COM S 573: Machine Learning
Homework #1 Solution

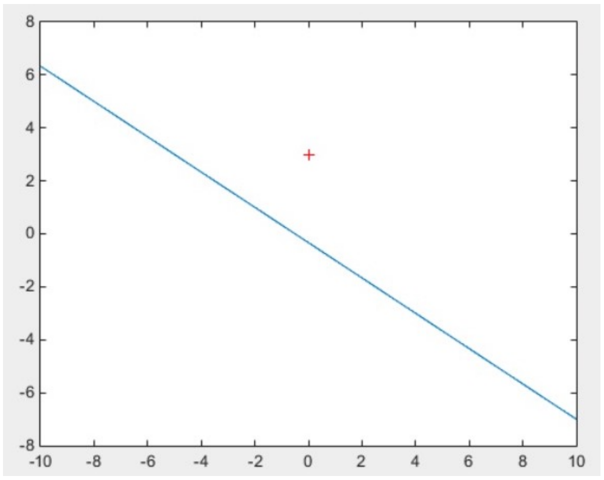
1. (10 points) Consider the perceptron in two dimensions: $h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x})$ where $\boldsymbol{w} = [w_0, w_1, w_2]^T$ and $\boldsymbol{x} = [1, x_1, x_2]^T$. Technically, $\boldsymbol{x}$ has three coordinates, but we call this perceptron two-dimensional because the first coordinate is fixed at 1.

   (a) Show that the regions on the plane where $h(\boldsymbol{x}) = +1$ and $h(\boldsymbol{x}) = -1$ are separated by a line. If we express this line by the equation $x_2 = ax_1 + b$, what are the slope a and intercept b in terms of w_0, w_1, w_2?

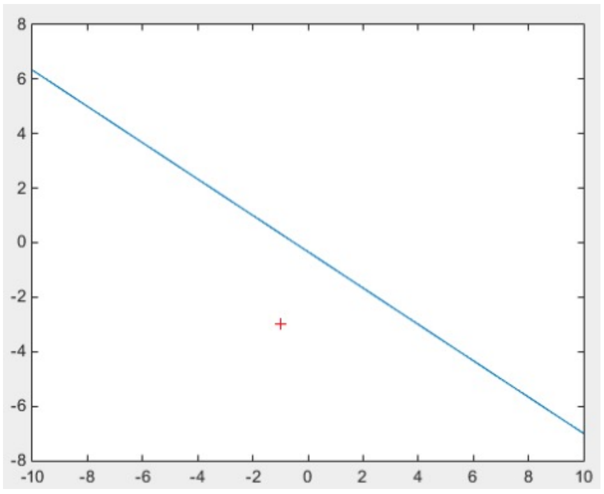
   Solution:
   The boundary is the line given by $x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}$.
   So the slope is $a = -\frac{w_1}{w_2}$ and $b = -\frac{w_0}{w_2}$.

   (b) Draw pictures for the cases $\boldsymbol{w} = [1, 2, 3]^T$ and $\boldsymbol{w} = -[1, 2, 3]^T$. Label the positive and negative prediction regions on the pictures.

   Solution:
   When $\boldsymbol{w} = [1, 2, 3]^T$, we have $x_2 = -\frac{2}{3}x_1 - \frac{1}{3}$. The figure looks like:



   When $\boldsymbol{w} = -[1, 2, 3]^T$, we have $x_2 = -\frac{2}{3}x_1 - \frac{1}{3}$. The figure looks like:



2. (30 points) In logistic regression (labels are {1, -1}), the objective function can be written as

$$E(w) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n w^T x_n}\right).$$

Please

(a) (10 points) Compute the first-order derivative $\nabla E(w)$. You will need to provide the intermediate steps of derivation.
Solution:

$$\begin{aligned}\nabla_w E_{in}(w) &= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{\left(1+e^{-y_n w^T x_n}\right)} \nabla_w \left(1 + e^{-y_n w^T x_n}\right) \\ &= \frac{1}{N}\sum_{n=1}^{N} \frac{e^{-y_n w^T x_n}}{\left(1+e^{-y_n w^T x_n}\right)} \nabla_w \left(-y_n w^T x_n\right) \\ &= \frac{1}{N}\sum_{n=1}^{N} \frac{e^{-y_n w^T x_n}}{\left(1+e^{-y_n w^T x_n}\right)} \left(-y_n x_n\right) \\ &= -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n x_n}{1+e^{y_n w^T x_n}}\end{aligned}$$

(b) (10 points) Once the optimal w is obtain, it will be used to make predictions as follows:

$$\text{Predicted class of } x = \begin{cases} 1 & \text{if } \theta(w^T x) \geq 0.5 \\ -1 & \text{if } \theta(w^T x) < 0.5 \end{cases}$$

where $\theta(z) = \frac{1}{1+e^{-z}}$.
Explain why the decision boundary of logistic regression is still linear, though the linear signal $w^T x$ is passed through a nonlinear function θ to compute the outcome of prediction.
Solution:

$$\hat{y}_i = sigmoid(\boldsymbol{w}^T \boldsymbol{x}_i) = \frac{1}{2} \rightarrow \frac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}_i}} = \frac{1}{2} \rightarrow e^{-\boldsymbol{w}^T \boldsymbol{x}_i} = 1 \rightarrow \boldsymbol{w}^T \boldsymbol{x}_i = 0$$

