Comparison of Support Vector Machines to Other Supervised Learning Algorithms in Leukemia Cancer Classification

Winnie Ngo WNGO1@SWARTHMORE.EDU
500 College Avenue, Swarthmore, PA 19081

Eric Oh EOH1@SWARTHMORE.EDU
500 College Avenue, Swarthmore, PA 19081

Abstract

A large number of supervised learning algorithms have been created in the past decade and widely applied to the field of bioinformatics. In this paper, we explore the accuracy of some of the most popular data classification methods: Support Vector Machines (SVMs), Decision Trees (DT), Random Forests (RF), Extra Trees (ET), and Naive Bayes (NB).

We present the results of an empirical comparision conducted on the performance of these algorithms on the classification of leukemia classes using gene expression data of patients. Patients were either classified as having acute myeloid leukemia (AML) or acute lymphoblastic leukemia (ALL). To assess their performaces, we trained and evaluated each of our models using labeled leukemia patients data and n-fold cross validation. We tested and compared various values of n and parameters specific to each algorithm.

Our results indicate that the SVM with a linear kernel yielded the highest accuracies over each n-fold cross validation, with the Naive Bayes classifier following close behind with accuracies abovr 90%. Over a range of n values for performing n-fold cross validation, most of the algorithms achieved their highest respective accuracies at $n = 7$. To compare the differences in accuracy amongst the best classifiers, we ran paired t-tests on several combinations of algorithms. With a few exceptions, the SVM with a linear kernel outperformed all other algorithms. Our conclusions provide a sense of the efficacy of supervised learning algorithms and guidelines to follow when choosing a classification method.

CPSC 68 Bioinformatics Final Report Proceedings, Swarthmore College, Fall 2014.

1. Introduction

One of the most important and difficult challenges in bioinformatics is the classification and prediction of biological data. In particular, cancer classification provides a unique challenge and numerous applications. One of the main difficulties surrounding cancer treatment is being able to successfully target specific therapies to known and distinct cancer types in order to maximize effectiveness. So being able to correctly classify cancer classes is a key to advances in cancer treatment. Historically, cancer classification has relied on biological insights, rather than unbiased computational approaches for distinguishing between cancer types. In this paper, we chose to classify different forms of acute leukemia from gene expression data using a computational machine learning approach. [6]

Supervised learning is the machine learning task of learning the link between two datasets: the observed data X and an external variable y that one wants to predict. Supervised learning algorithms train a classification model on labeled data. This resulting trained model is then tested on novel data.

For our project, we are interested in how classification models implemented using different supervised learning algorithms perform when used to classify patients with AML or ALL leukemia. The main supervised learning algorithm we aimed to study were Support Vector Machines (SVM), as they have been shown before to perform well in many areas of biological analysis. The other supervised learning algorithms that we implemented and used to compare to SVMs are Decision Trees, Random Forests, Extra Trees, and Gaussian Naive Bayes.

We test these different algorithms on the classification leukemia patients as having either AML or ALL type

leukemia. We using n-fold cross validation to test our predictions on our training set. We report that SVMs outperform the other algorithms on average and in addition, we discover the n at which our classifiers have the highest accuracy. We further validate our results by running t-tests to confirm that our differences in accuracy are indeed statistically significant.

The rest of the paper is organized as follows: In Section 2, previous work done on comparing classification algorithms is discussed. Section 3 then discusses the algorithm methodology, with Section 4 detailing the results of applying the classification methods to the leukemia cancer data. Section 5 then evaluates the results and investigates the efficacy of the classifiers. Finally, Section 6 concludes the paper and explores future work.

2. Related Work

Much work has been done previously on the comparison of various classification methods. Each of the following papers has compared various classifiers, and reported their results with regard to their test data and evaluation criteria.

Huang et al. [7] compared decision trees, SVMs, and naive bayes against each other using the Area Under Curve (AUC) metric. With their data set, it is shown that the AUC metric is a better criterion than accuracy. Using this metric, it is shown that decision trees performs better than both SVMs and naive bayes.

