DATA SCIENCE AND MACHINE LEARNING (MSC)

DAMA60: Algorithmic Techniques and Systems for Data Science and Machine Learning

Academic Year: 2023–2024

#5 Written Assignment	
Submission Deadline	**Wed, 15 May 2024 (23:59 EET)**
Student Name	***Kritikos Antonios***

Remarks

The deadline is definitive.

An indicative solution will be posted online along with the return of the graded assignments.

The assignment is due via the STUDY submission system. **You are expected to turn in a document (.DOC, .ODT, .PDF) and a file containing a Python program:**

- 1 document file (this document) with your name in the area indicated by "Student Name" and the answers to all the questions, along with the Python code snippets for Topic 5.
- 1 file with the complete Python program that answers Topic 5.

You should not make any changes in the written assignment file other than providing your own answers. You should also type all of your answers into Word or compatible word processors (such as Apache OpenOffice https://www.openoffice.org/ or LibreOffice https://www.libreoffice.org/) and not attach any handwritten notes as pictures into your work. Make sure to name all the files (DOC/DOCX/ODT files and Python program file) with **your last name first followed by a dash symbol and the names of each component at the end**. For example, for the student with last name Aggelou the files should be named as follows: Aggelou-HW5.doc (or Aggelou-HW5.pdf), Aggelou-Topic5.py.

Please ensure that all Python code you submit for this assignment opens, displays and executes successfully within the IDLE environment — the official Integrated Development and Learning Environment provided with Python 3. This consistency in the execution environment is necessary for a thorough and equitable evaluation of your work.

Topic	Points	Grades
1. Online Quiz	40	
2. Perceptron & Support Vector Machines	15	
3. Singular Value Decomposition	15	
4. Neural Nets	15	
5. Python Implementation of Logistic Regression	15	
TOTAL	**100**	**/100**

Topic 1: Online Quiz

(40 points) Complete the corresponding online quiz available at:

https://study.eap.gr/mod/quiz/view.php?id=27275

You have one effort and unlimited time to complete the quiz, up to the submission deadline.

Topic 2: Perceptron & Support Vector Machines

(15 total points) Suppose the next training dataset is provided, consisting of 2 classes with 3 instances each. The positive and negative class is signified by the green "plus" and the red "minus" sign, respectively. The coordinates of those 6 instances are also recorded in Table 1 explicitly. We are going to train some specific variants of the linear classifiers that are taught in Chapter 12 of "J. Leskovec, A. Rajaraman & J.D. Ullman (2020). Mining of Massive Datasets (3rd edition). Cambridge University Press", as they are described in each of the following questions.

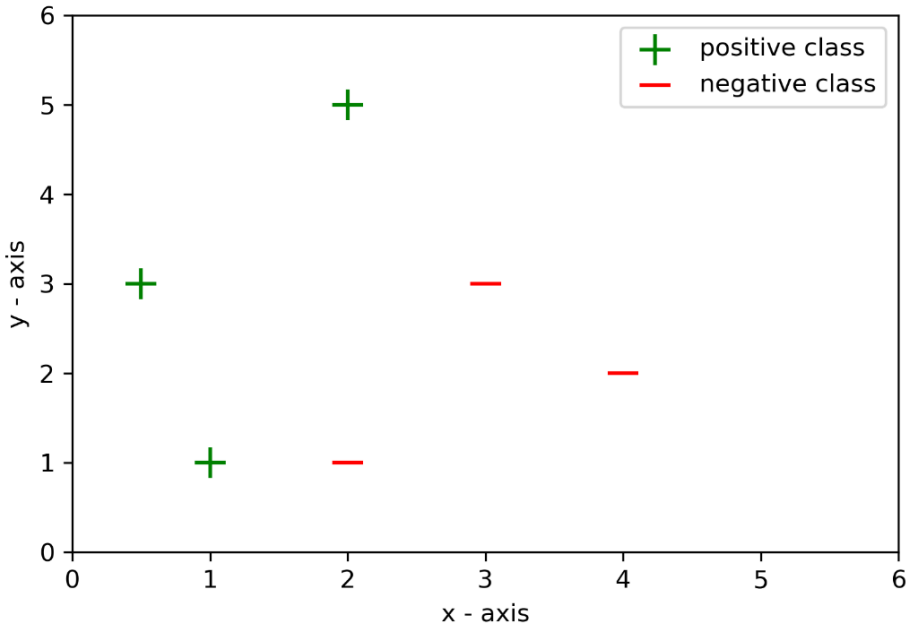

Figure 1: Scatter plot of provided training instances.

