INF442 : PROJET INFORMATIQUE 9 GDPR IN PRACTICE: DATA ANONYMIZATION

20 mai 2021

Yunhao Chen, Jiale Ning

1 INTRODUCTION

1.1 Background

We are in a big data era. A big amount of data is generated every day. However, one big problem in front of many Data Scientists is the protection of private data. Imagine that we have feedbacks for our products from customers. Our team wants to sell this data to Amazonia, which will do some fancy machine learning to predict customers' behavior, e.g., what they are going to buy next. The legal problem is that our data comes from many countries, which don't have the same legislation regarding private data, and some customers probably did not agree to give away their private data to Amazonia.

Our target here is to anonymize all these texts to remove any personal data before handing them out to Amazonia.

1.2 Problem description

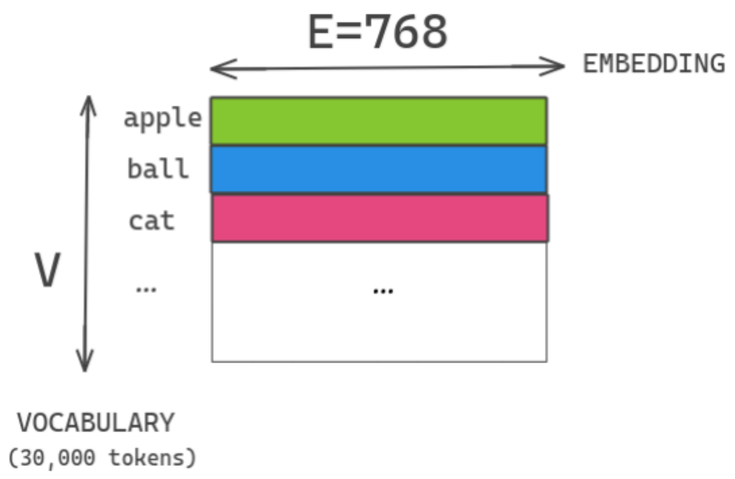
First, the classical CONLL 2003 dataset is used for training the model and for testing it. Then, the raw data was pre-processed and was transformed to a format suitable to our analysis methods. The pre-processing is as follows :

▶ The words are tokenized : we separate the root of each word ("playing" becomes "play" and "#ing" where # usually represents a word separation into several tokens).

▶ The previous step enables us to construct a dictionary : the list (of, say, length s) of all unique tokens in the document. We can consider this dictionary as the Larousse where each word would be uniquely indexed. We can notice that the previous step is very effective in reducing the size of the dictionary, which, in a way, encodes a degree of complexity of the problem.

▶ Each token x is converted to a dummy vector which size s is the same as the dictionary, where a '1' indicates the presence of that token and '0' its absence, i.e. each token is now represented by a flipped 'bit' at the location of its index in the dictionary and s-1 0s ;

▶ All tokens, now represented as dummy vectors go through some very big neural network. In this project, we focus on BERT, which has 24 layers, 16 attention heads, 340M parameters. This network "captures" the meaning of the sentence and embeds each of its tokens in a vector space. We can consider it as a transformation f where f(x) is the BERT representation of the token x.



This figure illustrates the vector representation of tokens. Each token is represented by a row vector. The V represents the volume of our dictionary.

We want to predict if a token is associated to a personal entity, and should thus be removed from the text. All tokens are paired with their Named-Entity-Recognition (NER) tags, such as : O (nothing in particular), MISC, PERS (our tag of interest) and LOC (a location).

1.3 Sub-problems

We consider several sub-problems in which different methods are applied.

▶ Sub-problem 1 :
For now, the ConLL2003 dataset are already pre-processed so that each token (sub-word) is represented as a numeric vector. Moreover, each token is associated to a tag. Since we have to anonymize, we first choose to convert the labels to PERS (say, class '1') vs rest (say, class '0'). Then, all things left is to apply a supervised binary classification algorithm to train a model and use it to predict the tag of a token given.

