Statistical Machine Learning 2017

Assignment 4
Deadline: 14th of January 2018

Instructions:

- You can work **alone or in pairs** (= max 2 people). **Write the full name and S/U-number of all team members on the first page of the report.**
- Write a **self-contained report** with the answers to each question, **including** comments, derivations, explanations, graphs, etc. This means that the elements and/or intermediate steps required to derive the answer have to be in the report. (Answers like 'No' or 'x=27.2' by themselves are not sufficient, even when they are the result of running your code.)
- If an exercise specifically asks for code, put **essential code snippets** in your answer to the question in the report, and explain briefly what the code does. In addition, hand in **complete (working and documented) source code** (MATLAB recommended, other languages are allowed but not "supported").
- In order to avoid extremely verbose or poorly formatted reports, we impose a **maximum page limit** of 40 (2×20) pages, including plots and code, with the following formatting: fixed **font size** of 11pt on an **A4 paper**; **margins** fixed to 2cm on all sides. All figures should have axis labels and a caption or title that states to which exercise (and part) they belong.
- Upload reports to **Blackboard** as a **single pdf** file: '`SML_A4_<Namestudent(s)>.pdf`' and one zip-file with the executable source/data files (e.g. matlab m-files). For those working in pairs, only one team member should upload the solutions.
- Assignment 4 consists of 4 exercises, weighted as follows: 6 points, 5 points, 5 points, and 4 points. Note that this assignment counts double compared to the previous assignments. The **grading** will be based solely on the report pdf file. The source files are considered supplementary material (e.g. to verify that you indeed did the coding).
- For any problems or questions, send us an email, or just ask.
  Email addresses: `tomc@cs.ru.nl` and `b.kappen@science.ru.nl`

Exercise 1 – Logistic regression (weight 6)

Part 1 – The IRLS algorithm

Many machine learning problems require minimizing some function $f(\mathbf{x})$. For this, an alternative to the familiar gradient descent technique, is the so called Newton-Raphson iterative method:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \mathbf{H}^{-1}\nabla f(\mathbf{x}^{(n)}) \tag{1}$$

where $\mathbf{H}$ represents the Hessian matrix of second derivatives of $f(\mathbf{x})$, see Bishop, §4.3.3.

1. Derive an expression for the minimization of the function $f(x) = \sin(x)$, using the Newton-Raphson iterative optimization scheme (1), and verify (using Matlab, just up to, e.g., five iterations) how quickly it converges when starting from $x^{(0)} = 1$. What happens when you start from $x^{(0)} = -1$?
   Hint: The Hessian of a 1-dimensional function $f(x)$ is just the second derivative f''. So, the Newton-Raphson iterative method reduces in 1-d to

$$x^{(n+1)} = x^{(n)} - \frac{f'(x^{(n)})}{f''(x^{(n)})} \tag{2}$$

We want to apply this method to the logistic regression model for classification (see Bishop, §4.3.2):

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) \tag{3}$$

For a data set $\{\phi_n, t_n\}_{n=1}^N$, with $t_n \in \{0, 1\}$, using $y_n = p(\mathcal{C}_1|\phi_n)$ the corresponding cross entropy error function to minimize is

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \tag{4}$$

With one basis function ϕ and the dummy basis function 1, the feature vector in (3) becomes $\boldsymbol{\phi} = [1, \phi]^{\mathrm{T}}$. The weight vector including the bias term is then also two dimensional, $\mathbf{w} = [w_0, w_1]^{\mathrm{T}}$. Expressions for the gradient $\nabla E(\mathbf{w})$ and Hessian $\mathbf{H}$ in terms of the data set are given in Bishop, eq.4.96-98. As both are implicitly dependent on the weights $\mathbf{w}$, they have to be recalculated after each step: hence this is known as the 'Iterative Reweighted Least Squares' algorithm.

Consider the following data set: $\{\phi_1, t_1\} = \{0.3, 1\}$, $\{\phi_2, t_2\} = \{0.44, 0\}$, $\{\phi_3, t_3\} = \{0.46, 1\}$ and $\{\phi_4, t_4\} = \{0.6, 0\}$, and initial weight vector $\mathbf{w}^{(0)} = [1.0, 1.0]^{\mathrm{T}}$.

2. Show using e.g. a Matlab implementation that for this situation the IRLS algorithm converges in a few iterations to the optimal solution $\widehat{\mathbf{w}}^{\mathrm{T}} \approx [9.8, -21.7]$, and show that this solution corresponding to a decision boundary $\phi = 0.45$ in the logistic regression model.
   (The IRLS algorithm should take about five lines of Matlab code inside a loop + initialization).

