EE 565 Machine Learning
Project 1 Report

Mehrdad Ghyabi

***Abstract*—This is a summarized report about implementation and evaluation of three different types of machine learning algorithms. Moreover, the effect of input parameters is investigated in each case. First, two famous clustering algorithms are defined and tested. After that, polynomial regression using least square solution is implemented. Last but not least, K-Nearest Neighbors algorithm is investigated in three different cases.**

***Index Terms*—K=Means, Polynomial regression, K-Nearest Neighbors.**

I. Introduction

This report is consisted of eight sections. After the introduction, in section II, different data sets are clustered using batch K-Means. Effect of number of data points on the precision of resulted centroids, and effect of number of clusters on performance of algorithm are also investigated in part II. In part III, implementation of Online K-means algorithm and effect of learning rate on its performance is investigated. In section IV, different scenarios of image segmentation are investigated using K-means algorithm. In section V, the problem of polynomial curve fitting is solved by using the least squares solution. Next, the effect of regularization on performance of polynomial regression is investigated in section VI. In the last case, different data sets from project 0 are classified by implementing K-nearest neighbors algorithm in section VII. All the reported runtimes in this text are resulted from running Python codes on a "Dell Precision T1700" workstation with an Intel Xeon E3-1240 v3 @ 3.40GHz CPU. Runtimes may differ according to the system specifications.

II. Problem 1: Batch K-means

A. Circular Symmetric Gaussian Data Set and K = 2

In this part, Batch K-means algorithm is used to group the input data set into two clusters. The input data set is consisted of two circular symmetric gaussian data sub-sets, centered at origin and (5,5), accumulated together.

As the number of data points is increased the performance of algorithm improves. The metric used to measure the performance of the algorithm is the sum of Euclidian distances from estimated centroids to actual ones, as described in (1) in which, μ is the estimated centroid and C is the actual one.

$$error = \sum_{i=1}^{2} \|\mu_i - C_i\| \tag{1}$$

As the data set is generated randomly each time, and also the initiation of centroids is a random process, the clustering process has been repeated 50 times for each number of datapoints to result in a smoother behavior. The resulting plot is presented in Figure 1. Runtime is about 20 seconds.

Initialization of the centers also affects the results. The algorithm will converge, as long as initial centers are selected from data points. But if one of initial centers are far away from actual centers the algorithm won't converge. This is shown in a piece of code with file name "problem1b.py".

to have a sense of number of iterations needed for convergence, the problem was solved 1000 times for datasets containing 50 data point. The results are shown in Table I.

TABLE I
Information about Number of Iterations

Quantity	Value
maximum number of iterations	8.00
minimum number of iterations	2.00
Average number of iterations	3.74

B. Circular Symmetric Gaussian Data Set and $K \geq 2$

With a fixed number of data points, the value of the objective function J decrease as the number of clusters K is increased from 2 to 20. The resulting plot is presented in Figure 2. The algorithm has been repeated 20 times at each value of K and the average J is reported.

Figure 3 to Figure 5 show three different cases of center initialization and the evolution of initial centers to estimated centers. Centers tend to move toward more crowded regions. Ii is interesting that initiated centers seem to be willing to stay in the same data sub-set that they have been initiated in.

III. Problem 2: Online K-means

The same dataset from Problem 1 has been clustered using online K-mean algorithm and the effect of learning rate on epoch iterations needed for convergence was investigated. Figure 6 shows the resulting plot of average number of epoch

iterations versus learning rate. The experiment was repeated 10 times for each of 15 learning rate values and the runtime was 80 seconds. The best learning rate turned out to be 0.0008 with average epoch iterations of 42.9.

IV. Problem 3: K-means Application: Image Segmentation

In this section the batch K-means algorithm was implemented on image segmentation problem. Three images were used as input and effect of cluster number on the performance of algorithm was investigated.

In case of "machine-learning-1" image it seems that *K*=5 is the minimum number of clusters with acceptable details in the resulting image. Figure 7 shows the resulting picture with *K*=4, Figure 8 shows the resulting picture with *K*=8 and Figure 9 shows the resulting picture with *K*=5.

In case of "Nature-brain" image it seems that *K*=4 is the minimum number of clusters with acceptable details in the resulting image. It was expected, since the dominant color in this image is green so only 4 colors can show enough details and adding to number of clusters just helps with the textures not the details. Figure 10 shows the resulting picture with *K*=4 and Figure 11 shows the resulting picture with *K*=8.

In case of "nature-1" image it seems that *K*=8 is the minimum number of clusters with acceptable details in the resulting image. The dominant color of image is blue but there is a big area of green in the image and it would not show up in the image until the number of clusters is increased to 8. Figure 12 shows the resulting picture with *K*=4, figure 13 shows resulting image with *K*=6 and Figure 14 shows the resulting picture with *K*=8.

