IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: Nov 3, 2021
Assignment 2 (MKN Section) — Due Date : Nov 12, 23:59

Roll No: 17011890 — Name: Thomas Bayes
Roll No: 17011892 — Name: Ronald Fisher
Team name:
References (if any):

- This assignment has to be completed in teams of two. Collaborations outside the team are strictly prohibited.
- Use LaTeX to write-up your solutions (in the solution blocks of the source LaTeX file of this assignment), and submit the resulting single pdf file at GradeScope by the due date. (Note: **No late submissions** will be allowed, other than one-day late submission with 10% penalty or four-day late submission with 30% penalty! Within GradeScope, indicate the page number where your solution to each question starts, else we won't be able to grade it!)
- For the programming questions, please submit your code directly in moodle (carefully following the file-name/folder/README conventions given in the questions/moodle), but provide your results/answers/Colab-link in the pdf file you upload to GradeScope. We will run plagiarism checks on codes, and any detected plagiarism in writing/code will be strictly penalized. **Please ensure that the link to your Colab notebook/code is private and give access to all TAs.**
- If you have referred a book or any other online material for obtaining a solution, please cite the source. Again don't copy the source *as is* - you may use the source to understand the solution, but write-up the solution in your own words.
- Points will be awarded based on how clear, concise and rigorous your solutions are, and how correct your code is. Overall points for this assignment would be **min(your score including bonus points scored, 60)**.
- Check the Moodle discussion/announcement forums regularly for updates regarding the assignment. Please start early and clear all doubts ASAP. Post your doubt only on Moodle Discussion Forum so that everyone is on the same page. Please note that the TAs can **only** clarify doubts regarding problem statements (they won't discuss any prospective solution or verify your solution or give hints).

1. (15 points) [SVM]

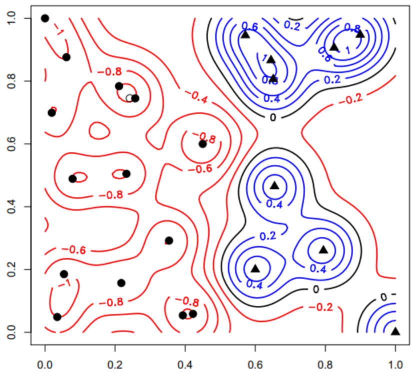
   (a) (3 points) A Gaussian or Radial Basis Function (RBF) kernel with inverse width $k > 0$ is

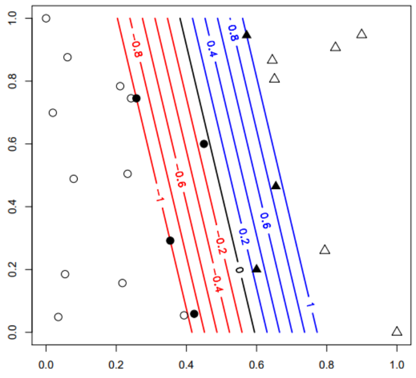
   $$K(u, v) = e^{-k||u-v||^2}.$$

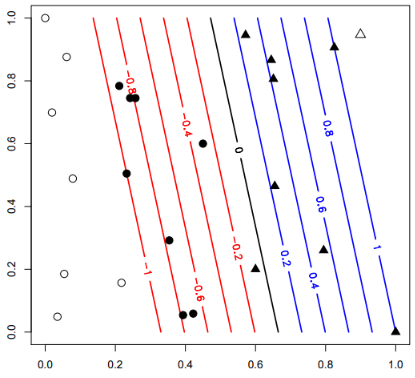
   Below figures show decision boundaries and margins for SVMs learned on the exact same dataset. Parameters used for the different runs are as follows:
   i) Linear Kernel with C = 1
   ii) Linear Kernel with C = 10
   iii) Linear Kernel with C = 0.1

