HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

MACHINE LEARNING

1st assignment

Ηλεκτρονικός Αυτοματισμός (Τμήμα Φυσικής)

Rodo Kasidiari **2019505**
Panagiotis Tziolos **2019512**
Charalampos Bezaitis **2019508**
Philippos Tziolos **2019511**

December 2020

Introduction

The aim of this report is to present and comment on results that emerged in the different problems of the first assignment. Every problem is accomplished according to the class' instructions and to Machine Learning theory (books and lecture material).
All exercises are implemented in Python 3. We used the following libraries:

- numpy
- random
- matplotlib
- os

The source code files accompany this report, stand alone and run independently (python3 *.py). The mathematical equations we used in order to perform 1st problem's requirements, are all included in the Appendix.

Problem 1

In the first problem it is asked to answer questions concerning a generalized linear regression problem defined by the following model:

$$y = \theta_0 x + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_5 x^5 + \eta$$

The following assumptions are made:

1. Components of the weight vector $\hat{\theta}$ are $\boldsymbol{\theta_0 = 0.2, \theta_1 = -1, \theta_2 = 0.9, \theta_3 = 0.7, \theta_5 = -0.2}$.
2. Training set samples have been created from **N** equidistant points in the interval [0,2].
3. Noise samples originating from a Gaussian distribution with **mean 0** and **variance** $\boldsymbol{\sigma_\eta^2}$.

Exercise 1

The first exercise is about applying the Least Squares method to estimate the parameter vector. In the beginning, x is created in the interval [0, 2] with N equidistant points. Then, 20 points of y are computed by the given theta vector $\hat{\theta}$ (1) and used for training. The next step is to add noise to y training points. From there, a 5th degree theta vector is calculated using the LS method. The new theta is used with the relevant polynomial for the testing set, y. The input of the test data are 1000 points following a random distribution again in the plane of [0,2]. So, an output we get is of this form:

$$Predicted\ Theta:\ \theta_0 = 0.64\ \ \theta_1 = -2.6\ \theta_2 = 3.0\ \ \theta_3 = -0.28\ \theta_4 = -0.14$$

In the same implementation Mean Square Error of the trained data is equal to *0.54* and the MSE of the tested data is equal to *0.13*, as shown in the Figure 1 below.

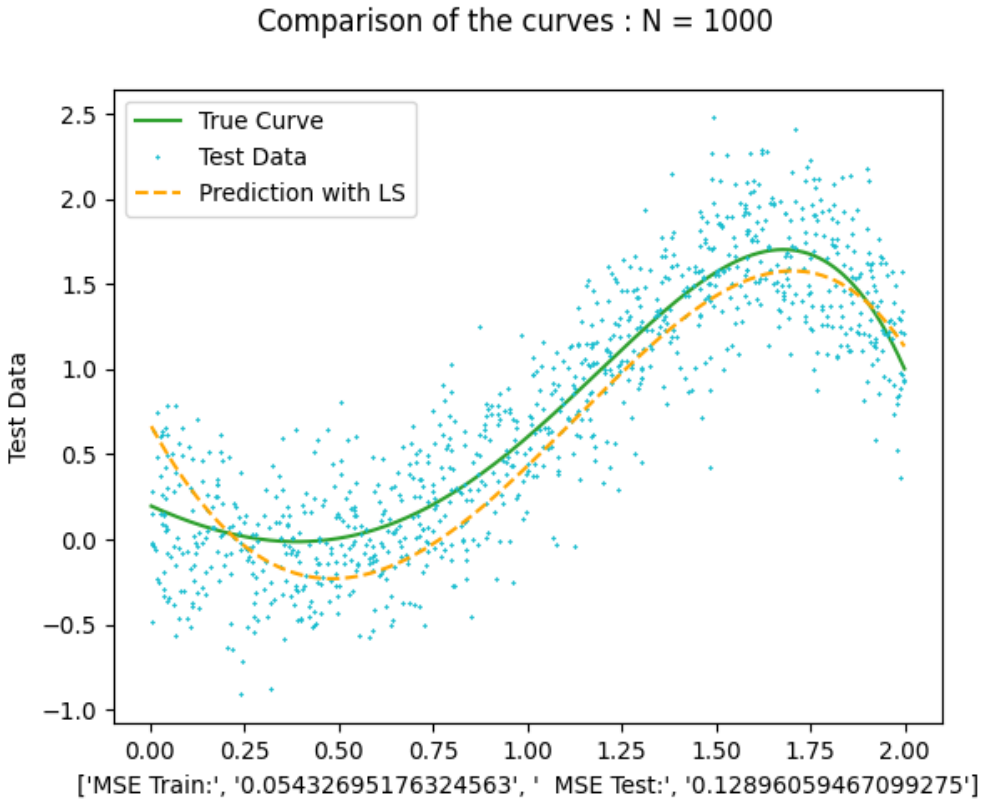

Figure 1: Least Square Method, Testing Points = 1000

Comments

Even with such a naive and simple method, the predicted curve is close enough to the true curve. The predicted curve is being drawn by all 1000 testing points. A different degree of theta vector is used so the differences with the real theta vector are expected as the LS method tries to “fit” to the real one as good as possible. We see that the MSE of the trained data is lower than the MSE of the test data as it was expected. It is important to say that these results are obtained with only 20 training points.

Exercise 2

In the second exercise, for N=20 and $\boldsymbol{\sigma_{\eta}^{2}}$= 0.1, it is requested to apply regression using the Least Squares method and a 2nd degree polynomial. After 100 experiments with different noise samples, the mean and the variance of y are calculated for each point of the training set. The whole process is repeated using a 10th degree polynomial. The two results are compared as shown in the following Figure 2.