Next, centroids from image "nature-1" was used to segment the other two pictures. In case of "machine-learning-1" only 6 clusters seem to be enough since the two pictures have a close color theme. Figure 15 shows "machine-learning" segmented with 4 centers from "nature-1", Figure 16 shows the same case with 6 centers and Figure 17 is the result with 8 centers. In case of "Nature-Brain" up to 10 clusters have to be used for the resulting image to show enough details since the two pictures have different color themes. Figure 18 shows "Nature-Brain" segmented with 4 centers from "nature-1", Figure 19 shows the same case with 8 centers and Figure 20 is the result with 10 centers.

In the next step, centers were extracted from "machine-learning-1" and were used to segment "nature-1". It seems that at least 10 centers are needed for an acceptable result. Figure 21 shows the resulting image with 10 centers. The process was repeated to extract centers from "Nature-Brain" and segment "nature-1" with those centers. A number of 8 centers seem to result in a good-enough image. The resulting image is presented in Figure 22.

V. Problem 4: Polynomial Curve Fitting

Polynomial curve fitting algorithm was coded into Python and results are presented in Table II, and Figure 23 to Figure 26 for polynomials of zero to 9 degree respectively.

TABLE II
Tuned Weights

	M=0	M=1	M=3	M=9
w_0^*	0.186	0.820	0.314	0.349
w_1^*		-1.268	7.985	232.124
w_2^*			-25.426	-5316.195
w_3^*			17.374	48517.719
w_4^*				-231402.305
w_5^*				639398.324
w_6^*				-1060748.64
w_7^*				1041382.08
w_8^*				-557145.342
w_9^*				125082.149

In the next step the effect of the degree of polynomial on the E_{RMS} of both training and test data was investigated. The resulting plot is presented in Figure 27. Here E_{RMS} is calculated using (2).

$$E_{RMS} = \sqrt{\frac{1}{2}\sum_{n=1}^{N}[y(x_n, \boldsymbol{w}^*) - t_n]^2} \qquad (2)$$

The fitting process was repeated using 15 noisy data point as training set (Figure 28) and 100 noisy data points as train set (Figure 29) using polynomials of ninth degree.

VI. Problem 5: Polynomial Regression with Regularization

Polynomial regression with regularization term was performed on the given data set as training set with two values of $\lambda = 1$ and $\lambda = e^{-18}$ and polynomials of degree nine. Results are provided in Figure 30, Figure 31 and Table III.

TABLE III
Tuned Weights

	$ln\ \lambda = -18$	$ln\ \lambda = 0$
w_0^*	0.349	0.460
w_1^*	5.768	-0.345
w_2^*	-14.965	-0.365
w_3^*	29.857	-0.245
w_4^*	-95.620	-0.123
w_5^*	35.172	-0.023
w_6^*	168.713	0.054
w_7^*	-18.035	0.113
w_8^*	-273.175	0.160
w_9^*	162.193	0.196

VII. Problem 6: Classification

K-nearest neighbors algorithm was used to classify data sets generated with three different distributions (Concentric

Gaussian, Double Moon and Gaussian XOR). Each case was repeated three times with $K = 1$, $K = 5$ and $K = 15$ to investigate the effect of that parameter on the results.

In case of data with Concentric Gaussian distribution (Figure 32), as K is increased the resulting boundary gets smoother but the precision of algorithm is decreased. Results are presented in Figure 33 to Figure 35.

In case of data with Double Moon distribution (Figure 36), as K is increased the resulting the resulting classification improves slightly. Results are presented in Figure 37 to Figure 39.

In case of data with Gaussian XOR distribution (Figure 40), the classifier has a poor performance changing the K changes the boundary but it doesn't have a significant effect on the performance of the classifier. Results are presented in Figure 41 to Figure 43.

.

