Statistical Machine Learning 2018

Assignment and answers 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).

   ANSWER: See figure 1. For implementation of this and later functions, see the matlab file a001.m.

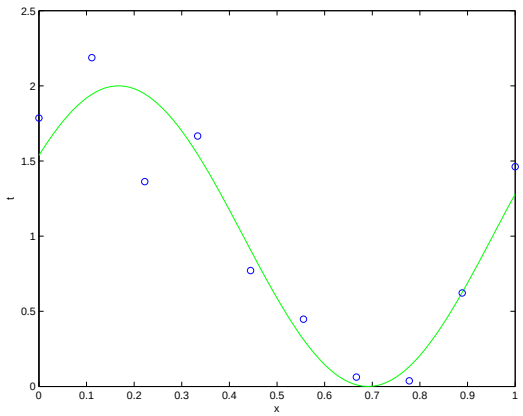

Figure 1: Left: The function $f(x) = 1 + \sin(6(x-2))$ (in green), with 10 training points (circles). Noise is Gaussian with mean 0 and standard deviation 0.3.

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.

ANSWER: In the Tutorial Exercises (Week 1, Exercise 5), you derived the following set of linear equations describing the minimum of the loss function:

$$\sum_{j=0}^{M} A_{ij} w_j = T_i$$

with

$$A_{ij} = \sum_{n=1}^{N} x_n^{i+j}, \qquad T_i = \sum_{n=1}^{N} t_n x_n^i.$$

In matrix notation, the minimum with respect to the parameter vector $\mathbf{w}$ is given by:

$$\mathbf{T} = \mathbf{A}\mathbf{w}$$

Solving such an equation can be done in MATLAB (in the least-squares sense) using the \-operator in the statement "$\mathbf{w} = \mathbf{A} \setminus \mathbf{T}$". For details, see the example implementation in a001.m.

3. For the given dataset $\mathcal{D}_{10}$, run the *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).

   ANSWER: We ran the *PolCurFit*() function for various orders $M = [0, \ldots, 9]$. See plots in figure 2 for results.

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.

   ANSWER: With a larger data set the (strong) overfitting for the highest order polynomial, resulting in zero training error but large(r) test error, no longer occurs. In other words, for a model with few degrees of freedom relative to a large data set, the danger of overfitting is much less severe, and so regularization becomes less important. See figure 3.

5. Modify the *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

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

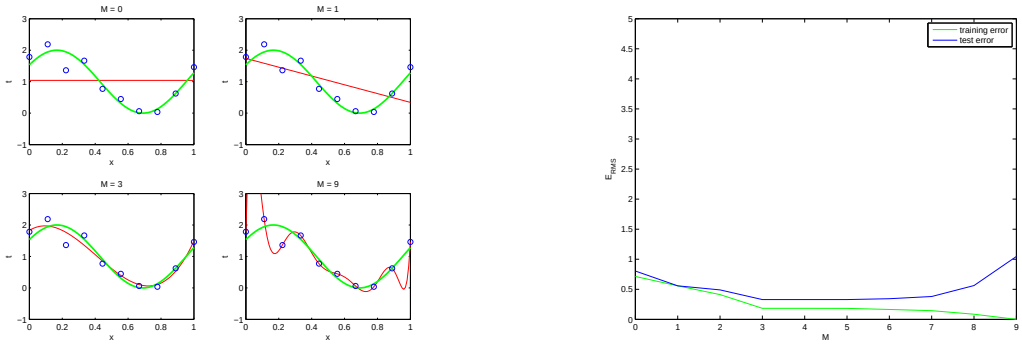

Figure 2: The function (in green), with 10 training points (circles), and 4 polynomials (in red) of increasing order trained on the training points. Right: Training and test error as a function of the order of the polynomial. Note the severe overfitting for high M.

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.

