the exam of this course. There are in total 3 mandatory pass/fail assignments on this course, which can be solved individually or in groups of no more than 3 participants. The course will end with a larger written exam assignment which must be solved individually and is graded (7-point scale).

The deadline for this assignment is **Tues. 17/02/2015**. You must submit your solution electronically via the Absalon home page. Go to the assignments list and choose this assignment and upload your solution prior to the deadline. If you choose to work in groups on this assignment you should only upload one solution, but remember to include the names of all participants both in the solution as well as in Absalon when you submit the solution. If you do not pass the assignment, having made a *serious* attempt, you may get a second chance of submitting a new solution.

A solution consists of:

- 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 in 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. If we cannot make your code run we will consider your submission incomplete and you may be asked to resubmit.

- A PDF file with notes detailing your answers to the questions, which may include graphs and tables if needed (**max. 10 pages** text including figures and tables). Do *not* include your source code in this PDF file.

Software tips

The choice of programming is up to you, but we recommend you choose a language you are familiar with.

If you have knowledge in C++, we recommend the Shark machine learning library `http://image.diku.dk/shark` for professional machine learning applications.

There are several Python learning libraries.

If you wish to use R, you may like the book by James et al. [4]. This book is recommended as supplementary reading material to all students.

If you wish to use Matlab, the University of Copenhagen has a license agreement with MathWorks that allows students to download and install a copy of Matlab on personal computers. On the KUnet web site you find a menu item called Software Library (Softwarebiblioteket) `https://intranet.ku.dk/selvbetjening/Sider/Software-bibliotek.aspx`, where you find a link to The Mathworks – Matlab & Simulink + toolboxes.

I.1 Online Math Quiz

You need basic linear algebra and calculus for understanding machine learning. To recall some of your math knowledge, go to Absalon, choose the Math Quiz under Assignments and answer the questions. Do the calculations by hand.

If you feel unsure about what a question means and how to answer it, this indicates that you are not fully comfortable with the mathematical skills that are assumed in this course. No worries. In this case, just take your time and go back to your notes from school or your first study years – or grab one of the numerous textbooks that are around.

Feel free to ask questions about the mathematical background in the exercise classes!

I.2 Probability and Parameter Estimation

We will use a multivariate Gaussian (or normal) distribution as running example in the following. To be able to visualize results, we will work in one and two dimensions. But notice that all the following questions also apply to higher dimensions. In these exercises, you should not use the built-in Gaussian functions in, e.g., Matlab, but generate your own.

I.2.1 Univariate Gaussian distributions

We start out by making a plot (or plots) of the 1-dimensional Gaussian distribution function (see e.g. section 2.3 in [2]) using the mean and standard deviation parameter values $(\mu, \sigma) = (-1, 1), (0, 2)$, and $(2, 3)$.

Deliverables: Source code; plots for the three different parameter settings

I.2.2 Sampling from a multivariate Gaussian distribution

In this question we will generate a data set consisting of $N = 100$ 2-dimensional samples $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$ drawn from a 2-dimensional Gaussian distribution $\mathscr{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean

$$\boldsymbol{\mu} = (1, 2)^T$$

and covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} 0.3 & 0.2 \\ 0.2 & 0.2 \end{pmatrix} .$$

For this you can use the Matlab function `randn`, the Python function `numpy.random.randn`, or similarly in R use `rnorm`, which can generate a matrix of random numbers sampled from a Gaussian distribution with mean zero and unit variance. For sampling from a univariate normal distribution you can use `shark::Normal`. As explained by Bishop ([2], page 528) and the lecture slides we can draw a sample $\mathbf{y}$ from $\mathscr{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ by the following linear transformation

$$\mathbf{y} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z} ,$$

where $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$ and $\mathbf{z}$ is a vector of independent zero mean unit variance normal random numbers. There exist a number of numerical linear algebra methods to find an $\mathbf{L}$ such that $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$. One of the most efficient is the Cholesky transformation (`chol` in both Matlab and R, but in matlab use second argument `'lower'` and be aware that R returns the transposed matrix). In Python you can use `numpy.linalg.cholesky`. Shark provide several variants of Cholesky decompositions, the basic one being `shark::blas::choleskyDecomposition`.

In this assignment, you may *not* use direct sampling from a multi-variate Gaussian distribution as provided in many languages, such as in C++ with Shark using `shark::MultiVariateNormalDistribution`, or in Python using either `random.gauss`, `random.normalvariate`, `numpy.random.normal`, or similar.