Amor et al. [2] compared decision trees and naive bayes on intrusion detection system data. This comparison's results show that the predictions made with naive bayes are better than those of decision trees. The same comparison was done later by Chen et al. [4] with SVMs and artificial neural networks (ANN). The results for that study show that SVMs outperform ANNs.

Amendolia et al. [1] compared k-nearest neighbors (kNN), SVMs, and ANNs for talasemi detection using the accuracy criterion. It was shown that ANN has better accuracy than the other two methods.

O'Farrell et al. [11] have compared kNN and ANN in the classification of spectral data. The results show that if the values of data deviate from real values, using the ANN is superior. However, with simple data, the kNN algorithm performs better.

Another criterion, root mean squared error (RMSE), was investigated by Kim [8] to compare decision trees, ANNs, and SVMs. Their results show that if the data contains any errors or if real values of attributes are not available, then SVM regression could act better than ANNs and its RMSE would be superior.

All of the papers mentioned have completed research that compares classifiers against each other in some way. A problem many of them share however, is that they are very data dependent and seem to not be generalizable. A few results actually seem to be contradicting each other, but that is the nature of the business when we are unable to test on all existing data. We hope that our results will give some intuition as to the choice of n for n-fold cross validation and some parameter choices when it comes to choosing classifiers for general data sets.

3. Methodology

3.1. Classification Algorithms

A classification algorithm requires two pieces of information: the set of features to be used as inputs to train the model and the class membership of each data point.

Support Vector Machines

SVMs try to find a function that maps all the input features, $\mathbf{x}$, to its class membership, $y \in \{-1, 1\}$ such that

$$y = f(\mathbf{x}, \mathbf{w}) \tag{1}$$

where $\mathbf{w}$ is the parameter vector, $f(\mathbf{x}, \mathbf{w})$ is the function, and y the output. SVMS are designed to maximize the margin that separates the two classes using a hyperplane to increase generalizability. The main advantage to using SVMs are compared to another algorithm is that the hyperplane is searched through maximizing the margin. This allows for more robustness and greater generalization. Training data is modeled as points in space, and new data is mapped into the same space with our function f and classified based on which side of the margin they fall on. In the process of finding the best hyperplane, SVMs find a set of the data that is the hardest to classify, namely those closest to the hyperplane. These points are referred to as the support vectors.

The trained SVM classifier is then a linear combination of the similarity between an input and the support vectors. To measure this similarity, SVMs use a kernel function $\psi(\mathbf{x}, \mathbf{x_i})$, where $\mathbf{x_i}$ is the i^{th} support vector. The class decision is then made using the following:

$$y = sgn\left\{\sum \alpha_i y_i \psi(\mathbf{x}, \mathbf{x_i})\right\} \tag{2}$$

where y_i is the class label for the i^{th} support vector and α_i is the positive parameter of the i^{th} support vector determined by the SVM. Two very popular kernel functions that were used in our experiments are the linear and radial basis functions (RBF). The linear kernel is given by:

$$\psi(\mathbf{x}, \mathbf{x_i}) = \mathbf{x}^T \mathbf{x_i} + c \tag{3}$$

where c is a constant and the RBF is given by:

$$\psi(\mathbf{x}, \mathbf{x_i}) = exp\left(-\gamma \left\|\mathbf{x} - \mathbf{x_i}\right\|^2\right) \tag{4}$$

for $\gamma > 0$, a kernel parameter [12].

Classification Trees

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. Tree models where the target variable can take a finite set of values are called classification trees. The leaves of these trees represent class labels and each node represents a feature that the tree is discriminating by. Decision Trees are a simple representation for classifying examples using a machine learning approach. They are require little data preparation compared to other algorithms and are relatively easy to understand and to interpret. However, decision trees tend to overfit on data with a large number of features.

