Homework 4
LOGISTIC REGRESSION, GRADIENT DESCENT

CMU 10-601B: Machine Learning (Fall 2017)
https://piazza.com/cmu/spring2017/10601
OUT: September 19, 2017 11:59 PM
DUE: September 26, 2017 11:59 PM
Authors: Praneet Dutta, Rhea Jain, Mo Li

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the collaboration policy on piazza for more information: https://piazza.com/cmu/fall2017/10601b/home
- **Late Submission Policy:** See the late submission policy here: https://piazza.com/cmu/fall2017/10601b/home
- **Submitting your work:** Homework assignments may consist of multiple parts. For this assignment, you will be asked to submit via Canvas, Gradescope, and Autolab.
  - **Canvas:** We will use an online system called Canvas for short answer and multiple choice questions. You can log in with your Andrew ID and password. (As a reminder, never enter your Andrew password into any website unless you have first checked that the URL starts with "https://" and the domain name ends in ".cmu.edu" – but in this case it's OK since both conditions are met) A link to the Homework 4 Canvas section is provided in Problem 1 below.
  - **Gradescope:** For written problems such as derivations, proofs, or plots we will be using Gradescope. You can access the site here: https://gradescope.com/. **Students are required to sign up using their andrew ids or else they may not receive credit for all of their work.** Each derivation/proof should be completed on a separate page. Submissions can be handwritten, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Upon submission, label each question using the template provided. Regrade requests can be made, however this gives the TA to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.
  - **Autolab:** You can access the 10601 course on autolab by going to https://autolab.andrew.cmu.edu/ Using All programming assignments will be graded automatically on Autolab using Octave 3.8.2 and Python 2.7. You may develop your code in your favorite IDE, but please make sure that it runs as expected on Octave 3.8.2 or Python 2.7 before submitting. The code which you write will be executed remotely against a suite of tests, and the results are used to automatically assign you a grade. To make sure your code executes correctly on our servers, you should avoid using libraries which are not present in the basic Octave install. For Python users, you are encouraged to use the `numpy` package. The version of `numpy` used on Autolab is 1.7.1. The deadline displayed on Autolab may not correspond to the actual deadline for this homework, since we are allowing late submissions (as discussed in the late submission policy on the course site).

Problem 1: Muliple Choice and True/False Questions [24 points]

This problem consists of a set of multiple choice and True/False questions, which is posted on Canvas at URL https://canvas.cmu.edu/courses/2650/assignments

Problem 2: Implementing Logistic Regression [60 points]

For this part of the assignment, you must submit your code on Autolab.

For this question, you will implement a logistic regression classifier using gradient descent for binary classification. You are given a dataset containing passages of movie reviews which has been used for the implementation of Naïve Bayes classifier in HW3 (in fact, the dataset you will be working on is a subset of IMDB Large Movie Review dataset used by Mass et. al, ACL 2011). Your task is to estimate appropriate parameters for the logistic regression model using the training data, and use it to predict reviews from test data, and classify each passage as either a positive or a negative review.

Background

As you have learned in the lecture, logistic regression can be seen as a generalized linear model that can be used to classify data with binary or discrete-valued outcomes by estimating probabilities using a logistic function. A common way to train logistic regression is to maximize conditional likelihood or a posteriori estimate of the training data by gradient descent. X is matrix that contains all n training examples. $\mathbf{x}^{(i)}$ is the ith row of X, corresponding to the ith training example. Each training example has d features, i.e. $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$. Of course, $X(i,j)$ is the jth feature of the ith training example. $\mathbf{y}$ is a vector of length n that contains the labels for the training examples, $y^{(i)}$, which is the ith element in $\mathbf{y}$, is the label for the ith training example.

Recall that the logistic regression hypothesis $h(.)$ is defined as

$$h_\theta(\mathbf{x}^{(i)}) = \sigma(\theta^\top \mathbf{x}^{(i)})$$

and $\sigma(.)$ is the sigmoid function defined as

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

The cost function $\mathcal{F}(\theta)$ we minimize when training logistic regression is a scalar value as follows:

$$\mathcal{F}(\theta) = \sum_{i=1}^{n}[-y^{(i)}\log(h_\theta(\mathbf{x}^{(i)})) - (1-y^{(i)})\log(1-h_\theta(\mathbf{x}^{(i)}))] + \frac{\lambda}{2}\|\theta\|_2^2$$

where the final term is a regularization term.

Recall that the ℓ_2 norm of a vector θ is defined as

$$\|\theta\|_2 = \sqrt{\sum_{j=1}^{d}|\theta_j|^2}$$

The gradient of the cost function is a d-dimensional vector, and its j^{th} element is

$$\frac{\partial \mathcal{F}(\theta)}{\partial \theta_j} = \sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)} + \lambda\theta_j \tag{1}$$

Note that in the code, we add a x_0 feature with value 1 for every training example, thus be careful not to regularize the θ_0 parameter, and so

$$\frac{\partial \mathcal{F}(\theta)}{\partial \theta_0} = \sum_{i=1}^{n}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) \tag{2}$$

Note that now X becomes a matrix with $d+1$ features, and the gradient becomes a $d+1$ dimensional vector.

