Statistical Machine Learning 2018

Assignment 1
Deadline: 7th of October 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 requires coding, put **essential code snippets** in your answer to the question in the report, and explain briefly what the code does. In addition, put the **full code listings** in a separate pdf file and hand in complete (working) 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 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 **Brightspace** as a **single pdf** file: '`SML_A1_<Namestudent(s)>.pdf`', in combination with one pdf file '`SML_A1_CODELISTING_<Namestudent(s)>.pdf`' for the code listings, and one zip-file with the executable source/data files (e.g. matlab m-files). For those working in pairs, it is sufficient if one team member uploads the solutions.
- Assignment 1 consists of 3 exercises, weighted as follows: 5, 2.5, and 2.5 points, respectively. The assignment also includes an optional exercise (worth 1 bonus point). The **grading** will be based on the report pdf file. The code listing and 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 – weight 5

Consider once more the M-th order polynomial

$$y(x; \mathbf{w}) = w_0 + w_1 x + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{1}$$

1. Create the function $f(x) = 1 + \sin(6(x - 2))$ in MATLAB. Generate a data set $\mathcal{D}_{10}$ of 10 noisy observations of this function. Take the 10 inputs spaced uniformly in range $[0, 1]$, and assume that the noise is Gaussian with mean 0 and standard deviation 0.3. $\mathcal{D}_{10}$ will be the training set. In a similar way, generate an additional test set $\mathcal{T}$ of 100 noisy observations over the same interval. Plot both the function and observations in $\mathcal{D}_{10}$ in a single graph (similar to Bishop, Fig.1.2).

2. Create MATLAB function $\mathbf{w} = \mathit{PolCurFit}(\mathcal{D}_N, M)$ that takes as input a data set $\mathcal{D}_N$, consisting of N input/output-pairs $\{x_n, t_n\}$, and a parameter M, representing the order of the polynomial in (1), and outputs a vector of weights $\mathbf{w} = [w_0, \ldots, w_M]$ that minimizes the sum-of-squares error function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n; \mathbf{w}) - t_n\}^2 \tag{2}$$

   Hint: use the results from the Tutorial Exercises (Week 1, Exercise 5), and the \-operator (backslash) in MATLAB to solve a linear system of equations.

3. For the given dataset $\mathcal{D}_{10}$, run the $\mathit{PolCurFit}()$ function for $M = [0, \ldots, 9]$, and,

   - Plot for various orders M (at least for $M = 0, M = 1, M = 3, M = 9$) the resulting polynomial, together with the function f and observations $\mathcal{D}_{10}$ (similar to Bishop, Fig 1.4)
   - For each order $M \in [0, \ldots, 9]$, compute the root-mean-square error

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N} \tag{3}$$

     of the corresponding polynomial, evaluated on both the training set $\mathcal{D}_{10}$ and the testset $\mathcal{T}$. Plot both as a function of M in a single graph. (see Bishop, Fig.1.5).

4. Repeat this procedure for a data set $\mathcal{D}_{40}$ of 40 observations (with the same noise level) and compare with the previous result.

5. Modify the $\mathit{PolCurFit}()$ function to include an additional penalty parameter λ, for a procedure that solves the minimization problem for a modified error function with quadratic regularizer (weight decay), given as

$$\tilde{E} = E + \frac{\lambda}{2} \sum_{j=0}^{M} w_j^2. \tag{4}$$

   Verify that the regularizer drives the weights of high order terms in the polynomial to zero, and see if you can reproduce and explain the effect observed in Bishop, Fig.1.8.

