Predicting Gene Function and Localization

By Ankit Kumar and Raissa Largman

CS 229 Fall 2013

I. INTRODUCTION

Our data comes from the 2001 KDD Cup Data Mining Competition. The competition had two tasks, to predict the function and localization of genes. Thus we are presented with a classification problem. The data set consists of a row per gene with the following information about each gene:

1. Gene ID 2. Essential 3. Class 4. Complex 5. Phenotype
6. Motif 7. Chromosome Number 8. Function 9. Localization

All the attributes are discrete variables, with most related to the proteins that are coded by that gene. The function of the gene describes the tasks/operations that the associated protein performs or is involved in such as protein synthesis, metabolism, and transcription. The localization attribute is the region of the cell in which the protein resides, such as ribosome, nucleus, cytoplasm, etc. Additionally, the data set contains interaction data for the genes, describing the relationships between pairs of genes. The data presents a challenge in that many genes have missing variables and it is not obvious how to best utilize the interaction data for the genes.

II. DATA

Each gene in the data set can have one of 15 localizations, and one or more of 13 functions. To classify the test data, the main goal is to find an appropriate feature set. The first data set, with the gene information, provides a starting point. Each gene has the above 9 attributes. However, many of the attributes can take on several discrete values, which are often strings. Therefore, we transformed all possible values of the 9 attributes into binary valued features. Furthermore, for the 9 attributes, many genes have ”?” as a value because data is missing, so we add a an unknown feature for each of the attributes. This leaves a total of 373 features for each of the 862 training genes.

We now have a large feature set compared to the number of training vectors. The data vectors are very sparse because we are given only 8 attributes for each gene, not including Gene ID*. Therefore feature selection could improve accuracy (refer to section IV).

*Note: Each gene can have more than one function, so although only 8 attributes are given besides Gene ID, the data vector for each gene can have more than 8 non-zero feature values

III. BASELINE IMPLEMENTATION

A. Classifiers

We first decided to implement Naive Bayes and Softmax with our binary feature set using only the relational gene data (no interaction data yet). We decided to use Naive Bayes because the data is binary rather than continuous, and furthermore, Naive Bayes and Softmax classification provides a good benchmark for basic training and testing error. We tested two types of Naive Bayes classifiers and two types of Softmax classifiers:

1. Multinomial Naive Bayes which does not penalize the nonoccurence of features
2. Bernoulli Naive Bayes which explicitly penalizes the nonoccurence of features
3. Softmax with an L1 norm
4. Softmax with an L2 norm

We expected these to have different results, considering our feature vectors are very sparse. Therefore, penalization of features that do not occur could affect classification output. We trained both classifiers on the training set of 862 genes and then tested them on the testing set of 1929 genes.

For function prediction we implemented 13 different classifiers, one for each different function. This is because each gene can have more than one function, and the scoring of success for function classification included all possible functions for a given gene. The score for function predicition is as follows:

$$Score = \frac{TP + TN}{FP + FN + TP + TN}$$

B. Results

Classifier Type	Training Error	Testing Error
Multinomial	30.1%	38.1%
Bernoulli	30.1%	39.4%
Softmax L1	23.1%	35.4%
Softmax L2	20.1%	35.1%

Table 1: Results of Classifiers on Localization Prediction

Classifier Type	Training Score	Testing Score
Multinomial	91.9%	87.4%
Bernoulli	91.5%	87.8%
Softmax L1	93.2%	92.0%
Softmax L2	94.1%	91.9%

Table 2: Results of Classifiers on Function Prediction

C. Analysis

Classification with Multinomial versus Bernoulli Naive Bayes classifier led to slightly lower training and testing error with Mutinomial Naive Bayes. This makes sense intuitively because the data vectors are sparse due to the fact that there are 373 features but each data vector will have only 8 to approximately 15 non zero feature values. Therefore, penalizing for all the features that do not occur penalizes the vast majority of features and would not improve performance.

However, the Softmax classifiers did much better than Naive Bayes, likely because the features are not actually conditionally independent. Therefore, we decided to use the Softmax L1 classifier for the remainder of our project.

The classifiers do much better on the function prediction as opposed to localization prediction because, as previously mentioned, a gene can have multiple functions, which increases chances of predicting at least one correct one.

IV. CLUSTERING AND INTERACTIONS

A. Graph Clustering

Our first approach after the baseline implementation was to try and incorporate the interaction data into the classification. We created a graph representation of the interactions data such that the nodes were the genes and the weight of connection between two nodes was the absolute value of the correlation coefficient between those two nodes. We then ran a community detection algorithm that was developed by Ankit Kumar for his CS224W final project to cluster that graph representation into 16 clusters.

