# CSE6250: Big Data Analytics in Healthcare
# Homework 2

Jeff McGehee

February 12, 2017

## 1 Logistic Regression

### 1.1 Batch Gradient Descent

**a.** Derive the gradient of the negative log-likelihood in terms of $\mathbf{w}$ for this setting.
Calculate the partial derivative of the negative log-likelihood wrt $\mathbf{w}$

$$\frac{\partial}{\partial \mathbf{w}} NLL(D, \mathbf{w}) = \left( y \frac{1}{\sigma(t)} - (1-y) \frac{1}{1-\sigma(t)} \right) \frac{\partial}{\partial \mathbf{w}} \sigma(t) \tag{1}$$

Calculate the partial derivative of the sigmoid function wrt $t$

$$\frac{\partial}{\partial t} \sigma(t) = \sigma(t)\big(1 - \sigma(t)\big) \tag{2}$$

Substitute (2) into (1)

$$\frac{\partial}{\partial \mathbf{w}} NLL(D, \mathbf{w}) = \left( y \frac{1}{\sigma(t)} - (1-y) \frac{1}{1-\sigma(t)} \right) \frac{\partial}{\partial \mathbf{w}} t \cdot \sigma(t)\big(1 - \sigma(t)\big) \tag{3}$$

Simplifying

$$\big(y - \sigma(t)\big) \frac{\partial}{\partial \mathbf{w}} t \tag{4}$$

### 1.2 Stochastic Gradient Descent

**a.** Show the log likelihood, $l$, of a single $(\mathbf{x}_t, y_t)$ pair.

$$l(D_t, \mathbf{w}) = (1 - y_t) \log(1 - \sigma(\mathbf{w}^{\mathbf{T}} \mathbf{x_t})) + y_t \cdot \log \sigma(\mathbf{w}^{\mathbf{T}} \mathbf{x_t}) \tag{5}$$

**b.** Show how to update the coefficient vector $\mathbf{w}_t$ when you get a patient feature vector $\mathbf{x}_t$ and physician feedback label $y_t$ at time $t$ using $\mathbf{w}_{t-1}$ (assume learning rate $\eta$ is given).

Using (4) and recalling $t = \mathbf{w}^{\mathbf{T}} x_t$

$$\mathbf{w_t} = \mathbf{w_{t-1}} + \eta\big(y_t - \sigma(\mathbf{w}^{\mathbf{T}} x_t)\big)x_t \tag{6}$$

**c.** What is the time complexity of the update rule from **b** if $\mathbf{x}_t$ is very sparse?

$$O(n)$$

**d.** Briefly explain the consequence of using a very large $\eta$ and very small $\eta$. Very large $\eta$ has the risk of overshooting the minima, while a very small $\eta$ will converge extremely slowly.

**e.** Show how to update $\mathbf{w}_t$ under the penalty of L2 norm regularization. In other words, update $\mathbf{w}_t$ according to $l - \mu\|\mathbf{w}\|_2^2$, where $\mu$ is a constant. What's the time complexity?

$$\mathbf{w_t} = \mathbf{w_{t-1}} + \Bigg(\eta\big(y_t - \sigma(\mathbf{w}^{\mathbf{T}} x_t)x_t\big) - \eta\mu\mathbf{w}^{\mathbf{T}}\Bigg) \tag{7}$$

time complexity is $O(n)$

**f.** When you use L2 norm, you will find each time you get a new $(\mathbf{x}_t, y_t)$ you need to update every element of vector $\mathbf{w}_t$ even if $\mathbf{x}_t$ has very few non-zero elements. Write the pseudo-code on how to update $\mathbf{w}_t$ lazily.

## 2 Programming

### 2.1 Descriptive Statistics [10 points]

**a.** Complete *hive/event_statistics.hql* for computing statistics required in the question. Please be aware that **you are not allowed to change the filename.**

**b.** Use *events.csv* and *mortality.csv* provided in **data** as input and fill Table 1 with actual values. We only need the top 5 codes for common diagnoses, labs and medications. Their respective counts are not required.

### 2.2 Transform data

**Deliverable: pig/etl.pig and pig/utils.py [20 points]**

### 2.3 SGD Logistic Regression

In this question, you are going to implement your own Logistic Regression classifier in python using the equations you derived in question **1.2.e**. To help you get started, we have provided a skeleton code. You will find the relevant code files in *lr* folder. You will train and test a classifier by running

| Metric | Deceased patients | Alive patients |
|---|---|---|
| Event Count<br>1. Average Event Count<br>2. Max Event Count<br>3. Min Event Count | | |
| Encounter Count<br>1. Average Encounter Count<br>2. Max Encounter Count<br>3. Min Encounter Count | | |
| Record Length<br>1. Average Record Length<br>2. Max Record Length<br>3. Min Record Length | | |
| Common Diagnosis | | |
| Common Laboratory Test | | |
| Common Medication | | |

Table 1: Descriptive statistics for alive and dead patients

1. cat path/to/train/data | python train.py -f <number of features>

2. cat path/to/test/data | python test.py

Training and testing data of this problem will be output from previous Pig ETL problem.

To better understand the performance of your classifier, you will need to use standard metrics like AUC. Similarly with Homework1, we provide *code/environment.yml* which contains a list of libraries needed to set environment for this homework. You can use it to create a copy of conda 'environment' (http://conda.pydata.org/docs/using/envs.html#use-environment-from-file). If you already have your own Python development environment (it should be Python 2.7), please refer to this file to find necessary libraries. It will help you install necessary modules for drawing ROC curve. The environment is tested in Vagrant VM. You may need to modify it if you want to install it somewhere else. Remember to restart the terminal after installation.

**a.** Update the **lrsgd.py** file. You are allowed to add extra methods, but please make sure the existing method names and parameters remain unchanged. Use standard modules of Python 2.7 only, as we will not guarantee the availability of any third party modules while testing your code. [10 points]

**b.** Show the ROC curve generated by test.py in this writing report for different learning rates $\eta$ and regularization parameters $\mu$ combination and briefly explain the result. [5 points]

**c.** [Extra 10 points] Implement using the result of question **1.2.f**, and show the speed up. Test efficiency of your approach using larger data set *training.data*, which has 5675 possible distinct features. Save the code in a new file lrsgd_fast.py. We will test whether your code can finish witin reasonable amount of time and correctness of trained model. The training and testing data set can be downloaded from:

**Deliverable: lr/lrsgd.py and optional lr/lrsgd_fast.py** [**15 points**]

## 2.4 Hadoop

**c.** Compare the performance with that of previous problem and briefly analyze why the difference. [5 points]

## 2.5 Zeppelin

**Deliverable:** `zeppelin\bdh_hw2_zeppelin.json`[**10 points**]