ANSWER: Final solution for weights $\mathbf{w}$ takes the form of:

$$\mathbf{w} = (\mathbf{A} + \lambda \mathbf{I})^{-1}\mathbf{T} \tag{5}$$

with matrix $\mathbf{A}$ and vector $\mathbf{T}$ defined as in the Tutorial (Week 1, Exercise 5). Run a001.m to see that weight decay drives higher order terms to zero ($\lambda = 0$ and $\lambda = 1$ is compared). Very little regularization (small λ) gives the overfitting we had before, adding a bit of regularization greatly improves the error on the test set (even though the error on the training set always increases), but at some point the 'damping' effect becomes too strong, and the predictive results will start to deteriorate. See figure 4, and compare Bishop, p.11.

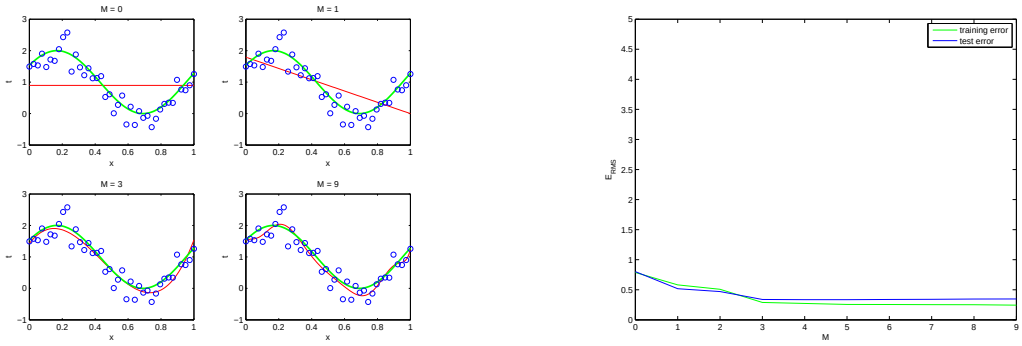

Figure 3: Left: The function (in green), with 40 training points (circles), and 4 polynomials (in red) of increasing order trained on the training points. Right: training and test error as a function of the order of the polynomial. The overfitting is now much less serious because the training set is larger.

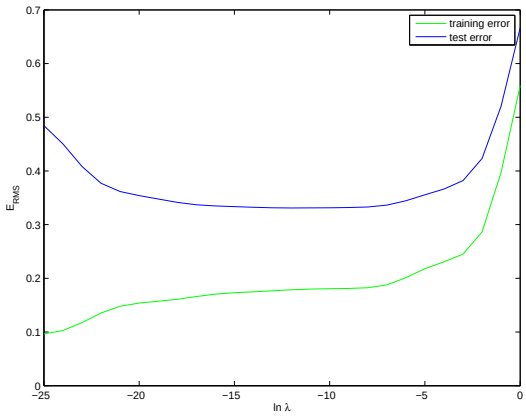

Figure 4: Training and test error as a function of $\ln \lambda$. We took $M = 9$, and the same 10 data points as earlier.

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:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{j=0}^{M} \left(\sum_{n_1+n_2+...+n_D=j} w_{n_1 n_2 ... n_D} x_1^{n_1} x_2^{n_2} ... x_D^{n_D} \right) \tag{6}$$

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 (6) 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.

ANSWER: We can write down the polynomial regression formula (6) for $D = 2$ as

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

or equivalently,

$$y(\mathbf{x}; \mathbf{w}) = \sum_{j=0}^{M} \left(\sum_{n=0}^{j} w_{n,j-n} x_1^{n} x_2^{j-n} \right).$$

Using the last expression, we can immediately see that for j-th order individual polynomial terms, we sum over $j+1$ different values of n, namely $\{0, 1, 2, ..., j\}$. This means that for j-th order terms, we have $j + 1$ different weights, namely $w_{0,j}, w_{1,j-1}, ..., w_{j,0}$. Since we include polynomial terms up to the M-th degree, the total number of parameters becomes:

$$\sum_{j=0}^{M} (j+1) \overset{k:=j+1}{=} \sum_{k=1}^{M+1} k = \frac{(M+1)(M+2)}{2}.$$

In the one-dimensional case, we needed $M + 1$ parameters (see Equation 1), which is a number linear in M. We have now shown that the number of parameters is quadratic in M in the two-dimensional case. In general, as the dimension D grows, the number of parameters grows exponentially, proportional to M^D.

