6.UAP

Vinayak Ramesh

vramesh@mit.edu

Improved Disease Prediction using Integrated Gene Expression Data

Abstract

In this paper we are interested in predicting a particular disease based on gene expression data. In order to do this, we use gene expression data sets to build Bayesian Networks as predictors of disease. We show that it is possible to improve the predictive ability of a Bayesian Network initially trained on a limited number of specific samples. We do this by integrating samples from several experiments associated with a disease. In this paper, we build a scalable software system for performing this analysis and look into the conditions under which this prediction accuracy improvement occurs.

Introduction

The explosion of microarray, sequencing, and other high throughput data sources is facilitating new ways of studying human disease and biology. These technologies produce vast data sets from which new insights, patterns, and knowledge can be learned. It has been shown that, individually, these technologies can exhibit a large amount of variance [1]. In order to tackle this, the fusion of data sets across many platforms is commonly performed [4] [5]. In validating the effectiveness of data integration techniques, the significant genes identified in the integration procedure is often compared to some Gold Standard of genes which is constantly changing [2]. To reduce reliance on this Gold Standard, one approach to validating integration techniques is by using the results to try and predict genes or phenotypes in other experiments [3].

Being able to predict disease based on gene expression data is highly desirable, and in particular, we are interested in prediction of a particular disease given a limited number of gene expression samples. A situation where one may be limited to a few very specific samples is when one is using a new and expensive form of array or sequencing technology. In this paper we investigate using more general samples to supplement our limited set and improve the ability of our predictor. In the case of having a new and expensive technology, one can rely on previously collected data about different but related disease to supplement the limited and specific data acquired by this particular technology.

Methods

Our goal is to create predictors for diseases based on gene expression data. Given a limited number of specific samples, we want to see how adding in heterogeneous sample sets affects the performance of our predictor. In particular, we consider Bayesian Networks for this. Bayesian Networks are a natural choice as they can act as probabilistic classifiers and can naturally encode the interdependencies between attributes. In this particular case, the interdependencies arise from the prevalence of particular genes and their association with a given phenotype. In addition to investigating how to improve our predictor, it is also valuable to see when adding in extra data starts decreasing the accuracy of our predictor. This can be regarded as introducing noise in our system - we want to know how much data causes this, and what type of data causes this.

In order to accomplish this, we construct classifiers according to 2 types of training schemes:

- Single
- Multiple (integrated)

Single:

In this training scheme, both the training and test data sets come from a single experiment which is associated with a particular phenotype or disease. The purpose of this training scheme is to establish a baseline from which we can understand how well a single experiment with a limited number of samples can predict the phenotype of other similar samples. From this baseline, we can see how adding additional samples increases or decreases the accuracy of our predictor. In order to generate the training data set, we designate some fraction t of the total number of samples in this experiment and assign it to the training data set. We then pick some fraction 1-t of the total number of samples and assign these to the test data set. For each experiment associated with some disease, this process is repeated for various values of t.

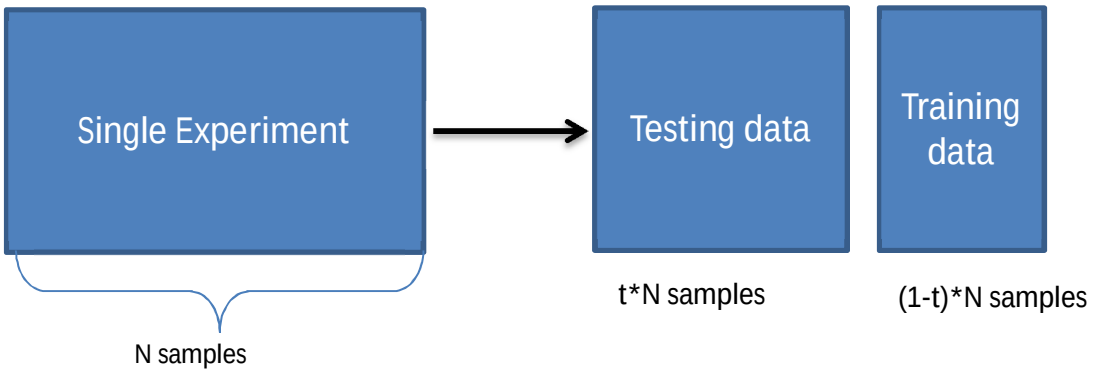

Figure 1. Single training scheme

Multiple (integrated):