(c) (5 points) Is the decision boundary still linear if the prediction rule is changed to the following? Justify briefly.

$$\text{Predicted class of } x = \begin{cases} 1 & \text{if } \theta(w^T x) \geq 0.9 \\ -1 & \text{if } \theta(w^T x) < 0.9 \end{cases}$$

Solution:

$$\hat{y}_i = sigmoid(\boldsymbol{w}^T \boldsymbol{x}_i) = 0.9 \rightarrow \frac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}_i}} = 0.9 \rightarrow e^{-\boldsymbol{w}^T \boldsymbol{x}_i} = \frac{1}{9} \rightarrow \boldsymbol{w}^T \boldsymbol{x}_i = \log\frac{1}{9}$$

(d) (5 points) In light of your answers to the above two questions, what is the essential property of logistic regression that results in the linear decision boundary?
Solution: Sigmoid is a Monotonic function.

3. (10 points) Given

$$X = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^{m\times n}$$

where $x_i \in \mathbb{R}^m$ for all i, and

$$Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix} \in \mathbb{R}^{n\times p}$$

where $y_i \in \mathbb{R}^p$ for all i. Show that

$$XY = \sum_{i=1}^{n} x_i y_i^T.$$

Solution:

$$(XY)_{kj} = \sum_{i=1}^{n} x_{ik} y_{ij}$$

$$(\sum_{i=1}^{n} x_i y_i^T)_{kj} = \sum_{i=1}^{n} (x_i y_i^T)_{kj} = \sum_{i=1}^{n} x_{ik} y_{ij}$$

4. (10 points) We show that maximizing log-likelihood is equivalent to minimizing $RSS(\boldsymbol{w})$. In particular,

$$\log \mathcal{L}(\boldsymbol{w}|\boldsymbol{X}) \;\; = -\frac{1}{2}\left(\frac{1}{\sigma^2} RSS(\boldsymbol{w}) + n\log\sigma^2\right) + const$$

(a) (5 points) Please derive the optimal $\boldsymbol{w}^*$ and σ^*.

Solution:

$$\frac{\log \mathcal{L}(\boldsymbol{w}|\boldsymbol{X})}{\partial \boldsymbol{w}} \equiv \frac{\partial RSS(\boldsymbol{w})}{\partial \boldsymbol{w}} = -2\boldsymbol{X^T y} + 2\boldsymbol{X^T X w} \;\; = 0 \rightarrow \boldsymbol{w}^* = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$$

$$\frac{\log \mathcal{L}(\boldsymbol{\sigma^2}|\boldsymbol{X})}{\partial \boldsymbol{w}} \equiv -\frac{1}{2}\left(\frac{N}{\sigma^2} - \frac{RSS(\boldsymbol{w})}{(\sigma^2)^2}\right) = 0 \rightarrow (\sigma^*)^2 = \frac{1}{N} RSS(\boldsymbol{w})$$

(b) (5 points) What do you observe from the optimal σ^*?

Solution: the noise variance coincides with the average RSS.

5. (10 points) Show that sigmoid function and softmax function are the same in the binary case.

$$\text{sigmoid}(y) = \frac{1}{1+e^{-y}}$$

$$\text{softmax}(y_j) = \frac{e^{y_j}}{\sum_{i=1}^{c} e^{y_i}}$$

Solution: for class 1 in softmax

$$\text{softmax}(y_1) = \frac{e^{y_1}}{\sum_{i=1}^{c} e^{y_i}} = \frac{e^{y_1}}{e^{y_1} + e^{y_2}} = \frac{1}{1+e^{-(y_1-y_2)}}$$

By setting $y = y_1 - y_2$ in the sigmoid function, two functions are the same.

6. (30 points) **Perceptron for Handwritten Digits Recognition**: The handwritten digits files are in the “data” folder: train.txt and test.txt. The starting code is in the “code” folder. In the data file, each row is a data example. The first entry is the digit label (“1” or “5”), and the next 256 are grayscale values between -1 and 1. The 256 pixels correspond to a 16×16 image. You are expected to implement your solution based on the given codes. The only file you need to modify is the “solution.py” file. You can test your solution by running “main.py” file. Note that code is provided to compute a two-dimensional feature (symmetry and average intensity) from each digit image; that is, each digit image is represented by a two-dimensional vector before being augmented with a “1” to form a three-dimensional vector as discussed in class. These features along with the corresponding labels should serve as inputs to your Perceptron algorithm. You are expected to use Python3.

   (a) (5 points) Familiarize yourself with the data by completing the *show_images* function. Include the images you plotted in your report.