In the lecture you learned the gradient ascent algorithm to maximize conditional log likelihood in logistic regression. Gradient descent actually works in the same manner if we convert the maximization of conditional log likelihood problem into the problem of minimizing the negative of the conditional log likelihood which is the cost function $\mathcal{F}(\theta)$. As a result, the gradient descent update step is

$$\theta_{new} = \theta_{old} - \alpha * \text{gradient} \tag{3}$$

where α is the step size or learning rate.

Instead of simply running gradient descent for a specific number of iterations, we will stop it after the solution has converged (for safety, we will set the maximum number of iterations). You can detect this convergence when

$$\|\theta_{new} - \theta_{old}\|_2 \leq \epsilon \tag{4}$$

for some small ϵ.

After training the trained logistic regression model, you can then use it to predict whether the passage is a positive or negative review.

Programming Instructions

You will implement some functions for training and testing a logistic regression classifier for this question. You will submit your code online through the CMU autolab system, which will execute it remotely against a suite of tests. Your grade will be automatically determined from the testing results. You can choose to code either in **Python 2.7** or **Octave 3.8.2**.

To reduce the load of Autolab server, you are allowed to submit at most 30 times to Autolab. If you use up all the submissions, we will grade on your last submission regardless of whether it is finished or not. So we suggest you thoroughly test your codes locally and then make the submission.

To get started, you can log into the autolab website (`https://autolab.andrew.cmu.edu`). From there you should see 10-601 in your list of courses. Download the handout for Homework 4 (Options → Download handout) and extract the contents (i.e., by executing `tar xvf hw4.tar` at the command line). In the archive you will find three folders. The `data` folder contains the data files for this problem. The `python` folder contains a `logreg.py` file which contains empty function templates for each of the functions you are asked to implement. Similarly, the `octave` folder contains separate `.m` files for each of the functions that you are asked to implement.

To finish each programming part of this problem, open the `LogReg.py` or the function-specific `.m` template files and complete the function(s) defined inside. When you are ready to submit your solutions, you will create a new tar archive of the files you are submitting the `hw4` directory. Please create the tar archive exactly as detailed below.

If you are submitting Python code:

```
tar cvf hw4_handin.tar LogReg.py
```

If you are submitting Octave code:

```
tar cvf hw4_handin.tar sigmoid.m calculateGradient.m update_weight.m check_conv.m train.m
predict_label.m calculateAccuracy.m logreg.m
```

If you are working in Octave and are missing **any** of the function-specific `.m` files in your tar archive, you will receive zero points.

We have provided all of the data for this assignment as `csv` files in the `data` folder in your handout. You can load the data using the `numpy.genfromtext` function in Python or the `csvread` and `csv2cell` (`io` package) functions in Octave. We have provided a `hw4_script` file for each language to help get you started and load data into your work space (note for `Octave` users: loading the dataset may take a little extra time depending on your computer specs). You should build on the `hw4_script` to test out the functions we are asking you to implement and turn in, but you will not submit the script itself. After loading the data, you should have 4 variables: `XTrain`, `yTrain`, `XTest`, `yTest`, .

- `XTrain` is a $n \times d$ dimensional matrix describing the n documents used for training your logistic regression classifier.
- `yTrain` is a $n \times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i)` is 0 if the i^{th} passage is a negative review and 1 if it is a positive one.
- `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.

Code

Preliminary Note

For this programming assignment, you are not allowed to use any machine learning packages in `python` or `octave` (e.g., `scikit-learn`). For `python` users, you are restricted to use only the `math`, `numpy`, and `scipy` libraries. For `octave` users, you are not allowed to use any external libraries.

Preliminary Note for Python Users

- In `numpy`, to initialize an $n \times 1$ dimensional vector, you should assign `y = numpy.array([y1,y2,...,yn])`, and to check its dimension, we use `y.shape` which yields `(n,)`. You **should not** initialize a vector by means of initializing a matrix, that is `X = np.ones((n,1))`, which will output a matrix with dimension `X.shape = (n,1)`. To initialize a $m \times n$ matrix with all entries equal to 1, we use `np.ones((m,n))`
- Numpy 2-D array is used to represent matrix `Xtrain, Xtest` and Numpy 1-D array is used for `Ytrain, Ytest`
- Attribute axis in `numpy.linalg.norm` is not supported on Autolab.
- `Index` starts from `0`.