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

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \nabla E(\mathbf{x}_n) \tag{7}$$

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

$$h(x, y) = 100(y - x^2)^2 + (1 - x)^2 \tag{8}$$

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?

   ANSWER: This is Rosenbrock's so called "banana function", see figure 5. (The colours have been chosen to accentuate the fruity aspect). The following MATLAB code can be used to create such a plot:

```
% define the function h(x,y)
h = @(x,y) 100 * (y - x.^2).^2 + (1 - x).^2;
% define x-values
x = [-2:0.1:2];
% define y-values
y = [-1:0.1:3];
% plot h(x,y)
surf(x,y,h(ones(41,1)*x,y'*ones(1,41)));
```

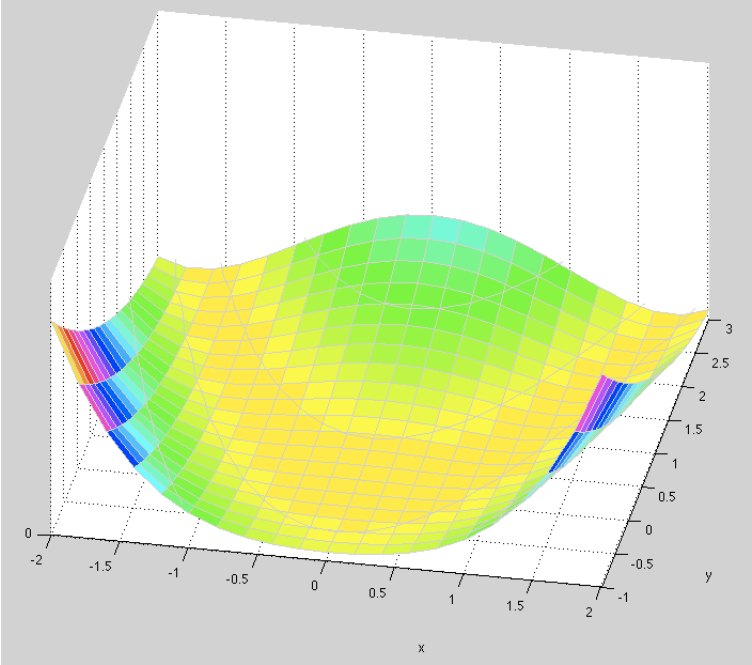

Figure 5: The Rosenbrock "banana function".

   At every point in the graph the gradient is largely oriented towards the nearest point of the "curved valley" in fig.5. The gradient along the curved bottom part towards the minimum is very small compared with the gradient across it, see fig.6. Steep in one direction: the learning rate cannot be too big otherwise it will diverge; flat in the other: learning will slow down. This means gradient descent is in trouble.

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.

ANSWER: The gradient ∇h is given by

$$\begin{aligned} \frac{\partial h}{\partial x} &= 400x^3 - 400yx + 2x - 2 \\ \frac{\partial h}{\partial y} &= 200y - 200x^2 \end{aligned}$$

Setting equal to zero gives

$$\begin{aligned} 400x^3 - 400yx + 2x - 2 &= 0 \\ 200y - 200x^2 &= 0 \end{aligned}$$

The second gives $y = x^2$, substituting this in the first results in

$$400x^3 - 400x^2x + 2x - 2 = 2x - 2 = 0$$

which implies $x = 1$ and so also $y = 1$.

We have shown that $(1, 1)$ is a critical point, but we still have to show that this point is a minimum. A function attains a local minimum at a critical point if its Hessian (the square matrix of second-order partial derivatives) is positive definite at that point. In the univariate case, this is equivalent to the second derivative being (strictly) positive at that point.