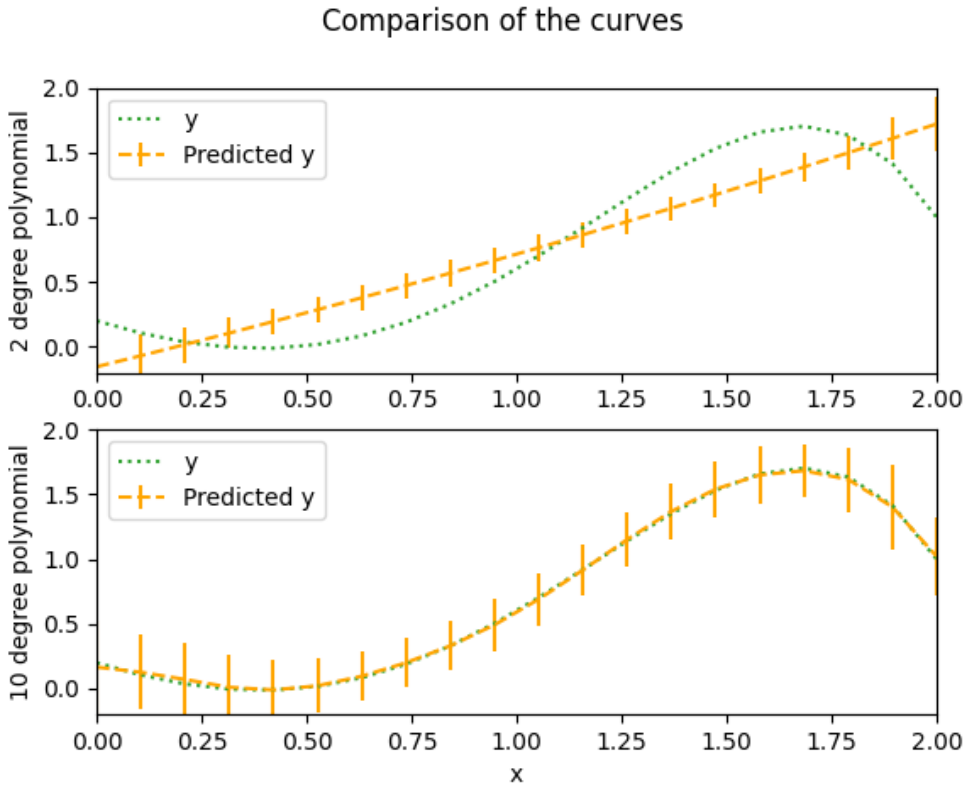

Figure 2: Comparison of fitting 2nd and 10th degree polynomial

Comments

As we can see from the plots shown in the above figure, using a 2nd degree polynomial when the real curve is represented by a 5th degree polynomial, causes bad/ under - fitting. On the other hand, using a 10th degree polynomial instead, causes the so-called *overfitting*. The cause of these "conflicts" is the *bias-variance tradeoff*. The basic dilemma is that by increasing the complexity (more free parameters) of the adopted estimator (10^{th} degree) the bias is reduced but there is an increment of the variance. Oppositely, while using many training points when the model is of a low complexity (2^{nd} degree), as the variance decreases, the bias term gets higher. This tradeoff is shown below, in Figure 3.

```
Calculating with the 2nd degree polynomial
The bias is
[-0.37749436 -0.19381584 -0.0339098   0.09771279  0.19716129  0.26147558
  0.28893616  0.27937439  0.2344826   0.1581243   0.0566443  -0.0608211
 -0.18203393 -0.29134441 -0.36938078 -0.39273914 -0.33367325 -0.15978442
  0.16628872  0.6871803 ]
While the variance is
[0.03230439 0.02131872 0.01432066 0.01033037 0.00849062 0.00806673
 0.00844662 0.00914077 0.00978226 0.01012672 0.0100524  0.00956009
 0.00877317 0.00793761 0.00742194 0.00771728 0.00943732 0.01331836
 0.02021922 0.03112136]


Calculating with the 10th degree polynomial
The bias is
[ 0.01905743  0.03380511 -0.0016878  -0.01722142 -0.0104654   0.00105121
  0.00311972 -0.00771286 -0.02559598 -0.04070395 -0.04495634 -0.03591965
 -0.01776531  0.0008775   0.01143013  0.01057509  0.00352326  0.00156502
  0.00880232 -0.00864146]
While the variance is
[0.1010741  0.08971941 0.08707036 0.05647804 0.04636842 0.03994314
 0.03911819 0.0465985  0.04308032 0.04066343 0.03906225 0.0387944
 0.04065761 0.03454314 0.04299067 0.0548246  0.05154545 0.07300332
 0.09385289 0.09642477]
```

Figure 3: Bias - Variance Dilemma / Implementation Output

Exercise 3

In this exercise, the Ridge Regression method is applied for 20 training points and 1000 tested points. Each time, a different lambda is used. We start from lambda equal to zero which actually refers to the LS method (Exercise 2). Then, the calculations are continuing with other lambdas for the interval of [0, 5].
The output of the program consists of the result of the RR method and the LS method result for comparison, as shown in Figure 4 below.

```
theta real = [[ 0.2 -1.   0.9  0.7 -0.2]]
theta predicted is : [[ 0.36751591 -3.29389394  4.99664535 -1.45252361 -0.03760547]]
MSE of trained data is: 0.06533283799655193
MSE of test data is:  0.11894523913972678
Minumum mse for tested is for λ =
0.479999999999938
Theta of rigde regression =
[[ 0.11748409]
 [-0.56629939]
 [ 0.23795189]
 [ 0.94452085]
 [-0.19566442]]
while the mse_tested is =
0.09641152551343246
```

Figure 4: Ridge Rigression Method / Implementation Output

Comments

The MSE is only a little smaller from the one calculated with the LS method. This, and the fact that the lambda is close to zero, prove that for this specific problem the RR tends to have *almost* the same results with the LS. It is important to say that lambda values are not the same in every implementation because of the noise interference so, there is no standard / optimal lambda that generalizes our model. There are only some that give the minimum MSE every time we run our program. Thus, in our case LS cannot be further generalized.

Exercise 4

The fourth exercise is about performing full Bayesian Inference in order to evaluate output y. Our prior knowledge for the unknown parameter vector is encoded via a Gaussian distribution *G(θ)* with mean θ_0 equal to the true parameter vector $\hat{\theta}$ (1) and covariance matrix $\Sigma_\theta = {\sigma_n}^2 I$ ${\sigma_n}^2 = 0.1$. Using the true model's structure, we evaluated y for 20 randomly selected test data points within the interval [0,2]. The experiment is performed with two different values of σ_n^2 (0.05 and 0.15). The results are presented below in Figure 5.

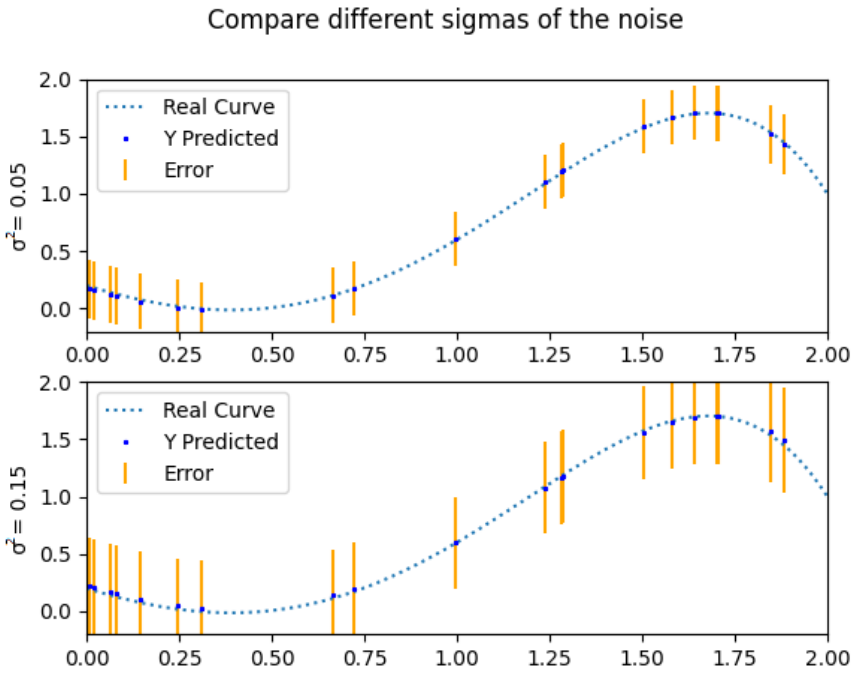

Figure 5: Full Bayesian Inference with different noise variance

Comment

The fact that we choose the mean value of theta to be the true one gives us the privilege of a good prior knowledge of theta. Thus, both our cases have descent results. Both overestimation and underestimation of the noise variance give us convenient fitting for some values of y even though their higher variance. It is important to say that the number of training points is small, so the uncertainty increases. As a result, if we use more training points the estimation of the y is even closer to the real curve.