In the multiple/integrated training scheme, our training data set is composed of multiple experiments, unlike in the single case. These experiments are all associated with the same specific MeSH term (phenotype). The purpose is of this scheme is to understand how much prediction improves by adding similar (homogenous) samples, but from different experiments. Similar to the single-specific training scheme, we pick some single experiment and partition it into training and testing sets based on some threshold value t. The training set of this experiment is then combined with the samples of all other experiments to create a much larger training set than we had in the single case. The composition of test data in this scheme remains the same as in single. We still test on a fraction of the experiment that was split. Again, this training scheme is done for different thresholds, t, of different experiments.

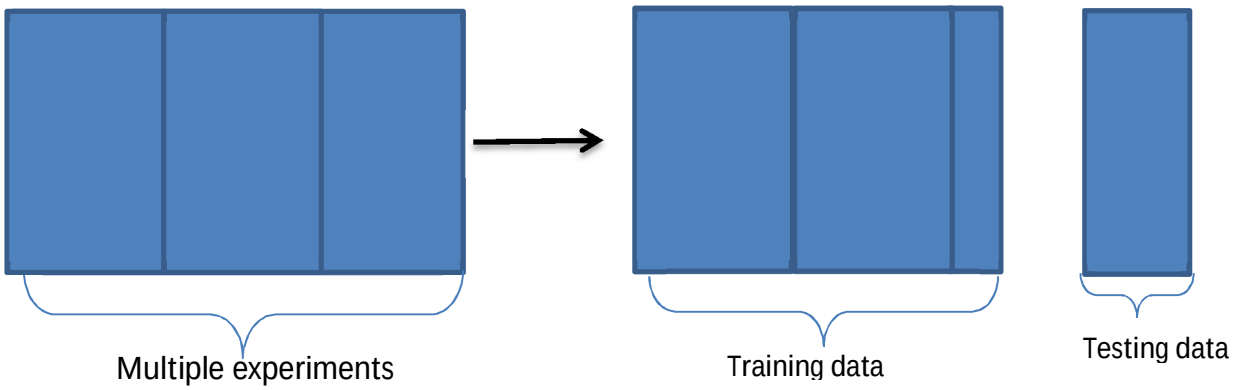

Figure 2. Conceptual illustration of multiple experiments

Data:

The diseases (phenotypes) considered in this paper were taken from the MeSH (Medical Subject Headings) vocabulary. With over 26,000 terms which are controlled and widely used, the MeSH vocabulary forms a good common reference. These terms are arranged in a hierarchical ontology, where more specific terms (diseases) are subclasses of more general terms. These can be accessed at http://www.nlm.nih.gov/mesh/

The data sets used in this paper were obtained from GEO (Gene Expression Omnibus). Particularly, we use their Gene Expression Data sets (GDS) files which can be accessed at http://www.ncbi.nlm.nih.gov/gds . The GDS files come from user-submitted experimental data. The information encoded in these data sets include the genes expressed in the different samples, information about each sample, their phenotype, and information about the particular platform/technology used to obtain samples. The platforms include high throughput genomics technologies including microarrays and different sequencing methods.

We now explain the different software components of our system and how they interact with each other.

Software System Architecture:

Our system for cleaning, processing, and analyzing the data sets consists of two main components: the Preprocessing engine and the Bayesian Analysis engine. The Preprocessing engine converts the GEO data sets into .arff files. These files are fed into the Bayesian Analysis engine which returns the final results in terms of classifier performance over the different disease, threshold, and feature parameters.