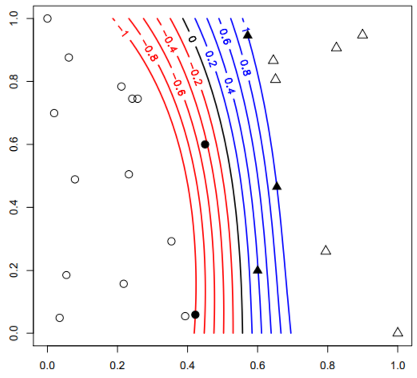
iv) RBF Kernel with $k = 1, C = 3$
v) RBF Kernel with $k = 0.1, C = 15$
vi) RBF Kernel with $k = 10, C = 1$

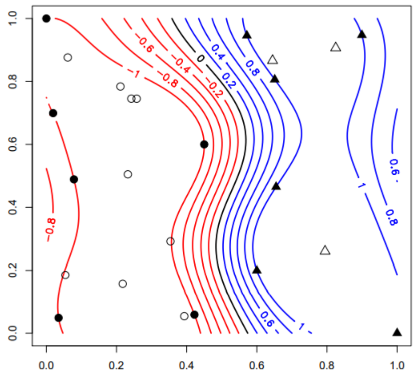
Find out which figure plot would have resulted after each run mentioned above. Justify your answer.

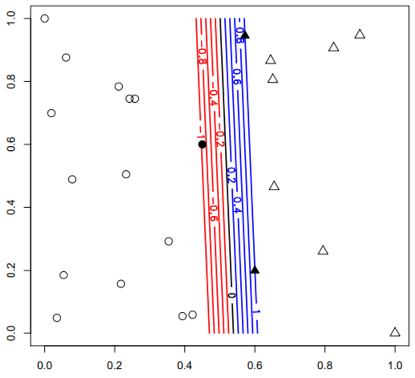

(a)


(b)


(c)


(d)


(e)


(f)

Circle and Triangles denotes class 1 and class 2 respectively, solid points are support vectors.

Solution:

(b) (12 points) Consider $(x_1, y_1), ..., (x_n, y_n)$ as a training data where $x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$. Epsilon-sensitive loss function for regression is given below:

$$L_\epsilon(x, y, f) = |y - f(x)|_\epsilon = max(0, |y - f(x)| - \epsilon).$$

Here x is the input, y is the output, and f is the function used for predicting the label. The cost function of Support Vector Regression or SVR is:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} L_{\epsilon}(x_i, y_i, f)$$

where $f(x) = w^T x$, and $C, \epsilon > 0$ are parameters. Now answer these questions on SVR, with expressions simplified as much as possible.

i. (2 points) What happens when $C \to \infty$? Write the primal form of SVR for this case. (Hint: You may use two constraints per data point to capture the modulus operation.)

Solution:

ii. (2 points) The rest of the questions are for any general value of C. Extend the above primal form for SVR to handle any general C using appropriate slack variables.
(Hint: Try two slack variables per data point in SVR, instead of the one in SVM.)

Solution:

iii. (2 points) Provide the Lagrangian function for the above primal. Can this function take infinite values, and if so under what conditions?

Solution:

iv. (2 points) Derive the Dual function from the above Lagrangian. Can this function take infinite values, and if so under what conditions?

Solution:

v. (2 points) Give the dual form of SVR, and comment on whether quadratic optimization solvers can be used to solve the dual problem?

Solution:

vi. (2 points) Write the KKT conditions that can be used to link the primal and dual solutions.

Solution:

2. (15 points) [ANN]

(a) (3 points) Consider the Artificial Neural Network (ANN) in figure 1 involving a single hidden neuron for solving the XOR problem. Show that the network solves the XOR problem by constructing decision regions, and a truth table for the network.

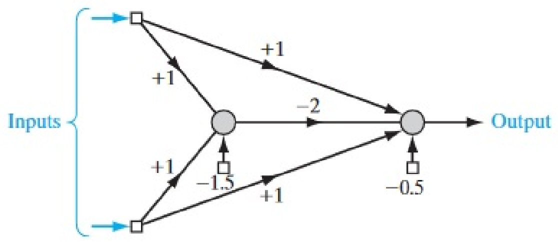


Figure 1: ANN for XOR problem

Solution:

(b) (3 points) Show, for a feed-forward neural network (FFN) with **tanh** hidden unit activation functions and a sum-of-squares error function, that the origin in weight space is a stationary point of the error function.