Exercise 5

In this exercise, we repeat the same steps as Exercise 4 but this time, using mean vector for G(**θ**): $\boldsymbol{\theta}_0 = [-10.54, 0.465, 0.0087, -0.093, -0.004]^T$. The variance of the noise is determined $\boldsymbol{\sigma^2_n}$**=0.05.** The experiment is performed four times according to the following assumptions:

1. N = 20 and $\sigma_\Theta{}^2 = 0.1$
2. N = 20 and $\sigma_\Theta{}^2 = 2$
3. N = 500 and $\sigma_\Theta{}^2 = 0.1$
4. N = 500 and $\sigma_\Theta{}^2 = 2$

Figure 6 and Figure 7 on the next page, show the four cases that were quoted above. The predicted points of y for a different number of training points and different variance of the theta vector are represented with blue dot – points and their error with orange lines.

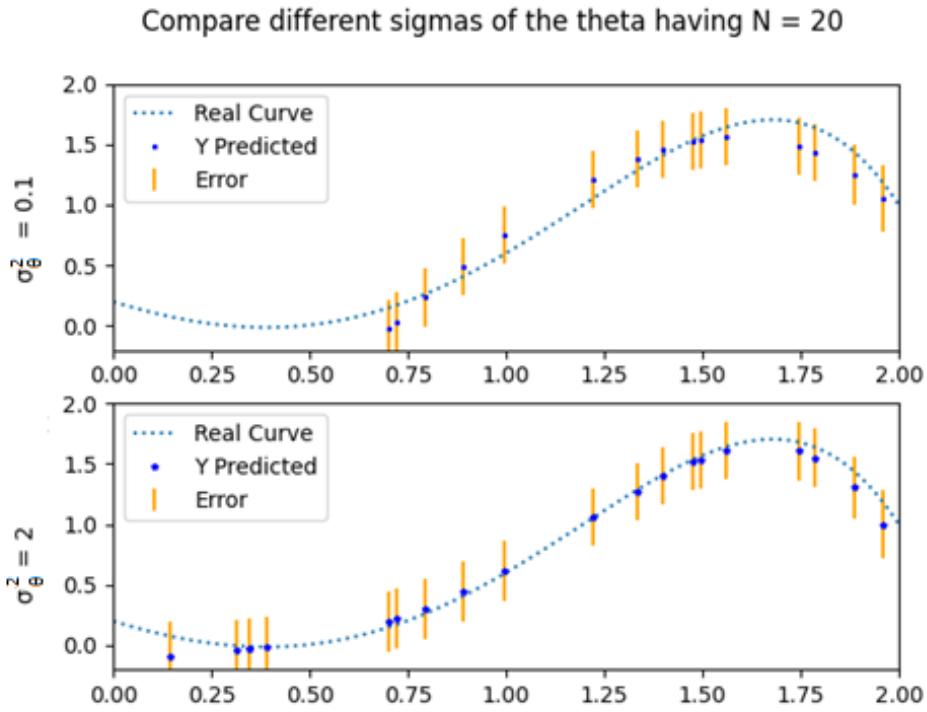

Figure 6: Bayesian Inference with N = 20 and different $\sigma_\Theta{}^2$

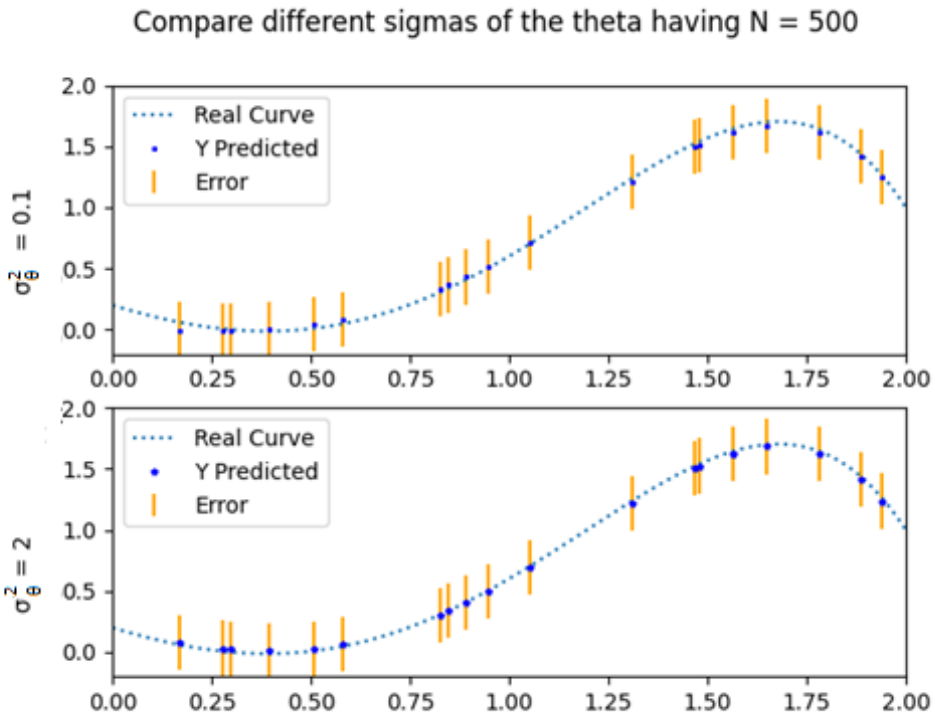

Figure 7: Bayesian Inference with N = 500 and different $\sigma_\theta{}^2$

Comments

As it was expected the more training points the better fitting to the real curve we get. Although, the mean value of the theta vector does not correspond to the real one, our results are close to the true ones. Consequently, our estimation of theta's variance is accurate and additionally, we use a quite large number of training points. In particular, in the first case (1), our prior knowledge is bad, both variance as well as the number of training points are small and as a result y – points do not match the real curve. In the second case (2), our results are better but still as described in the 1st case we lack in achieving optimal results. In both third (3) and fourth (4) cases, our results are better no matter our prior knowledge of theta. However, for $\sigma_\theta{}^2 = 2$, y – points are placed on the real curve since we have both satisfactory prior knowledge and more training points.

Exercise 6

Exercise 6 is about recovering the true variance of the noise by applying the Expectation-Maximization method. The training set is constructed with N = 500 points and the variance of the noise is $\sigma_n{}^2 = 0.05$. The algorithm is initialized with $a = \sigma_\theta{}^{-2} = 1$, $b = \sigma_\eta{}^{-2} = 1$, where a, b are the latent values that will help us estimate y and their errors. For that purpose, the test set we used contains 20 points randomly selected in the interval [0,2].

