Cancer diagnostics using Support Vector Machine based classification

By

Akhilesh Sudhakar (2013A7TS173P)
Manesh Narayan (2013B1A7542P)

BITS F464: ***Machine learning***

Submitted to:

Dr. Navneet Goyal,
Department of Computer Science,
BITS Pilani - Pilani campus

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(October, 2016)

Introduction

This project aims to use machine learning to classify a blood sample as cancerous or healthy based on gene features. Data is available for 6 types of cancer, and the classifier aims to classify new samples as cancerous or healthy for each of the 6 types of cancer. As a preprocessing step, feature selection is performed after which Support Vector Machine is used for classification.

Dataset Details

The dataset is composed of 285 individual samples, each with 57,736 genes. The 285 samples are composed of one healthy subset and 6 other subsets, each of those encompassing a particular type of cancer.

There are 55 samples from healthy people, 39 samples from people with Breast cancer, 42 samples of Colorectal cancer (CRC), 40 of Glioblastoma (GBM), 14 of Hepatobiliary Cancer (HBC), 60 of Non Small Cell Lung Cancer(NSCLC) and 35 samples from people with Pancreatic cancer (PAAD).

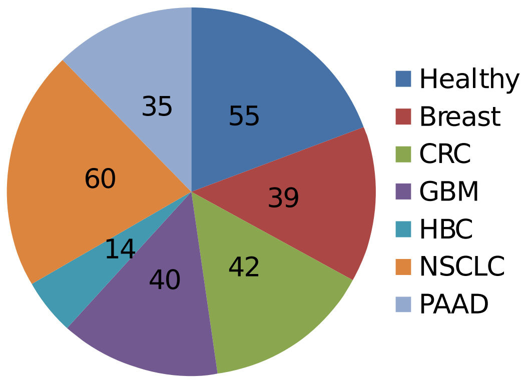

Figure 1 : Distribution of samples

The samples were acquired from both men and women and the donors were distributed across a range of ages (21 – 64 years old). The dataset was obtained from https://www.ncbi.nlm.nih.gov/

Methodology

For all the analyses, the R programming language was used, due to a massive number of libraries and the ease of use. RStudio was the development environment adopted.

Genes with a raw count of less than 5 across all samples are expected to contribute more to noise and do not have much predictive value due to their low variance. Hence all such genes were eliminated before any model building or feature selection. This resulted in a drastic fall from 57,736 to 15,674 in the number of genes on an average (Actual numbers vary slightly with each of the 70:30 partitions). The remaining 42,062 genes were excluded from all analyses.

For all evaluations, the dataset was split in a 70 : 30 ratio for model training and validation respectively. To account for a chance bias due to the partitioning, 10 different splits were made, and the model was trained on each of the 10 training sets and evaluated using the respective validation subsets. The final accuracy reported is the average of these 10 models. Each partition was created after providing a seed value, so as to allow for reproducibility of results.

Raw counts were transformed to their counts per million(cpm) counterparts to account for sequencing depth differences and to make the values comparable and exportable. “cpm” function from the edgeR package was used for the same. All model building was done on these cpm values.

The Training cohort is used for feature selection and model building. The algorithm never sees the Validation cohort to avoid bias, and uses it only for prediction.

The dataset has highly unbalanced classes. A possible implication of this is that despite seemingly good accuracy results, the classifier may have no actual predictive value due to very high false positive rates.

In case of a really small negative class, even a false positive rate of 100% may give a very high accuracy. Keeping this in mind, all results are listed with both the accuracy and the true negative percentage (% of healthy people predicted to be healthy by the classifier).

Selection of package to be used

While a lot of packages exist for dealing with count data, their assumptions about the data and the way they interpret it may not be accurate. RNA Seq, which gives count data, should be represented by a discrete probability distribution and can be modeled with Poisson or Negative Binomial distributions. A few packages assume the data to have a Poisson distribution. However Poisson distribution assumes the mean and the variance to be the same. Hence it can't account for the over-dispersion observed in the RNA count data. This is better modeled by an assumption of Negative Binomial count distribution, which can have independently varying mean and variance (Poisson is like a special case of Negative Binomial distributions).