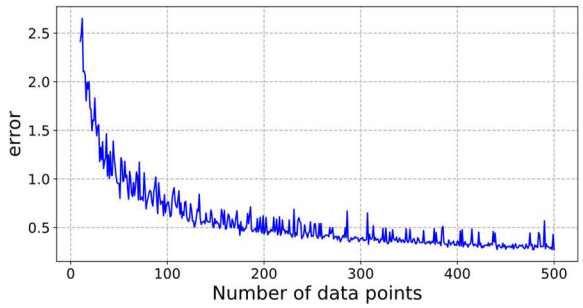

Fig. 1 Decreasing trend of "error" as the number of data points is increased

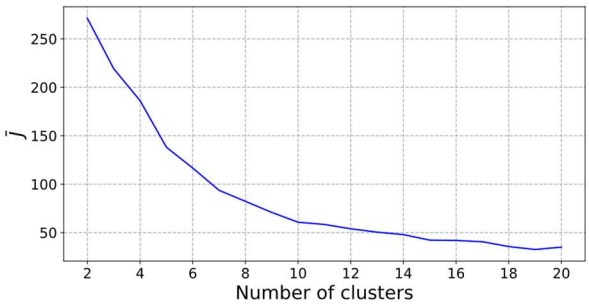

Fig. 2 Objective function decreases as number of clusters increases

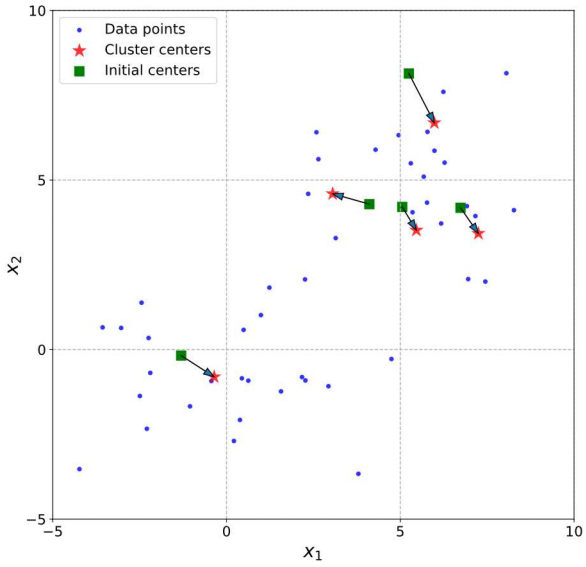

Fig. 3 Effect of initialization: Case 1

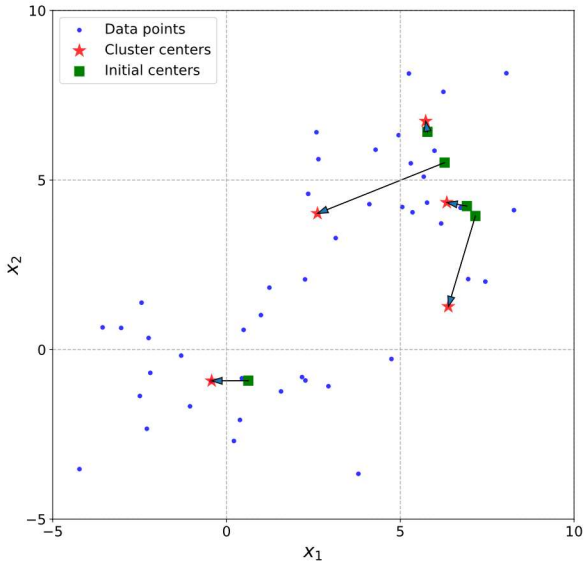

Fig. 4 Effect of initialization: Case 2

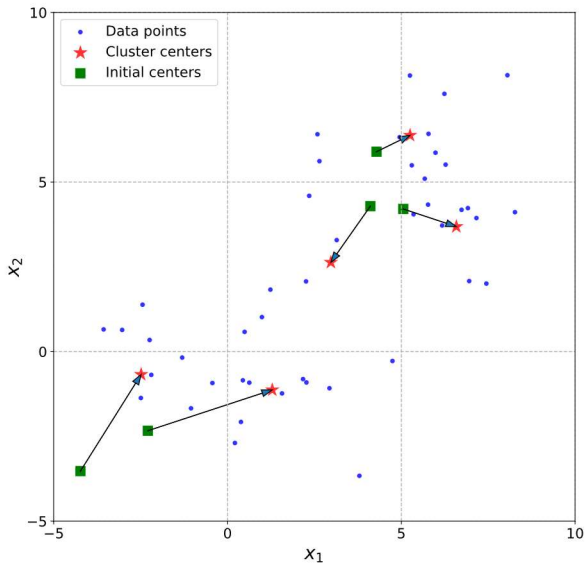

Fig. 5 Effect of initialization: Case 3

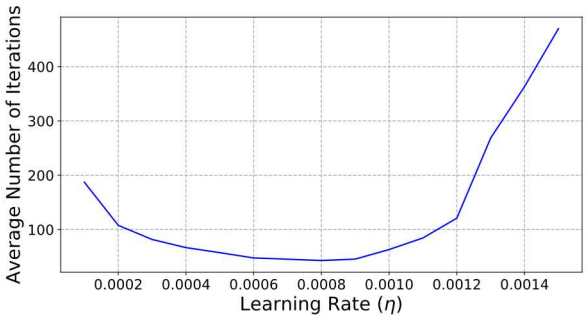

Fig. 6 Average number of epoch iterations versus learning rate

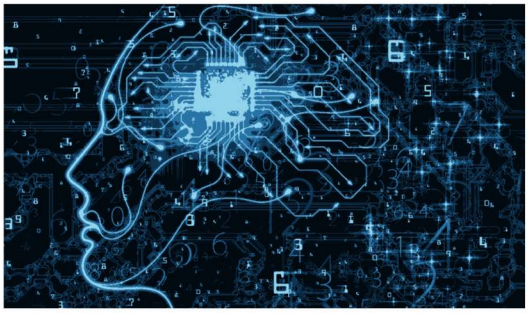

