MA559 HW#3

I pledge my honor that I have abided by the Stevens Honor System

Dimitrios Haralampopoulos

2023-03-27

: You shall submit a zip file named Assignment3_LastName_FirstName.zip which contains:

- python files (.ipynb or .py) including all the code, comments, plots, and result analysis. You need to provide detailed comments in English.
- pdf file including (a) solutions for questions 1 and 2, (b) descriptions of observations for questions 3,4,5.

Table 1: Data set for Question 1

data	x_{i1}	x_{i2}	y_i	α_i
$\mathbf{x}_1$	4	2.9	1	0.414
$\mathbf{x}_2$	4	4	1	0
$\mathbf{x}_3$	1	2.5	-1	0
$\mathbf{x}_4$	2.5	1	-1	0.018
$\mathbf{x}_5$	4.9	4.5	1	0
$\mathbf{x}_6$	1.9	1.9	-1	0
$\mathbf{x}_7$	3.5	4	1	0.018
$\mathbf{x}_8$	0.5	1.5	-1	0
$\mathbf{x}_9$	2	2.1	-1	0.414
$\mathbf{x}_{10}$	4.5	2.5	1	0

(20 points) Given 10 points in Table1, along with their classes and their Lagranian multipliers (α_i), answer the following questions:

(a) (7pts) What is the equation of the SVM hyperplane $h(x)$? Draw the hyperplane with the 10 points.

Solution vector $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$
To obtain b: $\alpha_i[y_i(\mathbf{w}^T\mathbf{x} + b) - 1] = 0$ for any of the support vectors
Largest margin hyperplane yields: $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

```
import numpy as np
import matplotlib.pyplot as mp
xi = np.asarray([4, 4, 1, 2.5, 4.9, 1.9, 3.5, 0.5, 2, 4.5])
xk = np.asarray([2.9, 4, 2.5, 1, 4.5, 1.9, 4, 1.5, 2.1, 2.5])
xx = np.asarray([[4, 2.9],[4,4],[1,2.5],
[2.5,1],[4.9,4.5],[1.9,1.9],[3.5,4],
[0.5,1.5],[2,2.1],[4.5,2.5]])
```

```
y = np.asarray([1, 1, -1, -1, 1, -1, 1, -1, -1, 1])
alpha = np.asarray([0.414, 0, 0, 0.018, 0, 0, 0.018, 0, 0.414, 0])
w0 = np.sum(np.multiply(np.multiply(alpha, y), xx.T), axis=1)
print(w0)
```

```
## [0.846  0.3852]
```

```
b = (1 - np.dot(w0.T, xx[0]))
print(b)
```