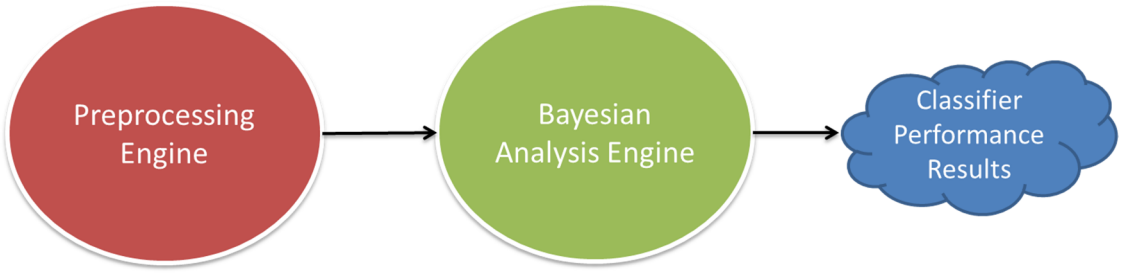

Figure 3. Diagram of how the different components of our system interact with each other

Preprocessing Engine:

The preprocessing system obtains the GEO data, selects the relevant experiments, and creates an appropriate .arff files to feed into the Bayesian Analysis engine. The different components of this system are controlled by a single python program. The user inputs the disease of interest and this engine creates the .arff file for single experiment analysis and for integrative experiment analysis. The Python controller access a Disease to Experiments mapping file, which gives the GDS files associated with a particular disease. This information is then fed into an R preprocessor program. This downloads the needed GDS and GPL files and stores them for quick future access. GDS experiments make it through the R preprocessing pipeline online if they contain only a control and a non-control group for their samples. Control groups refer to those samples without a particular disease, and non-control are those samples with the particular disease. For those experiments which make it through the pipeline, the top 50% of genes are determined and used as features in the resulting .arff files. Each sample (corresponding to a patient in the experiment) is represented as a vector of expression values of each gene acting as a feature. For .arff files corresponding to the integration of several experiments, the intersection of genes across all experiments is taken as the feature set.