Point	A	B	C	D	E	F
x-axis	1	1/2	2	2	3	4
y-axis	1	3	5	1	3	2
Label	+1	+1	+1	-1	-1	-1

Table 1: Coordinates of provided training instances.

(a) (6 points) Assuming that the learning rate (η) equals to $\frac{1}{4}$, consider the training of a perceptron classifier with zero threshold. The initial value of weight vector ($\boldsymbol{w}$) is $[-1, 1]$. We are going to apply the learning steps examining the provided points of the training set sorted lexicographically. Fill in the 3 missing values indicated by the numbered question marks. The first two correspond to the weight vector after processing all the provided points for one iteration ($\boldsymbol{w}^{(1)}$), while the third one counts the number of the correctly classified points after executing the first iteration.

Your answer:

$\boldsymbol{w}^{(1)}$ = [**-3/2, 1/2**]

Number of correctly classified points (out of 6) = **4**

(b) (9 points) In the provided dataset of Figure 1, apply Support Vector Machines classifier using the gradient descent method for 2 iterations. Utilize all the instances of the provided dataset as a batch. The objective loss function that should be minimized is Equation 12.4 of Section 12.3.3 in "J. Leskovec, A. Rajaraman & J.D. Ullman (2020). Mining of Massive Datasets (3rd edition). Cambridge University Press", defining an optimization problem involving regularization. Please note that the second term of Equation 12.4 is the hinge loss function (penalty function).

Thus, we set the regularization parameter (C) to 2, the learning rate (η) to $\frac{1}{4}$, while the initial weight vector ($\boldsymbol{w}$) and the bias term ($bias$) are set to $[-1, 0]$ and -1, respectively. Fill in the 4 missing values indicated by the numbered question marks.

Note that the numerical missing values should be filled in by using reduced fractions when needed, while accepted answers in the first to complete column may include 'x' or 'o' for wrong and correct decisions, respectively.

Your answer:

Iteration	Decisions regarding (ABCDEF) points	w^{new}	$bias^{new}$
1	xxxooo	[**1, 9/2**]	$\frac{3}{4}$
2	**oooxxx**	$[-\frac{15}{4}, \frac{3}{8}]$	**-15/16**

Topic 3: Singular Value Decomposition

(15 total points) The following matrix (Table 2) represents ratings of six movies given by eight users. Each row represents the users' set of ratings and each column represents the movies. The labels for users and movies are provided for every row and column. Finally, every cell in the matrix represents the user's rating for a movie.

	Sci-Fi				Romance	
	Matrix	Alien	Star Wars	The Avengers	Love Actually	Titanic
Joe	5	3	5	5	2	1
Jim	4	4	4	4	1	0
John	3	3	4	5	0	0
Jack	4	2	5	4	0	0
Jill	1	0	1	0	5	5
Jenny	0	0	1	2	4	5
Jane	0	0	0	0	4	3
Jullie	1	0	1	1	5	4

Table 2: Ratings of movies by users.

The SVD decomposition of the rating matrix is computed as follows:

$$USV^T =$$

$$\begin{bmatrix} -0.54361 & 0.12524 & 0.19457 & 0.12919 & -0.34957 & -0.63062 \\ -0.44337 & 0.20515 & 0.31233 & -0.57657 & 0.37801 & 0.082004 \\ -0.41082 & 0.24411 & -0.47207 & -0.27179 & -0.071147 & 0.21159 \\ -0.41889 & 0.24188 & -0.0087909 & 0.68432 & 0.089370 & 0.48797 \\ -0.21332 & -0.52096 & 0.28868 & 0.20015 & 0.58667 & -0.096525 \\ -0.22997 & -0.44830 & -0.70856 & -0.029325 & 0.17125 & -0.17213 \\ -0.11112 & -0.38690 & 0.18487 & -0.25997 & -0.29858 & 0.51746 \\ -0.23007 & -0.44952 & 0.14862 & 0.014539 & -0.50912 & 0.077369 \end{bmatrix}$$

$$\begin{bmatrix} 17.099 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11.813 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.2133 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.7794 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.91052 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.41469 \end{bmatrix}$$

$$\begin{bmatrix} -0.45867 & -0.32016 & -0.52064 & -0.52114 & -0.29895 & -0.23473 \\ 0.18422 & 0.20422 & 0.18741 & 0.19375 & -0.65501 & -0.65012 \\ 0.54585 & 0.18039 & 0.0084403 & -0.65146 & 0.35839 & -0.34146 \\ 0.26764 & -0.76737 & 0.48300 & -0.18328 & -0.22589 & 0.14699 \\ -0.015609 & 0.47077 & 0.19249 & -0.44004 & -0.48623 & 0.55769 \\ -0.62118 & 0.11302 & 0.65069 & -0.19811 & 0.25631 & -0.27022 \end{bmatrix}$$

As we only have two different movie genres we keep the first two singular values, reducing the number of dimensions from six to two.