Plot your dataset. For this you can use the function `plot` in Matlab, Python, or R, or a tool such as gnuplot.

Deliverables: Source code; plot of dataset

I.2.3 Means

Estimate the maximum likelihood sample mean of the data set using Eq. 2.121 [2]. Plot the sample mean and distribution mean as points in a 2-dimensional plot together with the data points. Quantify how much the sample mean deviate from the distribution mean. Why do you see a deviation from the distribution mean?

Deliverables: Plot of data and the two means; quantitative evaluation of deviation and explanation of what the quantity means; concise discussion of the reason for the deviation

I.2.4 Covariance: The geometry of multivariate Gaussian distributions

Estimate the maximum likelihood sample covariance $\mathbf{\Sigma}_{\mathbf{ML}}$ of the data set using Eq. 2.122 from [2], and compute the eigenvalues λ_i and normalized eigenvectors $\boldsymbol{e}_i$ of $\mathbf{\Sigma}_{\mathbf{ML}}$. This can be done with the function `eig` in Matlab or `eigen` in R. In Python you can use either `numpy.linalg.eig` or `numpy.linalg.eigh`. The tutorial "LinAlg: Eigenvalues, Inverses and Systems of Linear Equations" briefly explains how to compute eigenvectors and eigenvalues using Shark.

Plot the scaled and translated eigenvectors

$$\boldsymbol{\mu} + \sqrt{\lambda_i}\boldsymbol{e}_i$$

on top of the sampled data from Question I.2.2. Discuss how to interpret the eigenvectors and eigenvalues of $\mathbf{\Sigma}_{\mathbf{ML}}$.

Rotation matrices are of the form

$$\boldsymbol{R}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

and for any vector $\mathbf{z} = [z_1, z_2]^T$, the matrix-vector-product $\boldsymbol{R}_\theta\mathbf{z}$ returns the vector $\mathbf{z}$ rotated by θ degrees around the origin.

Generate new, rotated covariance matrices

$$\mathbf{\Sigma}_\theta = \boldsymbol{R}_\theta^{-1} \cdot \mathbf{\Sigma}_{\mathbf{ML}} \cdot \boldsymbol{R}_\theta.$$

for $\theta = 30, 60$ and 90 degrees, and resample the rotated Gaussian distribution $\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \mathbf{\Sigma}_\theta)$ as in Question I.2.2 (you can keep the same $\mathbf{z}$). Plot the samples from the rotated distributions in different colors in a combined plot. Explain what you see.

Can you find an angle ϕ such that the main direction of the data sampled from the Gaussian distribution $\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_\phi)$ points along the x axis?

Deliverables: Covariance $\boldsymbol{\Sigma}_{\mathbf{ML}}$, plot of data and eigenvectors; discussion of interpretation of eigenvectors and eigenvalues of $\boldsymbol{\Sigma}_{\mathbf{ML}}$; combined plot of rotated datasets; angle ϕ; source code for computing all of the above

I.3 Classification

Please download the data sets `IrisTrain2014.dt` and `IrisTest2014.dt` from the course homepage. These data sets have been generated from the famous Iris flower data set [1], which has been used as an example for classification algorithms since the work of Ronald Fisher [3]. 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.

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.

I.3.1 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 `IrisTrain2014.dt` as training data and report the accuracy (one minus the 0-1 loss) of the corresponding classifier on both the training data and the test data in `IrisTest2014.dt` for $k = 1, 3, 5$.

Deliverables: Source code of your nearest neighbor classifier; training and test results of your k-NN classifier for $k = 1, 3, 5$; short discussion of the results

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

I.3.2 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 (see section 1.3 in [2]). 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 `IrisTrain2014.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 `IrisTest2014.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.

To estimate the generalization performance, build a k_{best}-NN classifier using the complete training data set `IrisTrain2014.dt` and evaluate it on the independent test set `IrisTest2014.dt`.

Deliverables: Source code; a short description of how you proceeded (e.g., did the cross-validation); number of neighbors as suggested by hyperparameter selection; training and test error of k_{best}-NN

I.3.3 Data normalization

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

Consider the *training* data in `IrisTrain2014.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 I.3.2 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; 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

References

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

[2] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

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

[4] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning with Applications in R*, volume 103 of *Springer Texts in Statistics.* Springer, 2013.