▶ Sub-problem 2 :
We build and compare classifiers for the original Named Entity Recognition (NER) problem in ConLL2003, i.e. not just for PERS vs rest. Instead of a binary classification, the problem is treated as a multinomial classification. We compare the accuracies of these two cases.

2 ALGORITHMIC ISSUES

In order to anonymize the dataset, we have firstly implemented logistic regression (LR) for binary classification, that is, to identify personal data so that we can then remove them from the dataset. In particular, we decided to implement regularized logistic regression combined with Newton-Raphson's method, which by adding un hyperparameter λ handles the degenerate cases (singular Hessian) during convergence.

To precise our LR model, we define $\delta_1 = \sigma([1\ \mathbf{x}^T]\beta)$ for $x \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^{d+1}$ and $\delta_0 = 1 - \delta_1$. We assume $\mathbb{P}(Y = y|X = x)$ (y=0,1) follows logistic regression such that $\mathbb{P}(Y = 1|X = x) = \delta_1$ and $\mathbb{P}(Y = 0|X = x) = \delta_0$ where y=1 is the label for "PER" data and y=0 is the label for the rest.

We fit our model by maximimum likelihood $\mathcal{L}(y_i; x_i, \beta) = \mathbb{P}(Y = y_i|X = x_i; \beta)$:

$$\begin{aligned}\hat{\beta} &= \underset{\beta}{argmax}\ l(\beta) - \lambda\|\beta\|_2^2 \\ &= \underset{\beta}{argmax}\ log\mathcal{L}((y_i)_{i=1}^n; (x_i)_{i=1}^n, \beta) - \lambda\|\beta\|_2^2 \\ &= \sum_{i=1}^n \mathbb{1}_{y_i=1}[1\ \mathbf{x}^T]\beta - log(1 + exp([1\ \mathbf{x}^T]\beta)) - \lambda\|\beta\|_2^2\end{aligned}$$

We obtain the following formula :

$\bigtriangledown l(\beta) = \sum_{i=1}^n (\mathbb{1}_{y_i=1} - \sigma([1\ \mathbf{x}^T]\beta) \begin{bmatrix} 1 \\ x_i \end{bmatrix}) - 2\lambda\beta$

$\bigtriangledown^2 l(\beta) = -\sum_{i=1}^n \sigma([1\ \mathbf{x}^T]\beta)(1 - \sigma([1\ \mathbf{x}^T]\beta)) \begin{bmatrix} 1 \\ x_i \end{bmatrix} [1\ x_i^T] - 2\lambda\mathbf{I}_{d+1}$

We use them to implement regularized logistic regression with Newton-Raphson's method :

```
INIT : $\hat{\beta} \leftarrow 1$ //arbitrary vector
REPEAT : $\hat{\beta} \leftarrow \hat{\beta} - (\bigtriangledown^2 l(\hat{\beta}))^{-1} \bigtriangledown l(\hat{\beta})$
UNTIL CONVERGENCE ( $\|\hat{\beta}_{updated} - \hat{\beta}\| > defaultEps$)
```

Secondly, we have also solved the original Named Entity Recognition (NER) problem which is a multi-class problem that we convert into multiple binary problem through the "one-vs-all" approach. Based on the logistic regression model for binary classification we have build in the first part, we applied the "one-vs-all" approach by the steps as follows :

One − vs − all :

►train 1 classifier $\beta \in \mathbb{R}^{769}$ for each class y = 0,1,2,3,4 (there are 5 classes in our dataset : "O"-0, "PER"-1, "MISC"-2, "LOC"-3, "ORG"-4) to discriminate this class (assigned label 1) from the rest of the data (assigned label 0).

►assign each new observation $\mathbf{x} \in \mathbb{R}^{768}$ to the class $\underset{y=0,1,2,3,4}{argmax}\ \sigma([1\ \mathbf{x}^T]\beta)$ where σ(t)=$\frac{1}{1+exp(t)}$.