The Hessian matrix of h is computed in Equation 9. We have to show that the Hessian matrix of h is positive definite at point $(1, 1)$.

$$\mathbf{H}_h(x, y) = \begin{bmatrix} \frac{\partial^2 h}{\partial x^2} & \frac{\partial^2 h}{\partial x \partial y} \\ \frac{\partial^2 h}{\partial y \partial x} & \frac{\partial^2 h}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{bmatrix} \tag{9}$$

One way to do this is to show that all the eigenvalues of $\mathbf{H}_h(1, 1)$ are (strictly) positive. Alternatively, we can show that all the leading principal minors – The k^{th} leading principal minor of a matrix is the determinant of its upper-left k by k sub-matrix. – of $\mathbf{H}_h(1, 1)$ are (strictly) positive. For this two-dimensional function, it is easier to use the second method.

$$\mathbf{H}_h(1, 1) = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$$

Since $\big|802\big| = 802 > 0$ and $\begin{vmatrix} 802 & -400 \\ -400 & 200 \end{vmatrix} = 160400 - 160000 = 400 > 0$, it means that all the leading principal minors of $\mathbf{H}_h(1, 1)$ are (strictly) positive, hence the matrix is positive definite and the critical point $(1, 1)$ is a local minimum. In the case of this particular function, the local minimum also happens to be the global minimum. (Can you explain why?)

3. Write down the gradient descent iteration rule for this function.

   ANSWER: The gradient descent iteration rule is:

$$\begin{aligned} x_{n+1} &= x_n - \eta(400x_n^3 - 400y_nx_n + 2x_n - 2) \\ y_{n+1} &= y_n - \eta(200y_n - 200x_n^2) \end{aligned}$$

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.

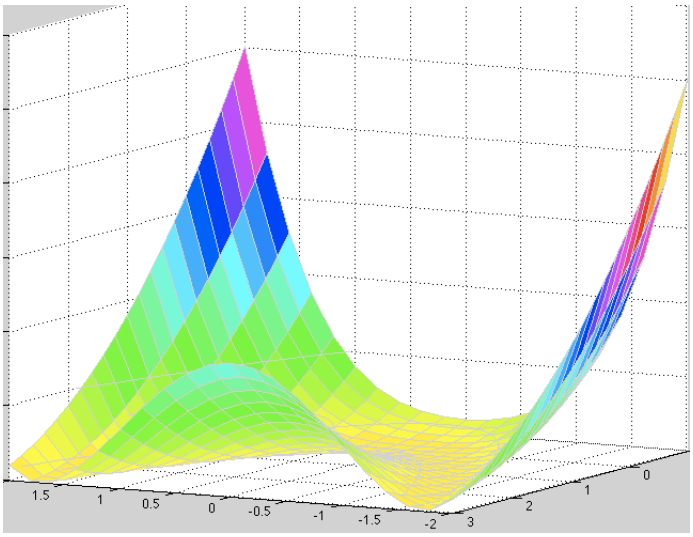

Figure 6: Near-zero gradient along bottom part of banana.

ANSWER: An example MATLAB implementation is provided in the file a018.m. Some trajectories for different learning rates are plotted in Figure 7. For this starting point $(1, -1)$, the gradient descent algorithm seems to converge for $\eta = 0.0001$ and $\eta = 0.001$, but diverges for $\eta = 0.003$. The convergence is very slow: even after 10000 steps, the step size is still considerable.

At every point in the graph the gradient is largely oriented towards the nearest point of the 'curved valley' in fig.5. The gradient along the curved bottom part towards the minimum is very small compared with the gradient across it, see fig.6. As a result for a small deviation from the bottom, the next gradient step is primarily back towards the bottom, i.e. perpendicular to the direction along the bottom of the valley you try to follow towards the minimum. More advanced methods, e.g. conjugate gradient descent, can take this into account and have no problem in finding the minimum in a few steps.

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?

