ECE M146 Homework 4 Solution
Introduction to Machine Learning
Instructor: Lara Dolecek
TA: Zehui (Alex) Chen, Ruiyi (John) Wu

Please upload your homework to Gradescope by April 30, 11:59 pm.
Please submit a single PDF directly on Gradescope
You may type your homework or scan your handwritten version. Make sure all the work is discernible.

1. In this section, you will use the k-NN classifier to predict whether a person survives or not on the Titanic. You will be using the same data set provided in HW3.

   Unlike what we did in the last homework, we are going to build the k-NN classifier from the first principle. You may **not** use **fitcknn** (**sklearn.neighbors.KNeighborsClassifier** for python) in this problem as you will get incorrect answer by using those built-in functions.

   The k-NN classifier classifies a data point with feature x_{test} based on a training set by performing the following procedures:

   - Compute the distance from x_{test} to the feature of all training points. We will use the Euclidean distance in this problem.
   - Find the k nearest neighbors of this point.
   - Classify this points as the majority class of its k nearest neighbors.

   We use the following two rules to handle ties:

   (a) Let d_k be the distance of the k-th nearest neighbor of x_{test}. If there are multiple training points that have distance d_k from x_{test}. Choose those points with the smallest index to be included in the k nearest neighbors.

   (b) For even k, among all k nearest neighbors of a data point, if the number of points from class 0 is the same as the number of points from class 0, classify this data point as y_{tie} deterministically.

   Answer the following questions:

   (a) With $y_{tie} = 1$, implement the k-NN algorithm. Find and plot the training and testing error for $k = 1, 2, \cdots, 15$.

   (b) With $y_{tie} = 0$, implement the k-NN algorithm. Find and plot the training and testing error for $k = 1, 2, \cdots, 15$.

   (c) Comment on the performance of the k-NN classifier in (a) and (b). How does larger k affect the training and testing error? How does even or odd k affect the performance of the k-NN classifier in (a) and (b), respectively? Are they contradictory to each other? Explain why. Hint: you may use your result from HW3 Q6 (a) to get some intuition.

Solution:

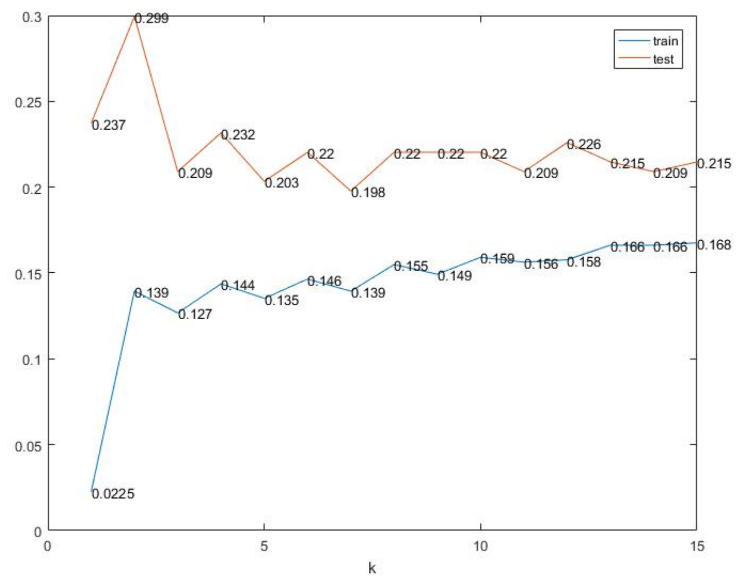


Figure 1: $y_{tie} = 1$

(a)