(a) (5 points) Compute the Retained Energy percentage (RE %) (as it is described in section 11.3.4 of "J. Leskovec, A. Rajaraman & J.D. Ullman (2020). Mining of Massive Datasets (3rd edition). Cambridge University Press") after the creation of the two-dimensional approximation matrix, and the Mean Square Error (MSE) between the original ratings and the reconstructed matrix, using the two larger singular values. Fill in the numbered question marks using 2 decimal digits without rounding.

Your answer:

RE (%) = **97.94%**

MSE = **0.19**

(b) (5 points) Assume a new user named George gives the following ratings to the movies he viewed: 4 for Alien, 5 for Star Wars and 1 for Titanic. Another user named Robert gives the following ratings to the movies: 1 for Alien and 4 for Love Actually.

What is the representation of both new users in the approximated concept space where only the 2 largest singular values have been kept? Using the cosine similarity metric calculate the similarity between George and Robert utilizing their approximated concept space vectors. Use 2 decimal digits of accuracy without rounding.

Your answer:

George: [**-4.11**, **1.10**]

Robert: [**-1.51**, **-2.41**]

$\cos(George, Robert) =$**0.29**

(c) (5 points) Provide a top-6 movie recommendation list for what George would like to watch utilizing his approximated concept space. Use alphabetical ordering if two or more movies have the same value in the concept space.

Your answer:

	Matrix	Alien	Star Wars	The Avengers	Love Actually	Titanic
ordering	**3**	4	**2**	**1**	**5**	**6**

Topic 4: Neural Nets

(15 total points) Make the necessary calculations and answer the following 2 questions regarding the full learning cycle of the following neural net with one hidden layer (Figure 2). Take as granted that the provided neural net does not include any bias term, while the selected activation functions σ_1 and σ_2, coincide with the identity activation function. The activation functions are applied in the hidden layer, where it holds that $h_1 = \sigma_1(z_1)$ and $h_2 = \sigma_2(z_2)$. The Loss function of that neural net has been set to the Mean Squared Error (MSE), following the convention that this is defined through the $\frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right)$, where y denotes the predicted value, $\hat{y}$ the actual value, and n the number of the training instances. Keep in mind that the coefficient of MSE ($\frac{1}{2}$) is used in neural nets for simplifying the calculations when we need to compute the derivatives of that specific Loss function. Moreover, the weights of the neural net have been randomly initialized to the next values:

Parameter	w_{11}	w_{12}	w_{21}	w_{22}	u_1	u_2
Value	0.10	0.20	0.05	-0.10	0.20	0.50

