INF442 Projet 09
data anonymization

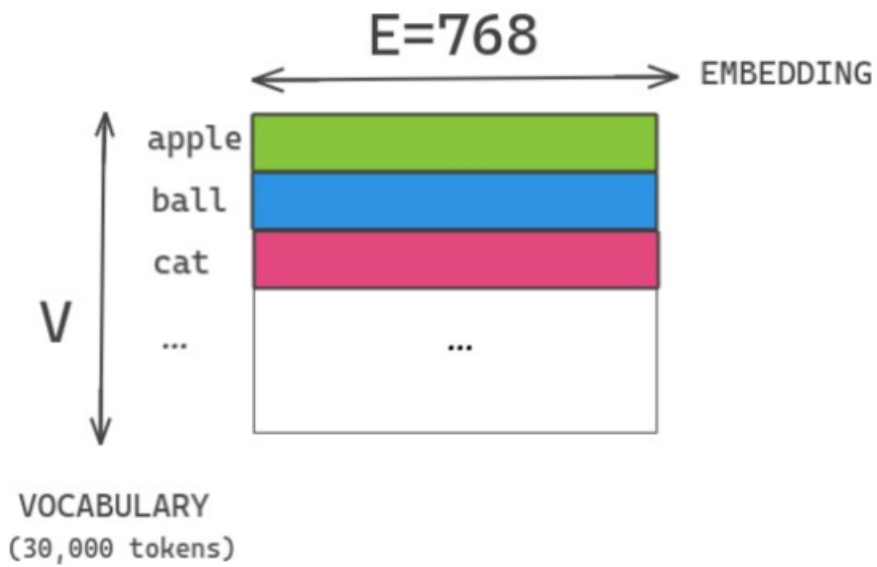
Jiale Ning & Yunhao Chen

May 2021

Table des matières

Description du problème

- données privées $\rightarrow$ problèmes légaux $\rightarrow$ données anonymes
- problème de classification
- série de données: CONLL 2003.
  Pré-traitement:
  vocabulaires $\xrightarrow{BERT}$ représentation vectorielle



Description du problème

Chaque mot accompagné par un Named-Entity-Recognition (NER) label, comme : O (rien en particulier), MISC, PERS (label qui nous intéresse) and LOC (localisation).

- sub-problème 1: label PERS contre le reste
- sub-problème 2: classification multinomiale

Algorithmes - Régression logistique

- Modèle (Classification binaire):
  On définit $\delta_1 = \sigma([1\ x^T]\beta)$ et $\delta_0 = 1 - \delta_1$ où $x \in \mathbb{R}^d$ et $\beta \in \mathbb{R}^{d+1}$.
  On suppose que, $\mathbb{P}(Y = 1|X = x) = \delta_1$ et $\mathbb{P}(Y = 0|X = x) = \delta_0$ où y=1 est le label pour "PER" et y=0 pour le reste.
  On fit notre modèle par maximisation de la vraisemblance "régularisée" $l(\beta)$:

$$\hat{\beta} = \underset{\beta}{argmax}\ l(\beta)$$

$$= \underset{\beta}{argmax} \sum_{i=1}^{n} log\ \mathbb{P}(Y = y_i|X = x_i; \beta) - \lambda\|\beta\|_2^2$$

  On obtient l'expression de $\bigtriangledown l(\beta)$ et $\bigtriangledown^2 l(\beta)$.

- Régression logistique régularisée avec méthode de Newton :
  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 ($\|\hat{\beta}_{updated} - \hat{\beta}\| > defaultEps$)

Algorithmes - One-vs-all

Le problème original : Named Entity Recognition (NER)

- Plusieurs labels : il y a 5 classes dans le base de données : "O", "PER", "MISC", "LOC", "ORG", dont les labels qu'on donne sont 0, 1, 2, 3, 4 respectivement.
- Classification en classes multiples : l'approche "one-vs-all".
- Étapes de "one-vs-all" :
  - entraîner 1 classifieur $\beta \in \mathbb{R}^{d+1}$ pour chaque classe y pour discriminer cette classe (label 1) du reste de données (label 0).
  - attribuer chaque nouvelle observation $x \in \mathbb{R}^d$ à la classe $\underset{y=0,1,2,3,4}{argmax}\ \delta_1 = \underset{y=0,1,2,3,4}{argmax}\ \sigma([1x^T]\beta)$ où $\sigma(t)=\frac{1}{1+exp(t)}$.

