Introducción a la ciencia de datos

Manuel Felipe Pineda L 1093223607

November 30, 2016

Abstract

In this lab I will try to train a system with given metadata for emails sent to users in order to predict whether or not a future email will be opened for each user.
The data set contains 53 features for 486048 users.
The project is based in a real competition that can be found here

1 Data attributes

The attributes are fully described in the file "attributes.pdf", and they are divided in four main sections.

Contest logins The attributes in this section have to do with the number of contests that a user actually logged in to HackerRank to compete in.

Contest Participation The attributes in this section have to do with the number of contests that a user signed up for.

Account Information This section contains miscellaneous attributes associated with a user's account and their participation in the HackerRank community.

Email Attributes This section contains attributes associated with emails sent to a user.

2 Experiments

For each experiment I used the classification report provided in scikit-learn [1], I also used the cross-validation technique with the k-fold method, with k equals to ten, to evaluate the estimators performance.

Each experiment was evaluated based in the f1-score and the accuracy-score, were run in 4 cores [1] and all were tested using the methods "fit" and "predict" provided by the scikit-learn API [2].

[1] Intel(R) Core(TM) i5-4300U CPU and 4 GB of RAM

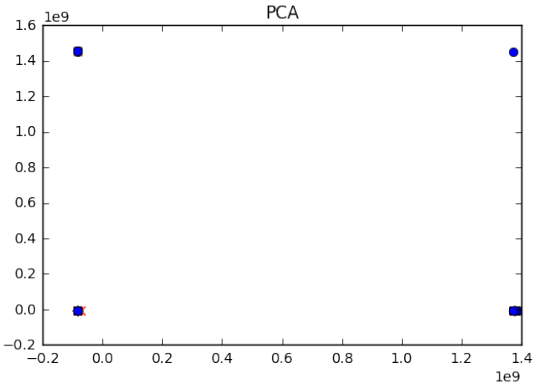

Figure 1: PCA

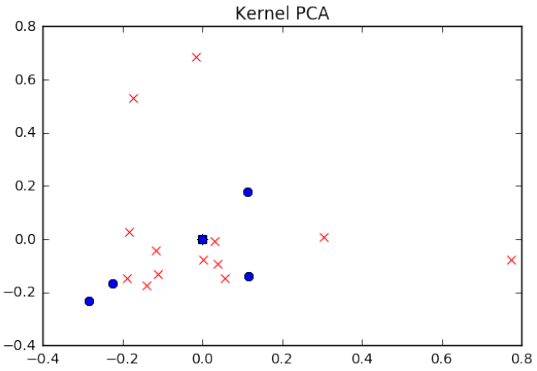

Figure 2: Kernel - PCA

3 Visualization

I used Principal Component Analysis (PCA) as dimensionality reduction in order to plot the data, I also used Kernel PCA as non-linear dimensionality reduction to see if it is able to find a projection of the data that makes data linearly separable.

The figure 1 shows that the data is not linearly separable but give us information about the groups of users in the platform, which could be very useful for other applications. In the figure 2 we can observe how the transformations could help to solve the problem making the classes more separable.

4 Estimators

4.1 Random guess

I use a random guess to get a lower bound and evaluate the estimators.

```
Accuracy: 0.50 (+/- 0.00)
F1 score: 0.40 (+/- 0.00)
```

Table 1: Comparison between estimators

Method	Accuracy	F1 Score	Time (s)
Linear models			
Perceptron	0.60 (+/- 0.35)	0.37 (+/- 0.21)	15.18
Logistic Regression	0.72 (+/- 0.00)	0.28 (+/- 0.01)	68.48
Stochastic GD	0.68 (+/- 0.23)	0.30 (+/- 0.25)	15.49
SGD log reg as loss	0.60 (+/- 0.35)	0.39 (+/- 0.22)	15.78
Passive Aggressive Classifier	0.56 (+/- 0.38)	0.37 (+/- 0.21)	23.27
Non linear transformation			
SVC	0.67 (+/- 0.01)	0.02 (+/- 0.02)	28.05
NuSVC	0.67 (+/- 0.00)	0.01 (+/- 0.02)	29.39
Random Trees Embbedings	0.72 (+/- 0.00)	0.28 (+/- 0.01)	141.27
Extra Trees Classifier [2]	0.70 (+/- 0.03)	0.41 (+/- 0.06)	1.25
Naive Bayes	0.68 (+/- 0.00)	0.41 (+/- 0.01)	41.89
RBF Sampler (Kernel approx)	0.64 (+/- 0.03)	0.16 (+/- 0.08)	1.54
Manifold Learning			
K Neighbors Classifier	0.61 (+/- 0.04)	0.41 (+/- 0.05)	1.58
Radius Neighbors Classifier	0.67 (+/- 0.00)	0.00 (+/- 0.00)	12.97
ANN			
Multilayer Perceptron	0.33 (+/- 0.00)	0.50 (+/- 0.00)	11.18

4.2 Testing estimators

I tested several methods based in different approaches, the table 1 contains the evaluation of the estimators with the default hyperparameters in order to compare them.

5 Tuning hyperparameters

I used the Exhaustive Grid Search to optimize best the classifiers.
The best parameters with KFold and cross validation under the f1 score are the following:

5.1 Optimizing Multilayer Perceptron

```
{
  'tol': 1e-05,
  'activation': 'identity',
  'alpha': 1.0,
  'learning_rate': 'adaptive',
  'hidden_layer_sizes': (100, 33),
  'solver': 'lbfgs',
  'max_iter': 400
}
```

With f1 score of 0.50, done in 11507.70 s

[2]This classifier is able to get perfect score using the whole data set

5.2 Optimizing K Neighbors

```
{
  'weights': 'uniform',
  'n_neighbors': 1
}
```

With f1 score of 0.41, done in 40.58 s

5.3 Optimizing Extra Trees Classifier

```
{
  'class_weight': 'balanced_subsample',
  'max_depth': None,
  'n_estimators': 1024,
  'min_samples_split': 256
}
```

With f1 score of 0.51, done in 1698.70 s

6 Results

The current data is very complex, as result, most of the classifiers get bad performance, even worse that random. However, it is possible to configure some estimators in order to receive better score that pure chance.

This process is very demanding in terms of time and processing because needs to explore a wide range of hyperparameters and the execution becomes exponential in the number of hyperparameters.

By the way, if we compare the score with respect the official leaderboard for the contest, this solution would result in the place 80 of 500.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.