Solution:

(c) (2 points) Sigmoid is one of the popular activation functions used in ANNs. Let's understand the drawbacks of using sigmoid as an activation function.

- **Sigmoid function causes gradients to vanish.** When is this statement true? Justify your answer clearly.
- **If a Sigmoid function is not zero-centered,** then what could be the possible consequence of this – can it prohibit efficient training of the ANN?

Solution:

(d) (2 points) Considering a simple ANN with two input neurons, 3 neurons in a single hidden layer and 1 output neuron. Assume that sigmoid is used as an activation function.

- Mathematically show why initializing all weights with the same value is a bad idea?
- What could be better strategies for weight initialisation for the above ANN architecture and why?

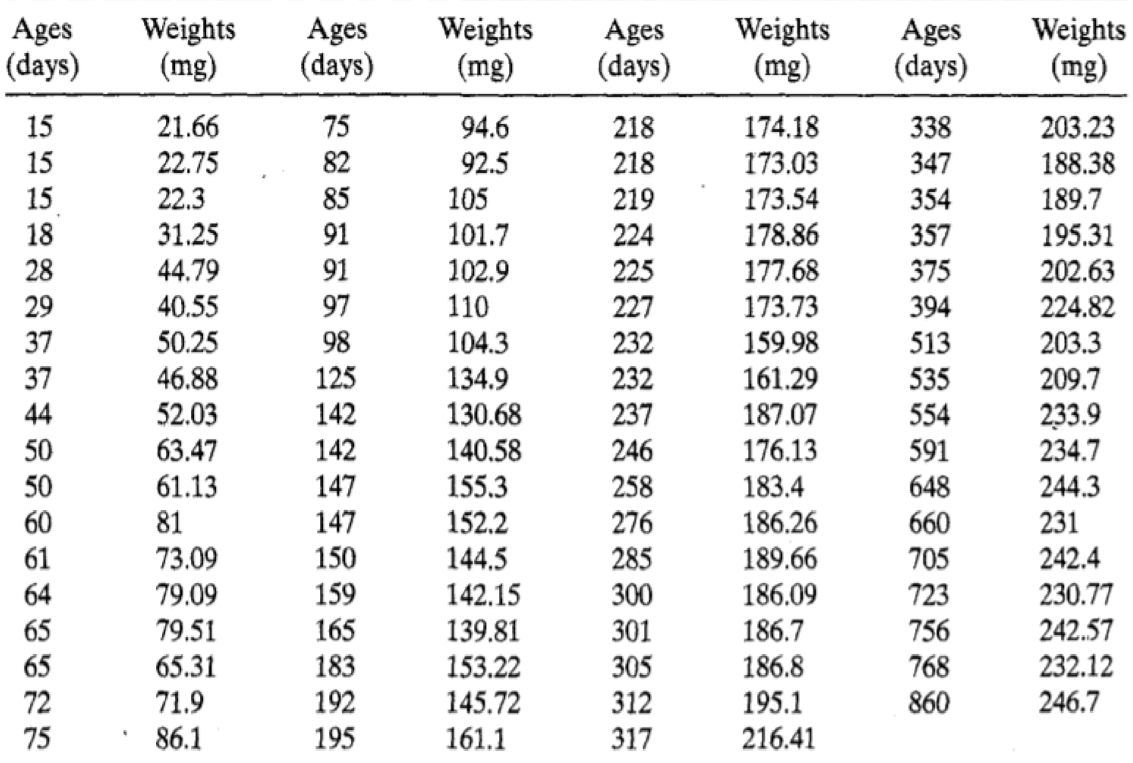
Solution:

(e) (5 points) In the planet Pandora, there is a creature called "Direhorse". The following table shows the **weights of eye lenses of direhorses as a function of age**. No simple analytical function can exactly interpolate these data because we do not have a single-valued function. Instead, we have a nonlinear least squares model of this data set, using a negative exponential

, as described by

$$y = 233.846 * (1 - \exp(-0.006042x)) + \epsilon$$

where ϵ is an error term.
You are appointed as the chief data scientist on the planet and the department of astrobiology has requested your assistance. Your task is briefly described below :
Using the back-propagation algorithm, design a feed-forward neural network (FFN, aka multilayer perceptron) that provides a nonlinear least-squares approximation to this dataset. Compare your results against the least-squares model given above. Briefly comment on when your designed FFN led to underfitting or overfitting, and how you fixed these issues.