   ANSWER: Requested is $p(F_1 = a|F_2 = g)$, which follows from:

$$\begin{aligned} p(F_1 = a|F_2 = g) =& p(F_1 = a, B = 1|F_2 = g) + p(F_1 = a, B = 2|F_2 = g) \\ =& p(F_1 = a|B = 1)p(B = 1|F_2 = g) + \\ & p(F_1 = a|B = 2)p(B = 2|F_2 = g) \end{aligned} \tag{10}$$

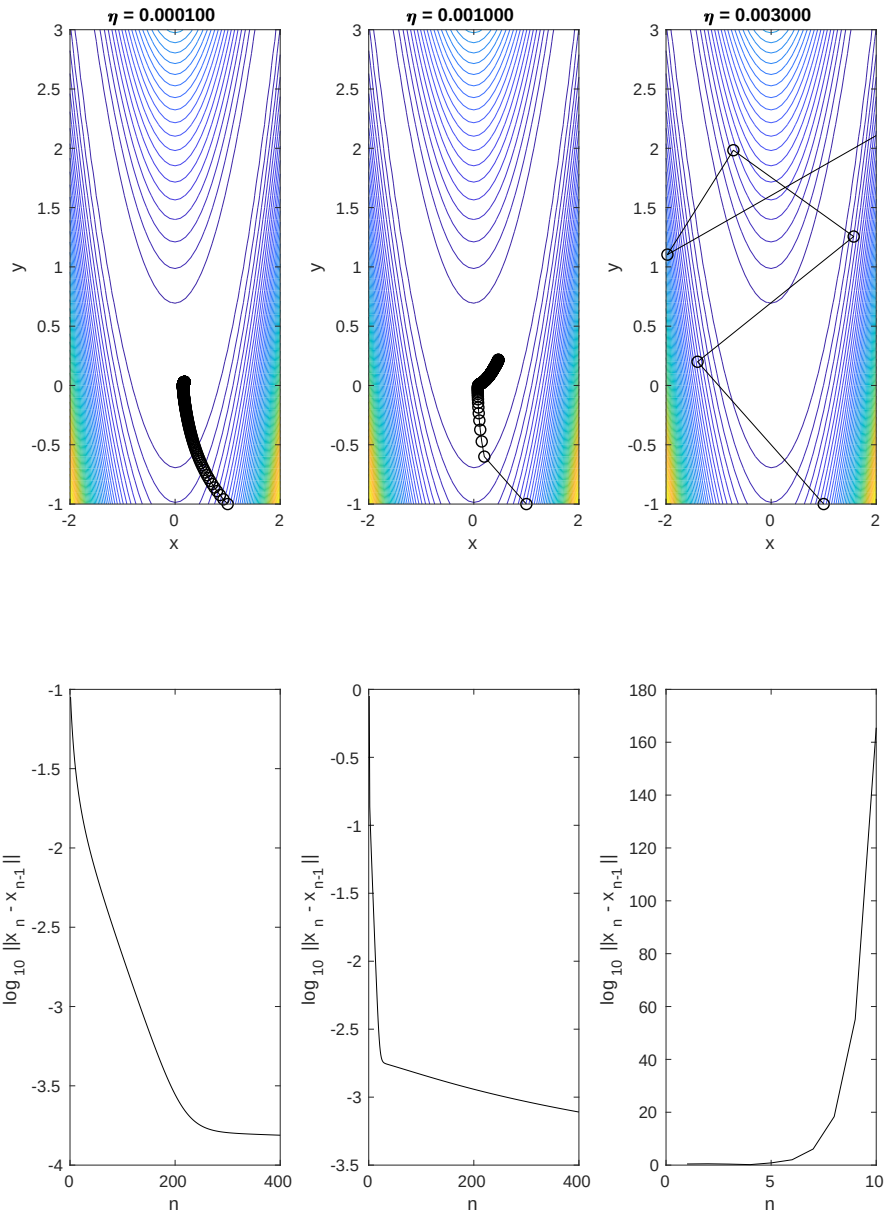

Figure 7: For three different learning rates η, a gradient descent trajectory has been plotted on top of a contour plot of the Rosenbrock function $h(x, y)$. For each trajectory, the $\log_{10}$ of the step size $||x_n - x_{n-1}||$ has been plotted as a function of the iteration n.

To arrive at the result of (10), we used the fact that $p(F_1|B, F_2) = p(F_1|B)$. Once we know the box from which the fruit were picked, the two picks become independent.

