**A Novel Hybrid Approach to Machine Learning**

**Muthukaruppan Annamalai, Ankur Jain, Vaishnavi Sannidhanam**

[{muthu,ankur,vaishu}@cs.washington.edu](mailto:%7b%7d@cs.washington.edu)

Department of Computer Science and Engineering,

University of Washington,

Seattle, Washington 98195

**Abstract**

Learning is one of the most powerful concepts in artificial intelligence research. It allows for a system to learn from its environment, and automatically modify its behavior to suit the needs. The world’s best computer backgammon player [10] that is on par with human champions is a computer program that learns by playing against itself. Computer learning is limited by the learning algorithm utilized and the data set available.

A number of techniques are available to perform machine learning. Decision trees [5-9], the naïve Bayes approach [11-17], and the more general Bayes net approach are a few of the choices. The naïve Bayes approach is an instance of the more general Bayes nets. This paper examines and analyzes the naïve Bayes and decision tree approaches to learning. Various techniques to avoid over-fitting, such as ensemble construction and cross-validation are also implemented and analyzed.

A novel approach that is a hybrid between the naïve Bayes approach and the decision tree method is presented. The hybrid approach produces a spectrum of options that could be used for learning by merely changing parameter values. At one end lies the naïve Bayes approach, while at the other lies the decision tree technique. The proposed hybrid scheme solves the problem of poor naïve Bayes performance in a domain with dependent attributes, and the memory consumption problem of the decision tree. We analyze this idea and show encouraging experimental data that backs the need for such a solution.

**1. Introduction**

Learning, being one of the most prominent fields of artificial Intelligence research, has been extensively worked on. The learning component of a system allows it to automatically learn from the environment and modify its behavior. This is different from the conventional notion of a deterministic program. Learning techniques have been employed in a wide range of domains. Current spam filtering systems [2] and the world’s best computer backgammon player [10] employ learning.

Traditional methods of learning include the naïve Bayes model, learning a Bayes net structure, and decision trees. The naïve Bayes approach uses a model that can be constructed very fast and is suited for learning when the attributes to be learnt upon are independent of each other. Performance is poor if the attributes exhibit dependencies. A more complicated Bayes net structure could be learnt in such a case if the independence property does not hold amongst attributes. This gives rise to more accurate predictions, but the actual construction takes large amounts of time. As for a decision tree, while construction is a simple and easy-to-understand process, it is rather time and memory consuming. Memory consumption problems are tackled with the use of pruning techniques.

Various kinds of approaches to enhance the performance of the above mentioned techniques exist. These include boosting, cross-validation, and ensemble construction. These techniques tend to solve the problem of over-fitting. This would arise when a learning algorithm is too concerned and takes a narrow approach to learning based on the data set used for training. This would lead to false predictions when the algorithm is allowed to predict on new unseen data.

In this paper, we implement and examine the naïve Bayes approach along with the decision tree approach. In order to study the effects of boosting, cross-validation and ensemble construction, we chose decision trees and implemented these techniques for them. A novel hybrid approach to learning using decision trees and naïve Bayes is proposed. As shown in Section 6, the hybrid approach displays a spectrum of solutions to learning. At one end of the spectrum lies the naïve Bayes method, while at the other end lies the decision tree approach. In other parts of the spectrum, the two techniques are combined by learning a decision tree using pruning methods, and including a Naïve Bayes model at the leaf nodes, consisting of the remaining attributes that have not been branched upon. This method takes advantage of the speed of the naïve Bayes approach, and uses the decision tree in order to break as many dependencies as possible in order to suit the data to the naïve Bayes assumption of attribute independence. Pruning on a decision tree is done due to memory concerns, and to avoid over-fitting. If the amount of data to be learnt, and the number of attributes are large, then pruning is required to keep the tree at a manageable size. This would result in the loss of data. In such a case, the use of a naïve Bayes model at the leaf nodes would allow for better predictions by recovering some of the lost data. It would be fast and not memory consuming, thus leading to an overall good predictor.

Section 2 examines related work, while Section 3 contains the details of decision trees and their construction process. Section 4 discusses the naïve Bayes approach and Section 5 discusses techniques used for improving the performance of the naïve Bayes and the decision tree techniques. Section 6 proposes our novel hybrid technique and explains the intuition and the need for such an approach. In section 7 we evaluate the naïve Bayes and Decision Tree classifier and the effect of various improvements. Experiments also show that the Hybrid inherits desirable properties from both classifiers. We conclude and discuss future work in Section 8.