Some of the popular R packages for the analysis of such data are Limma and EdgeR. Limma was originally intended for microarray data analysis and hence was rejected. DESeq being outdated, has been replaced by DESeq2 which isn't very widely adopted as of yet. EdgeR is very widely used, has many active users on online forums and is well documented. Hence EdgeR was chosen for this analysis.

The other commonly used packages for machine learning algorithms were caret and RWeka. RWeka provides an R interface for the very popular software, WEKA. Furthermore RWeka provides a lot of customization options for each algorithm and has a huge selection of algorithms available for usage.

Caret is useful primarily because it has internal tuning, which although not very good with larger datasets, gives a better performance on the downsampled sets.

Hence these 2 packages were used.

Foreach package was used for enabling parallel computation. Though resource intensive in itself, a single SVM evaluation or a single feature selection run wasn’t sufficient to saturate the RAM. This allowed for multiple cores to be used, each running a separate thread, speeding up operations by 5 times.

Baseline Prediction

To establish a baseline for the predictive strength, a few commonly used Machine Learning algorithms were used to predict the classes of validation set without doing any feature selection. The count data of all genes were transformed to their cpm values and this was used for model building and validation. The results observed with the Validation cohort are tabulated in Table 1. Due to a very high number of features, logistic regression and neural networks were incapable of processing the dataset at this point, and hence were excluded. The implementations of Support Vector Machine, Random Forest and Decision Tree as found in Weka were used for model building.

	SVM	Random Forest	Decision Tree
Accuracy	94.75	89.8167	82.9
True Negative Rate	84.41	63.5667	61.4833

Table 2 : Baseline Predictive Strengths (Averaged across 10 models)

Table 1 shows the better noise handling ability of Support Vector Machines(SVM) in comparison to other algorithms. Despite a small gap in the accuracies reported, the true negative rates vary greatly between SVM and Random Forest, highlighting the class imbalance and the need to retain true negative rate as one of the measures for a model’s predictive strength.

Feature selection

For feature selection, functions implemented in the edgeR package were used.

The normalization factors for the given dataset were first computed. Following this, a generalized linear model was fitted on the training set after estimating the dispersions of the data. Post this, a log likelihood ratio test was employed to aid in identifying differentially expressed genes. A likelihood ratio test is used to compare the goodness of fit of the null and alternate model on the data. This likelihood ratio is used to project the p value which can be corrected to obtain the false discovery rates. Likelihood ratio expresses how many times more likely the data is under one model than the other.

The p values computed from this test are then adjusted using “Benjamini and Hochberg” method, which gives us the False Discovery Rates (FDR). Given that a positive result is predicted, the FDR value gives the expectation of it being a false positive. Hence genes with high FDR values need to be rejected to control the number of false positives. The FDR values are used to filter out genes. The value used at this point was 0.00005, i.e. a 0.005% chance of a false positive given a positive. This leaves 1433 genes in the dataset and discards the others.

Any genes with a logCPM value of less than 3 were discarded, to ensure that only genes with sufficient count to provide meaningful results are retained. This leaves 1057 genes in the dataset.

To identify genes which are differentially expressed, a limit was imposed on the Fold Change(FC) value of the count data. With the healthy subset being used for making the intercept in the design, the fold change value provides the ratio of expression levels of the gene in the cancerous samples and the levels in the healthy samples. A 20% increment or decrement in the expression levels were used as the threshold for retention of a gene.

The counts of the genes which weren't discarded were then transformed to their cpm values used for model building and prediction.

The results for the same are tabulated in table 2.

	SVM	Logistic Regression	Random Forest	Decision Tree
Accuracy	93.4	89.35	90.23	81.75
True Negative Rate	90.625	88.93	77.03	59.38

Table 2 : Predictive strengths after feature selection (Averaged across 10 models)

The performance of all the classifiers is augmented to some extent. This is enough to justify feature selection on the datasets before using machine learning.

These results show that SVM has much better predictive power and justifies its adoption for further tuning.

Support Vector Machine

Support Vector Machine or SVM is a non probabilistic binary linear classifier. SVMs can efficiently perform a non linear classification by using a "kernel trick". SVM projects the samples into a higher dimensional space, wherein each feature is represented by an axis. Each of the training set records is positioned in this higher dimensional space according to their position vector (counts of every feature). During the training process, SVM tries to draw a hyperplane which can best separate the two classes and maximizes the space between them. Following this, the validation set has their class masked and the points are positioned in this higher dimensional space. Their positioning with respect to the hyperplane decides their predicted class. Only the final features used to draw the hyperplane, i.e. the ones closest to the hyperplane, decide its positioning and are called support vectors. This approach eliminates noisy features with ease.

