Homework #3

CS 260: Machine Learning Algorithms
Prof. Ameet Talwalkar
Due: 10/29/15, 8am
Please abide by the Academic Integrity Policy

Gradient Descent and Newton's Method

In this problem, you will implement (unregularized/regularized) logistic regression for binary classification problems using two different types of optimization approaches, namely, **batch gradient descent** and **Newton's method**. Two data sets are given; one of which is text data from which you will learn to construct features. For each of the problems, you need to *report your results on both of the datasets.* Please note that there are 9 problems in total. Also, *cross validation is NOT needed* for this programming assignment. *For any plot you make, you MUST at least provide title, x-label, y-label and legend (if there is more than one curves).* Please read submission instructions carefully before submitting your report and source codes.

1 Data

Ionosphere This dataset contains 351 instances and each instance has 34 attributes (features). All feature values are continuous and the last column is the class label ("bad" = b = 1, "good" = g = 0). Your goal is to predict the correct label, which is either "b" or "g". We already divided the dataset into training and test sets (`iono_train.dat` and `iono_test.dat`). Please use the training set to build your model and use the test set to evaluate your model. For more details about Ionosphere dataset, please refer to UCI website: `https://archive.ics.uci.edu/ml/datasets/Ionosphere`.

EmailSpam This dataset contains 941 instances and each of them is labeled as either a spam or ham (not spam) email. Each data instance is the text which is the content of the email (subject and body), and your goal is to classify each email as either a spam or a ham. We have already divided this dataset into training and test datasets, and each of these datasets is stored in two distinct folders (one corresponding to ham and the other corresponding the spam). In other words, to build your model, you need to iterate through all the text files within `/train/spam` and `/train/ham`, and to test your model you need to iterate through `/test/spam` and `/test/ham`.

2 Feature Representation

An essential part of machine learning is to build the feature representation for raw input data, which is often unstructured. Once we construct the features, each data instance $\mathbf{x}_i$ can be represented as:

$$\mathbf{x}_i = (x_{i1}, ..., x_{id})$$

where x_{ij} denotes the jth feature for ith instance.

Algorithm 1 Pseudocode for generating bag-of-word features from text

```
1: Initialize feature vector bg_feature = [0,0,...,0]
2: for token in text.tokenize() do
3:     if token in dict then
4:         token_idx = getIndex(dict, token)
5:         bg_feature[token_idx]++
6:     else
7:         continue
8:     end if
9: end for
10: return bg_feature
```

Ionosphere The dataset is well-formatted, so you can directly use the raw feature values to build your model. Please do not do any normalization. Also, since there is no categorical attribute, converting to binary features (which you did for HW #1) is not needed.

EmailSpam Since each data instance is text, you need to find a way converting it into a feature vector. In this homework, you will use the "Bag-of-Words" representation. More specifically, you will convert the text into a feature vector in which each entry of the vector is the count of words occur in that text. You will be provided with the predefined dictionary (`dic.dat`), and you should only consider words that appear in that dictionary and ignore all others.[1] Below is the pseudocode for generating bag-of-word features from text. For tokenization, please tokenize the string only using **whitespace and these three delimiters: '.,?'**. See below for an example:[2]

Email: *hey, i have a better offer for you, offer. better than all other spam filters. Do you like accepting offer?*

Pre-defined Dictionary: *[add, better, email, filters, hey, offer,like, spam,special]*

Bag-of-words feature vector: $[0, 2, 0, 1, 1, 3, 1, 1, 0]$

(Q1). After converting all training data into bag-of-words feature vectors, what are the 3 words that occur most frequently? Report the results using this format:

{(word1: # of occurrences), (word2: # of occurrences), (word3: # of occurrences)}

3 Implementation

The regularized cross-entropy function can be written as:

$$\varepsilon(\mathbf{w}, b) = -\sum_{i=1}^{n}\{y_i \log \sigma(b + \mathbf{w}^\top \mathbf{x}_i) + (1 - y_n) \log[1 - \sigma(b + \mathbf{w}^\top \mathbf{x}_i)\} + \lambda \|\mathbf{w}\|_2^2$$

where $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$, and $\sigma(\cdot)$ is the sigmoid function. λ is regularization coefficient and w_0 is bias parameter. Note that we don't regularize bias term b.

Stopping Criteria. For both algorithms, run for 50 iterations.

[1] In practice, you need to define the dictionary based on the specific requirements of your application. This process usually involves removing "stop words", "stemming", etc. In this homework, you don't need to consider these issues.

[2] `http://en.wikipedia.org/wiki/Bag-of-words_model` also provides a simple example.

Step size. For gradient method, you will use a fixed step size. Recall that Newton's method does not require a step size.

Initialization. For batch gradient descent, initialize the weight $\mathbf{w}$ to 0, and b to 0.1. For Newton's method, set initial weights to the ones we got from batch gradient descent after 5 iterations (when $\lambda = 0.05, \eta = 0.01$). (Please note that Newton's method may diverge if initialization is not proper).