   (b) (5 points) In this assignment, we have already extracted two features, symmetry and average intensity, to distinguish between 1 and 5. Familiarize yourself with the features by completing the *show_features* function and include the 2-D scatter plot into your report. For each sample, plot the two features with a red $*$ if the label is 1 and a blue $+$ if the label is 5.

   (c) (10 points) Complete the *Perceptron* class. You can test your accuracy results using the “test_accuracy” function in “main.py”.

   (d) (10 points) Complete the *show_result* function to plot the test data with the separators. Include the images you plotted into your report.

   Deliverable: You should submit (1) a report (along with your write-up for other questions) that summarizes your results and (2) the “solution.py” file to the Canvas.

   Note: Please read the “Readme.txt” file carefully before you start this assignment. Please do NOT change anything in the “main.py” and “helper.py” files when you program.

Solution: see the solution.py file.

```
import numpy as np
import sys
from helper import *


def show_images(data):
"""Show the input images and save them.

Args:
data: A stack of two images from train data with shape (2, 16, 16).
  Each of the image has the shape (16, 16)

Returns:
Do not return any arguments. Save the plots to 'image_1.*' and 'image_2.*' and
include them in your report
"""
### YOUR CODE HERE
for i,img in enumerate(data):
plt.clf()
fig = plt.figure()
plt.imshow(img)
fig.savefig('image_%d'%(i+1))
### END YOUR CODE


def show_features(X, y, save=True):
"""Plot a 2-D scatter plot in the feature space and save it.

Args:
X: An array of shape [n_samples, n_features].
y: An array of shape [n_samples,]. Only contains 1 or -1.
save: Boolean. The function will save the figure only if save is True.

Returns:
Do not return any arguments. Save the plot to 'train_features.*' and include it
in your report.
"""
### YOUR CODE HERE
n, _ = X.shape
fig = plt.figure()
for i in range(n):
if y[i] == 1:
plt.plot(X[i,0], X[i,1], 'r*')
else:
plt.plot(X[i,0], X[i,1], 'b+')
if save:
fig.savefig('train_features')

### END YOUR CODE
```

```
class Perceptron(object):

def __init__(self, max_iter):
self.max_iter = max_iter

def fit(self, X, y):
"""Train perceptron model on data (X,y).

Args:
X: An array of shape [n_samples, n_features].
y: An array of shape [n_samples,]. Only contains 1 or -1.

Returns:
self: Returns an instance of self.
"""
### YOUR CODE HERE
w = np.zeros([3])
for i in range(self.max_iter):
misclass = []

for idx in range(len(X)):
if not np.sign(np.dot(w, X[idx])) == y[idx]:
misclass += [idx]

if len(misclass) == 0:
print('All correct in training set.')
break

else:
idx = np.random.choice(misclass)
w = w + y[idx] * X[idx]

print('Fitting finished in %d iterations.' % (i+1))
self.W = w
### END YOUR CODE

return self

def get_params(self):
"""Get parameters for this perceptron model.

Returns:
W: An array of shape [n_features,].
"""
if self.W is None:
print("Run fit first!")
sys.exit(-1)
return self.W
```

```
def predict(self, X):
"""Predict class labels for samples in X.

Args:
X: An array of shape [n_samples, n_features].

Returns:
preds: An array of shape [n_samples,]. Only contains 1 or -1.
"""
### YOUR CODE HERE
preds = np.sign(np.dot(X, self.W.transpose()))

return preds
### END YOUR CODE

def score(self, X, y):
"""Returns the mean accuracy on the given test data and labels.

Args:
X: An array of shape [n_samples, n_features].
y: An array of shape [n_samples,]. Only contains 1 or -1.

Returns:
score: An float. Mean accuracy of self.predict(X) wrt. y.
"""
### YOUR CODE HERE
preds = self.predict(X)
score = sum(preds==y)/float(len(X))

return score
### END YOUR CODE




def show_result(X, y, W):
"""Plot the linear model after training.
   You can call show_features with 'save' being False for convenience.

Args:
X: An array of shape [n_samples, 2].
y: An array of shape [n_samples,]. Only contains 1 or -1.
W: An array of shape [n_features,].

Returns:
Do not return any arguments. Save the plot to 'test_result.*' and include it
in your report.
"""
### YOUR CODE HERE
show_features(X, y, save=False)
```

```
x = np.linspace(-1,0,2)
y = - W[1]/W[2]*x - W[0]/W[2]

plt.plot(x, y, linewidth=2.5, linestyle="-")
# plt.axis([-1, 0.1, -1, 0.5])
plt.savefig('test_result')
### END YOUR CODE



def test_perceptron(max_iter, X_train, y_train, X_test, y_test):

# train perceptron
model = Perceptron(max_iter)
model.fit(X_train, y_train)
train_acc = model.score(X_train, y_train)
W = model.get_params()

# test perceptron model
test_acc = model.score(X_test, y_test)

return W, train_acc, test_acc
```