Random Forests grows many classification trees.The algorithm constructs a large number of individual trees from each input and output as the class the mode of the class outputs for each decision tree. To train such a system, when the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority. [3] Random Forests runs efficiently on large data bases and can handle thousands of input variables without variable deletion. It does not overfit.

Extra trees are constructed in the same manner as random forests. However, for each decision tree, instead of picking the features with the most discriminative threshold, the threshold is drawn at random for each feature. This tends to reduce the variance but increase the bias of the model. Extra trees are generally cheaper to train from a computational point of view but can grow much bigger. [5]

Naive Bayes

A Naive Bayes classifier is a probabilistic classifier. It predicts classes using Bayes Theorem with the 'naive' assumption that each feature is independent of one another. Gaussian naive Bayes assumes that the values for each class follow a Gaussian distribution. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood. Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting.

3.2. Model Implementations

Due to our paper being experimental in nature, we chose not to create our own implementations of the algorithms, but rather to use existing implementations found in packages in Python and R. Using the Python package *scikit-learn*[1], we implemented and trained decision trees, random forests, extra trees, naive bayes, and SVMs classifcation models. In R, we accessed its interface to *libsvm*[2], a libray for Support Vector Machines, using package *e1071*.

SVMs: We use the following SVMs in scikit-learn: SVM with a linear kernel, SVM with a radial basis kernel with width $\left\{1 \times 10^{-6}, 1 \times 10^{-5}, .0001, .001, .01, .1, 0\right\}$, and linear SVM. We also vary the regularization parameter by factors of ten from 1 to $10,000$ with each kernel. We also used the R package e1071, which contains an implementation for libSVM, to run an SVM with a linear kernel.

Decision Trees: We use the default parameters for the decision trees in *scikit-learn*.

Random Forests: We use the default number of trees in the random forest, 10. There is no maximum depth, the minimum number of samples required to split an internal node is 1, and a random state seed of 0 for the random number generator.

Extra Trees: We fit a number of randomized decision trees on various sub-samples of the dataset using the same parameters as we did for random forests. We averaged the accuracies over the trees to improve efficacy and control over-fitting.

Naive Bayes: We used the Gaussian Naive Bayes implementation assuming the likelihood of each feature was Gaussian and all of the default parameters.

3.3. Parameters

Machine learning models are parameterized so that their behavior can be tuned for a given problem. We used *scikit-learn* to tune the parameters of each of the machine learning algorithms. The Python library utilizes grid search which is a approach to parameter tuning that methodically builds and evaluates a model for each combination of algorithm parameters specified in a grid and returns the best results.

[1] http://scikit-learn.org/

[2] http://www.csie.ntu.edu.tw/c̆jlin/libsvm/

3.4. Dataset

We compared the algorithms using their performance on the binary classification problem of predicting the cancer type of leukemia patients as having either acute myeloid leukemia (AML) or acute lymphocytic leukemia (ALL). The data was adopted from the same dataset used in Golub et al.'s paper running classification algorithms on leukemia patient data. We had information on 72 patients, with expression values for 7129 genes for each patient and class labels -1 for ALL and 1 for AML.

3.5. Performance Metric

Having trained the classifiers, we tested the models on our dataset using n-fold cross validation. We ran the cross validation using n values from 2 to 10. We calculated the accuracy of each by taking the average accuraciess across each fold. For further understanding, we calculated the confusion matrix per fold.

Having obtained our accuracies, we test whether any differences are statistically significant by running paired t-tests for the SVM with a linear kernel, libSVM in R, and decision trees against all other algorithms. A paired t-test is used to compare two population means where you have two samples in which observations in one sample can be paired with observations in the other sample.

4. Results

A summary of the performance of our classifiers is shown in Figures 1 to 4. Figure 1 shows the accuracy of the Gaussian Naive Bayes classifier using n-fold cross validation. For brevity, we grouped together the performance of the classification tree algorithms together in Figure 2. Figure 4 displays the respective accuracies the three implementations of a SVM using a linear kernel. Figure 3 shows the accuracy of the SVM using a rbf kernel.