3 IMPLEMENTATION STRUCTURE

We have implemented our algorithms using C++. There are several principal files in our project : Classification.cpp, LogistiqueRegression.cpp, Dataset.cpp, ConfusionMatrix.cpp and their corresponding headers (.hpp). There are also test_LR.cpp and test_LR_multinomial.cpp to test respectively binary classification and multinomial classification. Moreover, in order to facilitate the matrix operations, we have employed the Eigen package (in the folder 'eigen-3.4-rc1'). Now, Let's see together the implementation details of these classes.

▶ Dataset : It is used to represent the datasets. There is a two-dimensional matrix member in this class, called 'm_instances', to store the vector representation of all tokens. The number of samples and dimension of vector representation are stored respectively in 'm_nsamples' and 'm_dim'.

▶ Classification : It is a general class for classification problem. The member 'm_dataset' realized by a pointer to an instance 'dataset', represents the dataset. The member 'm_labels', realized by a vector, correspond to the true labels of this dataset. The member function 'Estimate' takes a vector of a sample and outputs the predict result (for a binary classification, it is 0 or 1).

▶ LogistiqueRegression : It inherits the class 'Classification'. It is designed to solve a binary classification problem using logistic regression. Apart from the members and member functions in 'Classification', it has a member function called 'SetCoefficients' which computes the coefficients beta, which is an important parameter in the algorithm logistic regression.

▶ ConfusionMatrix : This class is to evaluate the binary classification. It provides four forms of evaluation : precision, error rate, detection rate, false alarm rate. Therefore, we can compute the F-score which is a general evaluation for classifier.

3.1 How to use our programs

▶ test_LR.cpp : after the compilation, you can type './test_LR' to get the guide of utilization. In fact, the command form should be './test_LR <train_data_file> <train_label_file> <test_data_file> <test_label_file> <lambda>'.

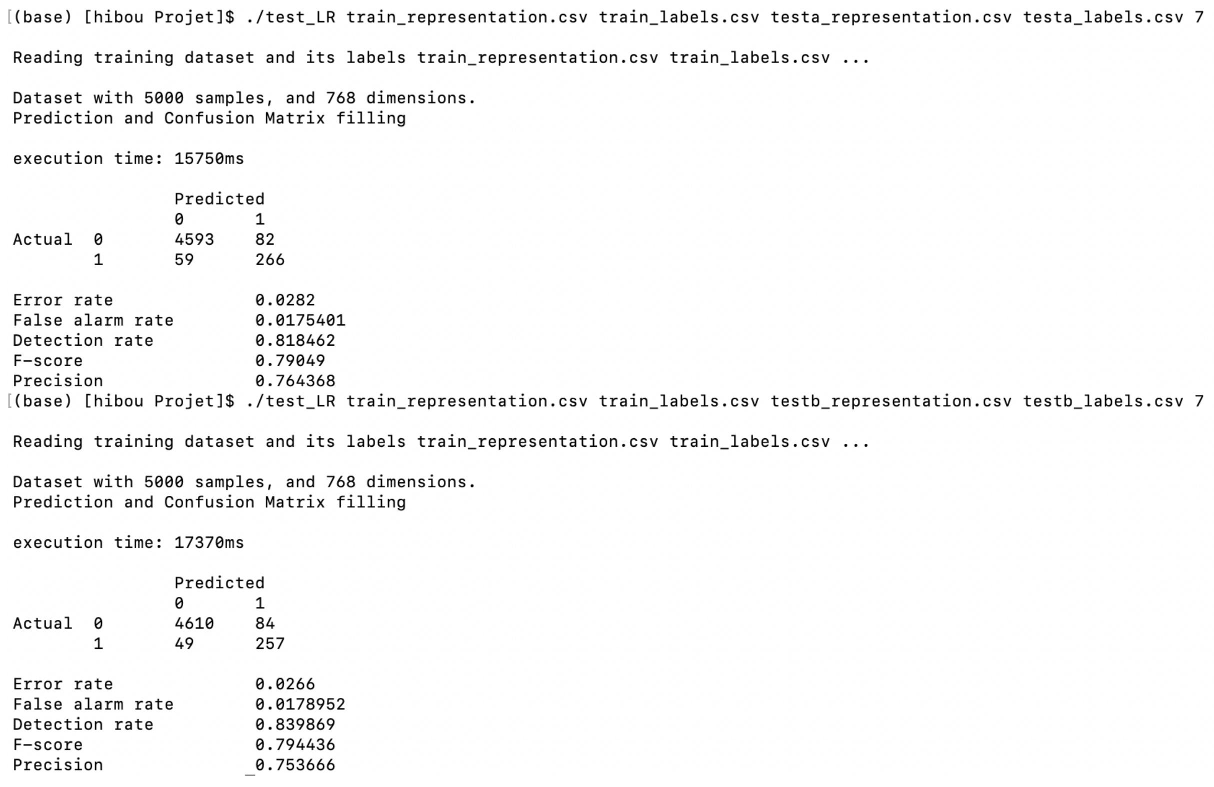
▶ test_LR_multinomial.cpp : in order to not type too much content in command line, we suppose that the program know in advance the name of all labels files (in this format : train_labels_O_vs_rest.csv or/and testa_labels_O_vs_rest.csv).

