MACHINE LEARING
FIRST ASSIGNMENT 2024-2025

- **Work in groups. The optimal number of students in a group is 3.**
- **Assignments should be handed in on eclass in a single file containing the code, results and comments. Use graphs wherever possible. Comment on your results and draw conclusions.**
- **Programming should be done in Python. The preferred format for the final handout is Jupyter Notebook, but other formats are also welcome.**
- **You can earn bonus points for sharp comments and creative thinking.**
- **Deadline: Monday, 13 January 2025.**

Problem 1

Consider the generalized linear regression problem generated by the following model:

$$y = f(x) + \eta \tag{1}$$

where

$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_5 x^5 \tag{1a},$$

η corresponds to white Gaussian noise and the components of the weight vector assume the following values:

$$\theta_0 = 0.2\,,\ \theta_1 = -1\,,\ \theta_2 = 0.9\,,\ \theta_3 = 0.7,\ \theta_5 = -0.2. \tag{2}$$

In every case below, we consider N equidistant points $x_1, x_2, \ldots, x_n$ in the interval [0,2] and use them to create samples for our training set:

$$y_n = \theta_0 + \theta_1 x_n + \theta_2 x_n^2 + \theta_3 x_n^3 + \theta_5 x_n^5 + \eta_n, \quad n = 1,2,\ldots,N \tag{3}$$

where η_n are i.i.d. noise samples originating from a Gaussian distribution with mean 0 and variance σ_η^2. This is the generating model for all training samples in what follows throughout problem 1.

We also create a test set using N_1 points randomly selected on the *x* axis in the interval [0,2]. As detailed below, we use several methods/models to train and to get estimates for the values of the dependent ordinate *y* for the points in our test set.

1) Generate $N = 20$ training points with $\sigma_\eta^2 = 0.05$. Apply the Least Squares (LS) method with the structure of the correct model (5th degree polynomial with the coefficient of the 4th power equal to zero) to estimate the parameter vector. Test with $N_1 = 20$. On the (*x,y*) plane plot the true curve and the estimates for *y* for the samples in the test set.
2) Again, generate $N = 20$ training points with σ_η^2=0.05. Apply LS assuming a 2nd degree polynomial structure. Perform 100 experiments using distinct noise samples in the training set for each experiment. For each point of the training set, calculate the mean and variance of the prediction for *y* over the 100 experiments and plot these quantities on the (*x,y*) plane along with the curve obtained by the true model. Repeat using a 10th degree polynomial. Compare your results obtained for the 2 different cases (2nd versus 10th degree polynomial) making special reference to the bias-variance dilemma.

3) Yet again, generate $N = 20$ training points with $\sigma_\eta^2 = 0.05$. We will encode our prior knowledge for the unknown parameter vector via a Gaussian distribution G(**θ**) with mean $\boldsymbol{\theta}_0$ equal to the true parameter vector in equation (2) and covariance matrix $\Sigma_\theta = \sigma_\theta^2 I,\ \sigma_\theta^2 = 0.1$. Use the structure and noise variance of the true model and perform full Bayesian Inference (FBI) in order to evaluate *y* for a test set with $N_1 = 20$. Plot your estimates and their errors on the (*x,y*) plane. Repeat for $\sigma_\eta^2 = 0.15$.
4) Repeat case (3) using the following mean vector for G(**θ**): $\boldsymbol{\theta_{0F}} = [-10.54, 0.465, 0.0087, -0.093, -0.004]^T$. With $\sigma_\eta^2 = 0.05$, perform the experiment four times, using two different values for σ_θ^2 (0.1 and 2) and two different values for N (20 and 500). Comment on your results.
5) Construct a training set with $N = 500$ and $\sigma_\eta^2 = 0.05$. Try to recover the true variance of the noise using the Expectation-Maximization (EM) method. Initialize the algorithm with $\alpha = \sigma_\theta^{-2} = 1, \beta = \sigma_\eta^{-2} = 1$. After convergence, estimate the *y*'s and their errors over a test set with $N_1 = 20$. Plot these quantities on the (*x,y*) plane, along with the true model curve.
6) And what if we don't know anything about the functional form of the generating model? Can a non-parametric method work? In what follows, we will be modelling probability density functions using Parzen windows with Gaussian kernels, as follows:

$$p(\boldsymbol{z}) = \frac{1}{N(2\pi h^2)^{L/2}} \sum_{i=1}^{N} exp\left[\frac{-\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2h^2}\right] \quad (4)$$

where N is the number of samples available to us and L is the dimension of the random variable vector **z**. We consider the regression problem of estimating *y* versus *x*, given training samples $(x_i, y_i),\ i = 1, \cdots, N$. Assuming that both the joint probability density $p(\mathrm{x}, \mathrm{y})$ and the marginal probability density $p(\mathrm{x})$ are accurately represented by the corresponding Parzen window representations, derive an analytical formula for the MSE optimal estimate $y = \hat{g}(x) = \mathbb{E}[\mathrm{y}|x]$. You can take into account that

$$\int_{-\infty}^{\infty} y\, exp[-a(y-b)^2]\, dy = b\sqrt{\frac{\pi}{\alpha}}. \quad (5)$$

Use $\sigma_\eta^2 = 0.05$ to generate $N = 20$ training samples. Estimate the *y*'s over a test set with $N_1 = 20$ with the formula that you have derived. Plot these quantities on the (*x,y*) plane, along with the true model curve. Repeat with $N = 500$ and $N_1 = 20$. Use the value $h = 1/\sqrt{N}$.