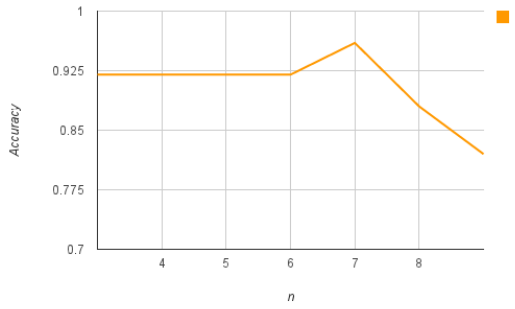

Figure 1. Accuracy of Naive Bayes using n-Fold Cross Validation

The accuracies for each classifier are plotted across the value n to better indicate which n best optimized the dataset and algorithm. For Naive Bayes, Decision Trees, and Random Forests, we see a peak and see that that $n = 7$ produces the best results. Extra Trees seem to plateau as n increases.

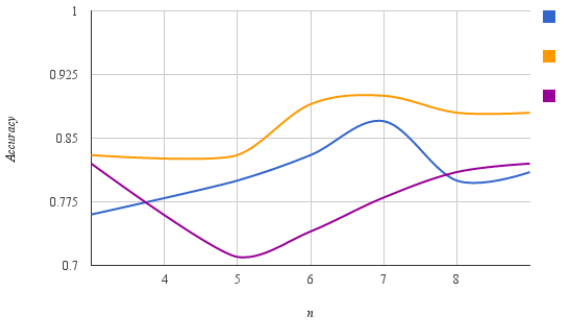

Figure 2. Accuracy of Decision Trees (blue), Random Forests (yellow), and Extra Trees (purple) using n-Fold Cross Validation

Overall, these four algorithms mentioned perform relatively well. Naive Bayes outperforms the three classification tree algorithms achieving the highest accuracy of 96% at $n = 7$.

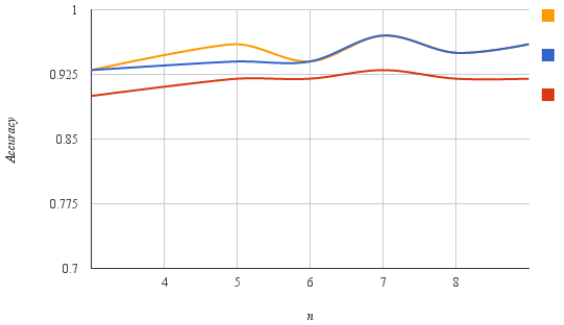

Figure 3. Accuracy of SVM linear (yellow), LinearSVM (blue), LibsVM (red) using n-Fold Cross Validation

Our SVM with a linear kernel performed the best out of all the algorithms we implemented and tested with the highest accuracy of 97%. Our results indicate that $n = 7$ also yielded the best results. *Scikit-learn*'s SVM with a linear kernel and Linear SVM implemtations perform similarly as see in Figure 4. Similar to SVM with a linear SVM, Linear SVM is implemented in terms of liblinear instead of libsvm so it has more flexibility in the choice of penalties and loss functions. Thus, it tends to scale better to large numbers of

