ECE M146 HW4 Melody Chen #705120273

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.KNeighborsClassifi** 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 indexes to be included in the k nearest neighbors. For example, let $k = 3$, if there is x_9 that is distance 1 away from x_{test}; x_1, x_3 and x_4 that is distance 2 away from x_{test}, then the 3 nearest neighbor of x_{test} are x_1, x_3 and x_9. Note that $d_k = 2$ in this example.

   (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.

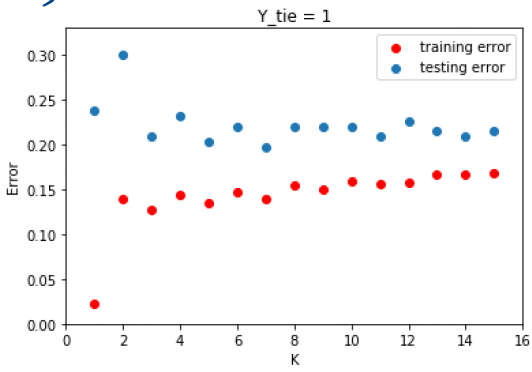
   Answer the following questions:

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

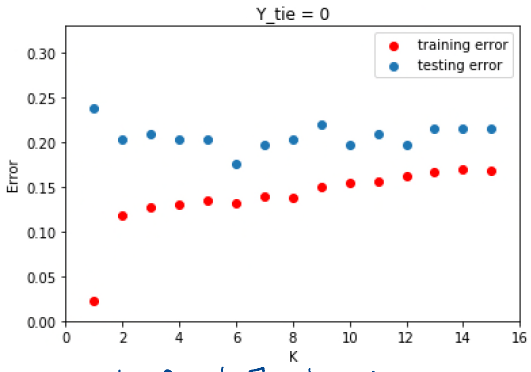
   Shown to the right.

   (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.

a)



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



Larger k doesn't appear to significantly affect training error except for k=1, training error for both value of y_tie is significantly lower than other ks, which makes sense as we're only looking at one neighbor for a training data. Larger k stabilizes and converges the testing error to what we previously calculated for both a) and b).
For a) even k appears to frequently cause surges in both training and testing error, which is likely due to frequent ties. For B), even k does not lead to surges which is likely due to y_tie = 0 being the more accurate guess in a tie.
Two results are not contradictory as surges for even k in a) should correspond to non-surges in B) as the prediction has only two possible outcomes.
By surges, I mean occasional bumps in testing error.

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:

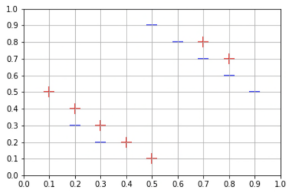


Table 1: KNN Example 1

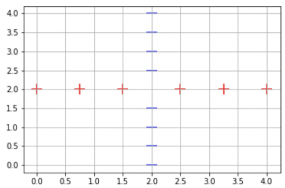


Table 2: KNN Example 2

   (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?

Table 1
k=1, 10 points misclassified. $\frac{10}{14}$ cross validation error.
k=3, 6 points misclassified. $\frac{6}{14}$ cross validation error.
k=5, 4 points misclassified. $\frac{4}{14}$ cross validation error.
k=7, 4 points misclassified. $\frac{4}{14}$ cross validation error.

k=5, k=7 minimizes Loo cross-validation error.

Table 2
k=1, 2 points misclassified. $\frac{2}{14}$ cross validation error.
k=3, 4 points misclassified. $\frac{4}{14}$ cross validation error.
k=5, at least 4 points misclassified.
k=7, at least 4 points misclassified.

k=1 minimizes Loo cross-validation error.

(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 $k = 13$.

k=1, LOO cross validation error: 10/14

k=13, LOO cross validation error: 14/14

The drawbacks of using too big of k is that decision regions become large uninterrupted segments, where the largest class could dominate the results.

The drawbacks of using too small of k is overfitting, as the decision boundary becomes very fine grained.

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 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 $\frac{\partial}{\partial Z} Tr(Z^T A Z) = (A^T + A)Z$.