# **2. Related Work**

Machine learning was a result of the quest for mimicking human intelligence. It has received tremendous amounts of attention in artificial intelligence research. [1] examines various approaches to machine learning, and classifies the approaches under three broad categories: (i) data mining, (ii) neural network and (iii) reinforcement learning techniques. Learning is applied to a variety of domains such as spam filtering [2,3] and games [10]. [2] describes SpamAssassin, which is a mail filter that uses text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases. A part of our experimental database was from [2].

This paper concentrates on decision trees [4-9] and the naïve Bayes approach [11-17]. Decision trees are an extremely easy-to-understand method of classifying training data, and using the classification to predict. [4,9] contain approaches to choosing attributes to split upon in decision trees. [4] presents a family of measures called C-SEP (Class Separation). The method utilizes the cosine of the angle between vectors associated with the nodes in the tree. [9] discusses an approach that splits on an attribute with maximum information gain.

Decision trees consume memory at rapid rates. Pruning [5-9] is required to keep memory utilization at modest levels. [6] is a backtracking approach that grows the tree and prunes as required. Though the final tree may be small, the process dictated in [6] is memory consuming during the tree growth process. [7] discusses pruning from the viewpoint of increasing simplicity. A complex yet accurate tree is pruned to increase its simplicity, hence enhancing understanding. Pruning should not tradeoff the accuracy beyond tolerable limits though. [8] takes the interesting approach of searching for pruned trees in a search space. [9] contains an approach that uses a statistical significance test, chi square value cut-offs, to prune trees. [5] is a general comparison of numerous approaches.

The Naive Bayes Classifier has been well-studied, even in the domain of spam. [11] advocates the use of specific domain knowledge to get better recall and precision. In the context of spam-classification, the authors manually identify about 20 non-phrasal domain specific features such as overemphasized punctuation, time of receiving, number of attachments, subject lines, etc. We use a similar set of rules (from [2]) to preprocess emails and parameterize them along 613 attributes. Along with [13], [11] does a detailed evaluation of Naïve Bayes-based spam classifiers. [12] proposes heuristics like Complement NB (to deal with skewed training data) and introducing weights (to deal with dependence) to improve the accuracy of Naïve Bayes text classifiers; however many of their heuristics assume multinomial attributes, which we already get rid off during our pre-processing stage. Finally, we implement in our Naïve Bayesian classifier many simple hacks suggested in [15-17] by authors out of practical experience of designing spam classifiers. These include setting priors, weighting to deal with skewed data, etc.

**3. The Decision Tree Approach**

Decision trees have been used for a variety of purposes such as pattern matching and machine learning. The reader is encouraged to examine the references provided for an in-depth discussion of decision trees. A brief description is provided here.

A decision tree essentially classifies the training data into sets. These sets are formed by branching on the attribute values that the examples in the training data**.** A naïve decision tree would sequentially branch on all attributes. Techniques to handle problems with the naïve decision tree are discussed in Section 5. A perfect decision tree would be structured such that all the training examples at a node in the tree are all of the same classification. But this rarely happens since there would always be a few outliers due to noise. The prediction at the node is then the majority of the classification of the various examples with biasing incorporated if required. When the formation of the tree is completed, prediction can take place. Given a piece of data, we traverse a path from the root to the leaf of the tree by taking edges corresponding to the value that the data depicts for the attribute at the node. The prediction of the leaf node is then the prediction that is returned.

As the number of attributes increases, it clearly is not possible to use the naïve technique. This is due to the explosion of nodes if we branch on every attribute. For example, if we have 20 binary attributes, then the total number of leaf nodes alone would be 220. The chi square value [9] is used as a means of testing for statistical significance. The attribute that is to be chosen for branching is usually chosen by trying to maximize on the amount of information gain.