A polynomial kernel function was used for the SVM. Poly kernel represents the similarity of vectors in a space over polynomials of the original features. This allows learning of non linear models. Poly kernels look not just at features but also at their combinations, or what is also known as interaction features. This is expected to hold specific significance in the case of biological data, as proteins and mRNA often influence the levels of other gene activity, either by direct interaction or by the gene product.

Feature selection optimization

Parameters like FDR and Fold change hold a lot of significance in the selection of features However, there is no accepted standard value for the same. To select for optimal values of the same, a graph of accuracy versus log of FDR value was plotted. The point of maxima for the FDR value as can be seen from the plot in figure 2A.

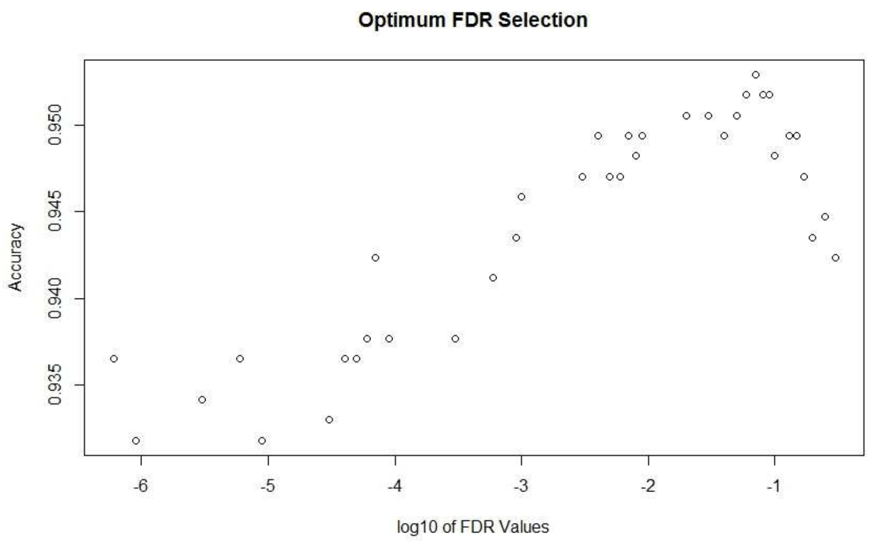

Figure 2A

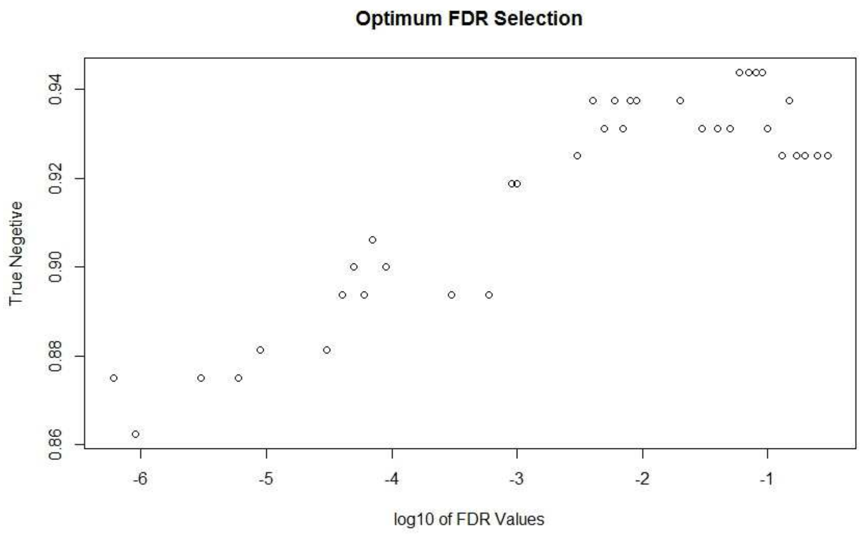

Figure 2B

The above plots show an accuracy of 95.29% and a true negative rate of 94.38%. for the optimal FDR value of 7E-02.