Preliminary Note for Octave Users

- Use only native Octave functions. Higher-level functions, such as `pdist2`, are not supported on Autolab.
- `Index` starts from `1`

Implementation

For **Python 2.7**, the Logistic Regression Class in logreg.py consists of the following functions:

- `init(alpha, regLambda, epsilon, maxNumIters)`: Intializes parameters for the logistic regression model. The constructor has already been intialized. Please do not change the values of these parameters.
  `alpha` : `learning rate (step size) for gradient descent, a scalar.`
  `regLambda` : `regularization parameter, a scalar.`

`epsilon` : `convergence threshold value, a scalar`.
`maxNumIters` : `maximum number of iterations for gradient descent, a scalar`.

- `calculateGradient(weight, X, Y, regLambda)`: Calculates the $(d+1)\times 1$ dimensional gradient for the given values of `X, Y, weight matrix and regLambda`.
  `weight :` $(d+1)\times 1$ `dimensional numpy matrix`
  `X :` $n\times(d+1)$ `dimensional numpy matrix`
  `Y :` $n\times 1$ `dimensional numpy matrix`
  `regLambda` : `regularization parameter, a scalar`.
- `sigmoid(z)` : Returns a $n\times 1$ dimensional matrix whose ith element is the value of the sigmoid function of z_i.
  `z :` $n\times 1$ `dimensional numpy matrix`
- `update_weight(X, Y, weight)`: Updates the $(d+1)\times 1$ dimensional weight matrix
  `weight :` $(d+1)\times 1$ `dimensional numpy matrix`
  `X :` $n\times(d+1)$ `dimensional numpy matrix`
  `Y :` $n\times 1$ `dimensional numpy matrix`
- `check_conv(weight, new_weight, epsilon)`: Checks if the weights have converged for gradient descent, returns True or False.
  `weight :` $(d+1)\times 1$ `dimensional numpy matrix`
  `new_weight :`$(d+1)\times 1$ `dimensional numpy matrix`
  `epsilon` : `convergence threshold, a scalar`.
- `train(X, Y)`: Trains the logistic regression model using training data and returns the weight trained as a $(d+1)\times 1$ dimensional numpy matrix. Part of this has been completed for you.
  `X :` $n\times d$ `dimensional numpy matrix`
  `Y :` $n\times 1$ `dimensional numpy matrix`
- `predict_label(X, weight)`: Predicts the labels for test data and return a $n\times 1$ numpy matrix of integer labels whose ith element is the predicted label for the ith training example.
  `weight :` $(d+1)\times 1$ `dimensional numpy matrix`
  `X :` $n\times d$ `dimensional numpy matrix`
- `calculateAccuracy(Y_predict, YTest)`: Computes the prediction accuracy for test data and returns the prediction accuracy on the test data as a scalar value between 0 to 100.
  `Y_predict :` $n\times 1$ `dimensional numpy matrix`
  `Y_test :` $n\times 1$ `dimensional numpy matrix`
- `LogReg.py` : This serves as the main function which loads the data and initializes the parameters. The above functions that you are required to complete are being called in LogReg.py. No need to modify this script.

For **Octave 3.8**, please complete all of the functions in the corresponding .m file in the following order.

- `calculateGradient(weight, X, Y, regLambda)`: Calculates the $(d+1)\times 1$ dimensional gradient for the given values of `X, Y, weight vector and regLambda`.

  `X :` $n\times(d+1)$ `matrix`
  `Y :` $n\times 1$ `matrix`
  `weight :` $(d+1)\times 1$ `matrix`
  `regLambda` : `regularization parameter, a scalar`
- `sigmoid(z)`: Returns a $n\times 1$ matrix whose ith element is the value of the sigmoid function of z_i.
- `update_weight(X, Y, weight, alpha, regLambda)`: Updates the $(d+1)\times 1$ weight matrix.
  `X :` $n\times(d+1)$ `matrix`
  `Y :` $n\times 1$ `matrix`
  `weight :` $(d+1)\times 1$ `matrix`

`regLambda` : `regularization parameter, a scalar`
`alpha` : `learning rate or step size for gradient descent, a scalar`.