Part 2 – Two-class classification using logistic regression

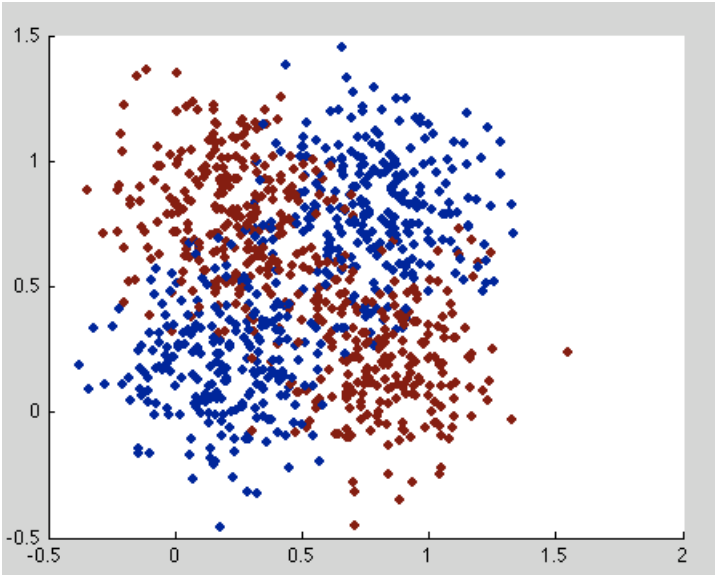

Figure 2.1 - Two class data for logistic regression.

Two-class classification using logistic regression in the IRLS algorithm. The data consists of 1000 pairs $\{x_1, x_2\}$ with corresponding class labels $\mathcal{C}_1 = 0$ or $\mathcal{C}_2 = 1$. Load it into Matlab using

```
data = load('a010_irlsdata.txt','-ASCII');
X = data(:,1:2); C = data(:,3);
```

1. Make a scatter plot of the data, similar to Figure 2.1. (Have a look at Matlab file `a010plotideas.m` in Blackboard for some ideas to make such a scatter plot and the plots later on.) Do you think logistic regression can be a good approach to classification for this type of data? Explain why.

2. Modify the Iterative Reweighted Least Squares algorithm from part 1 to calculate the optimal weights for this data set. Use again a dummy basis function. Initialize with the weight vector $\mathbf{w}^{\mathrm{T}} = [0, 0, 0]$. With these initial weights, what are the class probabilities according to the logistic regression model (i.e., before optimization)?

3. Run the algorithm. Make a scatter plot of the data, similar to Figure 2.1, but now with colors that represent the data point probabilities $P(C = 1|X_n)$ according to the model after optimization. Compare the cross entropy error with the initial value. Did it improve? Much? Explain your findings.

4. Introduce two Gaussian basis functions as features ϕ_1, ϕ_2, similar to Bishop, fig.4.12. Use identical, isotropic covariance matrices $\Sigma = \sigma^2 I$ with $\sigma^2 = 0.2$, and center the basis functions around $\boldsymbol{\mu}_1 = (0, 0)$ and $\boldsymbol{\mu}_2 = (1, 1)$. Make a scatter plot of the data in the feature domain. Do you think logistic regression can be a good approach to classification with these features? Explain why.

5. Modify the IRLS algorithm to use the features $\{\phi_1, \phi_2\}$ and the dummy basis function. Initialize with the weight vector $\mathbf{w}^{\mathrm{T}} = [0, 0, 0]$.

   Run the algorithm. Make a scatter plot of the data, similar to Figure 2.1, but now with colors that represent the data point probabilities $P(C = 1|X_n)$ according to this second model (after optimization). Compare the cross entropy error with the initial value. Did it improve? Much? Explain your findings.

Exercise 2 – EM and doping (weight 5)

In a certain hypothetical sport, banned substance 'X' has become popular as a performance enhancing drug, as its presence is hard to establish in blood samples directly. Recently, it has been discovered that users of the drug tend to show a strong positive correlation between concentrations of two other quantities, x_1 and x_2, present in the blood. In contrast, 'clean' athletes tend to fall in one of two or three groups, that either show no or a negative correlation between x_1 and x_2. Unfortunately, as each sample contains only a single, instantaneous, measurement for each variable, it is not possible to establish this correlation from the sample. However, in many cases it is possible to distinguish to which *class* a certain sample belongs by also looking at the values of two other measured variables, x_3 and x_4: certain combinations of measured values are often typical for one class but highly unusual for others.
After a high profile event, a large scale test has resulted in 2000 samples. Rumours suggest the number of positives could be as high as 20%. However, the exact relationship between different classes and typical $\mathbf{x}$ values is still not clear. This is where the EM-algorithm comes in ...

The blood sample measurements are modelled as a mixture of K Gaussians, one for each class

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{5}$$

where $\mathbf{x} = [x_1, x_2, x_3, x_4]$ represents the values for the measured quantities in the blood sample, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ and $\boldsymbol{\Sigma} = \{\Sigma_1, \ldots, \Sigma_K\}$ are the means and covariance matrices of the Gaussians for each class, and $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}$ are the mixing coefficients in the overall data set.