Fig. 7 machine-learning-1 segmented with 4 clusters

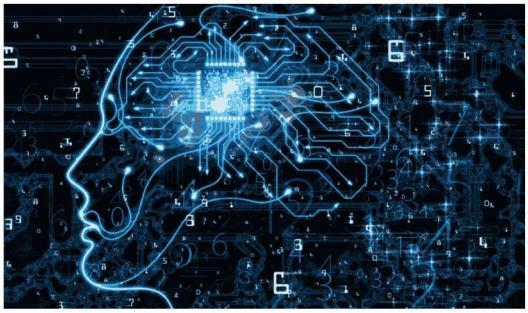

Fig. 8 machine-learning-1 segmented with 8 clusters

Fig. 9 machine-learning-1 segmented with 5 clusters

Fig. 10 Nature-Brain segmented with 4 clusters

Fig. 11 Nature-Brain segmented with 8 clusters

Fig. 12 nature-1 segmented with 4 clusters

Fig. 13 nature-1 segmented with 6 clusters

Fig. 14 nature-1 segmented with 8 clusters

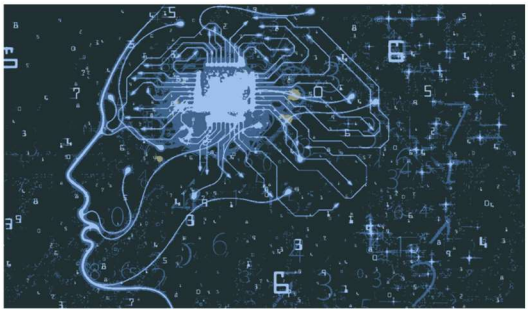

Fig. 15 machine-learning-1 segmented with 4 centers from nature-1

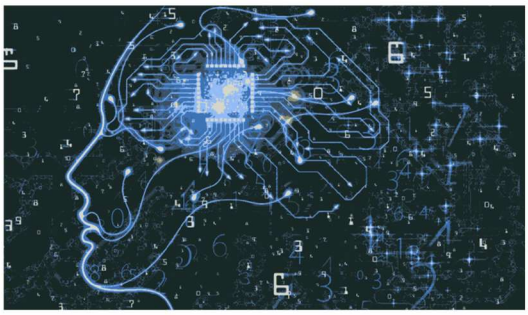

Fig. 16 machine-learning-1 segmented with 6 centers from nature-1