**4. The Naïve Bayes Approach**

Naïve Bayes classification is used widely because of its simplicity, efficiency and excellent performance in a large variety of applications, including text-classification and spam detection. Naïve Bayes estimates the probability that an instance *x* belongs to class *y* as

*P*(*y|x*)*=P*(*y*) *P*(x*|y*) (1)

*P*(*x*)

= *P*(*y*) Πi *P*(*xi*|*y*) (2)

*P*(*x*)

and predicts the class with the highest value of *P*(*y|x*).

Step (1) is simply the Bayes theorem and (2) is the Naïve Bayes assumption. The latter assumption is made because it is in general very difficult to learn *P*(*x*|*y*) for all *x*, and in contrast much easier to learn *P*(*xi|y*). For binomial attributes *xi*, this can be done by simply counting the number of occurrences in the training set S, of *xi* in each class *y*, and the number of instances in each class.

Even in this basic form, the naïve bayes classifier performs reasonably well. Its biggest weakness though lies in the very assumption that makes it so simple – that the attributes are independent of each other. There have been many heuristics that have been proposed in the literature [Section 2] to improve the classifier’s performance in domains where the assumption leads to bad performance. Its other drawback is that it does not work well when the training data set is skewed, for instance when we get no training instance for a particular class. For such cases, as suggested in [15], we assign *P*(*xi|y*) = ε>0.

Finally, in the spam-detection domain, there are only two classes (*i.e.* spam and ham) and a false positive is much more expensive (say λ-times) than a false negative. We predict a test example as spam when *P*(*y|x*)*/P*(*ŷ|x*) > λ. [13] suggests how to choose λ depending on the particular configuration in which the spam classifier is deployed; heeding to it we have used 9 for spam-classification. Note that λ=1 corresponds to choosing the class with the highest (“higher”, since there are only two classes) conditional probability.

**5. Improvements**

Over-fitting [9] is often a problem for learners. Given a set of training data, the learner would tend to take a narrow approach and learns such that its predictions for the given training set is accurate, while it fares poorly on unseen data. This is usually due to either outliers or insufficient representation of the various classes in the data that misleads the construction of the learner. Such a problem can be solved through a variety of approaches. The following subsections discuss a few: (i) cross-validation, (ii) ensemble and (iii) pruning. Pruning is only relevant in the context of decision trees. [9] discusses all these approaches in greater detail.

**5.1 Cross-Validation**

Cross-validation is the process of constructing a set of learners from a given training set and choosing the best of the generated set. A parameter, k, to indicate the number of learners to be constructed is required. The given training data set is divided into k different parts. To construct the learner, one of the k parts is used as the test data, while the rest k-1 are used for the training data. Each tree has a distinct test data set. In this manner, k learners are constructed, and the best of them is chosen.

The intuition behind cross-validation is that it would look for the most generic decision learner that performs well. In this manner, it is ensured that the learner does not learn unwanted patterns. It is to be noted that this is not a fool-proof technique, since a particular learner may perform very well on a given test set, but the test set may not be representative of the entire domain.

**5.2 Ensembles**

Another method to limit over-fitting is the ensemble construction method. This involves the construction of multiple learners, with each one contributing to every prediction. The AdaBoost [19] algorithm was used for the purposes of ensemble construction in this paper. This technique trains a learner based on the input training data. It then evaluates the performance of the learner on the same training data. The examples that were incorrectly predicted are then given a higher weight. Learning takes place again, and the learner construction process concentrates more on the accuracy of examples that have a high weight. This process carries on. Each learner has a weight associated with it, and when a prediction is required, the output of each learner is then weighed appropriately and a final prediction formed.

The ensemble method takes advantage of the property that the probability of a majority of learners being wrong is lower than that of a single learner being wrong. This property of course requires the errors to be independent in each learner, which is clearly not true. Theoretically, as the number of learners grows, the number of erroneous predictions decreases.

**5.3 Pruning**

Pruning serves two purposes while constructing Decision Trees: (i) it ensures that the final tree has not learnt unwanted patterns and (ii) it restricts the amount of resources consumed. Pruning can be done during the learning process or after the learning process (post-pruning).

There are various approaches to pruning [5-9]. This paper uses the one suggested in [9]. It uses a statistical significance test based on chi value cutoffs to determine whether branching on an attribute would provide benefits beyond a threshold. Excessive pruning would result in the loss of data. Such pruning would be required to fit a really large decision tree into memory.

**6. The Hybrid Approach**

The Hybrid classifier tries to capture the desirable properties of both the Naive Bayes and the Decision Tree based classifiers.