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

$$= \frac{1}{2}\frac{\partial}{\partial w} Tr\{(\tilde{X}\tilde{w})^T\tilde{X}\tilde{w} - (\tilde{X}\tilde{w})^T T - T^T(\tilde{X}\tilde{w}) + T^T T\}$$

$$= \frac{1}{2}\frac{\partial}{\partial w} Tr\{\tilde{w}^T\tilde{X}^T\tilde{X}\tilde{w} - 2T^T(\tilde{X}\tilde{w}) + T^T T\}$$

$$= \frac{1}{2}\frac{\partial}{\partial w}\left(Tr(\tilde{w}^T\tilde{X}^T\tilde{X}\tilde{w}) - Tr(2T^T(\tilde{X}w)) + Tr(T^T T)\right)$$

$$= \frac{1}{2}\left(2(\tilde{X}^T\tilde{X})\tilde{w} - 2\tilde{X}^T T\right)$$

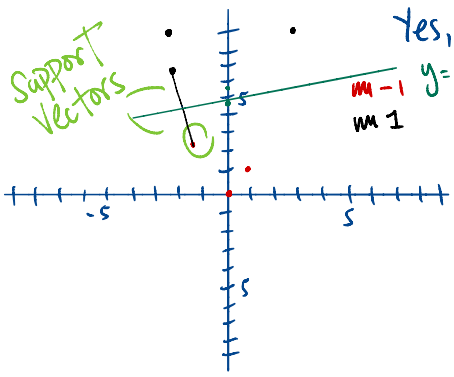
$$= (\tilde{X}^T\tilde{X})\tilde{w} - \tilde{X}^T T = 0$$

$$\boxed{\tilde{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T T}$$

4. You are given the following data set which is comprised of $\boldsymbol{x}^{(i)} \in \mathbb{R}^2$ 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?



Yes, data is linearly separable.
$y = \frac{1}{4}x + 4.75$

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

support vectors points: (-2.5, 6.25), (-1.5, 2.25)

equation of line between two points:

$m = \frac{6.25 - 2.25}{-2.5 + 1.5} = -4$

$y - 6.25 = -4(x + 2.5)$

$y = -4x - 10 + 6.25$

$y = -4x - 3.75$

Perpendicular Slope $= \frac{1}{4}$

midpoint: $(-2.5 + 0.5, 2.25 + 2) = (-2, 4.25)$

separating hyperplane: $y - 4.25 = \frac{1}{4}(x + 2)$

$y = \frac{1}{4}x + \frac{1}{2} + 4.25$

$y = 0.25x + 4.75$

w vector is perpendicular to separating hyper plane. So w vector should have slope of -4, and be pointing in direction of $[-1$

(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)?

We use dual form of quadratic program: $a_n \geq 0$ $\sum_{n=1}^{N} a_n t_n = 0$

$$\sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$

So, $a_2 = a_4$

$$= 2a_2 - \frac{1}{2}\left(a_2^2\|x_2\|^2 + a_2^2\|x_4\|^2 - 2a_2^2 x_2^T x_4\right)$$

$$= 2a_2 - \frac{1}{2}a_2^2\|x_2\|^2 - \frac{1}{2}a_2^2\|x_4\|^2 + a_2^2 x_2^T x_4$$

$$\frac{\partial}{\partial a_2}\left(2a_2 - \frac{1}{2}a_2^2\|x_2\|^2 - \frac{1}{2}a_2^2\|x_4\|^2 + a_2^2 x_2^T x_4\right)$$

$$= 2 - a_2\|x_2\|^2 - a_2\|x_4\|^2 + 2a_2 x_2^T x_4$$

$x_2: (-2.5, 6.25)$ $x_4: (-1.5, 2.25)$

$$2 - a_2\left((-2.5)^2 + 6.25^2\right) - a_2\left((-1.5)^2 + 2.25^2\right) + 2a_2(17.8125) = 0$$

$$= 2 - 45.3125a_2 - 7.3125a_2 + 35.63a_2 = 0$$

$$\boxed{a_4 = a_2 = \frac{2}{17}}$$

We solve for w.

$$w = \sum_{n=1}^{N} a_n t_n x_n$$

$$= \frac{2}{17}(1)\begin{bmatrix}-2.5\\6.25\end{bmatrix} - \frac{2}{17}\begin{bmatrix}-1.5\\2.25\end{bmatrix} = \boxed{\frac{2}{17}\begin{bmatrix}-1\\4\end{bmatrix}}$$

We solve for b.