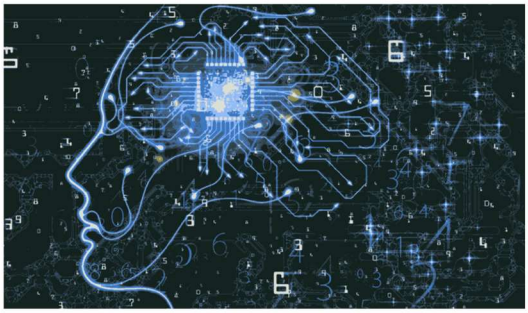

Fig. 17 machine-learning-1 segmented with 8 centers from nature-1

Fig. 18 Nature-Brain segmented with 4 centers from nature-1

Fig. 19 Nature-Brain segmented with 8 centers from nature-1

Fig. 20 Nature-Brain segmented with 10 centers from nature-1

Fig. 21 nature-1 segmented with 10 centers from machine-learning-1

Fig. 22 nature-1 segmented with 10 centers from Nature-Brain

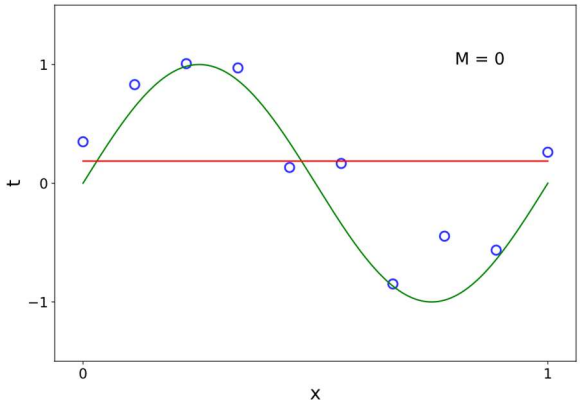

Fig. 23 Curve fitted with using a polynomial of zeroth degree

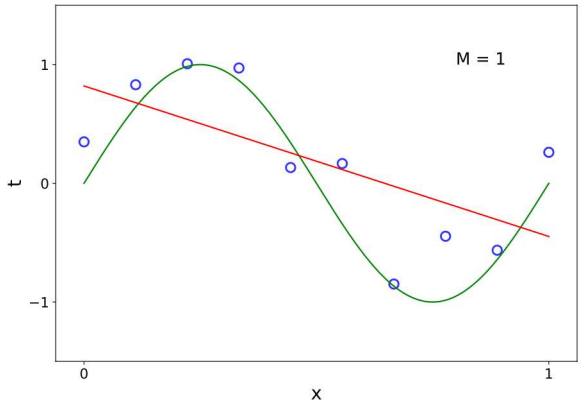

Fig. 24 Curve fitted with using a polynomial of first degree