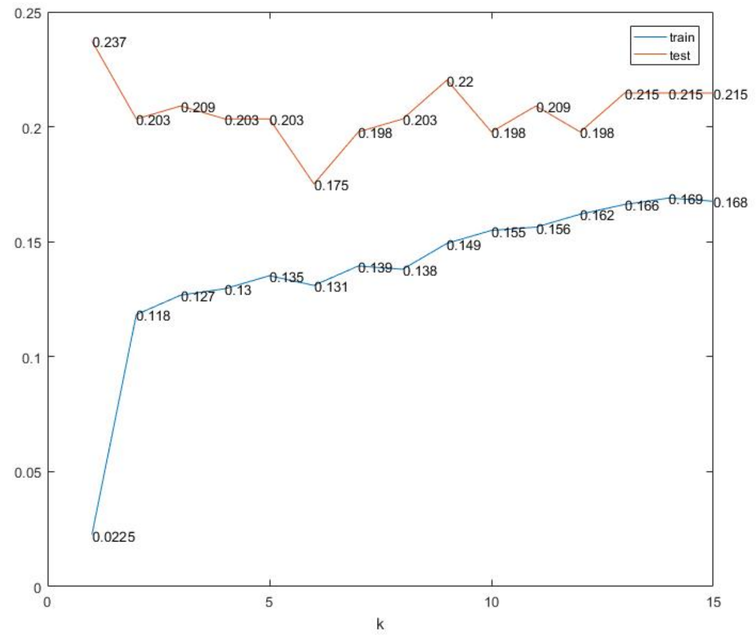


Figure 2: $y_{tie} = 0$

(b)

(c) With larger k, the training error gets larger and the testing error gets smaller. This shows that we are over-fitting with small k. For (a), both the training and testing errors increase with even k. For (b), both the training and testing errors decrease with even k. They are not contradictory to each other. This behavior is a result of tie breaking rule (b) as we deterministically choose the label when there is a tie for even k. Based on the answer of HW3 Q6 (a), the majority class of the training data is 0. As a result, predicting 0 when tie give us better result than predicting 0 when tie.

2. In this section, you will consider a k-nearest neighbor classifier using Euclidean distance metric on a binary classification task. We assign the class of the test point to be the class of the majority of the k nearest neighbors.

   Consider the following two datasets:

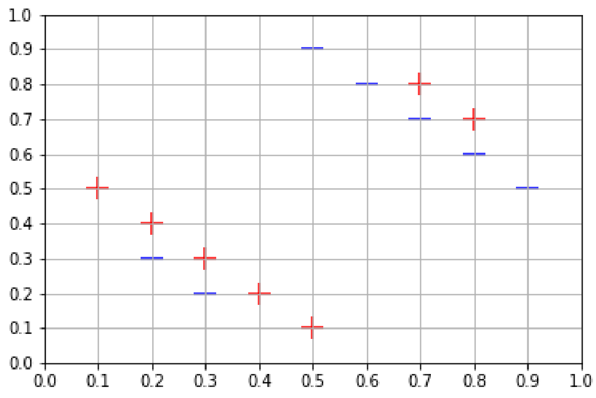


Table 1: KNN Example 1

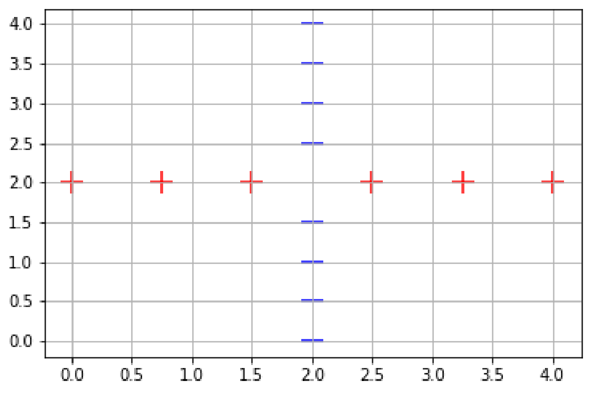


Table 2: KNN Example 2

   (a) For $k \in \{1, 3, 5, 7\}$, what values of k minimize leave-one-out cross-validation error for each dataset? What is the resulting validation error?

   Solution:

   For the first example, the best k are 5 and 7 with a validation error of $\frac{4}{14}$. If $k = 1$, with leave-one-out cross-validation, 10 points (despite the four points at $(0.1, 0.5), (0.5, 0.1), (0.9, 0.5) \text{and} (0.5, 0.9)$) are classified erroneously, resulting in a validation error of $\frac{10}{14}$. If $k = 3$, in addition to the 4 points that are seemingly out of place, the two points at $(0.3, 0.3)$ and $(0.7, 0.7)$ are also classified erroneously, resulting in a validation error of $\frac{6}{14}$.