Load the data using

```
X = load('a011_mixdata.txt', '-ASCII');
```

and set N to the number of datapoints and D to the number of variables in the data set X.

1. Try to give an estimate of the number, size and shape of the classes in the data, by plotting the distribution of the variables, e.g using `hist()`, `scatter()` or `scatter3()`.

2. Implement an EM-algorithm using the description and formulas given in Bishop, §9.2.2.
   Use variable K for the number of classes and choose a priori equal mixing coefficients π_k. Initialize the means $\boldsymbol{\mu}_k$ to random values around the sample mean of each variable, e.g. set $\mu_{k,1}$ to $\bar{x}_1 + [-1 \leq \epsilon \leq +1]$. Initialize the $\boldsymbol{\Sigma}_k$ to diagonal matrices with reasonably high variances, e.g. `4*rand()+2`, to avoid very small responsibilities in the first step. Make sure the EM-loop runs over at least 100 iterations. Display relevant quantities, at least the log likelihood (9.28), after each step so you can monitor progress and convergence. Write a plot routine that plots the x_1, x_2 coordinates of the data points, and color each data point according to the most probable component according to the mixture model.

3. Set $K = 2$, initialize your random generator and run the EM-algorithm on the data. Describe what happens. Try different random initializations and compare results.
   (Should converge within 50 steps to two clusters, accounting for ±1/3 resp. 2/3 of the data).
   Plot the x_1, x_2 coordinates colored according to the most probable component. Compute the correlation coefficients
   $$\rho_{12} = \frac{\mathrm{cov}[x_1, x_2]}{\sqrt{\mathrm{var}[x_1]\mathrm{var}[x_2]}} \tag{6}$$
   of each of the components (i.e., use their covariance matrices to compute variances and covariances in (6), see also (Bishop, eq. (2.93)). Does either class show the characteristic strong[1] positive correlation for $\{x_1, x_2\}$?

[1] According to Wikipedia, the correlation is none if $|\rho| < 0.1$, small if $0.1 < |\rho| < 0.3$, medium if $0.3 < |\rho| < 0.5$ and strong if $|\rho| > 0.5$

4. Increase the number of classes to $K = 3$ and rerun your algorithm on the data, again trying different random initializations. Plot the x_1, x_2 coordinates colored according to the most probable component and compute the correlation coefficients of each of the components. Check both your plot and your coefficients if one of the clusters now displays the strong positive $\{x_1, x_2\}$ correlation we are looking for.
   Increase to $K = 4$, do the same, and see if this improves your result (in terms of detection of the doping-cluster). Based on your findings, is the rumoured 1-in-5 estimate for users of X credible?

Having found the offending cluster in the data using the EM-algorithm, we are now presented with four samples $\{A, B, C, D\}$, with values for $[x_1, x_2, x_3, x_4]$ given as:

$$\begin{aligned} \text{A} &= [11.85, 2.2, 0.5, 4.0] \\ \text{B} &= [11.95, 3.1, 0.0, 1.0] \\ \text{C} &= [12.00, 2.5, 0.0, 2.0] \\ \text{D} &= [12.00, 3.0, 1.0, 6.3] \end{aligned}$$

One of these is from a subject who took drug X, and one is from a subject who tried to tamper with the test by artificially altering one or more of the x_i levels in his/her blood sample.

5. Identify which sample belongs to the suspected user and which one belongs to the 'fraud'.

Exercise 3 – Handwritten digit recognition (weight 5)

We use the EM-algorithm to tackle the problem of recognizing real, handwritten digits. The dataset contains 800 digital images of handwritten numbers '2', '3' and '4', derived from the MNIST dataset (Bishop, app.A). Each image consists of 28x28 binary pixels that are either black (= 1) or white (= 0). The labels belonging to the images are (initially) unknown.
The bitmapped images are modelled with a Bernoulli mixture model (Bishop §9.3.3):

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{(1-x_i)} \tag{7}$$

where x_i is the value of pixel i in an image, μ_{ki} represents the probability that pixel i in class k is black, and $\{\pi_1, \ldots, \pi_K\}$ are the mixing coefficients of classes in the data. We want to use this data set to classify new images of handwritten numbers '2', '3' and '4'.

Load the binary image data using the following commands:

```
N = 800; D = 28*28; X = uint8(zeros(N,D));
fid = fopen ('a012_images.dat', 'r');
for i = 1:N
 X(i,:)  = fread(fid, [D], 'uint8');
end;
status = fclose(fid);
```

1. Use `image();` to display some of the handwritten digits in the dataset (reshape vector `X(i,:)` into a square matrix and use '`BW_map=[1,1,1; 0,0,0]; colormap(BW_map);`'). Do you think it will be possible to classify them correctly with the right class prototypes?

