Homework Assignment 3

COGS 118A: Supervised Machine Learning Algorithms

Due: January 22nd, 2020, 11:59pm (Pacific Time).

Instructions: There are two parts to this homework. Part 1 is distributed here in this PDF. Part 2 is the code skeleton for questions 5, and it is distributed in a Jupyter notebook that can be downloaded from Canvas at `Modules -> Week 3 -> Homework files`. To answer the questions below you may need a mix of math and coding. To submit your answers you need to combine all of these into a single PDF document and upload it to Gradescope.

Part 1 : Your math can be **handwritten**. In that case use a scanner app to take photos of the pages and turn them into a PDF. Adobe, Evernote, Microscoft, and many others offer free scanner apps. Alternatively, your math may be done in LaTeX or in a Jupyter Notebook using LaTeX markdown and from there turned into PDF.

Part 2: Output from a Jupyter Notebook (math markdown, code, or plots) can be turned into PDF using `File->Print->Preview` or `File->Download as->PDF via LaTeX`

In the end: all of these different sources must be stitched together into **a single PDF file**. PDF Merge tool (e.g. smallpdf, comebinepdf) can do this among many other tools. Make sure to show the steps of your solution, not just the final answer. You may search information online but you will need to write code/find solutions to answer the questions yourself.

Late Policy: Late assignments are deducted 5% each day they are late. No late assignments are accepted after one week.

System Setup: For this class, please use **Python 3.6** or later for homework with recent copies of the libraries NumPy, SciPy, Pandas, Scikit-learn, Matplotlib, Seaborn, and Jupyter-Notebook.

To run a Jupyter Notebook you may use UCSD Datahub or Google Colab or your own local installation. If you decide on local installation, and you are creating your own setup for the first time we highly recommend you use Anaconda as it will come with all the required libraries and more.

If you are not feeling comfortable with the programming assignments in this homework, it might help to take a look at `https://github.com/UCSD-COGS108/Tutorials`

Grade: ____ out of 100 points

1 (10 points) Conceptual Questions

(1.1) Is the following statement true or false?

$f(x)$ is linear with respect to x, given $f(x) = w_0 + w_1 x + w_2 x^2$ where $x, w_0, w_1, w_2 \in \mathbb{R}$.

[True] [False]

(1.2) "One-hot encoding" is a standard technique that turns categorical features into general real numbers. If we have a dataset S containing n data points where each data point has 1 categorical variable. This variable encodes one of k possible categories for each sample. Thus, the shape of the one-hot encoding matrix that represents the dataset S is:

A. $k \times k$
B. $1 \times k$
C. $n \times k$
D. $n \times n$

(1.3) Assume we have a binary classification model:

$$f(\mathbf{x}) = \begin{cases} +1, & \mathbf{w} \cdot \mathbf{x} + b \geq 0, \\ -1, & \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases}$$

where there's a feature vector $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$, bias $b \in \mathbb{R}$, and a weight vector $\mathbf{w} = (w_1, w_2) \in \mathbb{R}^2$. The decision boundary of the classification model is:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

(a) If the predictions of the classifier f and its decision boundary $\mathbf{w} \cdot \mathbf{x} + b = 0$ are shown in Figure 1, which one below can be a possible solution of weight vector $\mathbf{w}$ and bias b?

A. $\mathbf{w} = (+1, 0), b = -1$.
B. $\mathbf{w} = (-1, 0), b = +1$.
C. $\mathbf{w} = (+1, 0), b = +1$.
D. $\mathbf{w} = (0, -1), b = -1$.

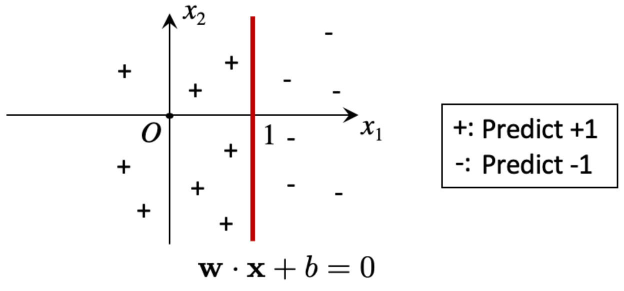


Figure 1: Decision Boundary 1

(b) If the predictions of the classifier f and its decision boundary $\mathbf{w} \cdot \mathbf{x} + b = 0$ are shown in Figure 2, which one below can be a possible solution of weight vector $\mathbf{w}$ and bias b?