After clustering the interactions graph we added the cluster numbers as additional features in the data set. We then performed classification with these additional features and were able to gain a substantial increase in performance. This is likely because genes that interact are in similar localizations and are involved in the same functions.

Classifier Type	Training Error	Testing Error
Softmax L1	19.1%	33.8%

Table 3: Results of Classifier with Graph Clustering on Localization Prediction

Classifier Type	Training Score	Testing Score
Softmax L1	94.5%	92.6%

Table 4: Results of Classifier with Graph Clustering on Function Prediction

B. K-means on Features

Our second approach involving clustering was to perform K-means clustering on the existing feature set, and then add those cluster numbers as additional features as well. This did not have any noticeable effect on performance. Due to the high dimensionality of the feature space, we performed PCA on the data in order to reduce the dimensionality, and then clustered the data. However this only led to a marginal increase in performance. This is likely becuse the clusters weren't meaningful. With only binary features the vector space created by our features was high dimensional and a point either had distance in a dimension (i.e the feature was 1) or didn't. Running PCA transformed the vector space, but likely did not find meaningful principal components to transform it into a space in which K-Means clustering in this way would be effective.

Classifier Type	Training Error	Testing Error
Softmax L1	18.8%	33.5%

Table 5: Results of Classifier with PCA and Kmeans on Localization Prediction

Classifier Type	Training Score	Testing Score
Softmax L1	94.7%	92.8%

Table 6: Results of Classifier with PCA and Kmeans on Function Prediction

C. KNN with Localization

The final approach was to implement a variation of K-Nearest Neighbors as a feature for our classifier. In particular, we added K features to the genes that represented, for each K, the localization of that gene's Kth nearest neighbor. The results were surprising, as the new classifier grossly overfitted on the training set and hurt test set performance considerably. We made sure that we weren't allowing a gene's nearest neighbor to be itself, so that in the training set we found a gene's K nearest neighbors by finding it's K nearest neighbors in the training set minus that gene. Therefore, we conclude that the only explanation for this massive failure of K-Nearest Neighbors is that in the training set, a gene's nearest neighbor is very indicative of it's localization, but in the test set, a gene's nearest neighbor is not very indicative. This could be the case if, for example, all the genes in the training set have neighbors that are very similar to it with the same localization, but genes in the test set aren't very similar to many genes in the training set, so it's "nearest neighbor" actually is not so near, and thus it is a bad predictor.

V. FEATURE SELECTION

A. Feature Ranking

The discrepancy between training error and testing error in the above tests, as well as the large number of features, suggested that the classifier was overfitting. Thus feature selection seemed like a natural next step. We employed a feature ranking algorithm which chooses the k highest scoring features based on univariate statistical tests. We chose k=200 as our final number of features since lower than this led to decreased performance, and higher than this had no change. However, feature selection was not very succesful because overfitting was not substantially reduced and performance was only marginally increased.

B. Results and Analysis

Classifier Type	Training Error	Testing Error
Softmax L1	19.0%	33.8%

Table 7: Results of Classifier with Feature Selection on Localization Prediction

Classifier Type	Training Error	Testing Error
Softmax L1	94.9%	93.0%

Table 8: Results of Softmax Classifiers with Feature Selection on Function Prediction

We likely did not see great improvement in performance or a decrease in overfitting because many data vectors are missing many features, which means that removal of features can cause some data vectors to have very few reamining features.

VI. CONCLUSION AND FUTURE DIRECTIONS

We were able to achieve results that were similar to those of the top scorers in the 2001 KDD Cup Data Mining Competition. The top scorer in localization classification had an accuracy of 72.2%, or a testing error of 27.8%. The top scorer in fuction classification had a testing score of 93.6%. These results were not achieved with machine learning as taught in 229, but instead with other techniques such as rule based learning.

In working with our data, including information from the graph of interactions data was the major contributer in helping improve accuracy. This is likely since genes that interact often has similar function and localization. However, the high dimensionality of the feature space led to a discrepancy in training and testing error that suggested overfitting. Our attempts to implement feature selection were only mildly successful due to the binary nature of our data and that many data vectors were missing a substantial number of features.

Future directions to take to improve performance could include natural language processing techniques since our data is similar in nature (as bag of word models have binary features as well), and also has high dimensionality.

VII. REFERENCES

(1) "KDD Cup 2001." KDD Cup 2001. KDD, 19 Sept. 2011. Web. 12 Dec. 2013.