A Naive Bayes Classifier is trivial to implement, requires little memory and is very quick to learn from a training set. Its biggest drawback however, is the large number of incorrect classifications it does when the attributes are dependent. If somehow the dependent attributes could be identified, and for each such pair, only one of them used both in the learning and the inferring stages, we might be able to cut down on the misclassifications.

Decision Trees classifiers, unless pruned, even for a small number of attributes require prohibitively large memory and running times. Pruning the tree by using moderate-to-large χ-values gets around this problem, but introduces another one - quite often a large number of training examples get cluttered into the same leaf node, because no matter which attribute one chooses, the information gain is never sufficient to split the node. Such leaves quite often have training instances of many different classes and therefore end up misclassifying test examples of all classes except the majority class. What is needed is a more ``intelligent'' inference mechanism at such nodes, that does a better job at classifying than just assigning the class with the maximum number of instances.

A side-effect of using the information-gain heuristic which we exploit is that if there are two attributes dependent on each other, then we can hope that one of them would be chosen to split a node of the decision tree. As a result, for attributes left in the leaf nodes, it would be more acceptable to make the independence assumption (that when all attributes were considered together), as other attributes that some of the attributes in the leaves were dependent on, would have been selected higher up in the tree to split a node.

The Hybrid Classifier combines the Decision Tree Classifiers' propensity to separate out dependent attributes, and the effective classification by the Naïve Bayes Classifier on independent attributes.

The idea is simple. In the learning phase, the Hybrid Classifier grows a tree exactly like the Decision Tree. The only difference is at the leaves, where a naive Bayes learner is now implemented. This Naive Bayes learner learns only on the training examples that arrive at that particular leaf, using only those attributes that have not been used by the Decision Tree along the path from the root to the leaf. During the inference phase, just as in Decision Trees, the attribute values of the test example determine the path that it takes down the tree and hence the particular leaf node that it reaches. The decision at the leaf node is taken by the Naive Bayes classifier based on the attributes of the

test which have still not been considered.

This classifier is parameterized by the χ-value used for cut-off. If it is very high, then there exists no attribute that will give an information gain sufficient to split even the root node; making the root node itself as the Naive-Bayesian leaf node. On the other hand, if the cut-off is zero, then we get exactly the same tree as is constructed by Decision Trees. Moreover, getting a zero information gain upon choosing any attribute means that the number of that all instances in the leaf node belong to the same class, in which case, the Naive Bayesian leaf node will just return the class that the leaf would have returned in a Decision Tree.

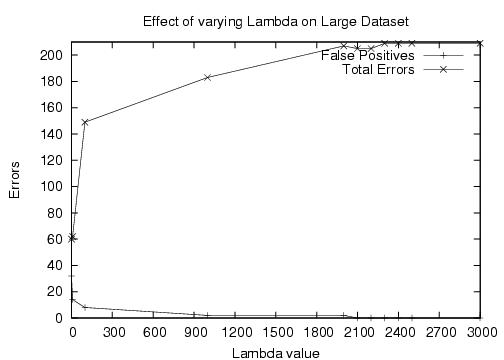
**7. Evaluation**

This section begins with the evaluation of the Naïve Bayes Classifier and the Decision Tree classifier. We then discuss the effect of using modifications such as cross-validation and ensembles. Finally, we evaluate our Hybrid Classifier.

We used the following two datasets in our experiments. Spamassasin[2]is an email corpus containing 18110 emails out of which around 10000 are spam and the rest ham. Spamassasin contains two datasets, a *SMALL* one and a *LARGE* one. The *SMALL* dataset constitutes of 2970 examples for training and 330 examples for testing. The *LARGE* dataset contains 16299 examples for training and 1811 examples for testing.

* 1. **Evaluating Naïve Bayes Classifier**

In its basic form, the Naïve Bayes classifier got 60 misclassifications (32 false positives and 28 false negatives) on the *LARGE* dataset, and 23 misclassifications (all false negatives) on the *SMALL* database.