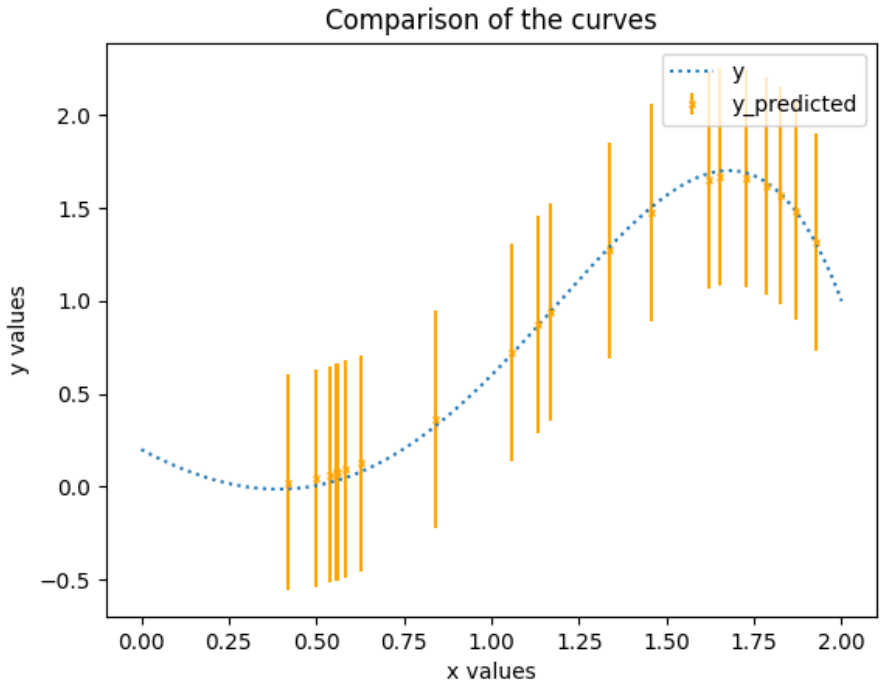

Figure 8: Expectation Maximization Method

Comments

After convergence we conclude that the large training set overcomes the lack of prior knowledge on our predictions and our results are really good. Specifically, a = 1.53 which corresponds $\sigma_\theta{}^2$ = 0.65 and b = 20.76 which corresponds $\sigma_\eta{}^2$ = 0.049. The latter is really close to the true value and thus our predicted model is close to the real one by small error.

Problem 2

In the second problem it is requested to perform different methods of classification in the two different datasets that follow.

I. **IRIS PLANT DATABASE** (Classification of three different kinds of iris plants).
II. **PIMA INDIANS DIABETES DATABASE** (Classification of pregnant Indians of the Pima tribe according to whether they suffer from diabetes or not).

Exercise 1

In the first exercise, we implemented the k Nearest Neighbor classifier for the Iris plant database and the Pima Indians database using 5-fold Cross Validation. Also, the training and testing sets were the same for all the experiments of different k. Although, the optimal method to calculate the distance is the Mahalanobis, for the features in both cases are correlated, we repeated the same experiments calculating the Euclidean distance as well, for further investigation. All the results are presented in the following pages.

Iris Plant database

In the case of the Euclidean distance, we found that the best values for k are 7 and 11, with 98% of success.

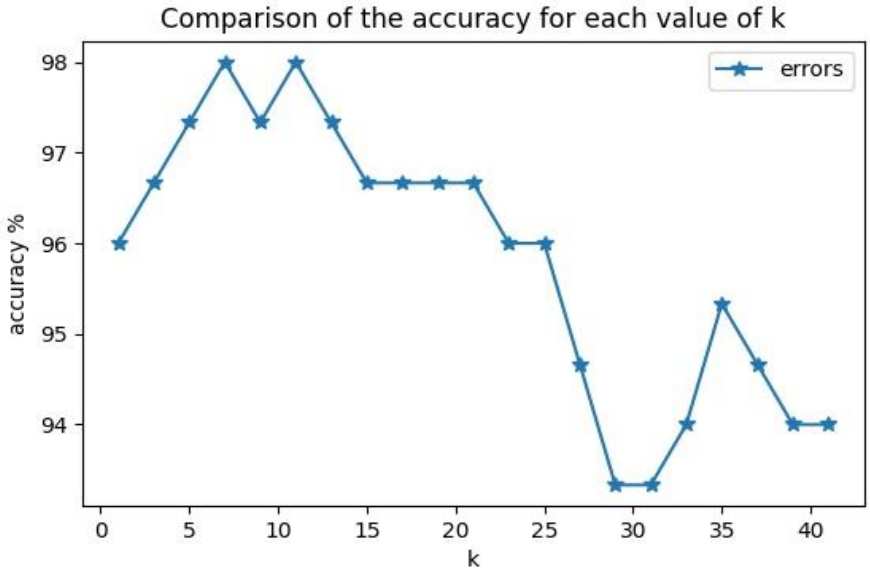

Figure 9: Accuracy of the k-nn classifier for different values of k using Euclidean distance

In the case of the Mahalanobis distance we found that the best values for k are 3, 5, 11, 15, 17, 19, 21, 23, 25, 29, 31, 33, 35, 39, with 98,7% of success.

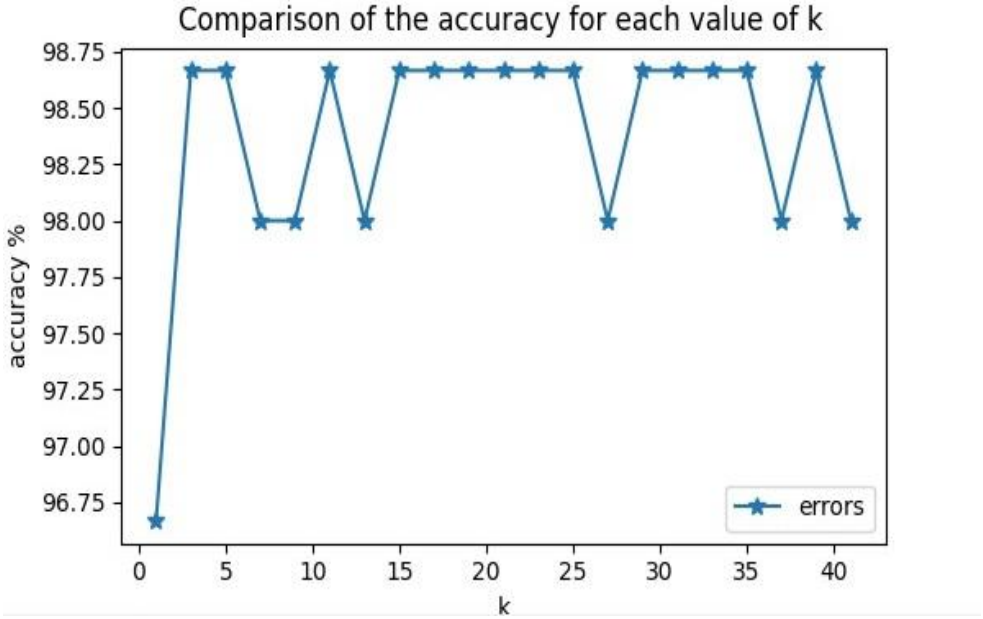

Figure 10: Accuracy of the k-nn classifier for different values of k using Mahalanobis distance

Pima Indians database

In the case of the Euclidean distance, we found that the best value for k is 35, with 75.3% of success.