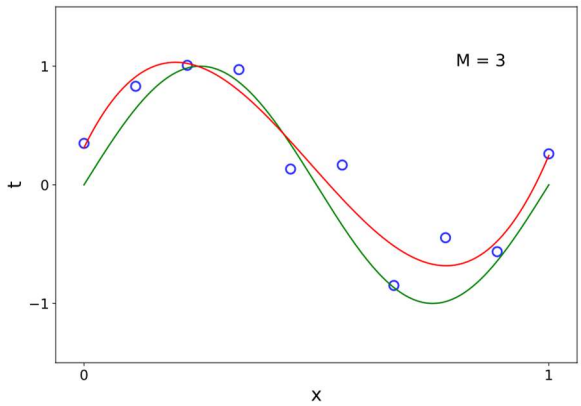

Fig. 25 Curve fitted with using a polynomial of third degree

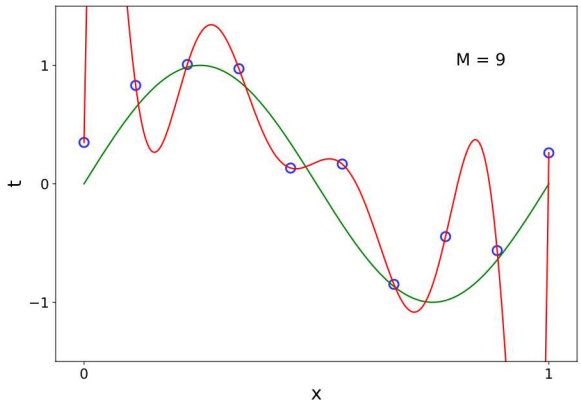

Fig. 26 Curve fitted with using a polynomial of ninth degree

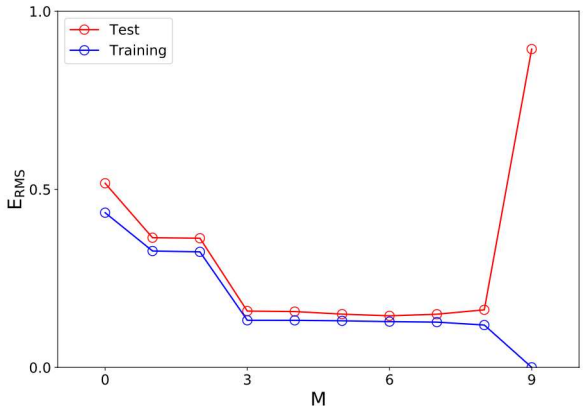

Fig. 27 Test and Train E_{RMS} versus degree of polynomial

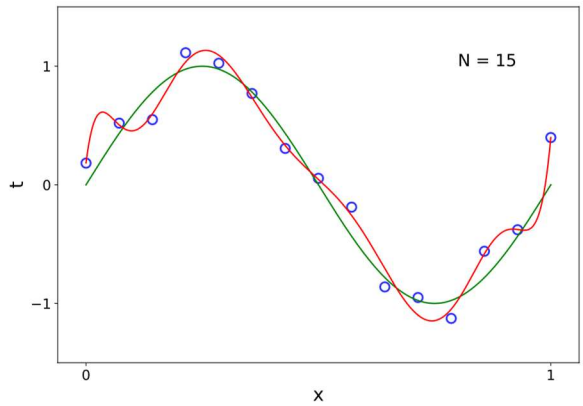

Fig. 28 Curve fitting using 15 noisy data points as training set

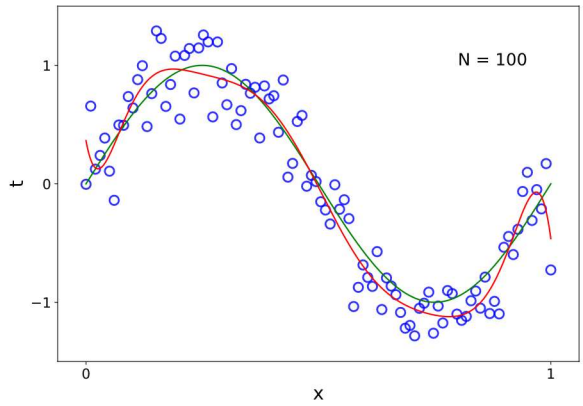

Fig. 29 Curve fitting using 100 noisy data points as training set

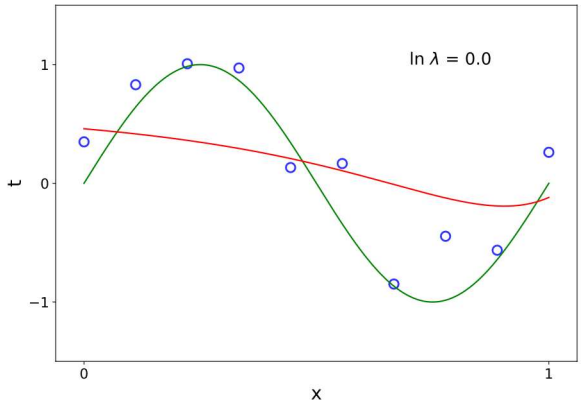

Fig. 30 Regularized curve fitting using $\lambda = 1$