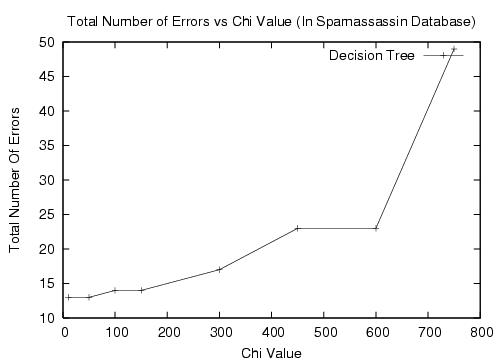


**Figure 1: This graph shows the variation of total misclassifications and false positives with λ. False negatives is the difference between the two curves.**

The cost of misclassifying is not symmetric in classifying spam – false positives are more expensive than false negatives. In order to tune the performance of the classifier, we study the effect of varying λ on the number of false positives and negatives. As expected, as λ grows, the number of false positives decreases, while the number of false negatives increases. Unfortunately, their sum (the total number of misclassifications) also increases – growing to almost 200 (11%) by the time the number of false positives comes down to 0.

* 1. **Evaluating Decision Trees**

The chi value [18] was set at 0.5 for these experiments. The decision tree had a total of 13 erroneous predictions for the *SMALL* dataset, out of which all 13 were false negatives. There were a total of 72 errors for the *LARGE* dataset with 41 of them being false negatives and the other 31 being false positives. Figure 2 shows the curve that represents the change in the total number of errors with χ when the SMALL dataset was used. The graph takes the shape



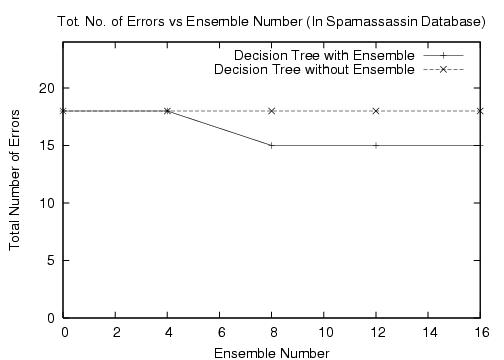
**Figure 2: This graph shows how the total number of errors varies with** χ **.**

of an exponential curve, with the exception of a plateau. We observed that as χ increases, the number of nodes pruned increases exponentially. Hence, the amount of information loss, which is proportional to node loss, is also exponential in nature, resulting in a large number of errors. The plateau in the graph is due to a region of χ values that have very few nodes between them. This is a property of the dataset.

* 1. **Effect of Improvements**

This section evaluates the performance of ensembles and cross validation. A naïve learner suffers from the problem of learning on a concentrated training set. This narrow approach would imply that the performance of the learner on unseen data might be poor. To avoid such over-fitting, we use techniques like ensembles and cross validation. We evaluate these approaches for decision trees.

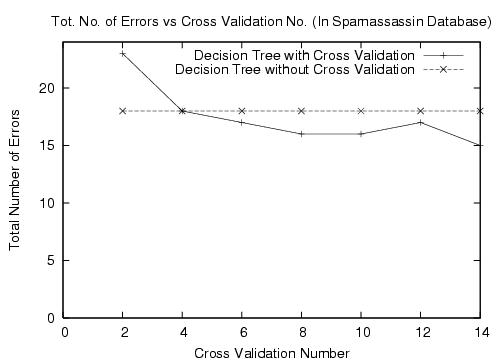
Figure 3 shows the effect of using ensembles. The *SMALL* dataset was used. Chi square values were set at 0.5 for all experiments in this section. The general



**Figure 3: This graph depicts how the number of errors varies as the ensemble number increases.**

trend to note is that of the reducing number of total errors. When ensemble number is less than or equal to 4, there are an insufficient number of trees for the ensemble to make a difference. This is due to the large number of trees required in order to correctly classify a hard-to-learn example. This is why the normal decision tree and the ensemble have identical performance till ensemble number 4. After 8 trees in the ensemble, the total number of errors remains constant. This indicates either one of two possible scenarios. One possibility is that the remaining misclassified examples are very difficult; hence require a large number of trees. This is not likely, since even with 16 trees no difference is observed. The second possibility can be due to outliers in the data set and other sorts of inconsistencies in the dataset which make fault-free prediction difficult.

