Statistical Methods for Machine Learning

January 31, 2013

Case 1: Foundations of statistical machine learning

This assignment is based on the content of C. M. Bishop *Pattern Recognition and Machine Learning* chapters 1, 2, and 11 (see lecture schedule in Absalon for details). The goal is for you to get familiar with programming for machine learning and some key concepts from statistics and probability theory needed throughout the course.

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 Tuesday 19/2 2012. 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 will get a second chance of submitting a new solution.

A solution consist of:

- Your solution source code (Matlab / R / Python scripts / C or C++ / Java code) with comments about the major steps involved in each Question (see below).
- 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.
- 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 non-programming 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.

New to the DIKU system?

If you wish you can use the DIKU system during this course, however this is not a requirement.

Access to DIKU e.g. from your laptop: All access to the DIKU system is achieved using SSH. You can connect to either of the three hosts:

```
ask.diku.dk
tyr.diku.dk
brok.diku.dk
```

From here you can log on to other DIKU machines. Execute the following command:

```
besthost kand
```

or

```
besthost bach
```

or

```
besthost app
```

Start matlab or R by executing the command:

```
matlab
```

or

```
R
```

Using your own computer

If you wish to use Matlab, the University of Copenhagen has a license agreement with MathWorks that allow 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/Sider/Software-bibliotek.aspx`. Under this menu item you can find a link to The Mathworks - Matlab & Simulink + toolboxes. Click this link and follow the instructions for how to install on your own computer.

If you wish to program your solutions in C++ we recommend the Shark machine learning library `http://image.diku.dk/shark/sphinx_pages/build/html/index.html` which should work on Windows, Linux and MacOSX. You need to have CMake and the boost library installed before you install Shark.

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.

Question 1.1

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

Question 1.2

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, with mean

$$\mu = (1, 1)^T$$

and covariance matrix

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

For this you can use the Matlab function `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. As explained in Bishop page 528 and the lecture slides we can draw a sample $\mathbf{y}$ by the following linear transformation

$$\mathbf{y} = \mu + \mathbf{Lz} ,$$

where $\mathbf{\Sigma} = \mathbf{LL^T}$ and $\mathtt{z}$ is a vector of independent zero mean unit variance normal random numbers. There exists a number of numerical linear algebra methods to find an $\mathbf{L}$ such that $\mathbf{\Sigma} = \mathbf{LL^T}$. 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).

Question 1.3

Estimate the maximum likelihood sample mean and sample covariance of the data set using Bishop Eq. 2.121 and 2.122. Plot the sample mean and correct mean as points in a 2-dimensional plot together with the data points. For this you can use the function `plot` in Matlab or R. Quantify how much the sample mean deviate from the correct mean. Why do you see a deviation from the correct mean?

The Gaussian distribution and its conditional and marginal distributions

Question 1.4 (based on CB Ex. 2.25)

In CB Sec. 2.3.1 and 2.3.2, we considered the conditional and marginal distributions for a multivariate Gaussian. More generally, we can consider a partitioning of the components of $\mathtt{x}$ into three groups $\mathbf{x}_a, \mathbf{x}_b$, and $\mathbf{x}_c$, with a corresponding partitioning of the mean μ and of the covariance $\mathbf{\Sigma}$ in the form

$$\mu = \begin{pmatrix} \mu_a \\ \mu_b \\ \mu_c \end{pmatrix} , \quad \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{aa} & \mathbf{\Sigma}_{ab} & \mathbf{\Sigma}_{ac} \\ \mathbf{\Sigma}_{ba} & \mathbf{\Sigma}_{bb} & \mathbf{\Sigma}_{bc} \\ \mathbf{\Sigma}_{ca} & \mathbf{\Sigma}_{cb} & \mathbf{\Sigma}_{cc} \end{pmatrix}$$

By use of the results of CB Sec. 2.3, find an expression for the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$ in which $\mathbf{x}_c$ has been marginalized out.

Non-parametric estimation – histograms

Question 1.5

Make a histogram based probability density estimate of each of the marginal distributions $p(x_1)$ and $p(x_2)$ of the data set generated in Question 1.2. For this you can use the `hist` function in Matlab or R. How does changing the bin width (or equivalently, the number of bins) affect the plot and the estimate of $p(x_1)$ and $p(x_2)$?

Question 1.6

Plot your histogram estimate of the marginal distribution $p(x_1)$ together with the analytical solution given the mean and covariance parameters stated in Question 1.2 (Hint: See CB section 2.3.2). Write the analytical expression for the marginal distribution $p(x_1)$.