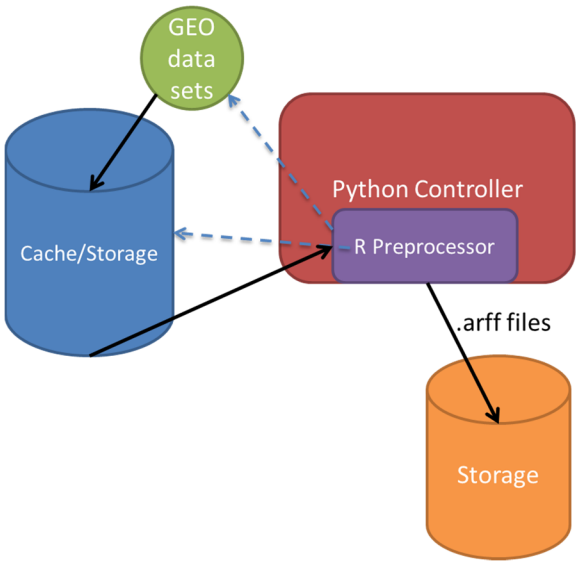

Figure 5. Schematic of the Preprocessing engine and all its components

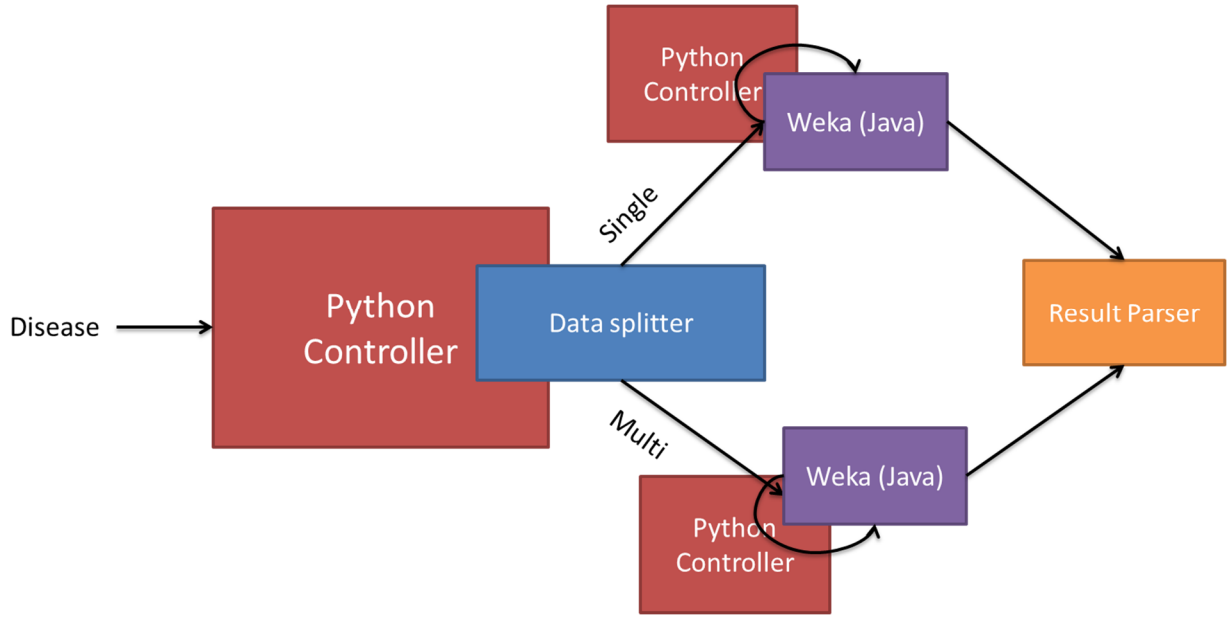

Figure 6. Schematic of the Bayesian Analytics engine and all its components

Bayesian Analysis Engine:

The Bayesian Analysis engine takes as input, the .arff files created by the Preprocessing engine. The entire Bayesian Analysis engine is handled by a Python controller. The Data Splitter creates training and test data files for various thresholds from the original .arff files created by the Preprocessing engine. In particular, our training thresholds range from 10% to 90%, in intervals of 10%. The resulting training and test data files are separated according to whether they correspond to a single experiment or multiple integrated experiments. Each set of training and test data are fed into an instance of a Python controller. This controller controls a Java program which runs the machine learning library, Weka. This set of python controller launches workers to run Bayesian networks for every threshold and for every set of features. In particular, we use the Tree Augmented Network algorithm for evaluating our Bayesian Network. Features are from 1 to 100 in intervals of 10, and from 100 to 1000, in intervals of 100. The classifier accuracy for each instance of a Bayesian network (for single and multiple experiments) are processed by the Result Parser. This parser stores all the data about classifier performance in a .csv file for further analysis.

Parallelization:

Since we want to run many diseases through the both the Preprocessing and Bayesian Analysis engines, it is important to parallelize their operation to improve efficiency. For the Preprocessing engine, each disease and each type of experiment set (single vs. multiple) can be spawned independently and assigned a worker. This allows the preprocessing for several diseases to be distributed among many machines. For Bayesian Analysis, the python controllers for single and multiple, associated with each

disease can be assigned their own worker. In this way, the single and multiple analyses can be parallelized for each disease. By assigning a worker to each of these processes, we can distribute these processes over several machines, and thus devote maximal resources to the Bayesian network analysis. After Bayesian network analysis is finished for each worker, the results can be combined using the Result Parser.

Data Binarization:

In order to avoid the problem of different gene expression scales, we binarize all gene expression data [6]. Binarization is necessary to avoid the problems arising from the differing scales of different platforms. For each experiment, we take the median of the gene expression values. All values above this median are converted to 1, and all values below are converted to 0. Binarizing the expression data allows for fair comparison among experimental results gathered from different platforms.