- `check_conv(weight, new_weight, epsilon)`: Checks if the weights have converged for gradient descent, returns true if converge or false if not converge.
  `weight :` $(d+1)\times 1$ `matrix`
  `new_weight :` $(d+1)\times 1$ `matrix`
  `epsilon` : `convergence threshold, a scalar`
- `train(X, Y, maxNumIters, alpha, regLambda, epsilon)`: Train the logistic regression model using the training data within maxNumIters (the maximum number of iterations for gradient descent). Return the weights trained with logistic regression model as a $(d+1)\times 1$ matrix.
  `X :` $n\times d$ `matrix`
  `Y :` $n\times 1$ `matrix`
  `maxNumIters` : `maximum number of iterations for gradient descent, a scalar`.
  `alpha` : `learning rate or step size for gradient descent, a scalar`.
  `regLambda` : `regularization parameter, a scalar`.
  `epsilon` : `convergence threshold value, a scalar`.
- `predict_label(XTest, weight_train)`: Predicts the labels for test data using the learned weight vector. Return the predicted labels in a $n\times 1$ matrix.
  `XTest :` $n\times d$ `matrix`
  `weight_train :` $(d+1)\times 1$ `matrix`
- `calculateAccuracy(Y_predict, yTest)`: Computes the prediction accuracy for test data. Return a scalar value in the range of $0-100$.
  `Y_predict :` n`×1 matrix`
  `Y_test :` n`×1 matrix`
- `logreg.m`: This serves as the main function which loads the data and initializes the parameters. The above functions that you are required to complete are being called in logreg.m. No need to modify this script.

Training Logistic Regression

1. [**4 points**] Complete the function `sigmoid(z)`. Given a $n\times 1$ matrix , the sigmoid function normalizes the values of the vector between 0 and 1 and should return a $n\times 1$ matrix.
2. [**12 points**] Complete the function `calculateGradient(weight, X, Y, regLambda)` according to equation (1) and (2) in the background section. The gradient returned will be a $d\times 1$ matrix .
3. [**6 points**] Complete the function `update_weight(X, Y)` (for Python user) or `update_weight(X, Y, weight, alpha, regLambda)` (for Octave user). You will make use of the gradient computed and the learning rate parameter to implement this function (refer to equation (3)).
4. [**6 points**] Complete the function `check_conv(weight, new_weight, epsilon)` (for Python user) or `check_conv(weight, new_weight, tol)` (for Octave user). Refer to equation (4). Return True if the weights have converged, i.e. the difference between the updated weights and the previous weights is below the given threshold.
5. [**14 points**] Complete the function `train(X, Y)` (for Python user) or `train(X, Y, maxNumIters, alpha, regLambda, epsilon)` (for Octave user). Part of the initialization has already been done for you. This should call the functions implemented in Question 3 and 4. Update the weights at every iteration. The training process will finish if the weights converge or the number of iterations exceeds the maximum iteration numbers.

Testing Logistic Regression

6. [**12 points**] Complete the function `predict_label(X)` (for Python user) or `predict_label(XTest, weight_train)` (for Octave user) which predict the labels for test data after training the logistic regression model. Return the predicted labels(either 0 or 1) in a $m \times 1$ matrix .

7. [**6 points**] Complete the function `calculateAccuracy(Y_predict, Y_test)` (for Python user) or `calculateAccuracy(Y_predict, yTest)` (for Octave user). Given the true labels `yTest` and predicted labels from predict_label function in above question, return the percentage of the prediction accuracy on a scale of 0 to 100.

Problem 3: Generative and Discriminative Classifiers [16 pts]

For this part of the assignment, you must submit your answer as PDF on Gradescope. Submissions can be handwritten, but should be labeled and clearly legible. Else, submissions can be written in LaTeX. Unlike in homework 3, you do not need to follow a pre-designed format in order to submit your work. Upon submission, label each question using the template provided by Gradescope.

In class, we learned that when Y takes Boolean values and X is a n dimensional vector of $X = \langle X_1 ... X_n \rangle$ continuous variables, where each $X_i, i = 1, \cdots, n$ is distributed normally (i.e. $P(X_i|Y = y_k) = N(\mu_{ik}, \sigma_i)$), then logistic regression is the discriminative equivalent of Naive Bayes under the Naive Bayes assumptions

$$P(Y = 1 \mid X) = \frac{1}{1 + exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

and

$$P(Y = 0 \mid X) = \frac{exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

1. Let's consider a case where Y is Boolean and $X = \langle X_1 ... X_n \rangle$ is a vector of Boolean variables. Prove that for this case also logistic regression is the discriminative equivalent of Naive Bayes under the Naive Bayes assumptions.

   Hints

   (a) Please refer to Mitchell's book on Naive Bayes and logistic regression (Exercise 3).

   (b) Simple notation that will help. Since the X_i are Boolean variables, you need only one parameter to define $P(X_i \mid Y = y_k)$. Define $\theta_{i1} \equiv P(X_i = 1 \mid Y = 1)$, in which case $P(X_i = 0 \mid Y = 1) =$ (1-θ_{i1}). Similarly, use θ_{i0} to denote $P(X_i = 1|Y = 0)$.

   (c) Notice with the above notation you can represent $P(Xi \mid Y = 1)$ as follows.

   $$P(X_i \mid Y = 1) = \theta_{i1}^{(X_i)} (1 - \theta_{i1})^{(1-X_i)}$$

   Note when $X_i = 1$ the second term is equal to 1 because its exponent is zero. Similarly, when X_i = 0 the first term is equal to 1 because its exponent is zero.