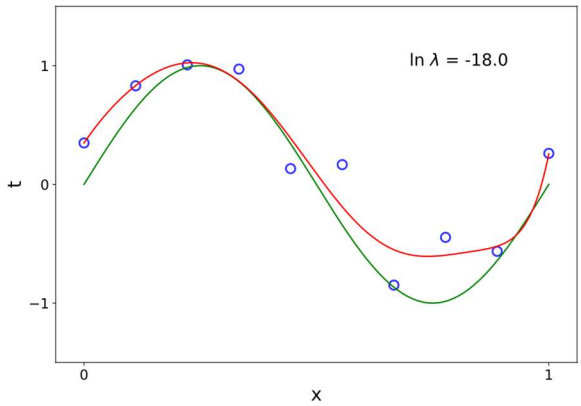

Fig. 31 Regularized curve fitting using $\lambda = e^{-18}$

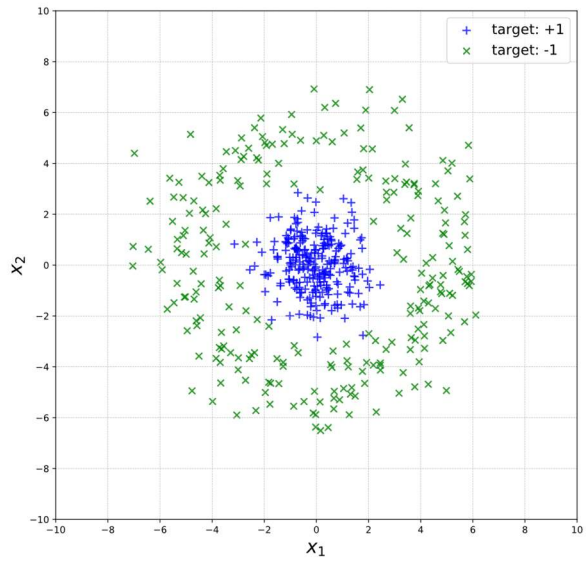

Fig. 32 Concentric Gaussian data

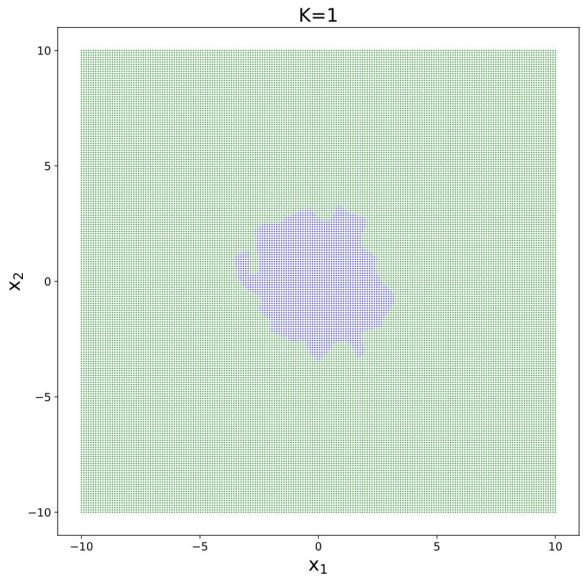

Fig. 33 Result of classifying Gaussian Concentric data with *K*=1

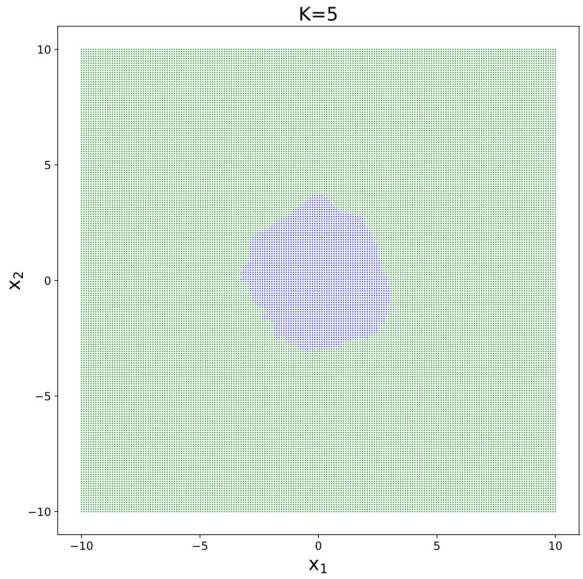

Fig. 34 Result of classifying Gaussian Concentric data with *K*=5

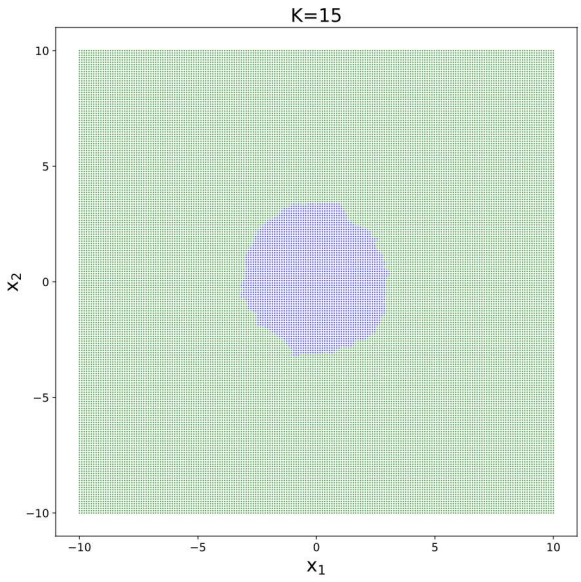

Fig. 35 Result of classifying Gaussian Concentric data with *K*=15

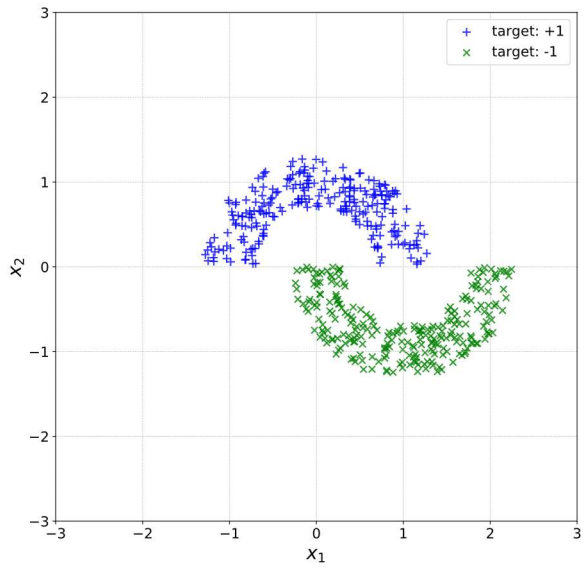

Fig. 36 Double Moon data

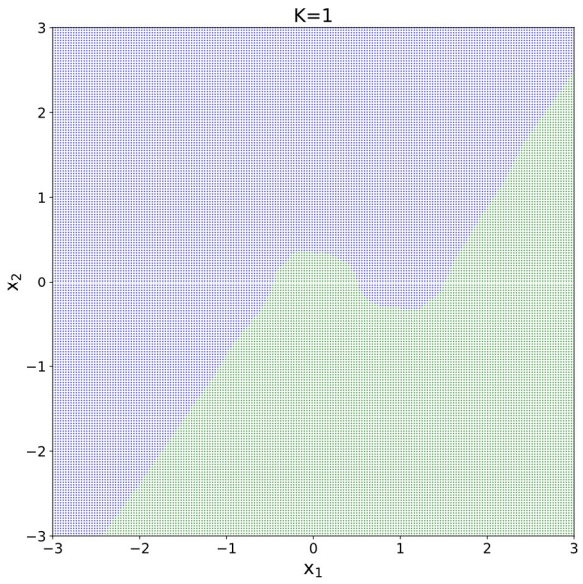