Extreme Condition. It is possible that when $\sigma(b + \mathbf{w}^T\mathbf{x})$ approaches 0, $\log(\sigma(b + \mathbf{w}^T\mathbf{x}))$ goes to -infinity. In order to prevent such case, please bound the value $\sigma(b + \mathbf{w}^T\mathbf{x})$ using small constant value, $1e - 16$. You can use the following logic in your code:

```
$tmp = \sigma(b + \mathbf{w}^T\mathbf{x})$
if $tmp < 1e - 16$ then
$tmp = 1e - 16$
```

3.1 Batch Gradient Descent

(Q2) Please write down the updating equation for $\mathbf{w}$ and b, for both unregularized logistic regression and regularized logistic regression. In particular, at iteration t using data points $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, where $\mathbf{x}_i = [x_{i1}, ..., x_{id}]$, and $y_i \in \{0, 1\}$ is the label, how do we compute $\mathbf{w}^{t+1}$ and b^{t+1} from $\mathbf{w}^t$ and b^t?

(Q3) For step sizes $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5\}$ and **without regularization**, implement Batch gradient descent (without cross-validation, use the whole training data for the gradient calculation).

(a) Plot the cross-entropy function value with respect to the number of steps ($T = [1, ..., 50]$) for the training data for each step size. Note: you need to make two plots, one for each dataset.

(b) Report the L_2 norm of vector $\mathbf{w}$ after 50 iterations for each step size η_i (fill in the table below)

L_2 norm (without regularization)	0.001	0.01	0.05	0.1	0.5
Ionosphere					
EmailSpam					

(Q4) For step sizes $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5\}$ and with regularization coefficients $\lambda = \{0, 0.05, 0.1, 0.15, ..., 0.5\}$, do the following:

(a) Given $\lambda = 0.1$, plot the cross-entropy function value with respect to the number of steps ($T = [1, ..., 50]$) for the training data for each step size using different step sizes. Note: you need to make two plots, one for each dataset.

(b) Given $\eta = 0.01$, report the L_2 norm of vector $\mathbf{w}$ after 50 iterations for each regularization coefficient λ_i (fill in the table below).

(c) Plot the cross entropy function value at $T = 50$ for different regularization coefficients, for both the training and test data. The x-axis will be the regularization coefficient and y-axis will be the cross entropy function value after 50 iterations. Each plot should contain two curves, and you should make 2 (two data sets) $* \times 5$ (five different step sizes) = 10 plots.

L_2 norm (with regularization, $\eta = 0.01$)	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Ionosphere											
EmailSpam											

3.2 Newton's method

Optimization also can be done using the 2nd order technique called "Newton's method". We define $\mathbb{H}$ as the Hessian matrix and $\nabla\varepsilon_t$ as the gradient of objective function at iteration t.

(Q5) Using the notation above, write down the updating equation for $\mathbf{w}$ and b at time t, for both unregularized logistic regression and regularized logistic regression.

(Q6) Implement Newton's method for logistic regression without regularization, and run it for 50 iterations on both of datasets.

(a) Plot the cross-entropy function value with respect to the number of steps ($T = [1, ..., 50]$) for the training data. Note: you need to make two plots, one for each dataset.

(b) Report the L_2 norm of vector $\mathbf{w}$ after 50 iterations.

(c) Report the cross-entropy function value for the test data.

(Q7) Repeat (Q6) for the regularized case using $\lambda = \{0, 0.05, 0.1, 0.15, ..., 0.5\}$ (report values for each λ setting).

3.3 Analysis and Comparison of Gradient Descent and Newton's Method

(Q8) Briefly (in no more than 4 sentences) explain your results from (Q3) and (Q4). You should discuss the rate of convergence as a function η, the change in magnitude of $\mathbf{w}$ as a function of λ, the value of the cross entropy function for different values of λ and η, and any other interesting trends you have observed.

(Q9) Briefly (in no more than 4 sentences) discuss the differences between gradient descent and Newton's method based on the results from (Q4) and (Q7). In particular, discuss the difference between gradient descent and Newton's method in terms of convergence and computation time.

Submission

Please provide the following as part of your submission:

- Provide your answers for all of the problems **in hard copy**. If you are printing it as black-white, please make sure that you are using different line styles and colors for each curve in your plot, if there is more than one curve. The papers need to be stapled and submitted at the beginning of class on the due date.
- Please put all of your code in a single folder named `[lastname]_[firstname]_hw3`, and submit a single `.zip` file containing this folder called `[lastname]_[firstname]_hw3.zip`. The only acceptable languages are MATLAB and Octave.
- You MUST include the main function called `CS260_hw3.m` in your root folder in your zip file. After running this main file, your program should be able to generate all of the results needed for this programming assignment, either as plots or console outputs. You can have multiple files (i.e your sub-functions), however, we require that once we unzip your folder and execute your main file, your program should execute correctly. Please double-check your program before submitting.
- Please submit this zip file by the due date by following the instructions on this form.
- You are encouraged to collaborate, but collaboration must be limited to discussion only and you need to write down / implement your solution on your own. You also need to list with whom you have discussed the HW problems.