Since the first piece of fruit is put back, the probabilities on the second pick are identical to the ones on the first pick, i.e. $p(F_2) = p(F_1)$, and therefore (Bayes' rule):

$$\begin{aligned} p(B=1|F_2=g) &= \frac{p(F_2=g|B=1)p(B=1)}{p(F_2=g)} \\ &= \frac{p(F_2=g|B=1)p(B=1)}{1-p(F_1=a)} \\ &= \frac{\frac{4}{12}\frac{1}{2}}{(1-\frac{3}{4})} \\ &= \frac{2}{3} \end{aligned}$$

Substituting in (10), using $p(B=2|F_2=g) = 1 - p(B=1|F_2=g)$, then gives

$$\begin{aligned} p(F_1=a|F_2=g) &= \frac{8}{12}\frac{2}{3} + \frac{15}{18}(1-\frac{2}{3}) \\ &= \frac{8}{18} + \frac{5}{18} \\ &= \frac{13}{18} \end{aligned}$$

Note that the probability of picking an apple the first time changes when information about a later event is known. In other words: a probability is not a fixed 'objective' property of a real world event, but quantifies a state of knowledge about that event.

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.

ANSWER: Requested is $p(F_1 = a|F_2 = g)$, which follows from:

$$\begin{aligned} p(F_1=a|F_2=g) =& p(F_1=a, B=1|F_2=g) + p(F_1=a, B=2|F_2=g) \\ =& p(F_1=a|B=1, F_2=g)p(B=1|F_2=g) + \\ & p(F_1=a|B=2, F_2=g)p(B=2|F_2=g) \end{aligned} \tag{11}$$

Note that it is no longer true that $p(F_1|B, F_2) = p(F_1|B)$, which we used previously in Equation (10). Even if we know the box from which we are picking fruit, the fact that the picked fruit is not replaced makes the two picks dependent. Using Bayes' rule, we have:

$$\begin{aligned} p(F_1=a|B=1, F_2=g) &= \frac{p(F_2=g|B=1, F_1=a)p(F_1=a|B=1)}{p(F_2=g|B=1)} \\ &= \frac{\frac{4}{11}\frac{8}{12}}{\frac{4}{11}\frac{8}{12} + \frac{3}{11}\frac{4}{12}} \\ &= \frac{8}{11} \end{aligned}$$

$$\begin{aligned} p(F_1=a|B=2, F_2=g) &= \frac{p(F_2=g|B=2, F_1=a)p(F_1=a|B=2)}{p(F_2=g|B=2)} \\ &= \frac{\frac{3}{17}\frac{15}{18}}{\frac{3}{17}\frac{15}{18} + \frac{2}{17}\frac{3}{18}} \\ &= \frac{15}{17} \end{aligned}$$

Even though the first piece of fruit is not returned to the box, the *marginal* probabilities of the second pick do not change, i.e. $p(F_2|B) = p(F_1|B)$. One way to see this is to imagine labeling the pieces of fruit in a random order. It is equally probable that an apple in a given box is labeled as 1 (first pick) or 2 (second pick). However, this only holds in the absence of additional information, such as which fruit were picked at different times. Confirm this for yourself.

Given that the marginals are the same, the derivation of $p(B = 1|F_2 = g)$ is the same as in the previous part:

$$\begin{aligned} p(B = 1|F_2 = g) &= \frac{p(F_2 = g|B = 1)p(B = 1)}{p(F_2 = g)} \\ &= \frac{p(F_2 = g|B = 1)p(B = 1)}{1 - p(F_1 = a)} \\ &= \frac{\frac{4}{12}\frac{1}{2}}{(1 - \frac{3}{4})} \\ &= \frac{2}{3} \end{aligned}$$

Substituting in (11) and using $p(B = 2|F_2 = g) = 1 - p(B = 1|F_2 = g)$ gives:

$$\begin{aligned} p(F_1 = a|F_2 = g) &= \frac{8}{11}\frac{2}{3} + \frac{15}{17}\frac{1}{3} \\ &= \frac{437}{561} \end{aligned}$$

Note that the probability $p(F_1 = a|F_2 = g)$ is higher than when we were sampling with replacement. Intuitively, the fact that we picked a grapefruit the second time reduces the probability that we picked a grapefruit the first time, as this would have taken one grapefruit out of the box. This in turn increases the probability of the first pick being an apple.