Figure 4 depicts the manner in which the total number of errors decreases as the number of trees in the cross validation increases. The general trend is as expected with the total number of errors decreasing with the number of trees in the cross validation algorithm. There are however two irregularities worth noting. Firstly, for

****

**Figure 4: This graph depicts how the number of errors varies as the cross validation number increases.**

cross validation number equal to 2, the number of total errors is higher than for a single simple tree. This is a property of the data set. When the data set is split randomly, it could happen that split sets are not representative of the domain. It should also be noted that the number of examples that a cross validation tree trains upon is fewer than the normal tree. This is due to the fact that we have to set aside a portion of the training data for testing during the cross validation process. The second irregularity occurs when cross validation number is equal to 12. This performs worse than with cross validation number equal to 10. This again is attributed to the random splitting of the data set. This random splitting in cross validation can lead to a small increase in the total number of errors. But from our experimentation, we observed that such increases in misclassifications are small in magnitude and are rid off when the cross validation number is increased further.

* 1. **Evaluating the Hybrid Classifier**

In this subsection, we evaluate the performance of the Hybrid Classifier. To best illustrate the effect of going in for the hybrid, we turn off cross-validation and ensemble.

Figure 5 shows how the number of incorrect predictions made by the Hybrid classifier varies with χ for the two datasets. It also plots a corresponding curve for decision trees and a horizontal line representing the Naïve Bayes numbers.

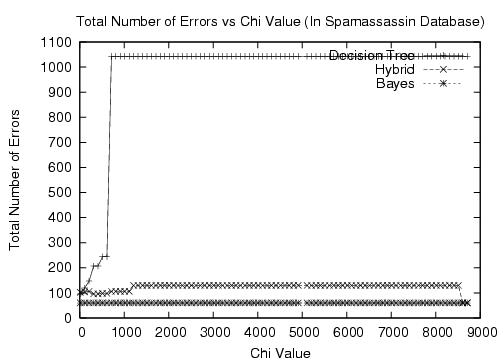
On the large dataset, we observed that Naïve Bayes performs better than Decision Trees, irrespective of the χ-value we choose. The curve for Hybrid classifier, however, performs exactly as the Naïve Bayes classifier for χ ~ 9000.

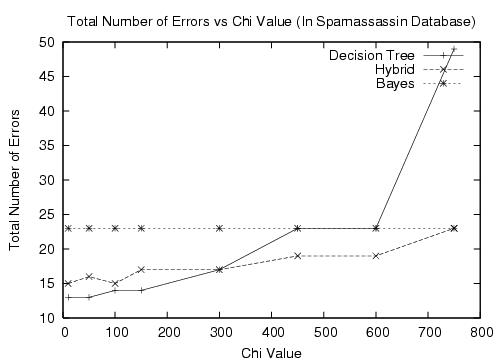
On the other hand, on the small dataset, Decision Trees perform better than the Naïve Bayes classifier, potentially due to greater dependencies in the attributes. Here, the Hybrid classifier behaves exactly like the Decision Trees for χ ~0.

Thus, the Hybrid classifier comes across as a very flexible and powerful classifier, which gives us the best of both the worlds – Naïve Bayes and decision trees – of course, depending on the choice of χ. We envision the Hybrid classifier to run *cross-validation* over different values of χ and choose the best value for a given dataset

Cross-validation would prove especially useful in datasets where neither Naïve Bayes nor Decision Trees perform well, and instead there is some intermediate value of χ for which the Hybrid classifier outperforms them both. Unfortunately though, neither of the two datasets that we considered showed this trend.

There are other advantages of using the Hybrid over these other classifiers. Decision trees experience an exponential blow-up in their memory requirement as the χ-value decreases and the tree depth increases. And so does the Hybrid classifier. Figure 6 shows this blow-up for both the datasets by plotting the number of nodes in the tree. Hybrid





**Figure 5: This graph depicts how the number of errors varies for the three classifiers for different χ values on the large (above) and small (below) spam datasets.**

classifiers however perform, nearly as well as decision trees, even for higher χ-values. Hence, given a decision tree with some χ value v1, we can obtain a Hybrid classifier, with χ value v2, such that v1 < v2 but the Hybrid still performs as well as the decision tree. Since v1 < v2, the hybrid would have fewer nodes than the corresponding decision tree. Thus if memory is a constraint, we get nearly the same performance as decision trees without running into the memory problems that the latter does.