7) Comparison and discussion time: Consider each of the experiments summarized in the table below, and perform it $M = 1000$ times, each time using different noise samples for your training set, also sampling different random points on the *x* axis for the test set. We treat the value of the Mean Squared Error Cost Function *over the points in the test set*

$$E = \frac{1}{N_1} \sum_{i=1}^{N_1} \left(y_i - \theta_0 - \theta_1 x_i - \theta_2 x_i^2 - \theta_3 x_i^3 - \theta_5 x_i^5\right)^2 \quad (6)$$

as a random variable and estimate its expectation value as the average over all $M = 1000$ test sets

$$\bar{E} = \frac{1}{M} \sum_{j=1}^{M} E_j \quad (7)$$

Compute this average for each one of the 20 experiments listed below and use the values to compare the different regression methods. Discuss the effectiveness of the

methods, keeping in mind that some methods have more access to information concerning the structure of the generating mechanism than others.

Experiments 1-10:

Generation of points in the training set: $N = 20$, $\sigma_\eta^2 = 0.05$, functional form given by equation (3) with parameter vector given by equation (2).
Specifications for the regression models:

Experi-ment #	Method	Functional form of the regression model	N_1	σ_η^2	Prior for $\boldsymbol{\theta_0}$	σ_θ^2	h
1	LS	True model (5th degree polynomial without 4th power)	20	-	-	-	-
2	LS	Full 5th degree polynomial	20	-	-	-	-
3	LS	2nd degree polynomial	20	-	-	-	-
4	LS	10th degree polynomial	20	-	-	-	-
5	FBI	True model	20	0.05	True eq. (2)	0.1	-
6	FBI	True model	20	0.05	False ($\boldsymbol{\theta_{0F}}$)	0.1	-
7	FBI	True model	20	0.05	True eq. (2)	2.0	-
8	FBI	True model	20	0.05	False ($\boldsymbol{\theta_{0F}}$)	2.0	-
9	EM	True model	20	-	-	-	-
10	Parzen windows	Sum of Gaussians	20	-	-	-	Optimal for $\bar{E}$ *

* Try different values of h and choose the one which gives the lowest value of $\bar{E}$.

Experiments 11-20: Same as above, but with larger training sets ($N = 500$).

Problem 2

1) Program and implement a k nearest neighbours classifier (k-NN). Use this classifier to solve the following problems:

   i. Rice_Cammeo_Osmancik.txt (Classification of rice grains into two classes according to morphological characteristics).
   ii. Any other classification task of your choice (for example from Kaggle). Make sure that the input features are integer or decimal numbers, not strings.

   Report on the percentage of correct classification as a function of the number of nearest neighbours. Use cross-validation to obtain the results.

2) For the problems above, implement Bayes classifiers under various different assumptions for the form of the probability density functions for each class, as detailed below. For each different assumption, compute classification accuracy using

cross validation (it goes without saying that for each cross-validation iteration, probability density functions will have to be calculated using only training set data for this question). For assumptions c) and d) below, we will obviously have a naïve Bayes classifier. Compare the performance of the Bayes classifiers to the performance of the k-NN classifier.

The assumptions for the pdfs are as follows:

a) Pdfs are gaussian. The covariance matrices are diagonal, with all diagonal elements equal. Mean and variance of the pdfs are estimated using Maximum Likelihood from the available data in the training set.
b) Pdfs are gaussian, with non-diagonal covariance matrices. Means and covariance matrices of the pdfs are estimated using Maximum Likelihood from the available data in the training set.
c) Components of the feature vectors are mutually statistically independent (the usual naïve Bayes approach). Marginal Pdfs are gaussian, with parameters (mean, variance) estimated using Maximum Likelihood from the available data in the training set.
d) Components of the feature vectors are mutually statistically independent (the usual naïve Bayes approach). Marginal pdfs are computed using 1-d Parzen windows with gaussian kernels. Take the width *h* of each window equal to $1/\sqrt{N}$, where N is the number of patterns in the training set.
e) Pdfs are estimated using Gaussian Mixtures, with the relevant parameters estimated using the Expectation-Maximization algorithm. In order to determine the number of Gaussians in the mixture, compute the Bayesian Information Criterion (BIC) (https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118856406.app5) and use its slope as explained in the following link: https://towardsdatascience.com/gaussian-mixture-model-clusterization-how-to-select-the-number-of-components-clusters-553bef45f6e4.