Feature Selection:

Before building the Bayesian networks for prediction, we perform discriminative gene selection [7] for choosing the top features in our model. In this case, features are the genes expressed in each experiment. The total set of features for each single experiment consists of the top 50% of total genes. For integrated experiments, the feature set consists of the intersection of genes from each individual experiment. We study the accuracy of prediction accuracy against the number of features used in the model, as well as the training threshold. The particular algorithm used for feature selection was a Ranker algorithm, with Information Gain ratio used as the comparison metric.

Ranker uses a sequential forward search algorithm to establish the set of top features. The gene with the high Information Gain ratio is selected as the top gene, the second highest Information Gain ratio as the second gene, and so on. Finding a globally maximal ordering of genes is computationally expensive - using this method, Ranker finds a locally maximum ordering.

Disease Selection Process:

Diseases were selected according to some criteria. In particular, we chose human (HS) diseases which had at least two experiments with a control and non-control group. Of those diseases, we picked those with at least a thousand features. We ran our experiments on the following diseases, from 100 features to 900 features:

Aneurysm

HIV

Adenoma

Endometriosis

Accuracy data:

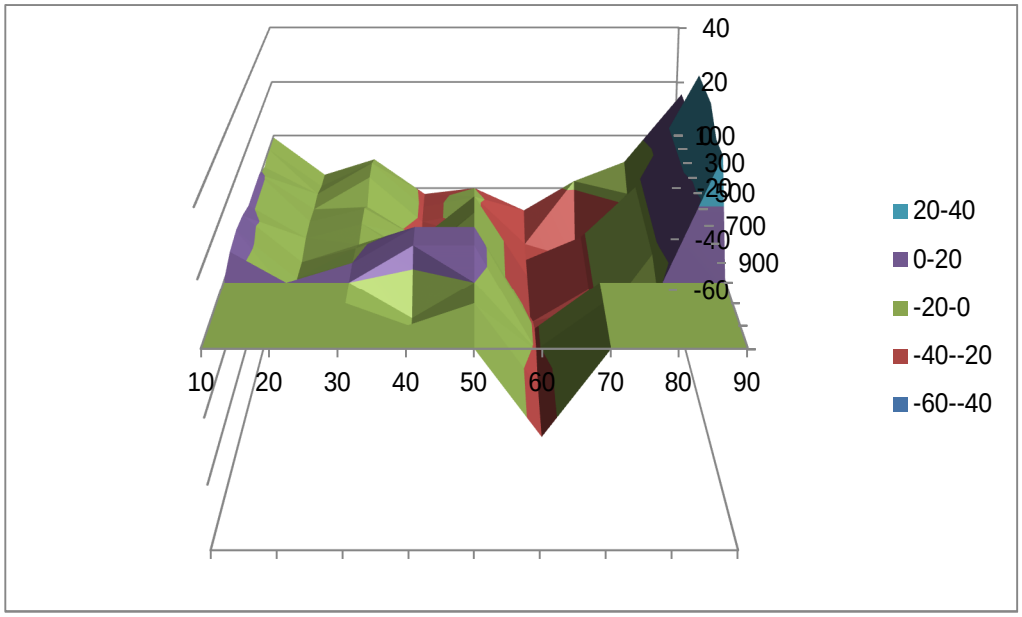

Figure 7. Graph of prediction accuracy difference (of integrated vs. single) vs. threshold vs. number of features

Here we plot the prediction accuracy vs. threshold vs. number of features. In figure 7, the prediction accuracy is the difference in accuracy between the integrated method and the single experiment method. Here we see that in the high feature and low threshold range, the integrative approach works better than the single experiment approach. In addition we also see that the high threshold region tends to behave better in the integrative case.

In order to verify the statistical significance of these results, we perform several T-Tests. All tests are performed with 1 tail, and with the assumption that the integrated case performs better than the single case.