For the second example, the best k are 1 with a validation error of $\frac{2}{14}$. The two misclassified points are $(1.5, 2)$ and $(2.5, 2)$. For $k = 3, 5$ and 7, there are at least two additional points that are classified erroneously. For $k = 3$, these two points are $(2, 1.5)$ and $(2, 2.5)$. For $k = 5$, these two points are $(1.75, 2)$ and $(1.75, 2)$. For $k = 7$, these two points are $(0, 2)$ and $(4, 2)$.

(b) In general, what are the drawbacks of using too big a k? What about too small a k? To see this, calculate the leave-one-out cross-validation error for the dataset in Figure 1 using $k = 1$ and $k = 13$.

Solution:

In general, too big causes underfitting since the largest class would dominate and too small causes overfitting. For Example 1, the validation error for $k = 1$ is $\frac{10}{14}$. The validation error for $k = 13$ is $\frac{14}{14}$ which is really bad.
Note although this statement holds true in general, for certain cases, e.g, our synthetic example 2, small k or large k may be preferred. Moreover, the nearest neighbor classifier, i.e., $k = 1$, is one commonly used calssifeir.

3. In this problem, we will derive the least square solution for multi-class classification. Consider a general classification problem with K classes, with a 1-of-K binary encoding scheme (defined latter) for the target vector $t, t \in \mathbb{R}^K$. Suppose we are given a training data set $\{x_n, t_n\}, n = 1, \cdots, n$ where $x_n \in \mathbb{R}^D$. For the 1-of-K binary encoding scheme, t_n has the k-th element being 1 and all other elements being 0 if the n-th data is in class k. We can use the following linear model to describe each class:

$$y_k(x) = w_k^T x + w_{k0},$$

where $k = 1, \cdots, K$. We can conveniently group these together using vector notation so that

$$y(x) = \tilde{W}^T \tilde{x},$$

where $\tilde{W}$ is a matrix whose k-th column comprises the $D+1$-dimensional vector $\tilde{w} = [w_{k0}, w_k^T]^T$ and $\tilde{x}$ is the corresponding augmented input vector $[1, x^T]^T$. For each new input with feature x, we assign it to the class for which the output $y_k = \tilde{w}_k^T \tilde{x}$ is largest. Define a matrix T whose n-th row is the vector t_n^T and together a matrix $\tilde{X}$ whose n-th row is $\tilde{x}_n^T$, the sum-of-squares error function can be written as

$$J(\tilde{W}) = \frac{1}{2} Tr \left\{ (\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T) \right\}.$$

Find the closed form solution of $\tilde{W}$ that minimizes the objective function $J(\tilde{W})$. Hint: You many use the following two matrix derivative about trace, $\frac{\partial}{\partial Z} Tr(AZ) = A^T$ and $\frac{\partial}{\partial Z} Tr(Z^T A Z) = (A^T + A)Z$.
Solution: We first expand the term in the trace operator and get

$$J(\tilde{W}) = \frac{1}{2} Tr \left\{ \tilde{W}^T \tilde{X}^T \tilde{X} \tilde{W} - 2T^T \tilde{X} \tilde{W} + T^T T \right\}.$$

