Stony Brook University
CSE512 – Machine Learning – Spring 21
Homework 2, Due: Mar 9 at midnight 11:59pm

This homework contains two questions. The first question is about parameter estimation. In the second question, you will implement logistic regression using stochastic gradient descent (SGD). The maximum number of points is 100 points. For this homework, you should review some material on parameter estimation, logistic regression, stochastic gradient decent, feature normalization, and performance measurement.

1 Question 1 – Parameter estimation (45 points)

This question uses a discrete probability distribution known as the Poisson distribution. A discrete random variable X follows a Poisson distribution with parameter λ if

$$p(X = k|\lambda) = \frac{\lambda^k}{k!}e^{-\lambda} \qquad k \in \{0, 1, 2, \dots\}$$

The Poisson distribution is a useful discrete distribution which can be used to model the number of occurrences of something per unit time. For example, it can be used to model the number of customers arriving at a bank per hour, or the number of calls handled by a telephone operator per minute.

1.1 Question 1.1 – MLE (15 points)

Assume the wait time for calling an Uber car is Poisson distributed (i.i.d) with parameter λ. You used Uber seven times and record the wait times in each trip.

Trip	1	2	3	4	5	6	7
Wait time	4	12	3	5	6	9	17

Let $\mathbf{X} = (X_1, \dots, X_n)$ be a random vector where X_i is the number of delay minutes for your i^{th} trip:

1. *(5 points)* Give the log-likelihood function of $\mathbf{X}$ given λ.
2. *(5 points)* Compute the MLE for λ in the general case.
3. *(5 point)* Compute the MLE for λ using the observed $\mathbf{X}$.

1.2 Question 1.2 – MAP (15 points)

Now let's be Bayesian and put a prior over the parameter λ. You talk to your party-goer friends, who tell you about the expected delays that they experience. You plot the distribution of the expected delays and your extensive experience in statistics tells you that the plot resembles a Gamma distribution pdf. So you believe a good prior distribution for λ may be a Gamma distribution. The Gamma distribution has pdf:

$$p(\lambda|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \lambda > 0. \tag{1}$$

$\Gamma(\alpha)$ is the Gamma function, which is the normalization factor that is introduced to have a proper probability density function (i.e., sum to 1). Do not worry if you don't know the explicit format of $\Gamma(\alpha)$; it is not important for this question. Also, if $\lambda \sim \Gamma(\alpha, \beta)$, then it has mean α/β and the mode is $(\alpha - 1)/\beta$ for $\alpha > 1$. Assume the prior distribution of λ is $\Gamma(\lambda|\alpha, \beta)$.

1. *(8 points)* Compute the posterior distribution over λ. Hint:

$$\lambda^{\sum_{i=1}^n X_i+\alpha-1}e^{-\lambda(n+\beta)}$$

looks like a Gamma distribution! Is the rest of the expression constant with respect to λ? Working out a messy integral can lead to the answer but it is unnecessary.

2. *(7 points)* Derive an analytic expression for the maximum a posterior (MAP) estimate of λ.

1.3 Question 1.3 – Estimator Bias (15 points)

In class, we saw that the maximum likelihood estimator for the variance of a Gaussian distribution:

$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

is biased, and that an unbiased estimator of variance is:

$$\hat{\sigma}^2_{unbiased} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

For the Gaussian distribution, these estimators give similar results for large enough N, and it is unclear whether one estimator is preferable to the other. In this problem, we will explore an example in which the maximum likelihood estimate is dramatically superior to any unbiased estimator.

Suppose we are not quite interested in estimating λ. We are more interested in estimating a quantity that is a *nonlinear* function of λ, namely $\eta = e^{-2\lambda}$. Suppose we want to estimate η from a single observation $X \sim Poisson(\lambda)$.

1. (5pts) Let $\hat{\eta} = e^{-2X}$. Show that $\hat{\eta}$ is the maximum likelihood estimate of η.
2. (5pts) Show that the bias of $\hat{\eta}$ is $e^{-(1-1/e^2)\lambda} - e^{-2\lambda}$. The following Taylor expansion may be useful:

$$e^x = \sum_{k=0}^{\infty}\frac{x^k}{k!}$$

3. (5pts) It turns out that $(-1)^X$ is the *only* unbiased estimate of η. Prove that it is indeed unbiased and briefly explain why this is a bad estimator to use.

