COMP5434 Big Data Computing

Project 1: Implement Logistic Regression
with Gradient Descent Backpropagation

February 8, 2018

Introduction

In Tutorial 1, we got start with Python and TensorFlow basics, and in LAB 1, we are going to have hands-on practices on Pyhton and TensorFlow. To consolidate the theoretical knowledge given in the lecture, in Project 1, we will implement the logistic regression with gradient descent backpropagation. Project 1 is an individual student project that student should work independently. In this project, basic programming skills are required, otherwise you may have difficulties in finishing it.

Logistic Regression

1. Key Points Recap

Logistic regression is the linear classification algorithm for binary classification problems, which can be expressed very much like linear regression. Input values, $\mathbf{x} = x_1, x_2, \cdots$, are combined linearly with parameters, $\boldsymbol{\theta} = \theta_1, \theta_2, \cdots$, to predict an output value, y. A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.

The logistic function is defined as,

$$h_\theta(x) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}, \text{ where } \boldsymbol{\theta}^T \mathbf{x} = \theta_0 + \sum_{i=1}^{m} \theta_j x_j. \tag{1}$$

We define the cost function as,

$$\mathcal{J}(\theta) = \mathcal{C}(h_\theta(x), y) = -\frac{1}{m} \left[\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i)) \right]. \tag{2}$$

The partial derivative of $\mathcal{J}(\theta)$ for x_j can be expressed as,

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = -\frac{1}{m} \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i. \tag{3}$$

Please refer to the detailed derivation in Appendix.
Gradient descent algorithm for logistic algorithm is calculated as in Eq. 2, which minimizes $\mathcal{J}(\theta)$ with parameter θ_j. Repeat the step for many times until it converges.

$$\theta_j \leftarrow \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i. \tag{4}$$

2. Logistic Regression for Binary Classification

Inputs: suppose you are the administrator of a university department and you want to determine each applicant's chance of admission based on their results on two exams. You have historical data from previous applicants that you can use as a training set for logistic regression. For each training example, you have the

applicants scores on two exams and the admissions decision.

Problem: build a logistic regression model to predict whether a student gets admitted into a university (i.e., build a classification model that estimates an applicants probability of admission based the scores from those two exams). If the probability $h_\theta(x) > 0.5$, the predicting class is 1, otherwise 0, if $h_\theta(x) < 0.5$.

Algorithm 1 shows logistic regression for binary classification step-by-step.

Algorithm 1: Logistic Regression for Binary Classification

- • Train a logistic regression classifier $h_\theta(x)$ for each class i to predict the probability that $y = i$ (i=0 and 1).
- **for** *iteration* k **do**
  - **for** *parameter index* j **do**
    - $\theta_j \leftarrow \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$
  - **end**
- **end**
- • On a new inputs, $\mathbf{x}' = x'_1, x'_2, \cdots$, pick the class that maximizes the hypothesis $\max_i h_\theta(x')$.
- $h_\theta(x') = \frac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{x}'}}$, where $\boldsymbol{\theta}^T \mathbf{x}' = \theta_0 + \sum_{i=1}^{m} \theta_j x'_j$.
- $y' = \begin{cases} 0, \text{ if } h_\theta(x') < 0.5; \\ 1, \text{ if } h_\theta(x') > 0.5. \end{cases}$

Project Description

1. Dataset Introduction

In this project, a dataset, called "*Pima Indians Diabetes Dataset*", is provided to you, which involves predicting the onset of diabetes within 5 years in Pima Indians given basic medical details. The dataset is owned by National Institute of Diabetes and Digestive and Kidney Diseases, and donated by Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu), Research Center, RMI Group Leader Applied Physics Laboratory, The Johns Hopkins University, Johns Hopkins Road Laurel, MD 20707 (301) 953-6231.

The dataset has 768 rows and 9 columns. All of the values in the file are numeric, specifically floating point values. Figure 1 illustrates a small number of samples of the first few rows of the dataset and Table 1 lists the meanings of each segment in each line of the dataset.