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.

   ANSWER: We have to show that the probability of the second pick being a grapefruit given the outcome of the first pick has the same value as the probability of the second pick:

$$\begin{aligned} p(F_2 = g) &= p(F_2 = g|B = 1)p(B = 1) + p(F_2 = g|B = 2)p(B = 2) \\ &= \frac{4}{24} \cdot \frac{1}{2} + \frac{3}{18} \cdot \frac{1}{2} = \frac{1}{6} = 0.1666 \\ p(B = 1|F_1 = g) &= \frac{p(F_1 = g|B = 1)p(B = 1)}{p(F_1 = g)} \\ &= \frac{\frac{4}{24}\frac{1}{2}}{\frac{1}{6}} = \frac{1}{2} = 0.5 \\ p(F_2 = g|F_1 = g) &= p(F_2 = g|B = 1)p(B = 1|F_1 = g) + p(F_2 = g|B = 2)p(B = 2|F_1 = g) \\ &= \frac{4}{24} \cdot \frac{1}{2} + \frac{3}{18} \cdot \frac{1}{2} = \frac{1}{6} = 0.1666 \end{aligned}$$

   Similarly, we can show that $p(F_2 = g|F_1 = a) = p(F_2 = g|F_1 = o) = p(F_2 = g)$.

   No: despite the equality $p(F_2 = g|F_1) = p(F_2)$, F_1 and F_2 are *not* independent. Independence only holds if $p(F_2 = y|F_1 = x) = p(F_2 = y)$ for *all possible values* x, y. In other words,

independence applies to random variables as a whole, not to specific outcomes only. F_1 and F_2 are not independent as is easily seen by considering, for example, $p(F_2 = a|F_1 = o)$: once the first pick was an orange you *know* it had to come from box 1 as the other box contained no oranges whatsoever. The probability of picking an apple from box 1 is $\frac{1}{3}$, while the probability of picking an apple from either box is $\frac{8}{24} \cdot \frac{1}{2} + \frac{15}{18} \cdot \frac{1}{2} = \frac{1}{2}$.

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. $X_1 \perp X_2$;

   ANSWER: We will show that $p(x_2|x_1) = p(x_2)$.

$$\begin{aligned} p(x_1, x_2) &= \sum_{x_3} \sum_{x_4} p(x_1, x_2, x_3, x_4) \\ &= \sum_{x_3} \sum_{x_4} p(x_1, x_4|x_2)p(x_2, x_3|x_1) \\ &= \left(\sum_{x_4} p(x_1, x_4|x_2) \right) \left(\sum_{x_3} p(x_2, x_3|x_1) \right) \\ &= p(x_1|x_2)p(x_2|x_1) \end{aligned}$$

   We now use the fact that $p(x_1, x_2)$ factorizes as $p(x_1|x_2)p(x_2)$ (or equivalently $p(x_2|x_1)p(x_1)$). Using the previous result, we get:

$$p(x_1|x_2)p(x_2) = p(x_1|x_2)p(x_2|x_1) \implies p(x_2) = p(x_2|x_1). \quad \square$$

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

   ANSWER: We will show that $p(x_3, x_4|x_1, x_2) = p(x_4|x_1, x_2)p(x_3|x_1, x_2)$.

$$\begin{aligned} p(x_3, x_4|x_1, x_2) &= \frac{p(x_3, x_4, x_1, x_2)}{p(x_1, x_2)} \\ &= \frac{p(x_1, x_4|x_2)p(x_2, x_3|x_1)}{p(x_1|x_2)p(x_2|x_1)} \\ &= \frac{p(x_1, x_4|x_2)}{p(x_1|x_2)} \cdot \frac{p(x_2, x_3|x_1)}{p(x_2|x_1)} \\ &= p(x_4|x_1, x_2)p(x_3|x_1, x_2). \quad \square \end{aligned}$$