6. The polynomial curve fitting procedure can be extended to the case of multidimensional inputs. Assuming an input vector of dimension D, namely $\mathbf{x} = (x_1, x_2, ..., x_D)$, we can write the regression function y as:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{j=0}^{M} \left(\sum_{n_1 + n_2 + \ldots + n_D = j} w_{n_1 n_2 \ldots n_D} x_1^{n_1} x_2^{n_2} \ldots x_D^{n_D} \right) \tag{5}$$

In the last expression, j refers to the order of the polynomial terms. The inner sum is over all the combinations of non-negative integers $n_1, n_2, ..., n_D$, such that the constraint $n_1 + n_2 + ... + n_D = j$ holds. The terms $n_1, n_2, ..., n_D$ correspond to the exponent for each variable $x_1, x_2, ..., x_D$ in their respective polynomial term.

Note that if $D = 1$, the above expression simplifies to the formula in Equation (1). The reason the second sum disappears is that there is only one combination of the non-negative integer n_1 for which the constraint $n_1 = j$ holds, which means that there is only a single term to sum over.

Fitting the polynomial curve to a multidimensional input vector works analogously to the one-dimensional case. However, the number of parameters (the size of $\mathbf{w}$) becomes much larger, even when $D = 2$. Write down the general polynomial curve equation in (5) for $D = 2$. How many parameters are needed in the two-dimensional case? Compare this to the number of parameters in the one-dimensional case.

Exercise 2 – weight 2.5

In this exercise, we consider the gradient descent algorithm for function minimization. When the function to be minimized is $E(\mathbf{x})$, the gradient descent iteration is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \nabla E(\mathbf{x}_n) \tag{6}$$

where $\eta > 0$ is the so-called learning-rate. In the following, we will apply gradient descent to the function

$$h(x, y) = 100(y - x^2)^2 + (1 - x)^2 \tag{7}$$

1. Make a plot of the function h over the interval $[-2 \leq x \leq 2] \times [-1 \leq y \leq 3]$.
   Tip: use MATLAB function `surf`. Can you guess from the plot if numerical minimization with gradient descent will be fast or slow for this function?
2. Knowing that a critical point of a function is a point where the gradient vanishes, show that $(1, 1)$ is the unique critical point of h. Prove that this point is a minimum for h.
3. Write down the gradient descent iteration rule for this function.
4. Implement gradient descent in MATLAB. Try some different values of η. Does the algorithm converge? How fast? Make plots of the trajectories on top of a contour plot of h. (Hint: have a look at the MATLAB example code `contour_example.m` on Brightspace for inspiration to plot contours of functions and trajectories). Report your findings. Explain why numerical minimization with gradient descent is slow for this function.

Exercise 3 – weight 2.5

Suppose we have two healthy but curiously mixed boxes of fruit, with one box containing 8 apples and 4 grapefruit and the other containing 15 apples and 3 grapefruit. One of the boxes is selected at random and a piece of fruit is picked (but not eaten) from the chosen box, with equal probability for each item in the box. The piece of fruit is returned and then once again from the *same* box a second piece is chosen at random. This is known as sampling with replacement. Model the box by random variable B, the first piece of fruit by variable F_1, and the second piece by F_2.

1. What is the probability that the first piece of fruit is an apple given that the second piece of fruit was a grapefruit? How can the result of the second pick affect the probability of the first pick?

2. Imagine now that after we remove a piece of fruit, it is not returned to the box. This is known as sampling without replacement. In this situation, recompute the probability that the first piece of fruit is an apple given that the second piece of fruit was a grapefruit. Explain the difference.

3. Starting from the initial situation (i.e., sampling with replacement), we add a dozen oranges to the first box and repeat the experiment. Show that now the outcome of the first pick has no impact on the probability that the second pick is a grapefruit. Are the two picks now dependent or independent? Explain your answer.

Exercise 4 – Bonus (weight 1)

Given a joint probability density function over the random vector $\mathbf{X} = (X_1, X_2, X_3, X_4)$ that factorizes as

$$p(x_1, x_2, x_3, x_4) = p(x_1, x_4|x_2)p(x_2, x_3|x_1),$$

show (using the sum and product rules for marginals and conditionals) that the following independence statements hold:

1. $X_1 \perp X_2$;

2. $X_3 \perp X_4 \,|\, X_1, X_2$.