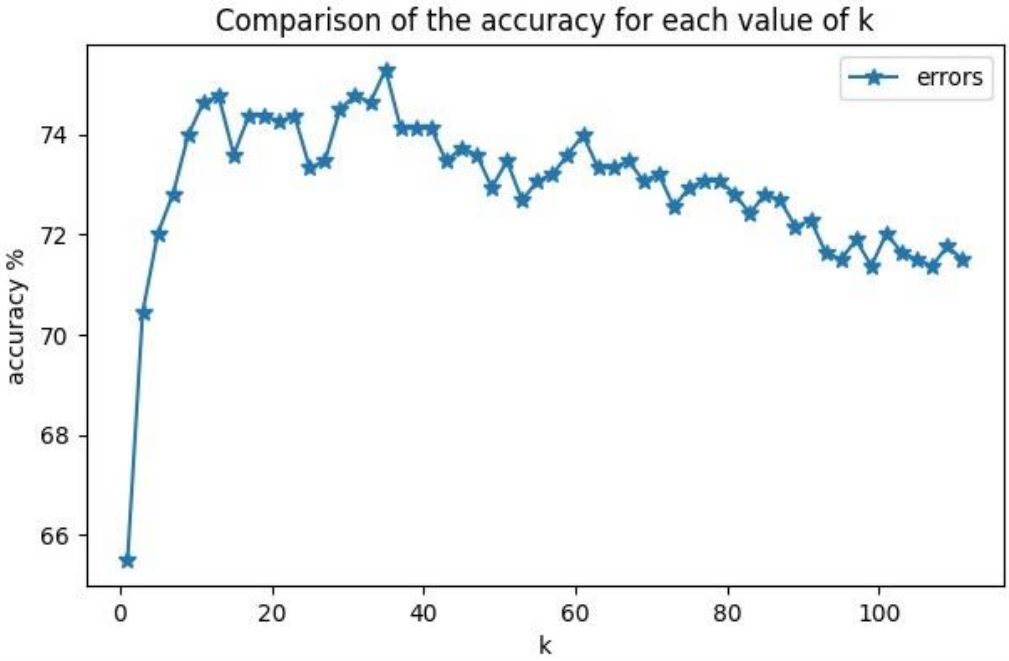

Figure 11: Accuracy of the k-nn classifier for different values of k using Euclidean distance

In the case of the Mahalanobis distance we found that the best values for k are 33, and 37 with 76.5% of success.

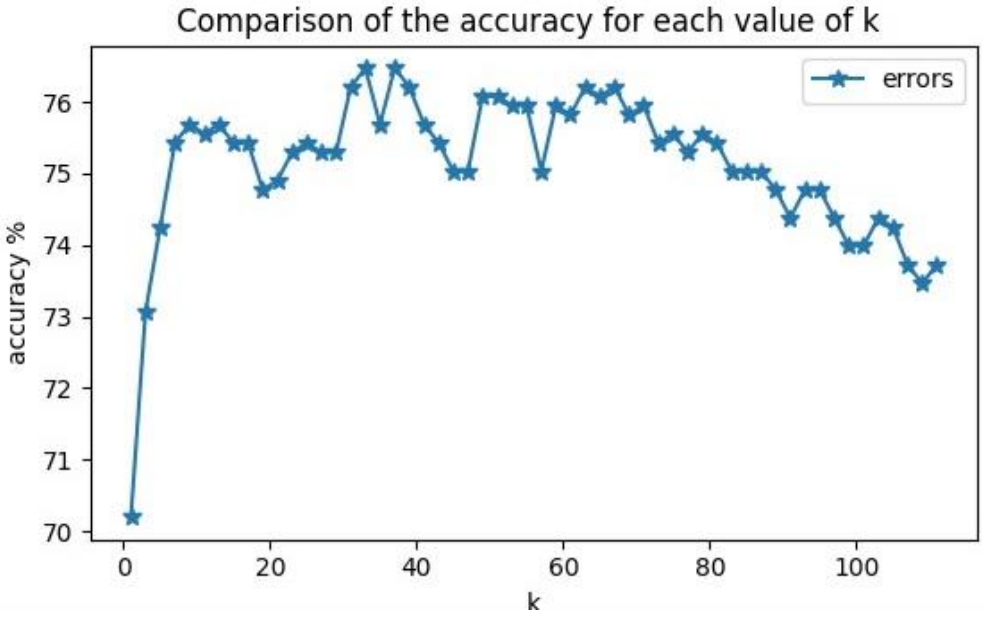

Figure 12: Accuracy of the k-nn classifier for different values of k using Mahalanobis distance

Exercise 2

In this exercise we are requested to estimate the probability density functions for each class regarding given assumptions. The (log)-likelihood function for each query is shown below:

a)

$$L = P(x_1, x_2, \dots, x_N) = \prod_{i=1}^{N} p(x_i) = \prod_{i=1}^{N} \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{\|x_i-\mu\|^2}{2\sigma^2}}$$

$$\log L = \sum_{i=1}^{N} \log p(x_i) = \sum_{i=1}^{N} -\frac{d}{2}\log(2\pi\sigma^2) - \frac{\|x_i-\mu\|^2}{2\sigma^2} = -\frac{dN}{2}\log(2\pi\sigma^2) - \sum_{i=1}^{N} \frac{\|x_i-\mu\|^2}{2\sigma^2}$$

b)

$$L = P(x_1, x_2, \dots, x_N) = \prod_{i=1}^{N} p(x_i) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_\iota-\mu)}$$

$$\log L = \sum_{i=1}^{N} \log p(x_i) = \sum_{i=1}^{N} -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_\iota-\mu)$$

c)

$$L = P(x_1, x_2, \dots, x_N) = \prod_{i=1}^{N}\prod_{j=1}^{d} p(x_{ij}) = \prod_{i=1}^{N}\prod_{j=1}^{d} \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{(x_{ij}-\overline{x_j})^2}{2\sigma^2}}$$

$$\log L = \sum_{i=1}^{N} \log p(x_i) = \sum_{i=1}^{N} \log\left(\prod_{j=1}^{d} \frac{1}{(2\pi{\sigma_i}^2)^{\frac{d}{2}}} e^{-\frac{(x_{ij}-\overline{x_{ij}})^2}{2{\sigma_i}^2}}\right) =$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{d} \log\left(\frac{1}{(2\pi{\sigma_i}^2)^{\frac{d}{2}}} e^{-\frac{(x_{ij}-\overline{x_j})^2}{2{\sigma_i}^2}}\right) = \sum_{i=1}^{N}\sum_{j=1}^{d} -\frac{d}{2}\log(2\pi{\sigma_i}^2) - \frac{(x_{ij}-\overline{x_j})^2}{2{\sigma_i}^2}$$

d)

