Check In: Investigating the Effects of Random Negative Sampling and Clustering on Positive and Unlabeled Data

Stephen Dove

November 2018

1 Results and Milestones

So far, I have completed all of my milestones for the first 3 weeks with one exceptions with the SVM which I will get into in the second section.

In the first week, I planned to create 20 different datasets for each different 20 Newsgroup[4] category. For this task, I am using an object that contains the the different partitions of a dataset where 1 category is labeled positive and the rest negative. These partitions are used to create the positive set of some fraction of the positive examples and the unlabeled set of the rest of examples. Choosing this data structure allows me to iterate through different datasets easily and track their unique partitions.

In the second week, I planned to create a semi-supervised SVM and the metrics to measure accuracy of the final classifier. The metrics are completed, but I am not seeing a large difference in accuracies between models. Each model (random negative sampling and Naive-Bayes) is between 97.6% and 98.1% accurate. The SVM that I implemented is currently a normal SVM, not transductive or semi-supervised.

In the third week, I planned to create models to create the set of reliable negatives. I currently have models for both random negative sampling and Naive Bayes that can feed into the SVM. As I noted above, these have preliminary accuracies between 97.6% and 98.1%.

2 Problems and Challenges

There were three challenges that I came across in the past few weeks while working on this project. The first was a technical challenge with separating the initial datasets since their compressed matrix form made seemingly normal row manipulations impossible. This has been remedied and is now working as expected even with keeping training time under a few minutes (or even a few seconds when I save the datasets to disk).

The second challenge is regarding the accuracy measurements. Right now, accuracy is very high for every model. Even if the model predicts negative every time, the accuracy will be close to 97.2%. One remedy to this issue is to remove some of the negative labels from the test set so that there is a higher proportion of positive labels to negative labels. This will make the accuracy numbers more apparent as to which models perform better.

The third challenge is with the semi-supervised SVM. When I initially read [3], I kept thinking about using a transductive or semi-supervised SVM instead of a normal supervised SVM. I misinterpreted the paper at first, but I still think that these models would benefit from semi-supervision. Since we only have a short amount of time, I looked for a package or implementation of a semi-supervised SVM, but finding these packages and implementations has been difficult. Following [9], Thorsten Jaochims released an implementation of the T-SVM, but I am still working on how to implement this since it is written in C. Following [11], Fabian Gieseke released an implementation of a Quasi-Newton Semi-Supervised SVM which looks promising.

This was written in Python, but is riddled with deprecated libraries. I am currently evaluating how I want to proceed with issue.

3 Changes to Research

The main change to this project is how I will approach the semi-supervised SVM. I feel that this extension will increase accuracy and would be a nice additional piece to evaluate. With the semi-supervised SVM, the final evaluation will change in that I will want to compare additional models being the three reliable negative creating models (Naive-Bayes, Random Negative Sampling, and Clustering) with both the SVM and the semi-supervised SVM.

As I stated above, I am still evaluating the best process for creating this semi-supervised SVM. The three options are:

1. Create a Python module from T. Jaochims' C implementation to act as some sort of black box taking in inputs and returning classifier
2. Debug and update F. Gieseke's Python implementation
3. Create my own semi-supervised SVM from scratch

I am currently leaning towards the third option of creating my own implementation to give myself the flexibility of making changes on the fly as I run into any issues down the road. This implementation would be in Python using NumPy and SciPy. My only concern is that I will need to cleverly deal with the sparse data arrays since I am sure that these previously implemented models efficiently take advantage of some compressed matrix format.

As an additional change, I will be looking into other metrics of evaluation besides accuracy specifically the modified F score from [3] if the reduced negative dataset technique does not differentiate accuracies as I expect.

References

[1] K. Bennett and A. Demiriz (1999). Semi-Supervised Support Vector Machines. *Advances in Neural Information Processing Systems*, 11, 368–374.

[2] K. Nigam, A. McCallum, and T. Mitchell (2006). Semi-supervised Text Classification Using EM. *Semi-Supervised Learning*. MIT Press.

[3] B. Liu, Y. Dai, X. Li, W. S. Lee and and P. Yu (2003). Building Text Classifiers Using Positive and Unlabeled Examples. *Proceedings of the Third IEEE International Conference on Data Mining*.

[4] K. Lang (1995). *Newsweeder: Learning to filter netnews.* ICML-95.

[5] A. Rajaraman and J.D. Ullman (2011). Data Mining. *Mining of Massive Datasets*. 1–17.

[6] A. Kyriakopoulou and T. Kalamboukis (2006). Text classification using clustering. *In ECML-PKDD. Discovery Challenge Workshop Proceedings.*

[7] J. Rocchio (1971). Relevance Feedback in Information Retrieval. *The Smart Retrieval System - Experiments in Automatic Document Processing.*

[8] W. S. Lee and B. Liu (2003). Learning with Positive and Unlabeled Examples using Weighted Logistic Regression. *Proceedings of the Twentieth International Conference on Machine Learning* (ICML-2003).

[9] T. Joachims (1999). Transductive Inference for Text Classification Using Support Vector Machines. *Proceedings of 16th International Conference on Machine Learning*. 200-209.

[10] R. Akbani, S. Kwek, and N. Japkowicz (2004). Applying Support Vector Machines to Imbalanced Data Sets. *in Proceedings of 15th ECML*. 39–50.

[11] F. Gieseke, A. Airola, T. Pahikkala, and O. Kramer (2014). Fast and Simple Gradient-Based Optimization for Semi-Supervised Support Vector Machines. *Neurocomputing (ICPRAM 2012 Special Issue)*. 23-32.