A graph of accuracy versus fold change values was also plotted to select an optimal value for fold change. In case of points signifying more than 50% change, the Fold change threshold for downregulation was kept at more than 50% and only the upregulation threshold was varied.

The optimal value was found to be a 50% change.

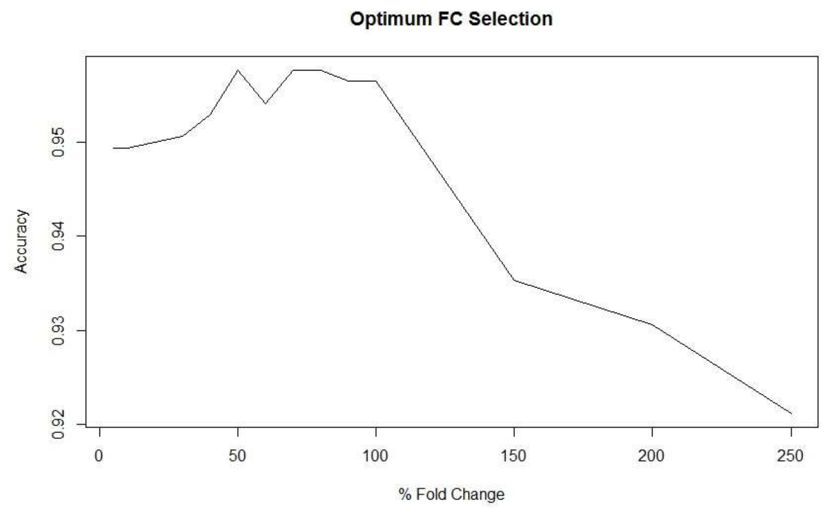

Figure 3A

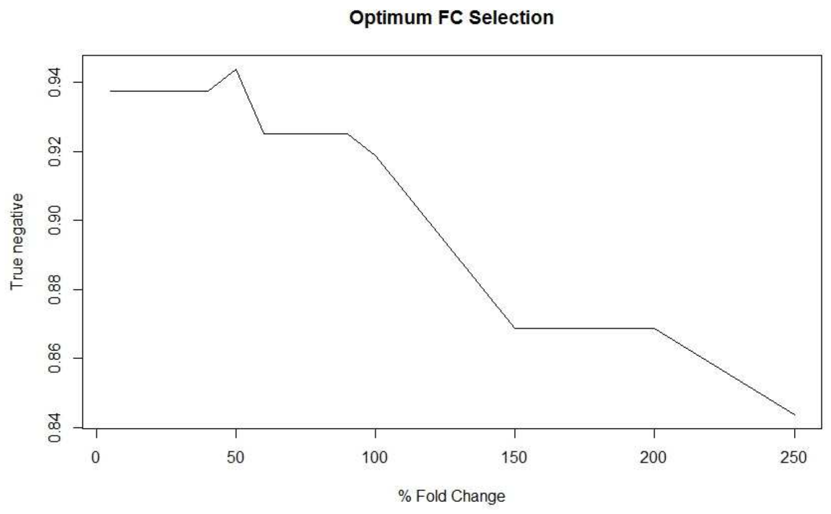

Figure 3B

At 50% fold change, the accuracy observed was 95.76% which is a small increment from using just the optimized FDR values for selection.

Classification of healthy vs one cancer class

Most diagnostic tests would be based on suggestive symptoms, which may hint at a particular type of cancer. In such a case, we only need to test for the presence of that type of cancer.

To determine the predictive strength of machine learning in such a scenario, healthy class and each cancer class were compared against each other. (Caret was used for these results)

	Accuracy	**True negative**
Healthy vs Breast Cancer	99.26	99.37
Healthy vs GBM	82.14	98.75
Healthy vs HBC	93.68	100
Healthy vs NSCLC	91.51	95.62
Healthy vs PAAD	98.85	100
Healthy vs CRC	96.15	100

Table 3 : One vs one classifier

The fact that wide variation is observed within GBM, may be a reason for the weaker predictive strength of the Healthy vs GBM classifier.