A. $\mathbf{w} = (+1, 0), b = -1$.
B. $\mathbf{w} = (-1, 0), b = +1$.
C. $\mathbf{w} = (+1, 0), b = +1$.
D. $\mathbf{w} = (0, -1), b = -1$.

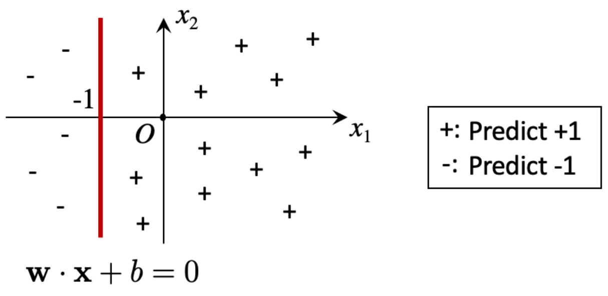


Figure 2: Decision Boundary 2

(1.4) Choose the **most** significant difference between **regression** and **classification**:

A. unsupervised learning vs. supervised learning.

B. prediction of continuous values vs. prediction of class labels.

C. features are not one-hot encoded vs features are one-hot encoded.

D. none of the above.

2 (25 points) Decision Boundary

2.1 (5 points)

We are given a classifier that performs classification in $\mathbb{R}^2$ (the space of data points with 2 features (x_1, x_2)) with the following decision rule:

$$h(x_1, x_2) = \begin{cases} 1, & \text{if} \quad x_1 + 2x_2 - 4 \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Draw the decision boundary of the classifier and shade the region where the classifier predicts 1. Make sure you have marked the x_1 and x_2 axes and the intercepts on those axes.

2.2 (15 points)

We are given a classifier that performs classification on $\mathbb{R}^2$ (the space of data points with 2 features (x_1, x_2)) with the following decision rule:

$$h(x_1, x_2) = \begin{cases} 1, & \text{if} \quad w_1 x_1 + w_2 x_2 + b \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Here, the normal vector $\mathbf{w}$ of the decision boundary is normalized, i.e.:

$$||\mathbf{w}||_2 = \sqrt{w_1^2 + w_2^2} = 1.$$

1. Compute the parameters w_1, w_2 and b for the decision boundary in Figure 3. Please make sure the predictions from the obtained classifier are consistent with Figure 3.

   Hint: Please use the intercepts in the Figure 3 to find the relation between w_1, w_2 and b. Then, substitute it into the normalization constraint to solve for parameters.

2. Use parameters from the above question to compute predictions for the following two data points: $A = (-3, -2)$, $B = (1, 1)$.

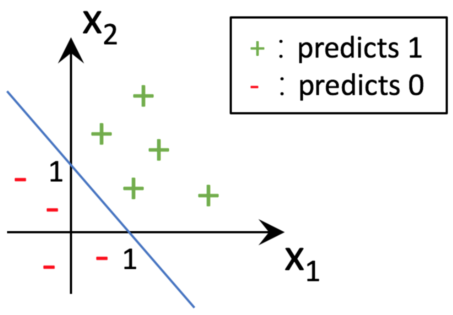


Figure 3: Decision boundary to solve for parameters.

2.3 (5 points)

We are given a classifier that performs classification in $\mathbb{R}^2$ (the space of data points with 2 features (x_1, x_2)) with the following decision rule:

$$h(x_1, x_2) = \begin{cases} 1, & \text{if} \quad x_1^2 + x_2^2 - 10 \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Draw the decision boundary of the classifier and shade the region where the classifier predicts 1. Make sure you have marked the x_1 and x_2 axes and the intercepts on those axes.