Ages (days)	Weights (mg)	Ages (days)	Weights (mg)	Ages (days)	Weights (mg)	Ages (days)	Weights (mg)
15	21.66	75	94.6	218	174.18	338	203.23
15	22.75	82	92.5	218	173.03	347	188.38
15	22.3	85	105	219	173.54	354	189.7
18	31.25	91	101.7	224	178.86	357	195.31
28	44.79	91	102.9	225	177.68	375	202.63
29	40.55	97	110	227	173.73	394	224.82
37	50.25	98	104.3	232	159.98	513	203.3
37	46.88	125	134.9	232	161.29	535	209.7
44	52.03	142	130.68	237	187.07	554	233.9
50	63.47	142	140.58	246	176.13	591	234.7
50	61.13	147	155.3	258	183.4	648	244.3
60	81	147	152.2	276	186.26	660	231
61	73.09	150	144.5	285	189.66	705	242.4
64	79.09	159	142.15	300	186.09	723	230.77
65	79.51	165	139.81	301	186.7	756	242.57
65	65.31	183	153.22	305	186.8	768	232.12
72	71.9	192	145.72	312	195.1	860	246.7
75	86.1	195	161.1	317	216.41		

Paste the training loss plot and report your results in the submissions.
Note: You can use any python library. **Share the link to your Colab notebook with appropriate output already displayed and make sure the notebook is private, with access given to TAs.**

Solution:

(f) (5 points) [OPTIONAL BONUS] ANN python libraries are banned in Pandora during the time of this project, and you are forced instead to use only the numpy library and any plotting libraries for this task. Code from scratch the backpropagation algorithm for a FFN with a single hidden layer but with variable number of hidden neurons, and report how the network performance is affected by varying the size of the hidden layer for the Direhorse dataset. Share the link to your Colab notebook with appropriate output already displayed and make sure the notebook is private, with access granted to TAs.

3. (15 points) [BOOST IS THE SECRET OF...]

   (a) (6 points) [ADABOOST PEN/PAPER] Say we have the following toy dataset in the form: x_1 : $(0,-1)$ $\quad \mathtt{label} : (-)$,$x_2 : (1,0)$ $\quad \mathtt{label} : (+)$,$x_1 : (-1,0)$ $\quad \mathtt{label} : (+)$,$x_1 : (0,1)$ $\quad \mathtt{label} : (-)$.

      i. (2 points) Using decision stumps as weak classifiers show how Adaboost works. Compute the parameters at each timestep. Compute upto $T = 4$.

      Solution:

      ii. (2 points) Draw the classifier at each timestep.

      Solution:

      iii. (2 points) What is the training error? Explain in short why Adaptive Boost outperforms a decision stump in this dataset.

      Solution:

   (b) (9 points) [ADABOOST CODE] In this question, you will code the AdaBoost algorithm from scratch. Follow the instructions in this Jupyter Notebook for this question. Find the dataset for the question here. (**NOTE: Test data should not be used for training.**)

      i. (2 points) Plot the training data.

      Solution:

      ii. (2 points) For training with k=5: (1) Plot the learnt decision surface. (2) Write down the test accuracy.

      Solution:

      iii. (2 points) For training with k=100: (1) Plot the learnt decision surface. (2) Write down the test accuracy.

      Solution:

      iv. (3 points) Paste the link to your Colab Notebook of your code. Make sure that your notebook is private and give access to all the TAs. Your notebook must contain all the codes that you used to generate the above results.