We first perform a T-test over all thresholds and all features. For this, we get the following p-value:

p = 1.49312E-05

For samples with thresholds less than 20% and higher than 600 features, we get the following p-value:
p = 0.01

For samples with thresholds greater than 90% and all features, we get the following p-value:
p = 0.00015

Discussion

From the results, we can see that including other related, but heterogeneous experiments can increase the predictive ability when initially given a limited number of specific samples.

Even though the diverse case tends to outperform the single-specific case most of the time, there tends to be some threshold value after which this is no longer true. Inspection of the data suggests this is around a threshold value of about 10-15%. This likely means, that at this value, the external data may be over or under-fitting the test data and thus decreasing the predictive ability of our Bayesian network. For small amounts of specific samples, we can seem to increase prediction accuracy by adding in samples from more general, related experiments, as long as we stick to this threshold and have a high number of features (over 600). We also see a significant spike in prediction accuracy around a high threshold value.

In order to further verify and understand these results, similar analysis should be performed on a large combination of gene expression data sets and ontology combinations. Many of the computations performed in this paper were quite expensive, and can take anywhere from minutes to hours to days to finish. This is especially true as the number of features in the Bayesian Network grows. The size of the Bayesian network grows as $O(2^n)$ where n is the number of features. In order to make this scale to larger amounts of data, some of the analysis software may have to be recast into a paradigm such as map-reduce and made to run across distributed clusters (for example, implementation with Hadoop and the Mahout machine learning library).

Conclusion

In this paper we showed that we can use predictive methods on gene expression data to verify whether an individual expresses a particular disease. This is important in further understanding the role that genes play in certain diseases. Though predictive power is improved for the most part using this technique, there does exist a point where integrating data from heterogeneous sources is not the best approach.

Future steps in this work would include:

1. Scaling the software platform – in order to process more data, the software has to be implemented in a scalable manner. This essentially means that a cluster of computers should be used for analysis rather than a single computer.
2. Verifying these results with more data and a higher number of features – The results in this paper would hopefully be strengthened by repeating the procedures with more data. However, in order to do this, the software platform must be scalable.
3. Experiment with different feature selection techniques. Picking the better feature sets could result in increased accuracy at a lower number of features.
4. It would be interesting to investigate whether integrating experiments from different but related diseases can produce this same effect of improve accuracy.

References

[1]. Joel T. Dudley, Robert Tibshirani, Tarangini Deshpande, and Atul J. Butte. Disease signatures are robust across tissues and experiments. Molecular systems biology, 5, September 2009.

[2]. Sangeeta B. English and Atul J. Butte. Evaluation and integration of 49 genome-wide experiments and the prediction of previously unknown obesity-related genes. Bioinformatics (Oxford, England), 23(21):2910–2917, November 2007.

[3]. Albert Wu, Amin Zollanvari, and Gil Alterovitz. Multinet Bayesian Networks for Integrative Genomic Discovery: Application to Genetic Epistatic Interactions in HIV

[4]. Daniel R. Rhodes and Arul M. Chinnaiyan. Integrative analysis of the cancer transcriptome. Nature genetics, 37 Suppl:S31–S37, June 2005

[5]. Stein Aerts, Diether Lambrechts, Sunit Maity, Peter Van Loo, Bert Coessens, Frederik De Smet, Leon-Charles Tranchevent, Bart De Moor, Peter Marynen, Bassem Hassan, Peter Carmeliet, and Yves Moreau. Gene prioritization through genomic data fusion. Nat Biotech, 24(5):537–544, May 2006.

[6]. Zhou, X., Wang, X. & Dougherty, E.R. Binarization of microarray data on the basis of a mixture model. *Molecular Cancer Therapeutics* **2**, 679-684 (2003).

[7]. Braga-Neto, U., Hashimoto, R., Dougherty, E.R., Nguyen, D.V. & Carroll, R.J. Is cross-validation better than resubstitution for ranking genes?*Bioinformatics* **20**, 253-258 (2004).