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
...
```

Figure 1: Screenshot of "*Pima Indians Diabetes Dataset*" samples.

Table 1: Meanings of each segment of the "*Pima Indians Diabetes Dataset*".

Index	Meanings of each segment of the dataset
1.	Number of times pregnant; 怀孕次数
2.	Plasma glucose concentration a 2 hours in an oral glucose tolerance test; 血糖浓度
3.	Diastolic blood pressure (mm Hg); 舒张压
4.	Triceps skin fold thickness (mm); 肱三头肌皮褶厚度
5.	2-Hour serum insulin (mu U/ml); 血清胰岛素
6.	Body mass index (weight in kg/(height in m)^2); BMI
7.	Diabetes pedigree function; 糖尿病家系函数
8.	Age (years);
9.	Class variable (0 or 1).

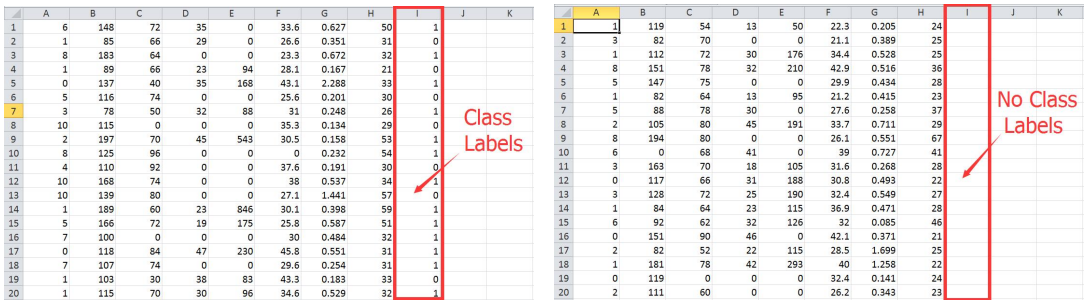
For more information about the dataset, please refer to *UCI Machine Learning Repository* at `https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes`.

2. Project Tasks

As we can see from the data structure of the "*Pima Indians Diabetes Dataset*", it can be formulated as a binary classification problem, where the prediction is either 0 (no diabetes) or 1 (diabetes).

In this project, we make use of the input data from index 1 to index 8 for predicting whether the patients have the disease of diabetes or not (in index 9). The dataset, "diabetes_dataset.csv", is split to training set, $\mathbf{x}$, and testing set, $\mathbf{x}'$, with the proportion 8:2, which means train your logistic regression model with 80% of the total samples, then test your model for the classification results using the remaining 20% samples. In training the logistic regression model, you have to update the value of parameters, $\boldsymbol{\theta}$, for many iterations until they reach the global minimum. You will be given a small set of testing samples, "test_samples.csv", without class labels (0 and 1) to test your model's performance.

In brief, the difference between "diabetes_dataset.csv" and "test_samples.csv" is that "diabetes_dataset.csv" contains the segment information of class labels while "test_samples.csv" does not, which is illustrated in Figure 2.



(a) "diabetes_dataset.csv" with class labels (b) "test_samples.csv" without class labels

Figure 2: Comparison of segment information between "diabetes_dataset.csv" and "test_samples.csv".

The classification results should be included in your project report, which contains two parts:
(1). The statistical classification error rate, $\mathcal{R}$, in percentage (%) of your model tested on the split testing set (20% of the total samples from "diabetes_dataset.csv") by comparing your predicting results with the given class labels.

Assume y and y' are the predicting class and the true class label, the absolute difference between y and y', denoted by Δ, should be $\Delta = |y - y'|$. We have,

$$\Delta = |y - y'| = \begin{cases} 0, & \text{correct classification,} \\ 1, & \text{wrong classification.} \end{cases} \tag{5}$$

Hence the total number of correct classification, $\mathcal{N}_c$, and wrong classification, $\mathcal{N}_w$, go to,

$$\begin{aligned} \mathcal{N}_c &= \#\{\Delta = |y - y'| = 0\}, \\ \mathcal{N}_w &= \#\{\Delta = |y - y'| = 1\}. \end{aligned} \tag{6}$$

The classification error rate, $\mathcal{R}$, can be calculated as,

$$\mathcal{R}(\%) = \frac{\mathcal{N}_w}{\mathcal{N}_c + \mathcal{N}_w} \times 100\%. \tag{7}$$

(2). The list of actual predicting classes in 0 and 1 of your model tested on the dataset given by "test_samples.csv". For instance, the output should be a series of 0 and 1, e.g., $[0, 0, 1, 0, 1, 1, \cdots, 0, 1, 1]$.

Requirement

1. You must work independently without seeking cooperation from others, but you can look for information and examples from Google and other sources;
2. You can use any programming language that you are skilled with, e.g., C, C++, Matlab, and Java, not only restricted to Python, to finish this project;
3. You must implement the algorithm step-by-step on your own. It is not allowed to use the existing libraries or functions from other tools or packages;
4. You can compare your classification results with Scikit-Learn and the example code is given below, **but it is not compulsory**.