2 Question 2 – Logistic Regression (55 points)

In this question, you will implement Logistic Regression using Stochastic Gradient Descent (SGD). Suppose the training data is $\{(X^1, Y^1), \cdots, (X^n, Y^n)\}$, where X^i is a column vector of d dimensions and Y^i is the target label. For a column vector X, let $\bar{X}$ denotes $[X; 1]$, the vector obtained by appending 1 to the end of X. $\boldsymbol{\theta}$ is the set of parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{k-1}$. Logistic regression for k classes assumes the following probability function:

$$P(Y = i|X; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}_i^T \bar{X})}{1 + \sum_{j=1}^{k-1}\exp(\boldsymbol{\theta}_j^T \bar{X})} \text{ for } i = 1, \cdots, k-1, \tag{2}$$

$$P(Y = 0|X; \boldsymbol{\theta}) = \frac{1}{1 + \sum_{j=1}^{k-1}\exp(\boldsymbol{\theta}_j^T \bar{X})}. \tag{3}$$

Logistic regression minimizes the negative average conditional log likelihood:

$$L(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\log(P(Y^i|\bar{X}^i; \boldsymbol{\theta})). \tag{4}$$

Here, $\log(\cdot)$ is the natural logarithm. To minimize this loss function, we can use gradient descent:

$$\boldsymbol{\theta}_c = \boldsymbol{\theta}_c - \eta \frac{\partial L}{\partial \boldsymbol{\theta}_c}, \quad \text{for } c = 1, \ldots, k-1 \tag{5}$$

$$\text{where } \frac{\partial L}{\partial \boldsymbol{\theta}_c} = -\frac{1}{n} \sum_{i=1}^{n} \frac{\partial \log(P(Y^i | \bar{X}^i; \boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_c} \tag{6}$$

This gradient is computed by enumerating over all training data. This gradient can be approximated using a batch of training data. Suppose $\mathcal{B}$ is a subset of $\{1, 2, \cdots, n\}$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_c} \approx -\frac{1}{card(\mathcal{B})} \sum_{i \in \mathcal{B}} \frac{\partial \log(P(Y^i | \bar{X}^i; \boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_c} \tag{7}$$

This leads to the following stochastic gradient descent algorithm:

Algorithm 1 Stochastic gradient descent for Logistic Regression

Inputs: $\{(X^i, Y^i)\}_{i=1}^n$ (for data), m (for batch size), η_{start} (initial learning rate), η_{end} (final learning rate), max_epoch (maximum number of epoches), ϵ (learning rate reduction criterion)
$(i_1, ..., i_n) = permute(1, ..., n)$
Divide $(i_1, ..., i_n)$ into batches of size m or $m+1$ ▷ n might not be divisible by m,
▷ so some batches might have size $m+1$
Randomly initialize $\boldsymbol{\theta}$
$\eta \leftarrow \eta_{start}$
for $epoch = 1, 2, ..., max\_epoch$ **do**
    $\boldsymbol{\theta}^{old} \leftarrow \boldsymbol{\theta}$
    for each batch $\mathcal{B}$ **do**
        Update $\boldsymbol{\theta}$ using Eqs. (5) and (7)
    end for
    if $L(\boldsymbol{\theta}^{old}) - L(\boldsymbol{\theta}) < \epsilon \times L(\boldsymbol{\theta}^{old})$ **then** // not much progress, try smaller learning rate
        $\eta \leftarrow \eta / 10$
    end if
    if $\eta < \eta_{end}$ **then** Break
    end if
end for
Outputs: $\boldsymbol{\theta}$.

2.1 Question 2.1 – Derivation (10 points)

Prove that:

$$\frac{\partial \log(P(Y^i|\bar{X}^i;\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}_c} = (\delta(c = Y^i) - P(c|\bar{X}^i;\boldsymbol{\theta}))\bar{X}^i. \tag{8}$$

where $\delta(c = Y^i)$ is the indicator function which takes a value of 1 if the class c equals the ground truth label Y^i, and 0 otherwise. Use Equation (8) to derive the gradient of the loss with respect to the parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{k-1}$.

2.2 Question 2.2 – Implementation (30 points)

Your task is to implement Logistic Regression with k classes using SGD.