3 (15 points) Derivatives

3.1 Function Defined by Scalars

1. Given a function $f(w) = (y_1 - wx_1)^2$ where $(x_1, y_1) = (1, 1)$ represents a data point, derive $\dfrac{\partial f(w)}{\partial w}$.

2. Given a function $f(w) = \sum_{i \in \{1,2\}} (y_i - wx_i)^2$ where $(x_1, y_1) = (1, 1), (x_2, y_2) = (2, 3)$ are two data points, derive $\dfrac{\partial f(w)}{\partial w}$.

3.2 Function Defined by Vectors

1. Given a function $f(w) = (\mathbf{y} - w\mathbf{x})^T(\mathbf{y} - w\mathbf{x})$ where $\mathbf{x} = [1, 3]^T$ and $\mathbf{y} = [1, 2]^T$, derive $\dfrac{\partial f(w)}{\partial w}$.
   Note: In $f(w)$, $w \in \mathbb{R}$ is still a scalar.

4 (30 points) Linear Regression

Assume we are given a dataset $S = \{(x_i, y_i), i = 1, \ldots, n\}$. Here, $x_i \in \mathbb{R}$ is a feature scalar (a.k.a. value of input variable) and $y_i \in \mathbb{R}$ is its corresponding value (a.k.a. value of dependent variable). In this section, we aim to fit data points with a line:

$$y = w_0 + w_1 x \tag{1}$$

where $w_0, w_1 \in \mathbb{R}$ are two parameters to determine the line. Next, we measure the quality of fitting by evaluating a sum-of-squares error function $g(w_0, w_1)$:

$$g(w_0, w_1) = \sum_{i=1}^{n} (w_0 + w_1 x_i - y_i)^2 \tag{2}$$

When $g(w_0, w_1)$ is near zero, it means the proposed line can fit the dataset and model an accurate relation between x_i and y_i. The best line with parameters (w_0^*, w_1^*) can reach the minimum value of the error function $g(w_0, w_1)$:

$$(w_0^*, w_1^*) = \arg\min_{w_0, w_1} g(w_0, w_1) \tag{3}$$

To obtain the parameters of the best line, we will take the gradient of function $g(w_0, w_1)$ and set it to zero. That is:

$$\nabla g(w_0, w_1) = \mathbf{0} \tag{4}$$

The solution (w_0^*, w_1^*) of the above equation will determine the best line $y = w_0^* + w_1^* x$ that fits the dataset S.

In reality, we typically tackle this task in a matrix form: First, we represent data points as matrices $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ and $Y = [y_1, y_2, \ldots, y_n]^T$, where $\mathbf{x}_i = [1, x_i]^T$ is a feature vector corresponding to x_i. The parameters of the line are also represented as a matrix $W = [w_0, w_1]^T$. Thus, the sum-of-squares error function $g(W)$ can be defined as (a.k.a. squared L_2 norm):

$$\begin{aligned} g(W) &= \sum_{i=1}^{n} (\mathbf{x}_i^T W - y_i)^2 && (5) \\ &= \|XW - Y\|_2^2 && (6) \\ &= (XW - Y)^T (XW - Y) && (7) \end{aligned}$$

Similarly, the parameters $W^* = [w_0^*, w_1^*]^T$ of the best line can be obtained by solving the equation below:

$$\nabla g(W) = \frac{\partial g(W)}{\partial W} = \mathbf{0} \tag{8}$$

(a) According to Eq. 6 and 7, compute the gradient of $g(W)$ with respect to W. Your result should be in the form of X, Y and W.

(b) By setting the answer of part (a) to $\mathbf{0}$, prove the following:

$$W^* = \arg\min_W g(W) = (X^T X)^{-1} X^T Y \tag{9}$$

Note: The above formula demonstrates a closed form solution of Eq. 8.