$$t_n(w^T x_n + b) = 1$$

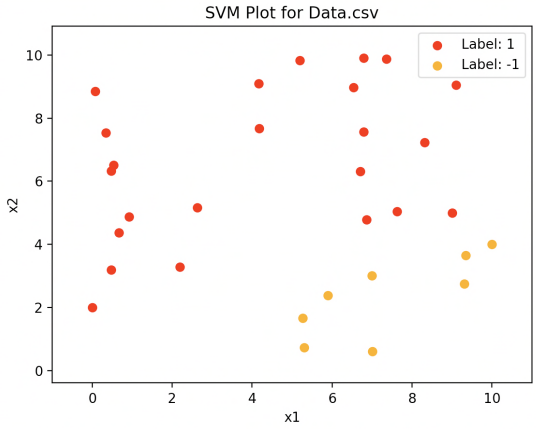
$$1\left(\left[-\frac{2}{17}, \frac{8}{17}\right]\begin{bmatrix}-2.5\\6.25\end{bmatrix} + b\right) = 1 \qquad \boxed{b = -\frac{38}{17}}$$

w is related to my solution in part b as it is perpendicular to the slope of my separating hyperplane, which is expected as two methods should yield similar w.

b is related to part B in that $\frac{-b}{w_2}$ = y intercept = 4.75. This makes sense as we plot hyperplane by $x_1 w_1 + x_2 w_2 + b = 0$.

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?



Yes, the data looks linearly separable.

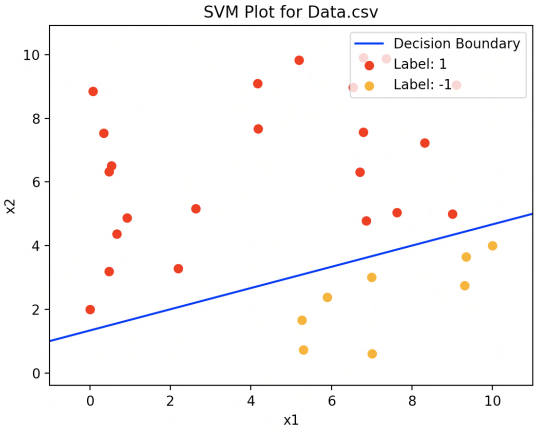
(b) **The Primal Problem** Use CVX to solve the primal problem of this form:

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

Report w and b. Plot the hyperplane defined by w and b.

$w = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}$ $b = -2$

plot →



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

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

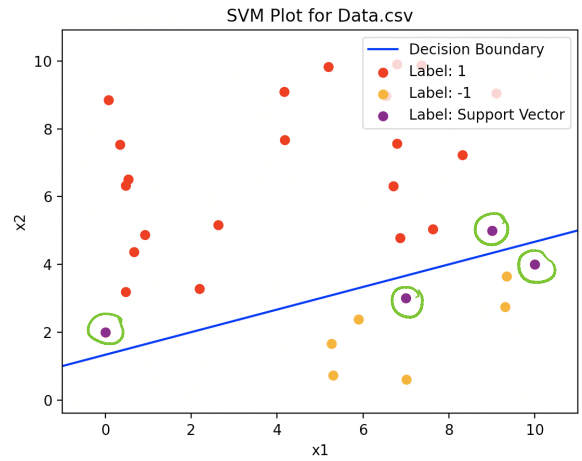
Use the resulting a to identify the support vectors on the plot. Report you non-zero $a_i's$. How many support vectors do you have? Circle those support vectors.

Note: The latter part of $W(a)$ is in quadratic form, i.e., $a^T P a$. 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.

I found 4 support vectors.

$a_{28} = 1.03178$ $a_{29} = 0.21821$ $a_{30} = 0.26059$

$a_{31} = 0.98941$



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 30 01:19:57 2020

@author: melodychen
"""

import numpy as np
import matplotlib.pyplot as plt
import csv
import cvxpy as cp

# global variables
x_train = []
y_train = []
x_1_1 = []
x_1_0 = []
x_2_1 = []
x_2_0 = []


def load_data():
    global x_1_1, x_1_0, x_2_1, x_2_0
    x_1_1 = []
    x_1_0 = []
    x_2_1 = []
    x_2_0 = []
    global x_train, y_train
    x_train = []
    y_train = []
    with open("data.csv") as csvfile:
        readCSV = csv.reader(csvfile, delimiter=',')
        for row in readCSV:
            if int(row[2]) == 1:
                x_1_1.append(float(row[0]))
                x_2_1.append(float(row[1]))
            else:
                x_1_0.append(float(row[0]))
                x_2_0.append(float(row[1]))
```