Question 1.7

Make a 2-dimensional histogram estimate of $p(\mathbf{x})$ and plot the probability distribution $p(\mathbf{x})$ of the data set using either the matlab function `hist3` or the `hist2d` function in R. Try with $N = 100, 1.000$, and 10.000. Try also to vary the number of bins by using 10×10, 15×15, and 20×20 for $N = 1.000$. Can you explain the results?

Sampling Methods

Sampling (or Monte Carlo) methods form a general and useful set of techniques that use random numbers to extract information about (multivariate) distributions and functions. In the context of statistical machine learning, we are most often concerned with drawing samples from distributions to obtain estimates of summary statistics such as the mean value of the distribution in question.

Question 1.8

When we have access to a uniform (pseudo) random number generator on the unit interval (`rand` in Matlab or `runif` in R) then we can use the transformation sampling method described in Bishop Sec. 11.1.1 to draw samples from more complex distributions. Implement the transformation method for the exponential distribution

$$p(y) = \lambda \exp(-\lambda y) \ , \ y \geq 0$$

using the expressions given at the bottom of page 526 in Bishop.

The crucial point of sampling methods is how many samples are needed to obtain a reliable estimate of the quantity of interest. Let us say we are interested in estimating the mean, which is $\mu_y = \frac{1}{\lambda}$ in the above distribution, we then use the sample mean $\widehat{y} = \frac{1}{L} \sum_{l=1}^{L} y^{(l)}$ of the L samples as our estimator. Since we can generate as many samples of size L as we want, we can investigate how this estimate on average converges to the true mean. To do this properly we need to take the absolute difference $|\mu_y - \widehat{y}|$ between the true mean μ_y and estimate $\widehat{y}$ averaged over many, say 1000, repetitions for several values of L, say 10, 100, 1000. Plot the expected absolute deviation as a function of L. Can you plot some transformed value of expected absolute deviation to get a more or less straight line and what does this mean?

Real data - simple color-based object detection

In this part we will work with real data in the form of photographic color images and make a simple color-based object detection algorithm. Here we define object detection as the task of localizing an object in an image (not necessarily inferring whether or not the object is present). That is, we want to answer the question - where is the object?

A digital color image is represented as an array of elements called pixels. At each pixel we represent the color of that pixel by a vector of RGB values. RGB is short for the Red, Green and Blue channel and each channel represents a continuous quantity related to the spectral energy in the channel. The RGB vector can therefore be represented by a 3-dimensional vector $\mathbf{x} = (x_r, x_g, x_b)^T \in \mathbb{R}^3$.

Lets assume that the object we are interested in detecting is reasonably uniformly colored and without texture or structure, and that the random color variation on the object can be described by a Gaussian distribution (a very crude assumption in practice). In this case we might assume that all pixels on the object are i.i.d. Gaussian with probability density (again a crude assumption)

$$p(\mathbf{x}|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}}) = \frac{1}{(2\pi)^{3/2}|\mathbf{\Sigma}_{\mathrm{rgb}}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_{\mathrm{rgb}})^T \mathbf{\Sigma}_{\mathrm{rgb}}^{-1}(\mathbf{x} - \mu_{\mathrm{rgb}})\right) \ .$$

Hence we assume the object has an average color μ_{rgb} and that the color variation across the object may be described by the covariance matrix $\mathbf{\Sigma}_{\mathrm{rgb}}$.

Given a training set of pixels on the type of object in question we can now learn a Gaussian probability model of the object color. Here we will consider the maximum likelihood estimate for the parameters $\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}}$ (just as in Question 1.4).

After estimating the model parameters we can for each pixel compute the probability density $p(\mathbf{x}|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}})$. If a pixel $\mathbf{x}$ has a high density value it means that its color is similar to the object being modeled.

Once we have this probability model we can perform various forms of inference: In this assignment we will use it to make a primitive form of object detection.

Aside: We could also make a simple segmentation (i.e. divide the image into perceptually meaningful regions) by pixel classification by assigning the label 1 if the probability is above a certain threshold, otherwise assign the label 0. However, we will return to this type of problems later in the course.

Question 1.9

Consider the image called `kande1` (use e.g. `imread` in Matlab and the `JPG` file, and use the `pixmap` package in R and the `pnm` file) shown in Fig. 1 and let us try to detect the red pitcher. In order to estimate the model parameters we need a training data set and we pick a square region inside the pitcher with a good representation of the color variation of the object. We will use the following region:

Let the image coordinate system have coordinate $(1, 1)$ at the upper left corner of the image and the y axis is inverted so that its positive direction points downwards (this is how Matlab handles images). Then

Training set: All pixels in the rectangle specified by the lower left corner $(328, 150)$ and upper right corner $(264, 330)$.

(See the region in Fig. 1 and the Matlab script `opg19.m` or the R script `opg19.R`.)