Fig. 37 Result of classifying Double Moon data with *K*=1

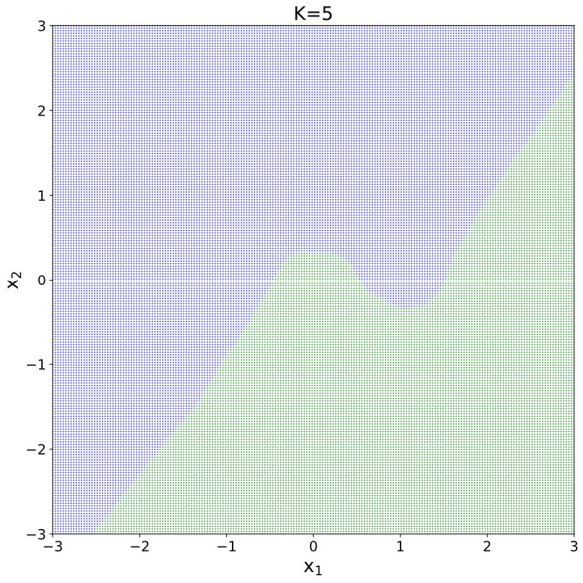

Fig. 38 Result of classifying Double Moon data with *K*=5

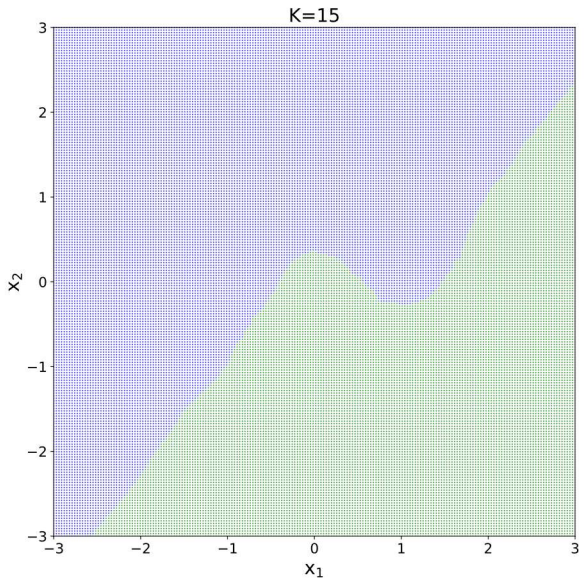

Fig. 39 Result of classifying Double Moon data with K=15

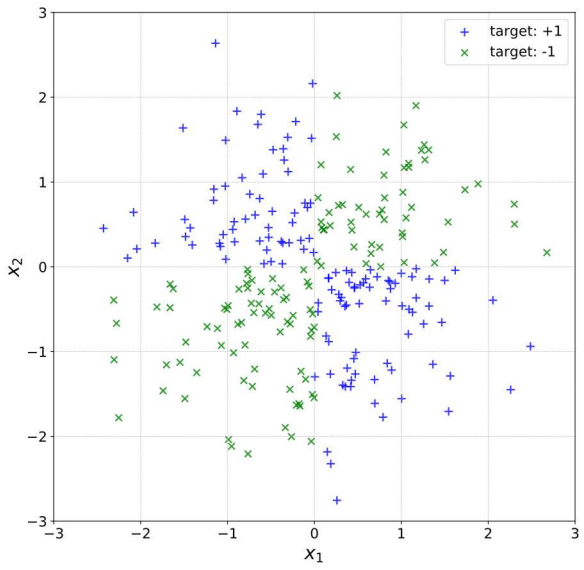

Fig. 40 Gaussian XOR data

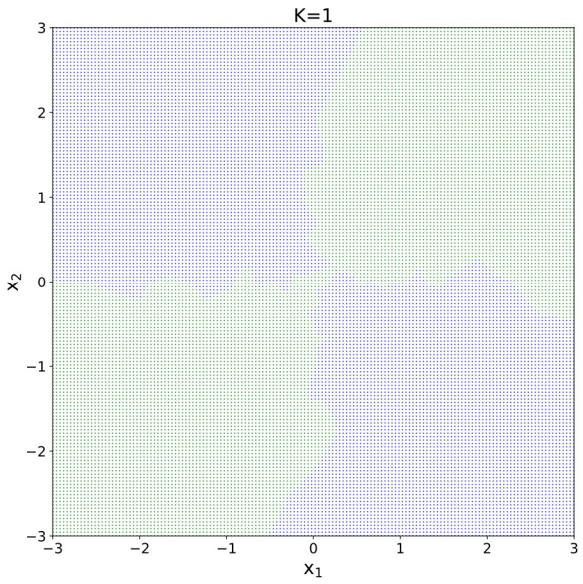

Fig. 41 Result of classifying Gaussian XOR data with K=1

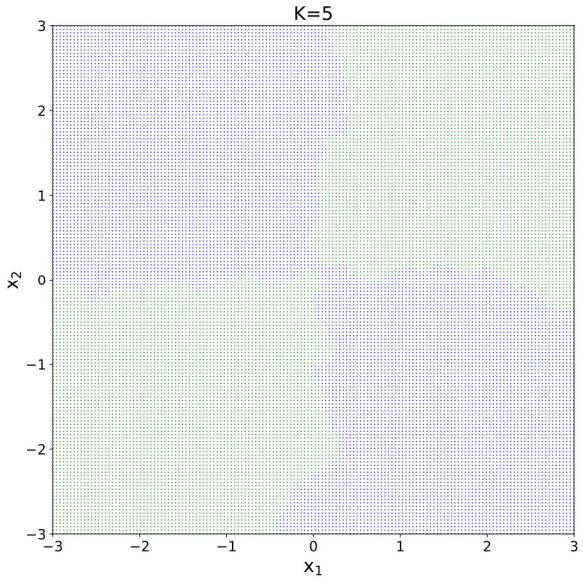

Fig. 42 Result of classifying Gaussian XOR data with K=5

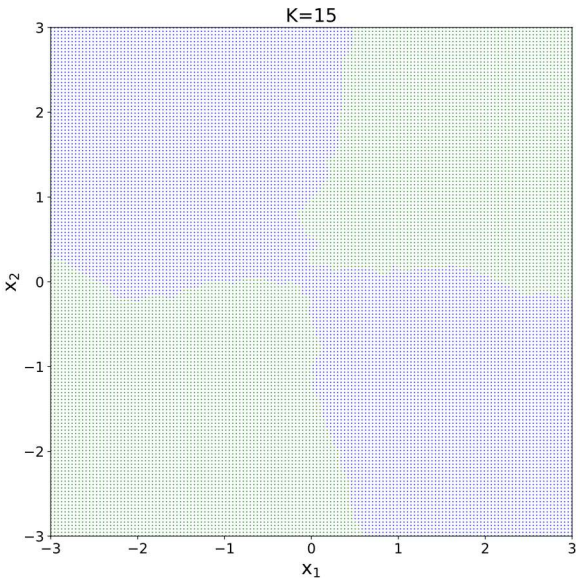

Fig. 43 Result of classifying Gaussian XOR data with *K*=15