Setting the derivative with respect to $\tilde{W}$ to 0, we get

$$0 = \tilde{X}^T \tilde{X} \tilde{W} - \tilde{X}^T T.$$

This shows that

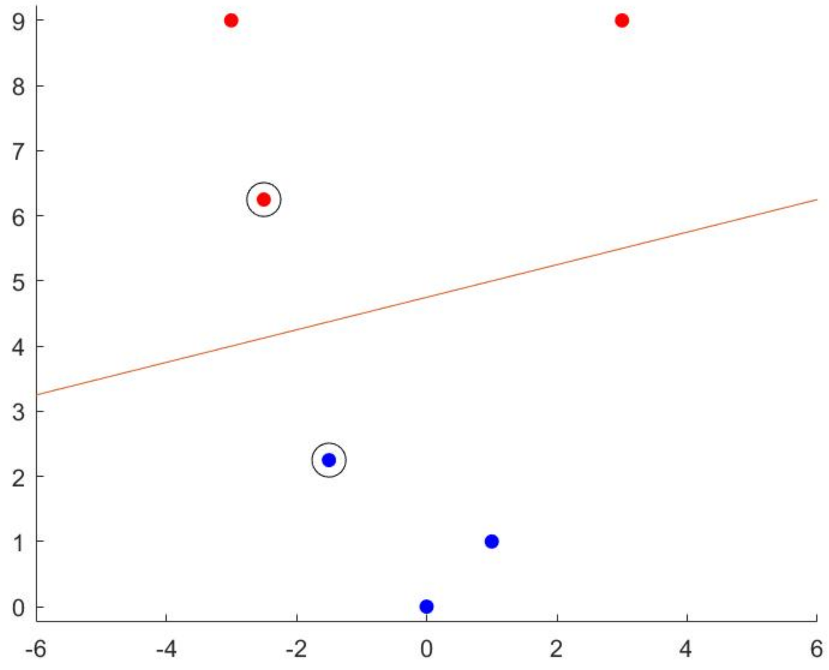
$$\tilde{W} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T T.$$

Note: This is the same solution for multi-dimensional least square if we treat t_n as the targeted value for x_n.

4. You are given the following data set which is comprised of $x^{(i)} \in \mathbb{R}^2$ and $y^{(i)} \in \{-1, 1\}$.

i	$x_1^{(i)}$	$x_2^{(i)}$	$y^{(i)}$
1	-3	9	1
2	-2.5	6.25	1
3	3	9	1
4	-1.5	2.25	-1
5	0	0	-1
6	1	1	-1

(a) Plot the data. Is the data linearly separable?
Solution: Yes.



(b) Look at the data and circle the support vectors by inspection. Find and plot the maximum margin separating hyperplane.

Solution: The two points that are closest are $(-1.5, 2.25)$ with negative label and $(-2.5, 6.25)$ with positive label. They are the support vectors and the maximum margin separating hyperplane is given by $-x_1+4x_2-19=0$ by finding a line that has normal vector $[-1, 4]^T$ and also passes through the mid-point of the support vectors, i.e., $(-2, 4.25)$. The line is drawn on the above figure.

(c) Find the α_i, w and b in

$$h(x) = \text{sign}\left(\sum_{n\in\mathcal{S}} \alpha_n y^{(n)} x^T x^{(n)} + b\right) = \text{sign}\left(w^T x + b\right),$$

where $\mathcal{S}$ is the index set of all support vectors. Do this by solving the dual form of the quadratic program. How is w and b related to your solution in part (b)?

Solution: Since we only have two support vectors, only the Lagrange multiplier corresponding to the support vectors are non-zero. Let α_1 denote the Lagrange multiplier for $x_1 = -1.5$ and similarly α_5 for $x_5 = -2.5$. From the condition $\sum_{i=1}^{6} \alpha_i y_i = 0$, we get $\alpha_1 = \alpha_5 = \alpha_0$. Write down the objective of the dual problem of SVM

$$
\begin{aligned}
W(\boldsymbol{\alpha}) &= \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{4} y_i y_j a_i a_j x^{(i)T} x^{(j)} \\
&= 2\alpha_0 - \frac{1}{2}\alpha_0^2 x^{(1)T} x^{(1)} + \alpha_0^2 x^{(1)T} x^{(5)} - \frac{1}{2}\alpha_0^2 x^{(5)T} x^{(5)} \\
&= 2\alpha_0 - 8.5\alpha_0^2.
\end{aligned}
$$