All you have to do is './test_LR_multinomial <train_data_file> <test_data_file> <lambda>'.

4 RESULTS

In the first part, we realized the binary classification and we have set the parameters λ=7 and defaultEps=7 in which case the testing results are presented below.

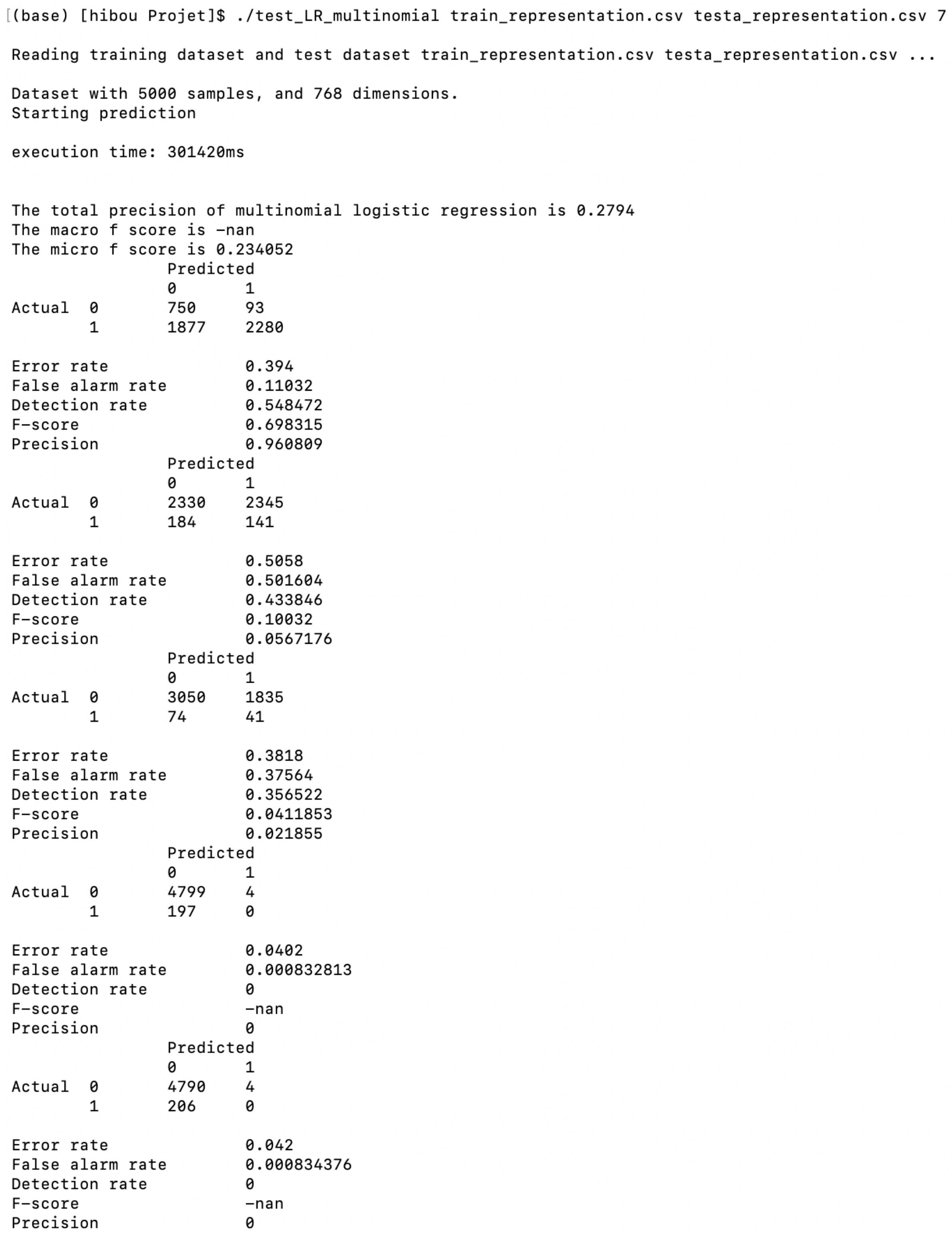
Moreover, we have tried λ and defaultEps with different values and we found that with values from 5 to 10 for both λ and defaultEps, the calculation of β converges with relatively satisfying results and with good convergence performance (quick convergence). In fact, when



```
[(base) [hibou Projet]$ ./test_LR train_representation.csv train_labels.csv testa_representation.csv testa_labels.csv 7

Reading training dataset and its labels train_representation.csv train_labels.csv ...

Dataset with 5000 samples, and 768 dimensions.
Prediction and Confusion Matrix filling

execution time: 15750ms

                Predicted
                0       1
Actual  0       4593    82
        1       59      266

Error rate              0.0282
False alarm rate        0.0175401
Detection rate          0.818462
F-score                 0.79049
Precision               0.764368
[(base) [hibou Projet]$ ./test_LR train_representation.csv train_labels.csv testb_representation.csv testb_labels.csv 7

Reading training dataset and its labels train_representation.csv train_labels.csv ...

Dataset with 5000 samples, and 768 dimensions.
Prediction and Confusion Matrix filling

execution time: 17370ms

                Predicted
                0       1
Actual  0       4610    84
        1       49      257

Error rate              0.0266
False alarm rate        0.0178952
Detection rate          0.839869
F-score                 0.794436
Precision               0.753666
```