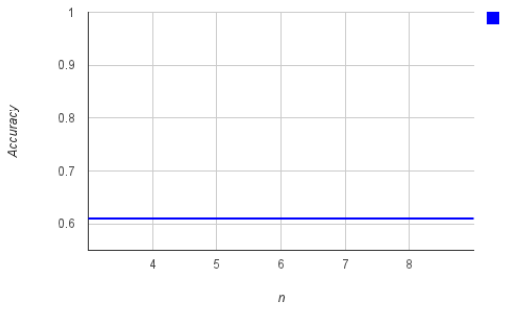

Figure 4. Accuracy of SVM RBF using n-Fold Cross Validation

Table 1. P-Values for the paired t-tests between selected algorithms

	SVM LINEAR	LIBSVM	DECISION TREES
LINEAR SVM	.3632	.000435	5.607×10^{-5}
LIBSVM	.0001709		.0002399
SVM RBF	3.184×10^{-8}	7.083×10^{-9}	3.978×10^{-5}
DECISION TREES	6.743×10^{-5}	.0002399	
RANDOM FORESTS	.000766	.004105	.001356
EXTRA TREES	.000434	.0011	.2958
NAIVE BAYES	.0655	.4791	.006225

samples. LibSVM implementation performs slightly worse compared to these two classifiers but still better than the classification tree algorithms.

Notably, the accuracy of the SVM with RBF kernel performed the worst. It consistently yielded an accuracy of 60%. Using different parameters for γ and C did not change the result as we saw when trying to tune the classifier. This suggests that our dataset is linear.

To see whether the differences in accuracies were statistically significant, we ran paired t-tests between the SVM with a linear kernel, libSVM, and decision trees against all of the other classification algorithms. The paired t-test determines the probability that our data supports the null hypothesis that the accuracies of the algorithms is the same against the alternative hypothesis that the accuracies are different. In Table 1, we report the various p-values for each paired t-test. A p-value of $p < 0.05$ allows us to reject the null hypothesis that the accuracies for the two methods are the same, whereas any $p \geq 0.05$, we cannot reject it. Table 1 shows that save for a few t-tests, the three algorithms have statistically significant differences in accuracy. SVM with a linear kernel and linear SVM have similar accuracies, along with SVM linear and libSVM with Naive Bayes, and decision trees and Extra Trees. For these, we cannot reject the null hypothesis that the accuracies are the same.

5. Conclusion

This paper has discussed and compared various classification algorithms, including decision trees, random forests, extra trees, naive bayes, and SVMs with different kernels and parameters. These classifiers were tested on data containing class labels for two tumor types for patients with leukemia.

The above analysis shows that for a small dataset, SVMs have higher accuracies classifying leukemia patients than do the other algorithms we tested. With excellent performance on the entire range of n-folds, SVMs with linear kernels was the best classification algorithm for our specificat dataset with the higest accuracy of 97%. Naive Bayes was a close second, followed by random forests and decision trees. The methods that performed poorest were SVM with an RBF kernel and Extra Trees.

6. Future (and Past) Work

For future work, additional comparisons can be made to other supervised learning algorithms. For example, SVM with linear kernel is usually comparable with logistic regression. It would be interesting to see how they compare. For our comparisons, we overlooked other algorithms like neural nets and bagged trees but should revisit for a complete comparison of all common classification algorithms.

A more interesting approach to future works would be to test and compare the performance of these algorithms on other datasets. With minimal tuning, all of our classifiers achieved accuracies of 60% or higher. Half of them yielded accuracies over 90% which is notably high with classification problems. This suggests that our dataset was particularly easy to classify. When exploring the dataset, we used SVM's feature ranking method to identify the most revelant features. Over 6000 of the 7000 gene expressions were discarded. It will be very interesting to conduct our experiment on more complicated gene expression data in cancer classification. In addition, SVMs can be used on multiclass classification problems such as comparative genomic hybridixation.

Another example of a complex classification problem involves the use of SVMs in the prediction of hot spot residues at protein-protein interfaces. We originally tried to tackle this problem as introduced by in the work done by Lise et. al. (2009) (2011) [9] [10]. In their study they used a machine learning approach, specifically support vector machines. As input variables, they used basic energy terms such as van der Waals, hydrogen bond, electrostatic and desolvation potentials derived from protein completx structures. They distinguished contributions from three different structural regions in protein complexes: side-chain inter-molecular, environment inter-

molecular and side-chain molecular. To each of them, they associated four input features corresponding to the energy terms jsut described resulting in a total of 12 input features per protein structure.