2. Implement an EM-algorithm for the Bernoulli mixture model, as described in Bishop, §9.3.3 (or better: copy and modify the one from exercise 2). Again, use variable $K = 3$ for the number of classes and choose a priori equal mixing coefficients π_k. Initialize the μ_{ki}, the pixel means for each of the class prototypes, to random values in the interval $[0.25, 0.75]$.

Let the algorithm run for at least 40 steps. To monitor progress and convergence, display the class images $\boldsymbol{\mu}$ after each step. Do this by mapping the pixel means to integer values in the range [0,255] and display as a greyscale image using `colormap(gray(255));`

Note: The likelihood terms per class in eq.9.56 can become extremely small. To counter this problem you can work with the logarithms, or you can process an image pixel-by-pixel for all classes and renormalize after every few steps.

3. Run the EM-algorithm on the image data set. Describe process and result. Compare the effect of different random initializations.

4. If you have the algorithm running, then try to

   - start with more/less classes and see what happens (= describe & explain)
   - use the image labels in 'a012_labels.dat' to identify some misclassified images and see if you understand why
   - initialize the three classes with the true values and see what happens

   How could you improve the overall performance for this dataset?

   Note: The image labels can be loaded, using

   ```
   fid = fopen ('a012_labels.dat', 'r');
   Z = fread(fid, N, 'uint8');
   ```

5. Create a handwritten digit image of your own and see if it is classified correctly by the class prototypes $\boldsymbol{\mu}$ from your EM-algorithm.

Exercise 4 – Gaussian processes for regression (weight 4)

We would like to apply Gaussian process models to the problem of regression (Bishop 6.4.2). We consider a noisy model of the form:

$$t_n = y_n + \epsilon_n, \tag{8}$$

where $y_n = y(\mathbf{x}_n)$ and ϵ_n are i.i.d. samples from a random noise variable on the observed target values. Furthermore, we assume that the noise process has a Gaussian distribution given by:

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{9}$$

One widely used kernel function for Gaussian process regression is given by the exponential of a quadratic form, with the addition of constant and linear terms (eq. 6.63 Bishop):

$$k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left(-\frac{\theta_1}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \theta_2 + \theta_3 \mathbf{x}^T \mathbf{x}' \tag{10}$$

We denote by $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3)$ the hyperparameter vector governing the kernel function k.

1. Implement the kernel given by Equation (10) in MATLAB as a function of $\mathbf{x}$, $\mathbf{x}'$ and $\boldsymbol{\theta}$.

2. For the parameter values $\boldsymbol{\theta} = (1, 1, 1, 1)$ and $N = 101$ equally spaced points $\mathbf{X}$ in the interval $[-1, 1]$, compute the Gram matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ (eq. 6.54 Bishop). (*Hint:* $\mathbf{x}$ and $\mathbf{x}'$ are univariate in this case; use `linspace` to determine the points $\mathbf{X}$)

3. What is the dimension of $\mathbf{K}$? How can we show that $\mathbf{K}$ is positive semidefinite?

4. We will now use the previously computed matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ to produce samples from the Gaussian process prior $\mathbf{y}(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$, with $\mathbf{X}$ being the previously determined N equally spaced points. Generate five functions $\mathbf{y}(\mathbf{X})$ with MATLAB and plot them against the N input values $\mathbf{X}$.

5. Repeat 2. and 4. for the hyperparameter configurations from Bishop, Figure 6.5:

$$\boldsymbol{\theta} \in \{(1, 4, 0, 0), (9, 4, 0, 0), (1, 64, 0, 0), (1, 0.25, 0, 0), (1, 4, 10, 0), (1, 4, 0, 5)\}.$$

   Describe the differences between the plots. Explain why and how the changes in the kernel parameters lead to these differences.

We now consider the following training data consisting of four data points:

$$\mathcal{D} = \{(x_1 = -0.5, t_1 = 0.5), (x_2 = 0.2, t_2 = -1), (x_3 = 0.3, t_3 = 3), (x_4 = -0.1, t_4 = -2.5)\} \quad (11)$$

6. Compute the Gram matrix of the training data for $\boldsymbol{\theta} = (1, 1, 1, 1)$. Then, taking $\beta = 1$ in Equation (9), compute the covariance matrix $\mathbf{C}$ corresponding to the marginal distribution of the training target values: $p(\mathbf{t}) = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$.

7. Using the previous results, compute the mean and the covariance of the conditional distribution $p(t|\mathbf{t})$ of a new target value t corresponding to the input $x = 0$. [Which equations from Bishop do you need?]

8. Does the mean of the conditional distribution $p(t|\mathbf{t})$ go to zero in the limit $x \to \pm\infty$? If so, explain why this happens. If not, how would you set the parameters $\boldsymbol{\theta}$ of the kernel function to make it happen?