```
from sklearn import linear_model
logreg = linear_model.LogisticRegression()
logreg.fit(y, y')
```

Project Report Writing

1. Use the report template to finish the project report and make some changes in the format and structure of the template if necessary;
2. Write the project report in an academic way. Generally speaking, an academic report should include several parts, *Abstract*, *Introduction*, *Problem Definition*, *Methodology*, *Experiments*, *Conclusion*, *Reference* and *Appendix*;
3. Well organize your project report following a clear logical order;
4. Present experiments in a nice way. Visualized (figures) and numerical (tables of statistic) results are highly appreciated. Compare the results with different

parameter settings if necessary. Say for example, you can compare the results using different numbers of training iterations, k, and different learning rate, α;
5. Plot and illustrate your system model's architecture if applicable;
6. Enclose you code in the last *Appendix* part with brief description and comments on the function of different parts of your code. Similarity of codes will be checked, and similar codes from different students will be penalized in the project scores.

Project Report Submission

The due date of submitting Project 1 Report is **8th March, 2018** and please submit the **hardcopy** with your name and student number on it to Dr. ZHAO during the lecture.

Appendix

Derivation: calculating the derivative of the cost function $\mathcal{J}(\theta)$.

$$\mathcal{J}(\theta) = \mathcal{C}(h_\theta(x), y) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))\right]. \quad (8)$$

Let f and g be differentiable at x with $g(x) \neq 0$. Then f and g is differentiable at the point x and based on the Quotient Rule for derivatives,

$$\left[\frac{f(x)}{g(x)}\right]' = \frac{g(x)f'(x) - f(x)g'(x)}{[g(x)]^2}. \quad (9)$$

Hence, the derivative of $h_\theta(x)$ can be calculated as,

$$\begin{aligned} h'_\theta(x) &= \left[\frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}\right]' \\ &= \frac{e^{-\boldsymbol{\theta}^T x}}{(1+e^{-\boldsymbol{\theta}^T\mathbf{x}})^2} \\ &= \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}) \\ &= h_\theta(x)(1-h_\theta(x)). \end{aligned} \quad (10)$$

According to the Chain Rule for calculating the derivative of polynomial, we have

$$\begin{aligned} J'(\theta) &= -\frac{1}{m}\sum_{i=1}^{m} \frac{y^i}{h_\theta(x^i)}\left(h_\theta(x^i)\right)' + \frac{1-y^i}{1-h_\theta(x^i)}\left(1-h_\theta(x^i)\right)' \\ &= -\frac{1}{m}\sum_{i=1}^{m}\left[\frac{y^i}{h_\theta(x^i)} - \frac{1-y^i}{1-h_\theta(x^i)}\right] h'_\theta(x^i). \end{aligned} \quad (11)$$

Here we assume the logarithm $\log x$ is the natural logarithm $\log_e x$ such that $(\log x)' = \frac{1}{x}$. Putting Eq. 10 and Eq. 11 together, we get

$$\begin{aligned} J'(\theta) &= -\frac{1}{m}\sum_{i=1}^{m} \frac{y^i h(x^i)(1-h(x^i))}{h(x^i)} - \frac{(1-y^i)h(x^i)(1-h(x^i))}{1-h(x^i)} \\ &= -\frac{1}{m}\sum_{i=1}^{m} y^i(1-h_\theta(x^i)) - (1-y^i)h_\theta(x^i) \\ &= -\frac{1}{m}\sum_{i=1}^{m} y^i - y^i h_\theta(x^i) - h_\theta(x^i) + y^i h_\theta(x^i) \\ &= -\frac{1}{m}\sum_{i=1}^{m} y^i - h_\theta(x^i). \end{aligned} \quad (12)$$

Finally, the partial derivative of $\mathcal{J}(\theta)$ with respect to θ_j can be expressed as,

$$\frac{\partial}{\partial\theta_j}\mathcal{J}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i. \quad (13)$$