2.2.1 (10 points)

Write a Python function with the signature:

$$[\mathbf{W}, \mathbf{b}] = logreg\_fit(\mathbf{X}, \mathbf{y}, m, eta\_start, eta\_end, epsilon, max\_epoch = 1000)$$

where

Inputs:

- $\mathbf{X}$: a two dimensional Numpy array of size $n \times d$, where n is the number of data points, and d the dimension of the feature vectors.
- $\mathbf{y}$: a Numpy vector of length n. $\mathbf{y}[i]$ is a categorical label corresponding to the data point $\mathbf{X}[i, :]$, $\mathbf{y}[i] \in \{0, 1, \ldots, k-1\}$. You can assume that the number of classes k is the maximum entry of $\mathbf{y}$ plus 1.
- $m, eta\_start, eta\_end, epsilon, max\_epoch$ are scalar parameters for the batch size, the starting learning rate, the ending learning rate, the learning rate reduction criterion, and the maximum number of epochs as described in the algorithm above.

Outputs:

- $\mathbf{W}$: a $(k-1)\times d$ Numpy matrix, $\mathbf{W}[i, :]$ is the learned weight vector $\boldsymbol{\theta}_{i+1}$.
- $\mathbf{b}$: a Numpy vector of length $k-1$ for the biases.

2.2.2 (10 points)

Write a Python function with the following signature.

$$\hat{\mathbf{y}} = logreg\_predict\_class(\mathbf{W}, \mathbf{b}, \mathbf{X})$$

This function takes the learned parameters $\mathbf{W}$ and $\mathbf{b}$ of a logistic regression classifier and test data $\mathbf{X}$ and output the predicted categorical label $\hat{\mathbf{y}}$ for the test data $\mathbf{X}$. See Question 2.2.1 for the expected formats for $\mathbf{W}, \mathbf{b}, \mathbf{X}$. The output $\hat{\mathbf{y}}$ should be a Numpy vector of length n, where n is the number of rows of data matrix $\mathbf{X}$. $\hat{\mathbf{y}}[i]$ is the predicted label for data point $\mathbf{X}[i, :]$.

2.2.3 (10 points)

Write a Python function with the following signature.

$$\mathbf{P} = logreg\_predict\_prob(\mathbf{W}, \mathbf{b}, \mathbf{X})$$

This function takes the learned parameters $\mathbf{W}$ and $\mathbf{b}$ of a logistic regression classifier and test data $\mathbf{X}$ and output the predicted probabilities. $\mathbf{P}$ is a Numpy matrix of size $n \times k$, where $\mathbf{P}[i, j]$ is the probability that $\mathbf{X}[i, :]$ belongs to class j.

2.2.4 Hints

Hint 1: You might want to implement Question 2.2.3 first. Questions 2.2.1 and 2.2.2 should make use of 2.2.3. You have to implement these functions yourself. You cannot use existing implementation in Python, e.g., sklearn.linear_model.LogisticRegression

Hint 2: To compute the probability values, you should use Eqs (2) and (3). To compute a probability value of the form $p = \frac{\exp(a_i)}{\sum_{j=0}^{k-1} \exp(a_j)}$, you can also use the equivalent formula: $p = \frac{\exp(a_i - a)}{\sum_{j=0}^{k-1} \exp(a_j - a)}$, where $a = \max\{a_0, \ldots, a_{k-1}\}$. In theory, the two formulations are the same, but the latter will help you avoid numerical problems in the calculation. Note also that, $\exp(0) = 1$.

Hint 3: Useful numpy functions that you can use for your implementation: $np.dot$, $np.exp$, $np.log$, $np.sum$, $np.mean$.

2.3 Question 2.3 – Running and reporting results (15 points)

In this question, you will train a Logistic Regression model using the functions you implemented in Question 2, in order to predict whether a patient will recover from COVID-19. The COVID-19 dataset was compiled using the data provided by the Stony Brook University Hospital. This dataset contains real data from patients that were diagnosed with COVID-19. For each patient, it includes a number of features, including age, gender, blood pressure and other lab results collected at the admission time. All these are the features can be used in order to forecast if a patient is likely to recover or not. This is a two-class classification problem, and we will refer to the Death Outcome as the positive class.

Step 1

The first step is to load your training and test data that are provided in the files $train.csv$ and $test.csv$ correspondingly. (Tip: To load the data from a csv file, you can use numpy.genfromtxt or csv.reader).

Inspect your data carefully: The first row of each csv file is the csv header. Each following row corresponds to a patient. The columns correspond to the different measurements per patient. The last column indicates if a patient died or not.