$$L = P(x_1, x_2, \dots, x_N) = \prod_{k=1}^{N} p(x_k) = \prod_{k=1}^{N}\prod_{i=1}^{d} \frac{1}{Nh}\sum_{j=1}^{N} \frac{1}{\sqrt{2\pi h^2}} e^{\frac{-(x_{ki}-x_{ij})^2}{2h^2}}$$

$$\log L = \sum_{k=1}^{N} \log p(x_k) = \sum_{k=1}^{N} \log\left(\prod_{i=1}^{d} \frac{1}{Nh}\sum_{j=1}^{N} \frac{1}{\sqrt{2\pi h^2}} e^{\frac{-(x_{ki}-x_{ij})^2}{2h^2}}\right) =$$

$$= \sum_{k=1}^{N}\sum_{i=1}^{d} \log\left(\frac{1}{Nh}\sum_{j=1}^{N} \frac{1}{\sqrt{2\pi h^2}} e^{\frac{-(x_{ki}-x_{ij})^2}{2h^2}}\right) = \sum_{k=1}^{N}\sum_{i=1}^{d} -\log(Nh) - \ln\left(\sum_{j=1}^{N} \frac{1}{\sqrt{2\pi h^2}} e^{\frac{-(x_{ki}-x_{ij})^2}{2h^2}}\right)$$

For all the above, in order to measure the goodness of our fit for each class, we used Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC).
As far as the Information Criterion is concerned, we calculated the second order Akaike information criterion as well as the Bayesian information criterion, the formulas of which are shown below. The reason why we use the second order Akaike information criterion instead of the usual is because it gives better results for small sample sizes while for large sample sizes it converges to the usual Akaike information criterion.

$$AIC = -2\log L(\Theta) + 2k + \frac{2k+1}{n-k-1} \qquad BIC = -2\log L(\Theta) + k\ \log n$$

Let f be the number of features in each example of the database.
The number of estimated parameters (k) in the model of query a is 2*f+1.
The number of estimated parameters (k) in the model of query b is (f+1) *f/2.
The number of estimated parameters (k) in the model of query c is 2*f.
The number of estimated parameters (k) in the model of query d is 1.

The notation we use hereinafter in this problem is:

- As ***mean*** we denote the mean values of the features for the examples of the class.
- As ***sigma estimator*** we denote the σ of query a.
- The ***Covariance Matrix*** is the one we use in query b.
- As ***variance*** we denote the σ^2 of the features for the examples of the class.

On the next pages we present our implemented results for each database.

Iris Plant database

```
________Class0_________

mean = [5.006 3.418 1.464 0.244]

sigma estimator = 0.3048080000000006

Covariance Matrix:
[[0.121764 0.098292 0.015816 0.010336]
 [0.098292 0.142276 0.011448 0.011208]
 [0.015816 0.011448 0.029504 0.005584]
 [0.010336 0.011208 0.005584 0.011264]]

Variance:
[0.121764 0.142276 0.029504 0.011264]

________Class1_________

mean = [5.936 2.77  4.26  1.326]

sigma estimator = 0.6123280000000002

Covariance Matrix:
[[0.261104 0.08348  0.17924  0.054664]
 [0.08348  0.0965   0.081    0.04038 ]
 [0.17924  0.081    0.2164   0.07164 ]
 [0.054664 0.04038  0.07164  0.038324]]

Variance:
[0.261104 0.0965   0.2164   0.038324]

________Class2_________

mean = [6.588 2.974 5.552 2.026]

sigma estimator = 0.8706000000000005

Covariance Matrix:
[[0.396256 0.091888 0.297224 0.048112]
 [0.091888 0.101924 0.069952 0.046676]
 [0.297224 0.069952 0.298496 0.047848]
 [0.048112 0.046676 0.047848 0.073924]]

Variance:
[0.396256 0.101924 0.298496 0.073924]
```

Figure 13: Iris Plant database / Output implementation

```
________Class0__________

The AIC for prob_a is : 190.21077152288444
The BIC for prob_a is : 199.52088655002518

The AIC for prob_b is : -75.32984165646073
The BIC for prob_b is : -49.39009100903811

The AIC for prob_c is : -19.261447119165958
The BIC for prob_c is : -4.379897222082253

The AIC for prob_d is : 2717.4602053911203
The BIC for prob_d is : 2719.3097283965485

________Class1__________

The AIC for prob_a is : 329.72797481665447
The BIC for prob_a is : 339.0380898437952

The AIC for prob_b is : 37.435509018122026
The BIC for prob_b is : 63.37525966554464

The AIC for prob_c is : 160.32232575650673
The BIC for prob_c is : 175.20387565359044

The AIC for prob_d is : 2718.072855507753
The BIC for prob_d is : 2719.9223785131812

________Class2__________

The AIC for prob_a is : 400.11088330908115
The BIC for prob_a is : 409.4209983362219

The AIC for prob_b is : 134.5791381030261
The BIC for prob_b is : 160.51888875044872

The AIC for prob_c is : 232.84309040106365
The BIC for prob_c is : 247.72464029814736

The AIC for prob_d is : 2718.585272735312
The BIC for prob_d is : 2720.4347957407404
```

Figure 14: Iris Plant database / Output Implementation / AIC & BIC criterions

Pima Indians database

```
________Class0__________

mean = [  3.298    109.98      68.184     19.664     68.792     30.3042
   0.429734  31.19    ]

sigma estimator = 11187.733555475173

Covariance Matrix:
[[ 9.08519600e+00  7.76796000e+00  7.23916800e+00 -5.30587200e+00
  -3.92920160e+01  3.81948400e-01 -7.20027320e-02  2.01233800e+01]
 [ 7.76796000e+00  6.81995600e+02  9.08536800e+01  6.22128000e+00
   9.10377840e+02  2.64314840e+01  7.45542680e-01  6.94078000e+01]
 [ 7.23916800e+00  9.08536800e+01  3.25622144e+02  5.02138240e+01
   1.33002272e+02  5.03454272e+01  1.47144944e-01  4.51570400e+01]
 [-5.30587200e+00  6.22128000e+00  5.02138240e+01  2.21267104e+02
   6.06452112e+02  5.01206112e+01  4.23028624e-01 -2.83981600e+01]
 [-3.92920160e+01  9.10377840e+02  1.33002272e+02  6.06452112e+02
   9.75479674e+03  1.92872674e+02  6.71014467e+00 -1.71800480e+02]
 [ 3.81948400e-01  2.64314840e+01  5.03454272e+01  5.01206112e+01
   1.92872674e+02  5.90156024e+01  1.62197517e-01  3.22980200e+00]
 [-7.20027320e-02  7.45542680e-01  1.47144944e-01  4.23028624e-01
   6.71014467e+00  1.62197517e-01  8.92731152e-02  1.45104540e-01]
 [ 2.01233800e+01  6.94078000e+01  4.51570400e+01 -2.83981600e+01
  -1.71800480e+02  3.22980200e+00  1.45104540e-01  1.35861900e+02]]

Variance:
[9.08519600e+00 6.81995600e+02 3.25622144e+02 2.21267104e+02
 9.75479674e+03 5.90156024e+01 8.92731152e-02 1.35861900e+02]

________Class1__________

mean = [  4.86567164 141.25746269  70.82462687  22.1641791  100.3358209
  35.14253731   0.5505      37.06716418]

sigma estimator = 21137.30579870758

Covariance Matrix:
[[ 1.39446425e+01 -6.49899755e+00  1.01704723e+01 -5.21675206e+00
  -4.06116061e+01 -4.30622633e+00 -9.60335821e-02  1.81918579e+01]
 [-6.49899755e+00  1.01633297e+03  4.69817192e+01  2.11629539e+01
   1.15345458e+03  1.16521079e+01  3.13677239e-01  3.44006182e+01]
 [ 1.01704723e+01  4.69817192e+01  4.60174468e+02  8.52675986e+01
   2.65379789e+02  2.08309674e+01  2.75236940e-01  6.16908833e+01]
 [-5.21675206e+00  2.11629539e+01  8.52675986e+01  3.11405881e+02
   1.11529561e+03  3.99210013e+01  1.79638806e+00 -1.77759523e+01]
 [-4.06116061e+01  1.15345458e+03  2.65379789e+02  1.11529561e+03
   1.91629021e+04  5.53069837e+01  5.22539925e+00  3.62871464e+01]
 [-4.30622633e+00  1.16521079e+01  2.08309674e+01  3.99210013e+01
   5.53069837e+01  5.25538622e+01  3.68475000e-01 -1.49215137e+01]
 [-9.60335821e-02  3.13677239e-01  2.75236940e-01  1.79638806e+00
   5.22539925e+00  3.68475000e-01  1.38130519e-01 -3.58541045e-01]
 [ 1.81918579e+01  3.44006182e+01  6.16908833e+01 -1.77759523e+01
   3.62871464e+01 -1.49215137e+01 -3.58541045e-01  1.19853698e+02]]

Variance:
[1.39446425e+01 1.01633297e+03 4.60174468e+02 3.11405881e+02
 1.91629021e+04 5.25538622e+01 1.38130519e-01 1.19853698e+02]
```