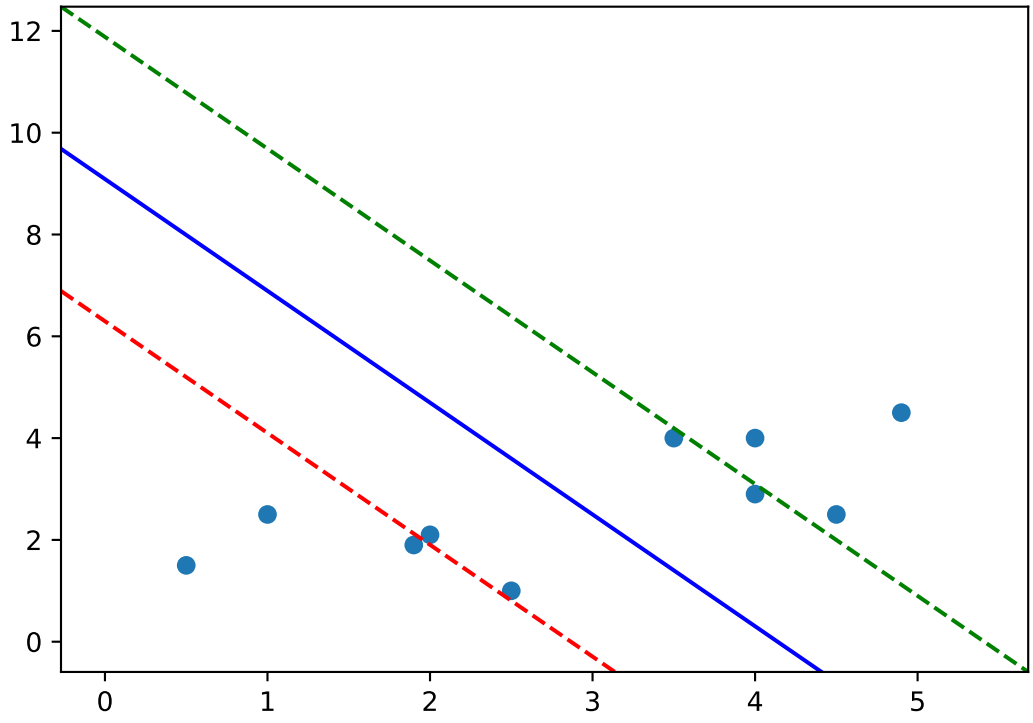
```
## -3.50108
```

```
def hx (w, x, b):
  return  np.dot(w.T, x.T) + b

def chx (w, x, b):
  return 1/np.sqrt(w[0]**2 + w[1]**2) * hx(w, x, b)

def sign (x):
  if x < 0:
    return -1
  elif x > 0:
    return 1
  else:
    return 0

mp.scatter(xi, xk)
p1 = (-b/w0[0], 0)
p2 = (0, -b/w0[1])
up1 = ((-(b - 1/np.sqrt(w0[0]**2 + w0[1]**2))/w0[0]), 0)
up2 = (0, (-(b - 1/np.sqrt(w0[0]**2 + w0[1]**2))/w0[1]))
lp1 = ((-(b + 1/np.sqrt(w0[0]**2 + w0[1]**2))/w0[0]), 0)
lp2 = (0, (-(b + 1/np.sqrt(w0[0]**2 + w0[1]**2))/w0[1]))
mp.axline(p1, p2, color = "b")
mp.axline(up1, up2, color = "g", linestyle = "--")
mp.axline(lp1, lp2, color = "r", linestyle = "--")
mp.show()
```




```
mp.close()
```

(b) (8pts) What is the distance of x_6 from the hyperplane? Write down the process of how the distance is calculated. Is it within the margin of the classifier?

$\mathbf{w}\mathbf{x}_1 + b = 0$, $\mathbf{w}\mathbf{x}_2 + b = 0 \rightarrow \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$
$\mathbf{w}^* = \frac{\mathbf{w}}{||\mathbf{w}||}$
$(\mathbf{w}^*)^T(\mathbf{x} - \mathbf{x}_0) = \frac{\mathbf{w}^T}{||\mathbf{w}||}(\mathbf{x} - \mathbf{x}_0) = \frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{x_0}) = \frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x} + b)$

```
chx(w0, xx[5], b)
```

```
## -1.2498290534100407
```

```
C = 1/np.sqrt(w0[0]**2 + w0[1]**2)
print(f"C margin limit: {C}")
```

```
## C margin limit: 1.075769541582063
```

This is outside the margin.

(c) (5pts) Classify the point $z = (3, 3)^T$ using $h(x)$ from above.

```
sign(hx(w0, np.asarray([3,3]).T, b))
```

```
## 1
```

(3,3) falls within the positive class

(20 points) The SVM loss function with slack variables can be viewed as:

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + \gamma \sum_{i=1}^{N} \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{Hinge loss}}$$

The hinge loss provides a way of dealing with datasets that are not separable.
(a) (8pts) Argue that $l = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$ is convex as a function of $\mathbf{w}$.

Given the definition of l, we can see that its values will always be on the interval [0,∞). Hinge loss can be represented by the system:

$$l = \max(0, 1 - y_i \cdot f(\mathbf{x}_i)) = \|_{z=0, \quad y_i f(\mathbf{x}_i) \geq 1}^{z=-y_i f(\mathbf{x}_i)+1, \quad y_i f(\mathbf{x}_i) \in (-\infty,\, 1)}$$

We can determine that this set is convex. Any linear function is convex by definition. 0 and $1 - y_i f(\mathbf{x}_i)$ are both linear functions. We can then say that $z = 0$ and $z = -y_i \mathbf{w}^T\mathbf{x} + 1$ are both convex. We then prove that l is convex because the maximum of two convex functions is also convex.

(b) (5pts) Suppose that for some $\mathbf{w}$ we have a correct prediction of f with $\mathbf{x}_i$, i.e. $f(\mathbf{x}_i)$ =sign($\mathbf{w}^T\mathbf{x}_i$). For binary classifications ($y_i = \pm 1$), what range of values can the hinge loss, l, take of this correctly classified example? Points which are classified correctly and which have non-zero hinge loss are referred to as margin mistakes.