To eliminate the need for True negative as a performance evaluation metric, the classes were down-sampled to the number of the samples in the smaller class. Hence while the smaller class retains all its samples numbering n, n random samples are drawn from the larger class. To account for possible bias, the draw is done 10 times. This provides 10 sets of balanced class datasets. Now each of these 10 sets are subset 10 times into a 70:30 ratio, giving a total of a 100 SVM runs for each classifier. The results summarized in table 4 are averaged over these 100 models.

	Breast	CRC	GBM	HBC	NSCLC	PAAD
Accuracy Caret	96.91	96.08	78.5	65.17	93.22	97.8
Accuracy RWeka	96.81	91.66	87.5	90	95	92
Number of features	5050	9656	1712	1585	4362	6623

Table 4 : One vs One classifier for downsampled set

The decrease in accuracy is possibly because of the reduction in the number of training samples, due to the subsampling. The classifiers with a drastic fall in the accuracy levels also have fewer features and have lesser samples within the class.

Boruta

Boruta is a package which provides the Boruta function. Boruta is an all-relevant selection wrapper algorithm. It finds the relevance of features by comparing their importance with that achievable at random by permuting them. Attributes with lesser importance than their permuted copies are removed. The ones with significantly higher importance are confirmed and the others are kept as tentative, till a future run either confirms or rejects it or till the number of iterations exceeds the maximum allowed.

Since the number of features selected by Boruta were low in each case, all the genes selected from each subset were used and since the number of genes left in the tentative list were less than 5, they were included in the selected features too. This gave a set of 146 genes. It gave an accuracy of 91.25% averaged over 10 models.

Principal Component Analysis

Principal Component Analysis (PCA) is a procedure that transforms a set of features into a subset of linearly uncorrelated features. Correlated features can cause overfitting of the models to training datasets, giving worse performance in the validation sets. PCA defines the components such that the first component has the largest variance in the entire dataset (to account for more variability in the data), and then each succeeding component has the highest variance possible provided that it is orthogonal to the preceding components. PCA is generally very good for selecting the important features.

PCA gives us 84 features and an accuracy of 91.25%. The features selected by PCA are a subset of the ones selected by boruta.

Conclusion

Feature selection helps the classifiers by eliminating a lot of features which could otherwise contribute to noise and potentially skew the results. It gives a good increase in performance.

SVM has by far emerged as the strongest binary classifier for RNA Seq data, which might have to do with its superior ability to deal with noise, given that we have thousands of features to select from. This gives it better predictive strength, both with and without feature selection.

When using a binary classifier for one cancer and one healthy class, the accuracy and true negative levels are very high, implying a really good classification ability. Only GBM and HBC have slightly lower accuracies, which was expected given that GBM has subclasses within itself. HBC has very few samples, which means fewer examples for training and testing as well. This can be easily adopted to diagnose suspected cancers and for monitoring post treatment.

Downsampling is useful to make the results comparable, however the number of samples is fairly low in certain cases, making it a bit difficult to get a good model. A higher number of samples would increase the predictive power massively.

Boruta and PCA give an accuracy of 91.25% each, which while on the lower side, uses only 146 and 84 genes respectively which is pretty good. These can be fairly useful for a quick diagnostic test.

Suggested Improvements and Pitfalls

There is a huge class imbalance in the groups. This ends up skewing the results such that high accuracies could still mean poor predictive strength. Balancing the classes isn’t a good idea as can be seen from the falling predictive strength due to lower number of samples.

Hence increase in the number of samples obtained should be a good way to move ahead.

Furthermore, this data has quite a few mislabeled samples. Though the final results reported have corrected for that, it did skew a lot of initial results and also puts a restriction on some of the other analyses that could have been conducted. Hence an independent data set should be gathered.

Also, edgeR, DESeq etc are designed to work with replicates. Their predictive strength increases a lot when using replicates as was observed using other data. However only a few samples have replicates in this data and none of them are labeled as such. In the event of a future sample gathering, this could be looked into.

The Foreach package is very good, however care should be taken while using packages whose performance with it hasn’t been evaluated. Foreach can give unknown and inexplicable errors in certain cases, especially while using for loops. Weka and Boruta are known to give errors on larger datasets if used inside a foreach loop.

R is a good language for this kind of work, due to the massive support on forums and due to a number of well maintained packages being available on CRAN. However, since R keeps all its objects and the environment for every session in memory, it puts a bottleneck on some algorithms, preventing them from running on larger datasets.