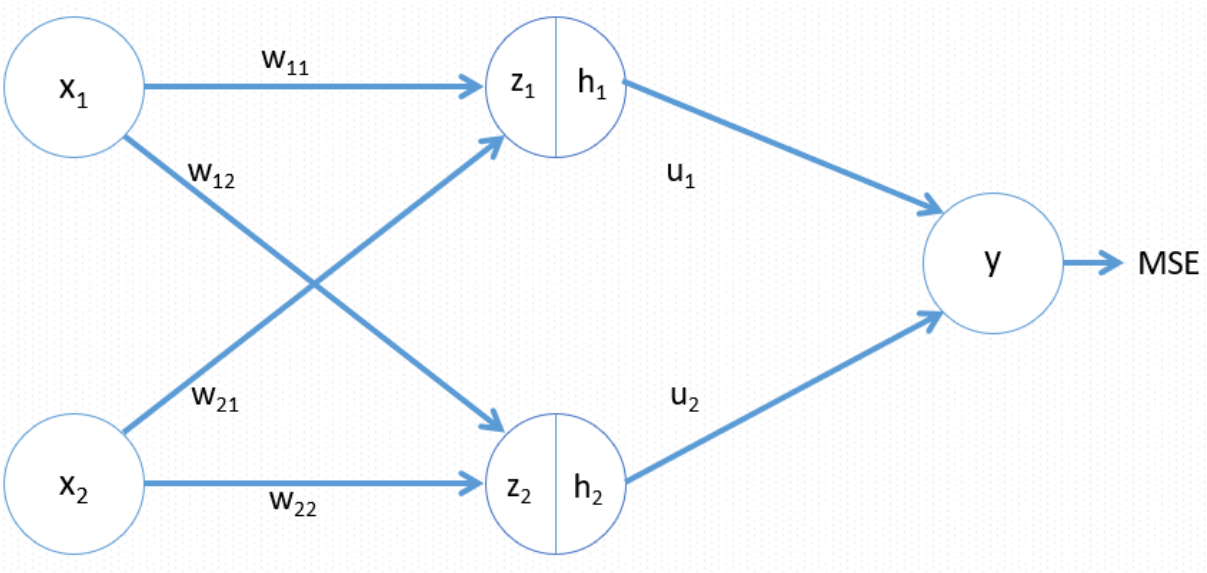

Figure 2: Architecture of examined neural net.

(a) (**6 points**) Let's assume that the input of the provided neural net is $(x_1, x_2) = (1, 5)$, whose ground truth value $\hat{y}$ is equal to 1. Execute only one forward pass for that input and compute the value of the network's Loss function, indicated by the numbered question mark. Use 3 decimal digits of accuracy without rounding. Note that $n = 1$ here, since we have only one instance.

Your answer:

$$L(y, \hat{y}) = \mathbf{0.583}$$

(b) (**9 points**) Apply one backpropagation pass in the neural net of Figure 2 and then associate the provided values that correspond to the updated weights with the names of actual parameters. Use the gradient descent method for updating the weights during that pass. Therefore, you need to fill the 3 gaps indicated by the numbered question marks, as depicted below. The learning rate parameter (lr) has been set equal to 0.10. Three of those pairs are already provided, helping you to verify your calculations. All the results have been computed with 3 decimal places of accuracy without rounding.

Key	Parameter		Value	Weights
0	w_{11}		**A**	0.170
1	w_{12}		**B**	0.237
2	w_{21}		**C**	0.121
3	w_{22}		**D**	0.467
4	u_1		**E**	0.158
5	u_2		**F**	0.254

Your answer:

Key corresponds to Value

0 corresponds to **C**

1 corresponds to **F**

2 corresponds to **E**

3 corresponds to **A**

4 corresponds to **B**

5 corresponds to **D**

Topic 5: Python Implementation of Logistic Regression

(15 total points) In this topic you will use popular libraries of Python to implement from scratch a variant of Logistic Regression classifier, a supervised learning algorithm that acts as a 1-layer neural net.

We are going to use a toy-dataset with a total of 3 columns, which describes a binary classification problem. The column 'label' is the class variable, while columns 'dim1' and 'dim2' denote the input features. After reading that toy-dataset and splitting it to training and test subsets, we scale its features using the corresponding component from sklearn library (StandardScaler). Next, we proceed with the actual part of the Logistic Regression, implementing its forward and backward pass, for a specified number of iterations. The cost function that we adopt here is the binary cross entropy, while we also apply the gradient descent algorithm for optimizing the learnable parameters of that network. Finally, we measure the performance of our classifier in the test subset.

File Topic5.py contains the incomplete Python code that you have to use to fill in your answers, while the Dataset.csv file includes the input data.

(a) (**3 points**) Complete the following Python code, by filling the 3 missing gaps indicated by the numbered question marks, to count the number of instances per class and assert that they sum up to the number of instances of the whole dataset.

Your answer:

```
#Required libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

# read the data and print some basic information
df = pd.read_csv('Dataset.csv')
print('Typical statistical information of the used data: \n\n ', df.describe(), sep='')

# keep the first 2 numeric columns as the feature space of the input
# and use the last one as the target variable (discrete)
X = df.loc[:, ['dim1','dim2']]
y = df.loc[:, 'label']

# count the instances of each class
count_class_0, count_class_1 = 0, 0
for i in y:
    if i == 0:
        count_class_0 += 1
    else:
        count_class_1 += 1
print('\n\nNumber of instances for class0: {:3} and class1: {:3}'.format(count_class_0,
count_class_1))

# verify that the sum of the values of the above 2 counters
# equals to the total number of dataset's instances
assert count_class_0 + count_class_1 == X.shape[0]
```