In most cases, a correctly classified point will have a $y_i \text{sign}(\mathbf{w}^T\mathbf{x}) = 1$, resulting in a hinge loss of 0. However, points with $y_i \text{sign}(\mathbf{w}^T\mathbf{x}) = 0$ are also possible. We can treat these as correctly classified examples, however their hinge loss becomes 1. In this case, we have a margin mistake. Hence, the hinge loss function can take on values between [0, 1].

(c) (7pts) Let $M(\mathbf{w})$ be the number of mistakes made by $\mathbf{w}$ on our dataset (in terms of classification loss). Show that:

$$\frac{1}{n} M(\mathbf{w}) \leq \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

In other words, the average hinge loss on our dataset is an upper bound on the average number of mistakes we make on our dataset.

For any given incorrect classification, we define as when $y_i \mathbf{w}^T \mathbf{x}_i < 1$ or sign($f(\mathbf{x}_i)$), indicating a violation of the margin. That means that in the loss function l, for every misclassification we have, it will result in loss ≥ 1. As for the function $M(\mathbf{w})$, we can simplify that to $\frac{1}{m} \cdot m$, where m is the number of misclassifications from our decision function. We can simplify this even further into $\frac{1}{m} M(\mathbf{w}) = \frac{1}{m} \cdot m = 1$.
We now have the equation $1 \leq \frac{1}{m} \sum_{i=1}^{n} \max(0,\ 1 - y_i \mathbf{w}^T \mathbf{x}_i)$
We know that since our loss will always be ≥ 1, we can say that our average will also be ≥ 1. Hence, the average hinge loss on our dataset will be an upper bound on the average number of mistakes on our dataset. Even if we change the coefficent back to $\frac{1}{n}$ from $\frac{1}{m}$, the proportion should remain the same since we are incurring zero loss from correctly classified examples and obviously adding no mistakes for $M(\mathbf{w})$ when an example is correctly classified.

(20 points) Implement a Decision Tree model for the Titanic data set (only use the train file). **Do not use packages for implementing decision trees.**

- (2 pts) Explain how you preprocess the features.
- (2 pts) Divide the train file into a training set and a test set.
- (8 pts) Build a tree on the training data and evaluate the classification performance on the test data.
- (4 pts) Try both Gini index and Information Gain as the attribute selection in building the decision tree; Compare their results on the test set.
- (2 pts) Report your best accuracy on the test data set.
- (2 pts) Give a brief description of your observations.

(20 points) Implement AdaBoost for the Titanic data set. You can use package/tools to implement your decision tree classifiers. The fit function of DecisionTreeClassifier in sklearn has a parameter: sample weight, which you can use to weigh training examples differently during various rounds of AdaBoost.

- (10 pts) Implement the AdaBoost algorithm.
- (4 pts) Plot the train and test errors as a function of the number of rounds from 1 through 500.
- (3 pts) Report your best accuracy on the test data set.
- (3 pts) Give a brief description of your observations.

(20 points) Apply a Neural Network (NN) model to predict a handwritten digit images into 0 to 9. The pickled file represents a tuple of 3 lists : the training set, the validation set and the test set. Each of the three lists is a pair formed from a list of images and a list of class labels for each of the images. An image is represented as numpy 1-dimensional array of 784 (28 x 28) float values between 0 and 1 (0 stands for black, 1 for white). The labels are numbers between 0 and 9 indicating which digit the image represents. The code block below shows how to load the dataset.

```
import cPickle, gzip, numpy

# Load the dataset
f = gzip.open('mnist.pkl.gz', 'rb')
train_set, valid_set, test_set = cPickle.load(f)
f.close()
```

- (6 pts) Build a neural network with one or two hidden layers using the training data and predict the labels of data samples in the test set. Choose a loss function for this classification problem. You can use packages for neural networks. If you write a neural network from scratch with forward pass and back-propagation, you will get extra credits.
- (3 pts) Tune your model hyper-parameters (number of hidden neurons in your hidden layers) on the validation set.
- (5 pts) Plot the train, validation, and test errors as a function of the epochs (x-axis: epochs, y-axis: errors.
- (2 pts) Report the best accuracy on the validation and test data sets.
- (2 pts) Apply early stopping using the validation set to avoid overfitting.
- (2 pts) Give a brief description of your observations.