STAT 530 Final Paper

Andrew Mehrmann

May 13, 2016

Biological problem

Data

Issues with Reproducing their Work

Their work involved learning a linear classifier of ER status using the FS_SVM algorithm. While the details of this algorithm are not within the scope of this paper, it can be said that FS_SVM is a feature-selecting version of the Support Vector Machine algorithm. Their final classifier was:

$ER + if\ 0.2488 \times Ex(AW97281) - 1.2934 \times Ex(CA12) - 2.2165 \times Ex(GATA3) < 1.8993$

$ER-$ Otherwise

The authors reported a training set cross-validation accuracy of $93.17 \pm 2.44\%$ and a test set accuracy of 22/23=95.65%.

The training and test sets were pre-determined by the authors and recorded in the metadata file. The first thing to note about their data is that 3 of the 23 test observations were all missing values. Even worse, 13,312 columns were only present in the testing data and not the training data. Removing these features from the entire data set resulted in 27,688 features, presumably the same 27,688 cited in the paper. They mention that they filtered the test set to only include the 27,688 present in the training data but it is not immediately clear why those features were distributed with the GSE file.

The authors describe the process by which the data were normalized:

> The value of each feature in each array was [N1] baselined to 0.1 (each value less than 0.1 was replaced with 0.1), [N2] normalized per array (each measurements on each array was divided by the 50th percentile value for that array) and then [N3] normalized per feature (each feature was divided by the median of its measurements in all samples). The N1–N3 normalization steps are performed by the GeneSpring software. We then [N4] transformed the data into z-scores, by subtracting from each feature value the mean for this j-th feature μ_j, then dividing by the standard deviation σ_j for that feature; hence each transformed feature has zero mean and unit variance over the dataset.

The authors write that data were normalized by [N1]-[N4] separately for training and test for their analysis. I think learning the statistics used in the normalization from the training set then applying them to the test set would have been a better choice. The most glaring issue from their work though is that one of the necessary features of their model, "A_32_P104334 (AW97281)", is one of those that is absent from the training set. It is important to point out that this was verified before performing any preprocessing on their data. To restate their feature filtering methods:

> Features were then filtered to include only those that were annotated with a GenBank accession number and were present in at least 44 of the E176 training samples; this produced a set of 27,688 features

How could they have produced that classifier if one of the three features it selected was not present in the *training* data? I have checked this assertion many times in different ways and still come up with this result. Nonetheless, I still wanted to see if I could reproduce their results on the test set. Afterall, I already had their closed-form classifier and the normalization process did not depend on the training data.

My Methods

As mentioned above, they normalized their training and testing sets separately so I did the same originally. When I began fitting my own models, I noticed they were predicting very well on the training set but very poorly on the test set. I believe there are two possible reasons for that:

1) I was overfitting the training data. But I used two methods (LASSO regression and Random Forest) that are designed to combat overfitting. LASSO in particular doesn't usually overfit, at least not this badly.

2) The data between training and testing were different. This could be because they actually were very different when they were given to me. If the data were from two different statistical populations, then a model fit on one population would certainly not generalize to another population. Another possibility was that the normalization process created differences in the data that the models could not overcome. To test this, I went back and normalized the data together and still had bad (albeit slightly better) results. Specifically...

`analyses.R`

Abandoning the methodology of the paper, I simply used their raw data, unnormalized. I thought methods that implement Bootstrapping would be best for this dataset with so few observations, so I did not proceed further with Naive Bayes. I settled on 3 algorithms to compare: LASSO, Random Forest, and Gradient Boosted Trees.

LASSO

Random Forest

Gradient Boosted Trees

Results

Conclusions

References

Bastani M, Vos L, Asgarian N, Deschenes J et al. A machine learned classifier that uses gene expression data to accurately predict estrogen receptor status. PLoS One 2013;8(12):e82144. PMID: 24312637

Davis, S. and Meltzer, P. S. GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. Bioinformatics, 2007, 14, 1846-1847

Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

Tianqi Chen, Tong He and Michael Benesty (2016). xgboost: Extreme Gradient Boosting. R package version 0.4-3. https://CRAN.R-project.org/package=xgboost

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.