the defaultEps is too small, the program doesn't terminate which means β doesn't converge. And λ is relevant to how good the results would be.

In the second part, we have used the logistic regression model built in the first part to realize multinomial classification. But we have obtained abnormal results showed below and we didn't figure out where the problem is though we have examined and tested many times our codes.



```
[(base) [hibou Projet]$ ./test_LR_multinomial train_representation.csv testa_representation.csv 7

Reading training dataset and test dataset train_representation.csv testa_representation.csv ...

Dataset with 5000 samples, and 768 dimensions.
Starting prediction

execution time: 301420ms


The total precision of multinomial logistic regression is 0.2794
The macro f score is -nan
The micro f score is 0.234052
                Predicted
                0       1
Actual  0       750     93
        1       1877    2280

Error rate              0.394
False alarm rate        0.11032
Detection rate          0.548472
F-score                 0.698315
Precision               0.960809
                Predicted
                0       1
Actual  0       2330    2345
        1       184     141

Error rate              0.5058
False alarm rate        0.501604
Detection rate          0.433846
F-score                 0.10032
Precision               0.0567176
                Predicted
                0       1
Actual  0       3050    1835
        1       74      41

Error rate              0.3818
False alarm rate        0.37564
Detection rate          0.356522
F-score                 0.0411853
Precision               0.021855
                Predicted
                0       1
Actual  0       4799    4
        1       197     0

Error rate              0.0402
False alarm rate        0.000832813
Detection rate          0
F-score                 -nan
Precision               0
                Predicted
                0       1
Actual  0       4790    4
        1       206     0

Error rate              0.042
False alarm rate        0.000834376
Detection rate          0
F-score                 -nan
Precision               0
```