Solution:

(c) (5 points) [OPTIONAL BONUS] In the AdaBoost algorithm, you have n points, (x_i, r_i) where r_i is the class label of x_i , $i = 1, \cdots, n$, let $h_t(x)$ be the weak classifier obtained at step t, and let α_t be its weight. Recall that the final classifier is

$$H(x) = sign(f(x)) \quad f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Show that the training error of the final classifier can be bounded from above by an exponential loss function :

$$\frac{1}{m}\sum_{i=1}^{m} I(H(x_i) \neq r_i) \leqslant \frac{1}{m}\sum_{i=1}^{i=m} e^{-f(x_i)r_i}$$

where I is the indicator function.

4. (15 points) [SINGULARLY PCA!] Consider a dataset of N points with each datapoint being a D-dimensional vector in $\mathbb{R}^D$. Let's assume that:

- we are in a high-dimensional setting where D >> N (e.g., D in millions, N in hundreds).
- the $N \times D$ matrix X corresponding to this dataset is already mean-centered (so that each column's mean is zero, and the covariance matrix seen in class becomes $S = \frac{1}{N}X^TX$).
- the rows (datapoints) of X are linearly independent.

Under the above assumptions, please attempt the following questions.

(a) (6 points) Whereas X is rectangular in general, XX^T and X^TX are square. Show that these two square matrices have the same set of non-zero eigenvalues. Further, argue briefy why these equal eigenvalues are all positive and N in number, and derive the multiplicity of the zero eigenvalue for both these matrices.
(Note: The square root of these equal positive eigenvalues $\{\lambda_i := \sigma_i^2\}_{i=1,\dots,N}$ are called the singular values $\{\sigma_i\}_{i=1,\dots,N}$ of X.)

Solution:

(b) (5 points) We can choose the set of eigenvectors $\{u_i\}_{i==1,\dots,N}$ of XX^T to be an orthonormal set and similarly we can choose an orthonormal set of eigenvectors $\{v_j\}_{j=1,\dots,D}$ for X^TX. Briefly argue why this orthonormal choice of eigenvectors is possible. Can you choose $\{v_i\}$ such that each v_i can be computed easily from u_i and X alone (i.e., without having to do an eigenvalue decomposition of the large matrix X^TX; assume $i = 1, \dots, N$ so that $\lambda_i > 0$ and $\sigma_i > 0$)?
(Note: $\{u_i\}, \{v_i\}$ are respectively called the left,right singular vectors of X, and computing them along with the corresponding singular values is called the Singular Value Decomposition or SVD of X.)

Solution:

(c) (4 points) Applying PCA on the matrix X would be computationally difficult as it would involve finding the eigenvectors of $S = \frac{1}{N}X^TX$, which would take $O(D^3)$ time. Using answer to the last question above, can you reduce this time complexity to $O(N^3)$? Please provide the exact steps involved, including the exact formula for computing the normalized (unit-length) eigenvectors of S.

Solution:

(d) (5 points) [OPTIONAL BONUS] Exercise 12.2 from Bishop's book helps prove why minimum value of the PCA squared error J, subject to the orthonormality constraints of the set of principal axes/directions $\{u_i\}$ that we seek, is obtained when the $\{u_i\}$ are eigenvectors of the data covariance matrix S. That exercise introduces a modified squared error $\widetilde{J}$, which involves a matrix H of Langrange multipliers, one for each constraint, as follows:

$$\widetilde{J} = \mathrm{Tr}\left\{\widehat{U}^T S \widehat{U}\right\} + \mathrm{Tr}\left\{H(I - \widehat{U}^T\widehat{U})\right\}$$

where $\widehat{U}$ is a matrix of dimension $D \times (D - M)$ whose columns are given by u_i. Now, any solution to minimizing $\widetilde{J}$ should satisfy $S\widehat{U} = \widehat{U}H$, and one <u>specific solution</u> is that the columns of $\widehat{U}$ are the eigenvectors of S, in which case H is a diagonal matrix. Show that any general solution to $S\widehat{U} = \widehat{U}H$ also gives the same value for $\widetilde{J}$ as the above specific solution.
(Hint: Show that H can be assumed to be a symmetric matrix, and then use the eigenvector expansion i.e., diagonalization of H.)