Previously, we define a sum-of-squares error function $g(w_0, w_1) = \sum_{i=1}^{n}(y_i - w_0 - w_1 x_1)^2$ and represent it in a matrix form $g(W) = \|XW - Y\|_2^2$. Actually, we can have multiple choices of the error function: For example, we can define a sum-of-absolute error function $h(w_0, w_1)$:

$$h(w_0, w_1) = \sum_{i=1}^{n} |w_0 + w_1 x_i - y_i| \tag{10}$$

and represent it in a matrix form $h(W)$ (a.k.a. L_1 norm):

$$\begin{aligned} h(W) &= \sum_{i=1}^{n} |\mathbf{x}_i^T W - y_i| \qquad (11) \\ &= \|XW - Y\|_1 \qquad (12) \end{aligned}$$

(c) According to the Eq. 11, compute the gradient of the error function $h(W)$ with respect to W. Your result should be in the form of $\mathbf{x}_i$, y_i and W.

Hint: Given a function $f(\mathbf{x}) \in \mathbb{R}$, we have:

$$\frac{\partial |f(\mathbf{x})|}{\partial \mathbf{x}} = \text{sign}(f(\mathbf{x})) \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$$

where

$$\text{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0. \end{cases}$$

You're done with this problem, but before going to the next one let's note that the gradient in (c) can also be represented as:

$$\nabla h(W) = \frac{\partial h(W)}{\partial W} = \Big(\big(\text{sign}(XW - Y)\big)^\top X \Big)^\top \tag{13}$$

where $\text{sign}(A)$ means performing element-wise $\text{sign}(a_{ij})$ over all element a_{ij} in a matrix A. This matrix form of gradient can be helpful in some following coding questions.

5 (20 points) Coding: Data Manipulation

Download the notebook from Canvas `Modules -> Week 3 -> Homework files` for the code skeleton seen here

In this question, we still use the Iris dataset from Homework 1 Question 6. In fact, you can see the shape of array X is $(150, 4)$ by running `X.shape`, which means it contains 150 data points where each has 4 features. Here, we will perform some basic data manipulation and calculate some statistics.

For this problem, we will be using Pandas (`https://pandas.pydata.org/`), the flagship Python package designed for data manipulation and analysis. In Pandas, data is stored in either a series or a dataframe instead of an array. These data structures offer more flexibility in dealing with datasets. For example, instead of just indixing and slicing an array by integer index values, we can use feature names (represented as column names in a dataframe to access and slice dataframes). This guide (`10 minutes to pandas`) can introduce you to Pandas and is a good introduction to start working on the library.

- Import several useful packages into Python:

```
import pandas as pd
from sklearn import datasets
```

- Load Iris dataset into Python:

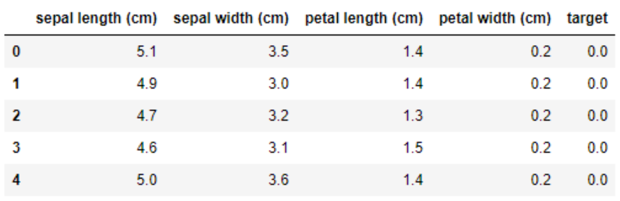
```
iris = datasets.load_iris()
```

- Create a Pandas dataframe that contains the data and labels:

```
iris_df = pd.DataFrame(data = np.c_[iris.data, iris.target],
                       columns = iris['feature_names'] + ['target'])
```

- We can use the method `.head()` to display the first 5 rows of the dataset:

```
iris_df.head()
```



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

1. Divide the dataframe `iris_df` evenly to five subsets of data points:

   Group 1: 1st to 30th data point,

   Group 2: 31st to 60th data point,

   Group 3: 61st to 90th data point,

   Group 4: 91st to 120th data point,

   Group 5: 121st to 150th data point.

   Then calculate the mean of feature vectors in each group. Your results should either be five 4-dimensional vectors (i.e. shape of `NumPy` array can be `(4,1)`, `(1,4)` or `(4,)`), or 5 pandas series (each of shape = `(4,)`) instead of Numpy arrays.

2. Remove 2nd and 3rd features from the dataframe `iris_df`, resulting in a dataframe of shape 150×3. Then calculate the mean of all feature vectors. Your result should be a pandas series of shape = `(2,)`.

3. Remove last 10 data points from the dataframe `iris_df`, resulting in a 140×5 dataframe. Then calculate the mean of feature vectors. Your result should be a 4-dimensional vector or a pandas series of shape = `(4,)`.