Figure 15: Pima Indians database / Output Implementation

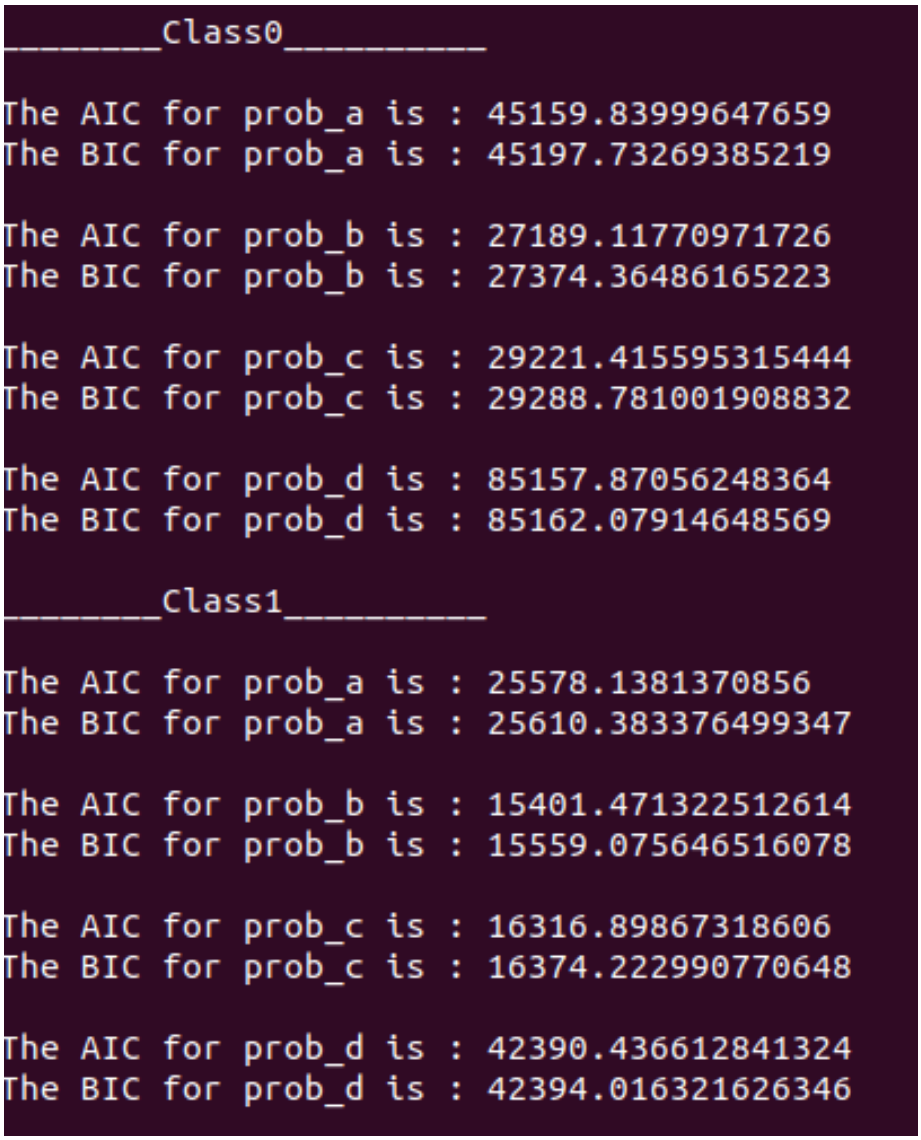

Figure 16: Pima Indians database / Output Implementation AIC & BIC criterions

Comments

From the above results we can conclude that in both cases the best model is that of *query b* as it seems to have less information loss. After b comes the model of *query c*, then the model of *query a* and last the model of *query d.*

Exercise 3

In exercise three, having in mind all the assumptions made in Exercise about the probability density function, we implemented Bayes classifier and calculated the accuracy of each one, on the Iris plant database and the Pima Indians database, using 5-fold cross validation.

Iris Plant database

```
In this problem we use 5-fold cross validation

The accuracy of problem's a method is 92.0
The accuracy of problem's b method is 95.33333333333334
The accuracy of problem's c method is 95.33333333333333
The accuracy of problem's d method is 86.0
```

Figure 17: Iris Plant database / Accuracy of models a, b, c, d

Pima Indians database

```
In this problem we use 5-fold cross validation

The accuracy of problem's a method is 63.26797385620914
The accuracy of problem's b method is 66.1437908496732
The accuracy of problem's c method is 74.50980392156863
The accuracy of problem's d method is 68.10457516339869
```

Figure 18: Pima Indians database / Accuracy of models a, b, c, d

Comments

Comparing the results to the previous findings we can say that although the criteria are not 100% accurate about the best model, they were close enough to the best ones which are c and b. Also, we can observe that although the method of query d (Parzen Windows) works well for a large sample, it does not fit well with small samples. Additionally, it's worth mentioning that, if the true distribution is a Gaussian density and we use a gaussian kernel then the optimal value of the bandwidth is $h^1 = 1.06*\sigma*N^{1/5}$.
In our problem the results of this bandwidth are shown below:

[1] http://csyue.nccu.edu.tw/ch/Kernel%20Estimation(Ref).pdf

Iris Plant database

The accuracy of problem's d method is 68.0

Pima Indians database

The accuracy of problem's d method is 68.10457516339869

Comments

In the iris plant database, we see that the accuracy decreases while the accuracy of the Pima Indians database increases, using the new bandwidth. Considering the reference *(1)*, we can conclude that the true distribution of the Iris Plant database is not Gaussian while the opposite is true for the Pima Indians database. Additionally, we can say that the k-nn method performed better than the Bayes classifiers as in Exercise 1 the results for the Iris Plant and Pima Indians database have shown greater accuracy.