We attempted to duplicate their work and compare the performace of SVMs to other machine learning algorithms. Although we were able to obtain the necessary protein structures from the Protein Data Bank [3] and their corresponding alanine data, we could not generate the necessary input features. Following their work done in (Lise, 2009), we attempted to use CHARMM19 force fields to obtain van der Vals interactions and electorstatic interactions energy. However, we lacked the needed cofiguration files and those parameters were not specified in the paper. In addition, Lise et. al. applied in-house methods to obtain energy values which we did not have access to. The third-party software and scripts we attempted to use such as the Mitchell Lab's Fast Atomic Density Evaluator (FADE) [4], the MDEnergy program [5] and the Volumne, Area, Dihedral Angle Reporter (VADAR) [6] failed to yield applicable results. Without features we were unable to implement an SVM modeled to predict protein hot spots. We encountered a problem inherent to bioinformatics—as increasingly large amounts of genomic information, including both genome sequences and expressed gene sequences, becomes available, more efficient, sensitive, and specific analyses become critical. Without applicable and effective analytical tools, the wealth of data available proves lackluster.

7. Acknowledgments

We would like to thank T.R. Golub for the idea behind this project, Jeff Knerr for his unwavering and dedicated technical support, and Ameet Soni for his guidance through this difficult project.

[3] http://www.rcsb.org/

[4] http://www.mitchell-lab.org/FADE.php

[5] http://www.ks.uiuc.edu/Development/MDTools/mdenergy/

[6] http://vadar.wishartlab.com/

References

[1] S.R. Amendolia, G. Cossu, M.L. Ganadu, B. Golosio, G.L. Masala, G.M. Mura, "A Comparative Study of K-Nearest Neighbor, Supporton Applied Computing, Cyprus, 2004.

[2] N.B. Amor, S. Benferhat, Z. Elouedi, "Naive Bayes vs Decision Trees in Intrusion Detection Systems," ACM Symposium on Applied Computing, Cryprus, 2004.

[3] L. Breiman, "Random Forests," Machine Learning, Vol. 45, No. 1, pp. 5-32, 2001.

[4] W.H. Chen, S.H. Hsu, H.P. Shen, "Application of SVM and ANN for Intrusion Detection," Journal of Computers and Operations Research, Vol. 32, Elsevier, 2005, pp. 2617-2634.

[5] P. Geurts, D. Ernst, L. Wehenkel, "Extremely randomized trees," Machine Learning, Vol. 63, No. 1, 2006.

[6] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander, "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," Science, Vol. 286, 1999.

[7] J. Huang, J. Lu, C.X. Ling, "Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy," Proceedings of the Third IEEE International Conference on Data Mining , 2003.

[8] Y.S. Kim, "Comparison of the Decision Tree, Artificial Neural Network, and Linear Regression Methods based on the Number and Types of Independent Variables and Sample Size," Journal of Expert Systems with Application, Elsevier, 2008, pp. 1227-1234.

[9] S. Lise, C. Archambeau, M. Pontil, D.T. Jones, "Prediction of hot spot residues at protein-protein interfaces by combining machine learning and energy-base methods", BMC Bioinformatics, Vol. 10, pp. 365-475.

[10] S. Lise, D. Buchan, M. Pontil, D.T. Jones, "Prediction of Hot Spot Residues at Protein-Protein Interfaces Using Support Vector Machines", PLoS One, Vol. 6, No. 2, 2011.

[11] M. O'Farrell, E. Lewis, C. Flanagan, W. Lyons, N. Jackman, "Comparison of k-NN and Neural Network Methods in the Classification of Spectral Data from an Optical Fiber-Based Sensors System used for Quality Control in the Food Industry," Journal of Sensors and Actuators B, pp. 354-362, 2005.

[12] Z.R. Yang, "Biological Applications of Support Vector Machines," Briefings in Bioinformatics, Vol. 5, No. 4, pp. 328-338, 2004.