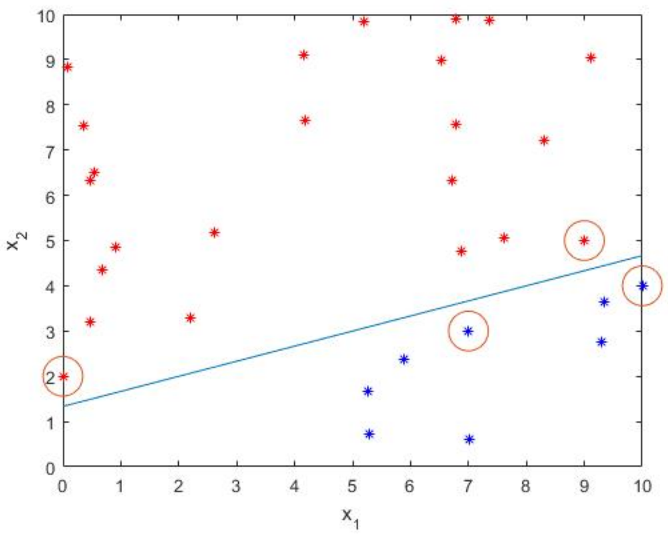
Maximizing $W(\alpha)$ over α_0, we get $\alpha_1 = \alpha_5 = \alpha_0 = \frac{2}{17}$. Using $w = \sum_{m \in \mathcal{S}} \alpha_m y_m \phi(x_m)$, we get $w = [-\frac{2}{17}, \frac{8}{17}]^T$. To find b, recall that

$$
y_i \left(w^T \phi(x_i) + b \right) = 1
$$

for any support vectors $\phi(x_i)$. Use any support vector, we can get $b = -\frac{38}{17}$. The w and b we find by solving the dual problem is a scaled version of $[w_1, w_2]^T$ and w_0 in part (d). These solutions therefore give the same separating hyperplane.

5. In this exercise, we will use MATLAB to solve both the primal and the dual problem of SVM. In *Data.csv*, the first two columns contain feature vectors $x^{(i)} \in \mathbb{R}^2$ and the last column contains the label $y^{(i)} \in \{-1, 1\}$. We will use CVX as the optimization solver in this problem. For help with CVX, refer to the CVX Users' Guide. Attach your code for submission. For Python user, feel free to use the following libraries: math, csv, numpy, matplotlib and cvxpy.

   (a) **Visulization** Use different color to plot data with different labels in the 2-D feature space. Is the data linearly separable?
   Solution: Yes, the data is linearly separable.



   (b) **The Primal Problem** Use CVX to solve the primal problem of this form:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \cdots, m \end{aligned}$$

   Report w and b. Plot the hyperplane defined by w and b.
   Solution: We get $w = [-0.5, 1.5]^T$ and $b = -2$. The hyperplane is show in the figure above.

   (c) **The Dual Problem** Use CVX to solve the dual problem of this form:

$$\begin{aligned} \max_{a} \quad & W(a) = \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} a_i a_j \langle x^{(i)}, x^{(j)} \rangle \\ s.t. \quad & 0 \leq a_i, i = 1, \cdots, m \\ & \sum_{i=1}^{m} a_i y^{(i)} = 0. \end{aligned}$$

   Use the resulting a to identify the support vectors on the plot. Report you non-zero $a_i's$. How many support vectors do you have? Circle those support vectors.

Note: The latter part of $W(a)$ is in quadratic form, i.e., $a^T Pa$. To use CVX, first find P and then use *quad_form(a,P)*. For Python user, you will need to add a small number to the diagonal of P matrix to make cvxpy work. i.e. Run the following code before using cvxpy: "P += 1e-13 * numpy.eye(31)", where 31 is the total number of data. Also, assume it is 0 if a number is less than 1e-9.
Solution: There are 4 support vectors.
The corresponding $a = [0.8533, 0.3967, 0.2011, 1.0489]^T$ ($a = [0.9894, 0.2606, 0.2182, 1.0318]^T$ for python, order does not matter). The support vectors are circled in the above figure. In order to use CVX to solve this problem, first find a matrix P where $P_{ij} = y^{(i)}y^{(j)}\langle x^{(i)}, x^{(j)}\rangle$. Then let CVX to maximize $\sum_{i=1}^{m} a_i - \frac{1}{2}a^T Pa$.