Load all the columns of $train.csv$ except the last one to your feature matrix X_{train} and the last column to your labels vector y_{train}. Both X_{train} and y_{train} should be numpy arrays. Convert X_{train} to float using $X_{train} = X_{train}.astype(np.float64)$ and y_{train} to int using $y_{train} = y_{train}.astype(int)$. Do the same for your test data.

Step 2

In general, features can have different types of values and ranges, which can significantly affect the performance of the classifier. As a result, the first step after loading the data is to pre-process them and scale the features in a similar range, using normalization techniques.

To normalize the features, you will use Z-score normalization (or Standardization). This method scales the features by removing the mean μ and dividing by their standard deviation σ:

$$X_{norm} = \frac{X - \mu}{\sigma}$$

You can use $sklearn.preprocessing.StandardScaler$ for this step. In that case, you should use $fit\_transform()$ for X_{train} to fit and scale your training data, and then $transform()$ for X_{test} to scale the test data based on the training mean and standard deviation. This is important, as the test data are seen as unknown during training, so you should not normalize them using knowledge from them.

Step 3

Train a Logistic Regression classifier using your normalized training data X_{train} and y_{train}, and the function $logreg\_fit()$ that you implemented in Question 2.2.1. Use $m = 256$, $eta\_start = 0.01$, $eta\_end = 0.00001$, $epsilon = 0.0001$ and $max\_epoch = 1000$.

2.3.1 Question

(a) Plot the loss function, as given in Eq. (3), against the number of epochs during training. Comment on what you see.

(b) Use the learned parameters (weights $\mathbf{W}$ and biases $\mathbf{b}$) to predict the labels $\hat{\mathbf{y}}_{train}$ for the training set $\mathbf{X}_{train}$ and $\hat{\mathbf{y}}_{test}$ for the test set $\mathbf{X}_{test}$. For each set, calculate and report the following performance metrics:

- The accuracy, i.e. the number of correctly predicted labels over the total number of samples. (You can use $sklearn.metrics.accuracy\_score$).
- The confusion matrix (You can use $sklearn.metrics.confusion\_matrix$).
- The accuracy computed based on the diagonal of the confusion matrix.

(c) Use the learned parameters (weights $\mathbf{W}$ and biases $\mathbf{b}$) to estimate the probabilities of the positive class ($y = 1$) for the test set. Calculate and report the following performance metrics for the test set:

- The average precision (you can use $sklearn.metrics.average\_precision\_score$).
- Plot the precision recall curve (you can use $sklearn.metrics.precision\_recall\_curve$).

2.3.2 Question

(a) Remove the feature normalization and train the classifier again. Plot the loss function against the number of epochs during training as in Question 2.3.1(a). Do you see any difference with scaling and not scaling the data (e.g. in terms of the speed of convergence, the number of epochs, the final performance)?

(b) With feature normalization, try different values for the hyperparameters eta_start, eta_end, max_epoch and m. Comment on what you see when increasing or decreasing these parameters. For example, what impact has a higher learning rate on the training, the number of epochs or the final performance?

(c) Choose your own values for eta_start, eta_end, max_epoch and m and train the classifier again. Plot the loss function against the number of epochs during training as in Question 2.3.1(a). Why did you choose these values?

3 What to submit

You will need to submit both your code and your answers to questions on Blackboard. Put the answer file and your python code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_Barack_Obama*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SUBID_FirstName_LastName.zip.

The answer file should be named: hw2-answers.pdf. You can use Latex if you wish, but it is not compulsory. The first page of the hw2-answers.pdf should be the filled cover page at the end of this homework. The remaining of the answer file should contain:

1. Answers to Question 1
2. Answers to Questions 2.1 and 2.3

Your Python code must be named hw2.py. It should contain the requested functions, exactly as described in Question 2. For automated testing, it should be possible for us to import your functions using 'from hw2 import ...'. You can submit other python/data files if necessary. Your code hw2.py can also include other functions if necessary.

Make sure you follow the instructions carefully. Your will lose points if:

1. You do not attach the cover page. Your answer file is not named hw2-answers.pdf
2. Your functions do not have the exact signatures as instructed
3. Your functions cannot be imported for automatic testing
4. Your functions crash or fail to run

4 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You cannot ask and discuss with students from previous years. You cannot look up the solution online.

Cover page for answers.pdf
CSE512 Spring 2021 - Machine Learning - Homework 2

Your Name:

Solar ID:

NetID email address:

Names of people whom you discussed the homework with: