Feature Selection for Noisy Data Binary Classification

Anonymous Author(s)
Affiliation
Address
email

This work is on binary classification of "bag-of-words" type data. The data is called Dexter and downloaded from the following website.

`http://archive.ics.uci.edu/ml/datasets/Dexter`

It is required to classify the Reuter's articles as "corporate acquisitions" or not. Overall there are 20,000 features out of which only 9947 are real features and the rest is a noise. Also 2572 real features are always zeros. There are several techniques we have tried so far for this classification task.

1 Support Vector Machine

Support Vector machine ([1]) is a binary linear classifier. It constructs a hyperplane in a high dimensional space to separate two classes. The hyperplane used in the algorithm has the largest functional margin which provides a lower generalization error. We used the built-in Python classifier *sklearn.svm.SVC* and on the validation data we got **50%** error. This is because the SVM algorithm treats all features as equally important whereas in our data set more then half of the features are irrelevant.

2 Random Forests

Random forests algorithm ([2]) uses decision trees for making predictions. It randomly chooses subsamples from dataset, constructs decision trees and uses averaging to improve prediction. The Python built-in function *sklearn.ensemble.RandomForestClassifier* has options for selecting the number of subsamples for constructing the decision trees. Without using this option on average we get **16.26%** error on the validation set by running the classifier 100 times. However, using the fact that we know the number of relevant features is give the number of features is 7375, we get on average **11.6%** error.

3 Lasso

Lasso ([3]) is an algorithm for linear regression which ignores some features (sets the coefficient to zero) depending on the value of the regularizer. As was said this is a regression algorithm which predicts numerical values but does not do classification. However, as we did in homework 2, we used the algorithm to try to zero out irrelevant features then use some threshold for classification. Again, we used the Python built-in function *sklearn.linear_model.Lasso* and used 0 as a threshold value. we tried the same here an for suitable choice of λ ($\lambda = 0.0134217728$), we were able to get **9.33%** accuracy.

4 Linear Support Vector Classification combined with L_1 regularization

There is an option in the Python built-in function *sklearn.svm.LinearSVC* ([4])where you can require to do a L_1 regularization. This basically combines SVM algorithm with Lasso. This can be done by setting $penalty =' l1'$ and by a suitable choice of lambda we were able to get **7.67%** accuracy.

5 Mean of each class

As we know most of the features in the dataset are useless, therefore we need to do some tricks to get rid of them somehow. The idea here was to compute the mean of each feature for two classes separately then compare them. If they differ by more than 0.1 the SVM error reduces from 50% to 15.67%. However there was no essential improvement in other algorithms even though we have tried differnet values for separation.

Acknowledgments

We especially thank Charles Delahunt for vary helpful discussions.

References

[1] Cortes, C.; Vapnik, V. (1995). ”Support-vector networks”. Machine Learning 20 (3): 273. doi:10.1007/BF00994018

[2] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 1416 August 1995. pp. 278282.

[3] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), JSTOR, pp. 267–288

[4] Documentation on LinearSVC, http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html