Write a program that implements the above sketched probability model using the maximum likelihood solution on the training set. That is, use the maximum likelihood estimators for μ_{rgb} and $\mathbf{\Sigma}_{\mathrm{rgb}}$. For each pixel in the image compute the probability density $p(\mathbf{x}|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}})$ and visualize it as an image. Comment on the result.

Question 1.10

Next lets try to detect the object using $p(\mathbf{x}|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}})$. We can make a simple algorithm by computing the weighted average position — the center of mass — of the image under the probability model and use this as our estimate of the object position.

Up to now we have referred to pixels $\mathbf{x}$ as just having a vector value — its color — but a pixel is also located somewhere in the image and therefore it also has a position. Let $q \in \mathbb{Z}^2$ denote a position and let $\mathbf{x}_q$ denote the color value of the pixel at that position. Using this notation we can compute the weighted average position as

$$\widehat{q} = \frac{1}{Z} \sum_q q \cdot p(\mathbf{x}_q|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}}) \ ,$$

where the sum runs over all pixels in the image and $Z = \sum_q p(\mathbf{x}_q|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}})$ is a normalization constant. Plot $\widehat{q}$ on top of the image and comment on the result.

We can also get an idea of the spread of the object by computing the spatial covariance

$$\mathbf{C} = \frac{1}{Z} \sum_q (q - \widehat{q})(q - \widehat{q})^T \cdot p(\mathbf{x}_q|\mu_{\mathrm{rgb}}, \mathbf{\Sigma}_{\mathrm{rgb}}) \ .$$

Plot $\mathbf{C}$ by plotting iso-probability curves of the corresponding 2 dimensional Gaussian distribution together with $\widehat{q}$ on top of the image and comment on the result. For this you may use the function `plot_results` (hand- out code for both matlab and R).

Question 1.11

Can your probability model generalize to other images of the same pitcher, but captured under different lighting? Investigate this by considering how well your model perform on the image called `kande2`. Explain what you have done and comment on the results.

Sami Brandt and Christian Igel, January 2013.

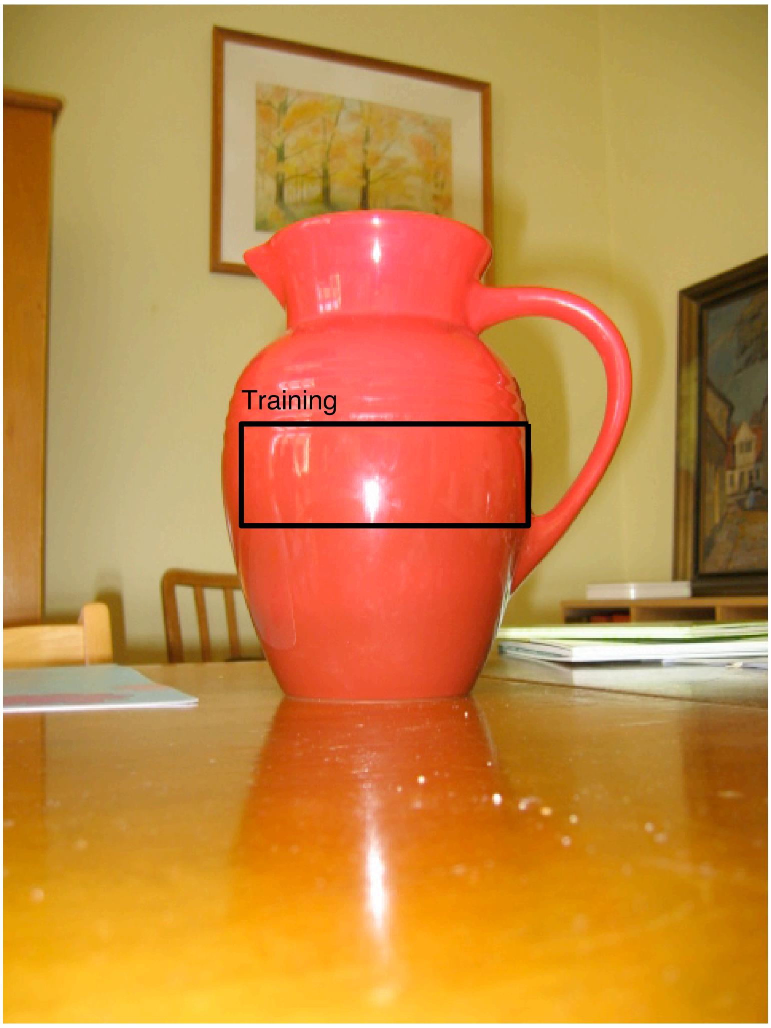

Figure 1: Visualisation of kande1 as well as the training region.