# 

**Figure 6: This graph plots the memory requirement for different χ values on the two datasets.**

**8. Conclusion and Future Work**

We have discussed, implemented and analyzed decision trees and the naïve Bayes approach to machine learning. We discussed problems with such learning techniques. Optimizations, such as cross validation, ensembles and pruning, to eliminate problems such as over-fitting, were discussed and implemented.

A novel hybrid approach to machine learning was also presented. The hybrid approach produces a spectrum of solutions that can be used to learn. At one end of the spectrum lies the naïve Bayes approach, while at the other end lies the decision tree approach. All points elsewhere in the spectrum constitute of a decision tree with a naïve Bayes model at the leaf nodes. This hybrid approach provides excellent learners for domains with a large number of dependent attributes. The large number of attributes would make the domain not suited to decision trees, while naïve Bayes will not work well either due to the lack of independence amongst attributes.

Future work can concentrate upon examining if a naïve Bayes approach is actually required at each leaf node, or if a naïve Bayes at just a subset of the leaf nodes would be sufficient. Another interesting question is whether the naïve Bayes model at a leaf node should contain all the remaining attributes, or if a large number of irrelevant attributes can be eliminated. There are numerous such questions that arise about the interaction of the hybrid technique with various parameters. The more the questions we answer, the greater the understanding obtained.

# **References**

[1] C. Burch, “A Survey of Machine Learning Techniques,” *Technical Report,* Pennsylvania Governor’s School for the Sciences, Pennsylvania.

[2] <http://spamassassin.apache.org/>

[3] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos and P. Stamatopoulos, “Learning To Filter Spam E-Mail: A Comparison of a Naïve Bayesian and a Memory-Based Approach,” *Technical Report,* University of Athens.

[4] U. Fayyad and K.B. Irani, “The Attribute Selection Problem in Decision Tree Generation,” *In the Proceedings of AAAI-92, 1992.*

[5] F. Esposito, D. Malerba, G. Semeraro, “A Comparative Analysis of Methods for Pruning Decision Trees,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 19, No.5, May 1997.

[6] S.B. Gelfand, C.S. Ravishankar, and E.J. Depl, “An Iterartive Growing and Pruning Algorithm for Classification Tree Design,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 13, No. 2, 1991.

[7] M. Bohanec and I. Bratko, “Trading Accuracy for Simplicity in Decision Trees,” *Machine Learning,* Vol. 15, No. 3, 1994.

[8] F. Esposito, D. Malerba, and G. Semeraro, “Decision Tree Pruning as a Search in the State Space,” *Machine Learning: ECML-93,* Berlin, 1993.

[9] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice-Hall International, Inc, 1995.

[10] G. Tesauro, “Temporal Difference Learning and TD-Gammon,” *Communications of the ACM*, 38(3), 1995.

[11] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz, “A Bayesian Approach to Filtering Junk E-Mail,” *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*.

[12] J. Rennie, “Improving multi-class text classification with Naive bayes,” *Proceedings of ICML*, 2003.

[13] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, “An Evaluation of Naive Bayesian Anti-Spam Filtering,” *Proc. of the workshop on Machine Learning in the New Information Age*, 2000.

[14] Yerazunis, W.S., "The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It", *MIT Spam Conference*, January 2004.

[15] Paul Graham, “A Plan for Spam,” Aug 2002. <http://paulgraham.com/spam.html>

[16] Paul Graham, “Better Bayesian Filtering,” Jan 2003.

<http://www.paulgraham.com/better.html>

[17] Greg Louis, “Refinements to Bayesian Filtering,” May 2003.

[18]http://www.richland.cc.il.us/james/lecture/m170/tbl-chi.html

[19] Shipp C.A. and L.I. Kuncheva. “An investigation into how AdaBoost affects classifier diversity,” *In the Proceedings of IPMU 2002, Annecy, France*, 2002.

# **Acknowledgements**

We would like to thank Dan Weld for his help in guiding the project in the right direction. We would also like to thank Sumit, for his advice on how to approach the spam classifier project.

# **Appendix**

All the code was written by us, and we did not download any code from anywhere.

The spam databases were obtained from spamassasin.org website.

We all worked on everything together, and did not split the tasks, as we felt the given time was sufficient, and that three brains at a task is much better than one.