```
41              x_train.append([float(row[0]), float(row[1
    ])])
42              y_train.append(float(row[2]))
43      # plot all points
44      plt.scatter(x_1_1, x_2_1, label='Label: 1', color=
    'red')
45      plt.scatter(x_1_0, x_2_0, label='Label: -1', color
    ='orange')
46      plt.xlabel('x1')
47      plt.ylabel('x2')
48      plt.title("SVM Plot for Data.csv")
49      plt.axis([min(x_1_0 + x_1_1) - 1, max(x_1_0 +
    x_1_1) + 1, min(x_2_1 + x_2_0) - 1, max(x_2_1 + x_2_0
    ) + 1])
50
51
52  def primal_problem():
53      # load data
54      x = np.zeros(shape=(len(x_train), 2))
55      y = np.zeros(shape=(len(x_train), 1))
56      # load into numpy array
57      for index, row in enumerate(x_train):
58          x[index][0] = row[0]
59          x[index][1] = row[1]
60          y[index] = y_train[index]
61      # variables we're using to minimize
62      w = cp.Variable(2)
63      b = cp.Variable(1)
64      # function we're trying to minimize
65      cost = cp.sum_squares(w)
66      # constraints for minimization
67      constraints = []
68      for index, row in enumerate(x):
69          constraints.append(y[index] * (w.T @ row + b
    ) >= 1)
70      # use cvxpy to do minimization
71      prob = cp.Problem(cp.Minimize(cost), constraints)
72      result = prob.solve()
73      # final values
74      print("w vector: "+str(w.value))
```

```
75      print("b: "+str(b.value))
76      # plot line perpendicular to vector w, decision
    boundary
77      x_line = np.linspace(min(x_1_0 + x_1_1)-1, max(
    x_1_0 + x_1_1)+1, 100)
78      # equation of line, we know that w1*x1 + w2*x2 +
    b = 0, x2 = (-w1*x1)/w2 - b/w2
79      y_line = (-(float(w.value[0])/float(w.value[1]))*
    x_line-(float(b.value)/float(w.value[1])))
80      plt.plot(x_line, y_line, color='blue', label='
    Decision Boundary')
81
82
83  def dual_problem():
84      # load data
85      x = np.zeros(shape=(len(x_train), 2))
86      y = np.zeros(shape=(len(x_train), 1))
87      for index, row in enumerate(x_train):
88          x[index][0] = row[0]
89          x[index][1] = row[1]
90          y[index] = y_train[index]
91      # variable used to maximize dual
92      alpha = cp.Variable(len(x_train))
93      # P matrix that represents part of latter part of
     W(a)
94      p = np.zeros(shape=(len(x_train), len(x_train)))
95      # Fill up P matrix with y_i*y_j*x_i^T*x_j
96      for r in range(len(x)):
97          for c in range(len(x)):
98              p[r][c] = y[r]*y[c]*x[r].transpose().dot(
    x[c])
99      # slight adjustment to P
100     p = p + 1e-13 * np.eye(31)
101     # function we're trying to maximize
102     cost = sum(alpha) - 0.5 * cp.quad_form(alpha, p)
103     # our constraints
104     constraints = []
105     for a in alpha:
106         constraints.append(0 <= a)
107     constraints.append(sum(alpha * y) == 0)
```

```
108     # using cvxpy to solve maximization problem
109     prob = cp.Problem(cp.Maximize(cost), constraints)
110     result = prob.solve()
111     print("Original Alphas: ")
112     print(alpha.value)
113     # we want to make very small values zero
114     alpha_float = []
115     non_zero_alpha = []
116     for index, num in enumerate(alpha.value):
117         if float(num) < 1e-9:
118             alpha_float.append(0)
119         else:
120             alpha_float.append(float(num))
121             non_zero_alpha.append((index, float(num
    )))
122     print("Cleaned up version of Alphas: ")
123     print(alpha_float)
124     print(non_zero_alpha)
125     # want to highlight these points
126     x_1_highlight = []
127     x_2_highlight = []
128     for tup in non_zero_alpha:
129         x_1_highlight.append(x[tup[0]][0])
130         x_2_highlight.append(x[tup[0]][1])
131     # highlights support vector in plot
132     plt.scatter(x_1_highlight, x_2_highlight, color='
    purple', label='Label: Support Vector')
133
134
135 if __name__ == "__main__":
136     load_data()  # part a
137     primal_problem()  # part b
138     dual_problem()  # part c
139     # show legend
140     plt.legend(loc='upper right')
141     # plot the graph
142     plt.show()
143
```