Exercise 4

In this exercise, the perceptron algorithm is implemented in order to perform classification on the Iris plant database and to examine if each class is linearly separable from the other two combined. The implementation results are the following:

```
_____Setosa_________
Iteration : 4
W = [-1, 1.299999999999999, 4.1, -5.200000000000001, -2.1999999999999997]
The number of errors is 0
The accuracy is 100.0%

_____Versicolor_____
Iteration : 100000
W = [102, 65.40000000234689, -62.599999999509976, -7.799999999991599, -143.09999999727052]
The number of errors is 8
The accuracy is 99.94666666666667%

_____Virginica______
Iteration : 100000
W = [1627, -105.29999999841132, -292.19999999571155, 427.699999998848, 666.300000001258]
The number of errors is 4
The accuracy is 99.97333333333333%
```

Figure 19: Perceptron Algorithm / Implementation Output

Comments

From the output of the algorithm, which is shown in Figure 19 we can conclude that only the Iris-Setosa class is linearly separable from the other two (perceptron's algorithm converges). For the other two classes, no matter how many epochs we train the network it will never be able to classify all examples correctly. In our case, we trained it for 100000 epochs, and it could not produce any values for the weights, which would correctly classify the examples of the classes.

Appendix

Regre-ssion method	Generalized Linear regression (white Gaussian noise)
Least Squares	$\hat{\theta}_{LS} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \boldsymbol{y}$
Ridge Regre-ssion	$\hat{\theta}_{RR} = \left(\Phi^T \Phi + \lambda \mathbf{I}\right)^{-1} \Phi^T \boldsymbol{y}$
Maximum Likelihood	• Noise model: $\boldsymbol{\eta} = \boldsymbol{y} - \Phi\boldsymbol{\theta} \rightarrow N(0, \sigma_\eta^2 I)$ $\hat{\theta}_{ML} = \hat{\theta}_{LS} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \boldsymbol{y}$
Maximum a Posteriori	• Noise model: $\boldsymbol{\eta} = \boldsymbol{y} - \Phi\boldsymbol{\theta} \rightarrow N(0, \sigma_\eta^2 I)$ • Prior: $p(\boldsymbol{\theta}) \rightarrow N(\boldsymbol{\theta}_0, \sigma_\theta^2 I)$ $\hat{\boldsymbol{\theta}}_{MAP} = \boldsymbol{\theta}_0 + \frac{1}{\sigma_\eta^2}\left(\frac{1}{\sigma_\theta^2} I + \frac{1}{\sigma_\eta^2}\Phi^T\Phi\right)^{-1} \Phi^T \left(\boldsymbol{y} - \Phi\boldsymbol{\theta}_0\right)$
Bayesian Inference	• Noise model: $\boldsymbol{\eta} = \boldsymbol{y} - \Phi\boldsymbol{\theta} \rightarrow N(0, \sigma_\eta^2 I)$ • Prior: $p(\boldsymbol{\theta}) \rightarrow N(\boldsymbol{\theta}_0, \sigma_\theta^2 I)$ • Posterior: $p\left(\boldsymbol{\theta} \mid \boldsymbol{y}\right) \rightarrow N(\boldsymbol{\theta} \mid \mu_{\theta\mid y}, \Sigma_{\theta\mid y})$ $\mu_{\theta\mid y} = \boldsymbol{\theta}_0 + \frac{1}{\sigma_\eta^2}\left(\frac{1}{\sigma_\theta^2} I + \frac{1}{\sigma_\eta^2}\Phi^T\Phi\right)^{-1} \Phi^T \left(\boldsymbol{y} - \Phi\boldsymbol{\theta}_0\right)$ $\Sigma_{\theta\mid y} = \left(\frac{1}{\sigma_\theta^2} I + \frac{1}{\sigma_\eta^2}\Phi^T\Phi\right)^{-1}$ $p\left(y \mid \boldsymbol{y}\right) \rightarrow N(y \mid \mu_y, \sigma_y^2)$ $\mu_y = \phi^T(x)\mu_{\theta\mid y},\ \sigma_y^2 = \sigma_\eta^2 + \sigma_\eta^2\sigma_\theta^2\phi^T(x)\left(\sigma_\eta^2 I + \sigma_\theta^2\Phi^T\Phi\right)^{-1}\phi(x)$

Regression method	Generalized linear regression (white Gaussian noise)

Expectation maximization

- Noise model: $\boldsymbol{\eta} = \boldsymbol{y} - \Phi\boldsymbol{\theta} \rightarrow N(0, \sigma_\eta^2 I)$
- Prior: $p(\theta) \rightarrow N(0, \sigma_\theta^2)$
- Posterior: $p(\boldsymbol{\theta}|\boldsymbol{y}) \rightarrow N(\boldsymbol{\mu}_{\theta|y}, \Sigma_{a,})$

$$\mu_{\theta|y} = b\Sigma_{\theta|y}\Phi^T\boldsymbol{y}, \Sigma_{a,} = \left(aI + b\Phi^T\Phi\right)^{-1}$$

- Goal is to estimate: $a = 1/\sigma_\theta^2, b = 1/\sigma_\eta^2$

Initialize: $a^{(0)}, b^{(0)}$

While $|a^{(j+1)} - a^{(j)}| > \varepsilon, |b^{(j+1)} - b^{(j)}| > \varepsilon$

E-step:

$$\mu_{a,}^{(j)} = b^{(j)}\Sigma_{a,}^{(j)}\Phi^T\boldsymbol{y}, \Sigma_{a,}^{(j)} = \left(a^{(j)}I + b^{(j)}\Phi^T\Phi\right)^{-1}$$

$$A^{(j)} = \left\|\boldsymbol{\mu}_{a,}^{(j)}\right\|^2 + \text{trace}\left\{\Sigma_{a,}^{(j)}\right\}$$

$$B^{(j)} = \left\|\boldsymbol{y} - \Phi\boldsymbol{\mu}_{a,}^{(j)}\right\|^2 + \text{trace}\left\{\Phi\Sigma_{a,}^{(j)}\Phi^T\right\}$$

$$Q(a, b, a^{(j)}, b^{(j)}) = \tfrac{K}{2}\ln a + \tfrac{N}{2}\ln b - \tfrac{a}{2}A^{(j)} - \tfrac{b}{2}B^{(j)} + \text{constant}$$

M-step:

$$\frac{\partial Q}{\partial a} = 0 \Rightarrow a^{(j+1)} = \frac{K}{A^{(j)}}$$

$$\frac{\partial Q}{\partial a} = 0 \Rightarrow b^{(j+1)} = \frac{N}{B^{(j)}}$$