User guide

Xie Li Yang 11/16/2014

Introduction

This is the guide for users who use our algorithm. If you want to share your data to others but still consider the privacy problem, our algorithm is a good choice. We use the idea of differential private to provide a solution to binary classification and you can tune the parameters freely according to different level.

Prepare

1.Operating system: Windows 64-bit with x64-based processor.

2.Python: 3.0 or higher version Please see `https://www.python.org/downloads/` for newest version

3.Software packages: to use our algorithm, download and install at least following software packages:

nose

numpy

scipy

sklearn

It is better if you also have the following software packages in your computer:

Dateutil

Pyparsing

Matplotlib

Six

Seaborn

The above software packages can be download at`http://www.lfd.uci.edu/~gohlke/pythonlibs/`, please download 64-bit version.

Algorithm description

We use the differentially private empirical risk minmization algorithm in paper `http://dl.acm.org/citation.cfm?id=2021036`. Here we use output perturbation method with logistic regression loss. Please see algorithm 1, Theorem 6, Corollary 11, Theorem 15 and Corollary 20 for details.

Steps

1.Input

First put the txt file which contains data set and py file which contains our program into python working directory. Our code expects as input an txt file, please write your training data set into a txt file with following format: Each line correspond to a feature vector (d dimension)followed by the label($+1$ or -1). All numbers are separate by single blank. Please write strictly according to the above format without any extra blank or symbol.

An example of an input file which works correctly with our code is provided in 'training data' txt file. Please see for details. Finally, put the txt file in the same file with py file contains algorithm.

2.Parameter setting

You are prompt to set several parameters while the code is running.

The setting of these parameters are determined by the data set you use and the privacy level. Roughly speaking, privacy parameter epsilon is recommended between 0.01 and 1 so that there is a good balance between accuracy and privacy. Too small(≤ 0.01) will cause unstable and huge error rate. Too large(≥ 2) will cause privacy leakage and no significant enhance in accuracy. Regularization parameter is recommended between 0.01 and 0.5. The value can be set by fix other parameters and take the value when accuracy take its maximum.

3.Running code

You should send the URL of your input .txt file to our program. URL should be something like: $r'C : \backslash Users \backslash < yourname > \backslash < yourdirectory > \backslash trainingdata.txt'$, your input data file here is 'training data.txt'.

4.Output

The code outputs one line, which are d floating point numbers correspond to private classifier trained on the training data set. The output file is a .txt file named as$outputclassifier.txt$, which will appear in the same directory with your input file. For further use, either you provide this classifier to others directly or you can get data set and compute result using this classifier and return the result to others.

Example code

We provide a simple test program to test our program. The test code is in 'oplrtest.py' and input data file is 'training data.txt'. Remember putting them together into python working directory if you want to use them.

Test code

1.Error rate

You can set the parameters according to error rate and tune them. We provide 'oplrer.py' file to compute the error rate. Your testing data set should be in the same directory and same format as training data set. We provide 'training data.txt' for test use. The output will display on IDE instead of store in file.

You should pay attention to the followings

1.Normalization: the norm of your data points should always less than on to match our algorithm. You need normalize your data before use it. You can either normalize by your own code or use our 'normalization.py'.

2.Dimension reduction: data dimension affect the accuracy of results and running time. Large dimension means accurate results and long running time and vice versa. You need do dimension reduction on your data before use it. You can either do it by your own code or use our 'dimensionRed.py'.

3.Data labels: data labels should always be +1 or -1.

4.Running time: This value is affected by data number and data dimension, both in linear relation, roughly speaking. Typically for a training data set with 20000 50-dimension data points, it takes about 30 minutes to finish computing the classifier. Other factors are your

computer conditions. High performance hardware is preferred so that running time won't bother your further work.