# Statistical Machine Learning 2018
# Assignment 1
# Deadline: 7th of October 2018

Christoph Schmidl
s4226887
c.schmidl@student.ru.nl

Mark Beijer
s4354834
mbeijer@science.ru.nl

September 26, 2018

## Exercise 1 - weight 5

Consider once more the M-th order polynomial

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

### 1.1

Create the function $f(x) = sin(6(x-2))$ in MATLAB. Generate a data set $\mathcal{D}_{10}$ of 10 noisy observation 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 observation in $\mathcal{D}_{10}$ in a single graph (similar to Bishop, Fig.1.2).

**Answer:**

### 1.2

Create a MATLAB function $w = PolCurFit(\mathcal{D}_N, M)$ 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 $w = [w_0, ..., w_M]$ that minimizes the sum-of-squares error function

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

Hint: use the results from the Tutorial Exercises (Week1, Exercise 5), and the \-operator (backslash) in MATLAB to solve a linear system of equations.

**Answer:**

### 1.3

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

- Plot for various orders $M$ (at least for $M = 0, M = 1, M = 3, M = 9$) the resulting ploynomial, together with the function $f$ and observation $\mathcal{D}_{10}$ (similar to Bishop, Fig 1.4)

- For each order $M \in [0, ..., 9]$, compute the root-mean-square error

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

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

**Answer:**

## 1.4

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:**

## 1.5

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

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

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

**Answer:**

## 1.6

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

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

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 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.

**Answer:**

# Exercise 2 - weight 2.5

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

$$x_{n+1} = x_n - \eta \nabla E(x_n)$$

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

$$h(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

## 2.1

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:**

## 2.2

Knowing that a critical point of a function is a point where the gradient vanishes, show that $(1, 1)$ is the unique critical point og $h$. Prove that this point is a minimum for $h$.

**Answer:**

## 2.3

Write down the gradient descent iteration rule for this function.
**Answer:**

## 2.4

Implement gradient descent in MATLAB. Try some different values of $\eta$. Does the algorithm converge? How fast? Make plots of the trajectories on top of a contour plot of $h$. (Hint: have look at the MATLAB example code *contour_example.m* on Brightspace for inspiration to plot contours of functions and trajectories). Report your findings. Explan why numerical minimization with gradient descent is slow for this function.

**Answer:**

# Exercise 3 - weight 2.5

Suppose we have two healthy but curiously mixed boxes of fruit, with one box containing 8 apples and 4 grapefruits and the other containing 15 apples and 3 grapefruits. 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 $\boldsymbol{B}$, the first piece of fruit by variable $\boldsymbol{F_1}$, and the second piece by $\boldsymbol{F_2}$.

## 3.1

**Question:** 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:**

Model the box by random variable **B**
The first piece of fruit by variable $F_1$
The second piece of fruit by variable $F_2$

- Box 1: 8 apples, 4 grapefruits =¿ 12 fruits in total
- Box 2: 15 apples, 3 grapefruits =¿ 18 fruits in total

$$B = \{b_1, b_2\}$$

$$P(F_1 = apple|F_2 = grapefruit)$$

Calculating the probability of picking a grapefruit from each box:

$$P(F = Grapefruit|B = 1) = \frac{4}{12} = \frac{1}{3}$$
$$P(F = Grapefruit|B = 2) = \frac{3}{18} = \frac{1}{6}$$

Calculating the total probability of picking a grapefruit:

$$P(F = Grapefruit) = P()$$

Calculating the probability of picking an apple from each box:

$$P(F = Apple|B = 1) = \frac{8}{12} = \frac{2}{3}$$
$$P(F = Apple|B = 2) = \frac{15}{18} = \frac{5}{6}$$

### 3.2

**Question:** 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:**

We want to find the following probability based on sampling without replacement:

$$P(F_1 = apple|F_2 = grapefruit)$$

### 3.3

**Question:** 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:**

## Exercise 4 - Bonus (weight 1)

Given a joint probability function over the random vector $X = (X_1, X_2, X_3, X_4)$ 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:

## 4.1

$$X_1 \perp\!\!\!\perp X_2$$

**Answer:**

In order to show that $X_1 \perp\!\!\!\perp X_2$, we have to prove that $p(x_1, x_2) = p(x_1)p(x_2)$. Figure 1 gives a visual representation of the factorization as a directed graph. We can already see that the statement $X_1 \perp\!\!\!\perp X_2$ does not hold.

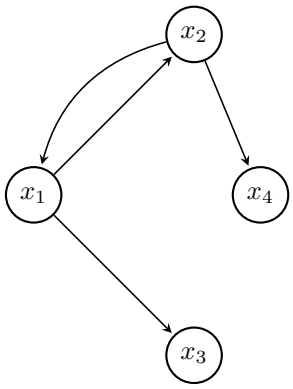


Figure 1: Factorization as directed graph

We can rewrite the factorization as follows:

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

We can now marginalize $x_1$ and $x_2$ and apply the sum and product rule at the end to get the joint probabilities:

$$\begin{aligned} p(x_1) &= p(x_1|x_2)p(x_2) = p(x_2, x_1) \\ p(x_2) &= p(x_2|x_1)p(x_1) = p(x_1, x_2) \end{aligned}$$

Therefore:

$$\begin{aligned} p(x_1)p(x_2) &= p(x_1, x_2)p(x_2, x_1) \\ p(x_1, x_2) &\neq p(x_1)p(x_2) \end{aligned}$$

Based on this contradiction the statement $X_1 \perp\!\!\!\perp X_2$ does **not** hold.

## 4.2

$$X_3 \perp\!\!\!\perp X_4|X_1, X_2$$

**Answer:**

In order to show that $X_3 \perp\!\!\!\perp X_4|X_1, X_2$, we have to prove that $p(x_3, x_4|x_1, x_2) = p(x_3|x_1, x_2)p(x_4|x_1, x_2)$. We already know from the previous exercise that

$$p(x_1)p(x_2) = p(x_1, x_2)p(x_2, x_1)$$

and based on the symmetry property we know that:

$$p(x_1, x_2) = p(x_2, x_1)$$

We can therefore replace the joint probabilities with marginal probabilities which makes it easier to prove the conditional independence:

$$p(x_1, x_2) = p(x_1)$$
$$p(x_1, x_2) = p(x_2)$$

We can rewrite the statements as follows when we replace $p(x_1, x_2)$ for $p(x_1)$:

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

Therefore the statement $X_3 \perp\!\!\!\perp X_4|X_1, X_2$ holds.