(b) (**4 points**) Here is the main part of the Logistic Regression classifier. First, we scale our data using the standardization method. Next, we introduce some parameters based on the structure of the input data, and then implement the training procedure. Complete the following Python code, by filling the 4 missing gaps indicated by the numbered question marks, carefully reading the comments of each line. Note that the binary cross entropy loss function is a variant of cross entropy as defined in "J. Leskovec, A. Rajaraman & J.D. Ullman (2020). Mining of Massive Datasets (3rd edition). Cambridge University Press". More specifically, its formula is $-\frac{1}{n}\sum_{i=1}^{n}[\hat{y}_i * \ln(p_i) + (1-\hat{y}_i) * \ln(1-p_i)]$, where $\hat{y}_i$ denotes the actual value (target variable) of i-th instance, p_i is the predicted probability of i-th instance to belong to the positive class, and n the number of training instances.

Your answer:

```
# split original dataset to training and test subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.85,shuffle=True,
                                                    stratify=y, random_state=2024)

# scale our data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# initialize the needed parameters for the Logistic Regression Classifier
lr = 0.03                               # define the learning rate
n_iter = 100                            # define the number of iterations
m = X_train.shape[0]                    # the number of rows of the training data
n = X_test.shape[0]                     # the number of rows of the test data
number_of_features = X_train.shape[1]   # the number of columns of the training or test data

# keep the weights of the Logistic Regression Classifier in a dictionary
network_params = {}
# initialize the input layer coefficients as zeros (vector with dimension (number of features,))
network_params["W"] = np.zeros(number_of_features)
# initialize the bias as 0 (scalar)
network_params["b"] = 0
print('\n\nInitial parameters: \nW = ', network_params['W'], '\nb = ',network_params['b'],'\n\n')

losses = []
for iteration in range(n_iter):

    # feed input X_train into the network to calculate the log-odds
    logits = np.dot(X_train, network_params['W']) + network_params['b']
    # apply the sigmoid function for transforming logits to probabilities
    prob = 1 / (1 + np.exp(-logits))
    # calculate the sum of loss for all instances based on the binary cross entropy function
    loss = np.sum(y_train * np.log(prob) + (1 - y_train) * np.log(1 - prob))
    # average the sum of loss with the number of training instances
    loss = -1/m * loss
    # add the loss of each iteration into a list
    losses.append(loss)
    # partial derivative of loss function with respect to weights
    dloss_dW = 1/m * np.dot(X_train.T, (prob - y_train))
    # partial derivative of loss function with respect to bias
    dloss_db = 1/m * np.sum(prob - y_train)
    # update rules based on gradient descent approach
    network_params['W'] -= lr * dloss_dW
    network_params['b'] -= lr * dloss_db
    # starting from zero, we repeat the next process per 25 iterations;
    # we print the value of the loss function at the current iteration
    if iteration % 25 == 0:
        print('Iteration {:3} Loss function ={:6}'.format(iteration, losses[iteration]))
```

(c) (**2 points**) Finally, we apply a typical threshold of 0.50 for discriminating the decisions of our classifier, measuring the performance of our trained classifier in the unseen test subset. We also print the values of the learned parameters. Fill the 2 missing gaps indicated by the numbered question marks, so as to compute that performance.

Your answer:

```
print('\n\nOptimized parameters: \nW = ', np.round(network_params['W'],3),\
                                    '\nb = ', np.round(network_params['b'],3))

# using the optimized parameters of the trained network
# infer that model over the unseen test subset
logits_test_set = np.dot(X_test, network_params["W"]) + network_params["b"]

# apply the sigmoid activation function on logits for transforming them to probabilities
prob_test_set = 1 / (1 + np.exp(-logits_test_set))

# measure the classification accuracy on the test subset (this should be a number in range [0,1])
y_pred = prob_test_set >= 0.5
y_pred = y_pred.astype("int")
print('\n\nClassification Accuracy (test subset) ={:6}'.\
                          format(np.round(np.count_nonzero( y_pred == y_test ) /n, 3)))
```

(d) (**3 points**) Follow the guidelines on the first page of this assignment and incorporate the .py file that answers this topic into your submission.

Your answer:

The code in your .py file with name **Kritikos-Topic5** conforms to the module's directions, and opens and displays correctly in IDLE.

(e) (**3 points**) Follow the guidelines on the first page of this assignment and incorporate the .py file that answers this topic into your submission.

Your answer:

The code executes correctly when run from your .py file with name **Kritikos-Topic5** using IDLE.