Détails de l'implémentation

```
class Dataset {
 private:
      int m_dim; //The dimension of the dataset.
      int m_nsamples; //The number of instances / samples.
      std::vector<std::vector<double> > m_instances;
    //The dataset is stored as a vector of double vectors.
};
```

```
class Classification{
protected:
   Dataset* m_dataset; //The pointer to a dataset.
   std::vector<int> m_labels; //a vector storing the labels
public:
   virtual int Estimate(const Eigen::VectorXd & x , double
       threshold=0.5) const = 0;
}; // pure virtual function
```

Détails de l'implémentation

```
class LogistiqueRegression : public Classification {
private:
   Eigen::VectorXd* m_beta; //The LogistiqueRegression coefficient.
public:
   void SetCoefficients(double lambda); //compute the coefficients
       m_beta
};
```

```
class ConfusionMatrix {
private:
    int m_confusion_matrix[2][2]; //The actual confusion matrix as a
        2 by 2 array.
};
```

Détails de l'implémentation

- **confusion matrix**: $C_{i,j} := \frac{1}{m} \#\{\text{points of class } j \text{ predicted as being in } i\}$
  - true positives (TP) for class i: points of this class correctly predicted in i
  - false positives (FP) for class i: points of other classes incorrectly predicted in i
  - true negatives (TN) for class i: points of $j \neq i$ predicted in $l \neq i$ (possibly with $l \neq j$)
  - false negatives (FN) for class i: points of i predicted in some $j \neq i$

Comment utiliser les programmes test_LR.cpp et test_LR_multinomial.cpp?

- ./test_LR (train_data_file) (train_label_file) (test_data_file) (test_label_file) (lambda)
- ./test_LR_multinomial (train_data_file) (test_data_file) (lambda)

Résultats - classification binaire pour identifier "PER"

Changer le paramètre λ en fixant defaultEps=7 :

execution time: 23630ms

		Predicted 0	Predicted 1
Actual	0	4515	160
	1	89	236

Error rate	0.0498
False alarm rate	0.0342246
Detection rate	0.726154
F-score	0.654646
Precision	0.59596

(a) $\lambda = 3$

execution time: 17820ms

		Predicted 0	Predicted 1
Actual	0	4584	91
	1	60	265

Error rate	0.0302
False alarm rate	0.0194652
Detection rate	0.815385
F-score	0.778267
Precision	0.744382

(b) $\lambda = 5$

execution time: 14740ms

		Predicted 0	Predicted 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

(c) $\lambda = 7$

execution time: 14720ms

		Predicted 0	Predicted 1
Actual	0	4611	64
	1	68	257

Error rate	0.0264
False alarm rate	0.0136898
Detection rate	0.790769
F-score	0.795666
Precision	0.800623

(d) $\lambda = 9$

execution time: 14530ms

		Predicted 0	Predicted 1
Actual	0	4627	48
	1	79	246

Error rate	0.0254
False alarm rate	0.0102674
Detection rate	0.756923
F-score	0.79483
Precision	0.836735

(e) $\lambda = 11$

execution time: 14830ms

		Predicted 0	Predicted 1
Actual	0	4632	43
	1	92	233

Error rate	0.027
False alarm rate	0.00919786
Detection rate	0.716923
F-score	0.775374
Precision	0.844203

(f) $\lambda = 13$

On voit que $\lambda \in [7, 9]$ nous donne une meilleure performance.

Results - classification binaire pour identifier "PER"

Changer le paramètre defaultEps en fixant λ=7 :

```
execution time: 69570ms

                Predicted
                0       1
Actual  0       4625    50
        1       135     190

Error rate              0.037
False alarm rate        0.0106952
Detection rate          0.584615
F-score                 0.672566
Precision               0.791667
```

(g) *defaultEps* = 3

```
execution time: 17620ms

                Predicted
                0       1
Actual  0       4608    67
        1       79      246

Error rate              0.0292
False alarm rate        0.0143316
Detection rate          0.756923
F-score                 0.77116
Precision               0.785942
```

(h) *defaultEps* = 5

```
execution time: 13850ms

                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
```

(i) *defaultEps* = 7

```
execution time: 15600ms

                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
```

(j) *defaultEps* = 9

```
execution time: 12340ms

                Predicted
                0       1
Actual  0       4573    102
        1       51      274

Error rate              0.0306
False alarm rate        0.0218182
Detection rate          0.843077
F-score                 0.78174
Precision               0.728723
```

(k) *defaultEps* = 11

```
execution time: 9570ms

                Predicted
                0       1
Actual  0       4554    121
        1       48      277

Error rate              0.0338
False alarm rate        0.0258824
Detection rate          0.852308
F-score                 0.766252
Precision               0.69598
```

(l) *defaultEps* = 13

Les résultats obtenus en prenant defaultEps $\in [7, 9]$ sont les mêmes et ont la F-score la plus haute. On obtient le seuil de convergence : [7,9].

Results - classification en multi-classes

Maleureusement, on n'a pas eu des résultats constructifs.

```
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
```

Conclusion

- source code partage sur Github
- méthode de travail
- une bonne expérience de projet