# Machine Learning Models For Lung Cancer Classification Using Array Comparative Genomic Hybridization

**C.F. Aliferis M.D., Ph.D.[1], D. Hardin Ph.D.[2], P. P. Massion M.D.[3]**
[1]Department of Biomedical Informatics, [2]Department of Mathematics,
[3]Department of Medicine, Vanderbilt University, Nashville, TN

## Abstract

Array CGH is a recently introduced technology that measures changes in the gene copy number of hundreds of genes in a single experiment. The primary goal of this study was to develop machine learning models that classify non-small Lung Cancers according to histopathology types and to compare several machine learning methods in this learning task. DNA from tumors of 37 patients (21 squamous carcinomas, and 16 adenocarcinomas) were extracted and hybridized onto a 452 BAC clone array. The following algorithms were used: KNN, Decision Tree Induction, Support Vector Machines and Feed-Forward Neural Networks. Performance was measured via leave-one-out classification accuracy. The best multi-gene model found had a leave-one-out accuracy of 89.2%. Decision Trees performed poorer than the other methods in this learning task and dataset. We conclude that gene copy numbers as measured by array CGH are, *collectively*, an excellent indicator of histological subtype. Several interesting research directions are discussed.

## Introduction

Array comparative genomic hybridization (array CGH) is a recently introduced technology that measures gene copy number changes of hundreds of genes in a single experiment[1]. Non-small cell lung cancers (NSCLC) have been studied with array CGH and determined distinct molecular signature between histological subtypes[2].

In general, recently-developed high-throughput genomic and proteomic methods such as cDNA arrays, oligonucleotide arrays and mass spectrometry[3] enable researchers to gather data on cellular processes at the molecular level with extraordinary speed and efficiency. At the same time, interpretation of high-density genomic data sets poses a number of computational challenges. These include signal processing to de-noise the data, imputation for missing values, and most importantly, data analysis per se. In this paper we address aspects of the data analysis task.

While significant experience has been gathered so far in the application of various statistical and data mining approaches in the analysis of gene expression Microarray Data[4] little is known about how well various machine learning methods perform with array CGH data.

The primary goal of this study was *to study the feasibility of using machine learning methods on such data to create models that classify non-small Lung Cancers (NSCLCs) as squamous carcinomas (SqCa) or adenocarcinomas (AdCa).* A related goal was to *compare several machine learning methods in this learning task.*

## Methods

1. Assays: DNA from tumors of 37 patients (21 squamous carcinomas, (SqCa) and 16 adenocarcinomas (AdCa)) were extracted after microdissection and hybridized onto a 452 BAC clone array (printed in quadruplicate) carrying genes of potential importance in cancer. Normalization of fluorescence ratios was conducted so that the mean of the middle third of ratios across the array was one[1].

2. Imputation: To address the problem of missing values we applied a K-Nearest Neighbors (KNN) method for imputation (i.e., filling-in of missing values) as follows: for each instance of a gene that had a missing value the case closest to the case containing that missing value (i.e., the closest neighbor) that *did* have an observed value for the gene was found using Euclidean Distance (ED). That value was substituted for the missing one. To compute distances between cases with missing values, if one of the two corresponding gene measurements was missing, the mean of the observed values for this gene across all cases was used to compute the ED component. When both values were missing, the mean observed difference was used for the ED component. We chose the above procedure because it is non-parametric and multivariate. We also experimented by imputing with more naïve approaches (such as imputing with the mean or with a random value from the observed distribution for the gene). Since these naïve approaches produced uniformly worse models, we will not discuss them further. We also experimented with the more sophisticated Caruana/Justresearch approach (whereby we iterated the KNN imputation until convergence was attained)[5], without further improvement over the non-iterative approach.

3. Statistical and Machine Learning Methods

3.1 Nominal regression: Nominal (a.k.a. "polychotomous") regression generalizes the Logistic Regression (LR) model of statistics to nominal response variables[6]. We used Nominal regression to sort genes by strength of univariate association with the outcome (cancer type) as explained in the Results section. In our experiments we used the SPSS implementation of Nominal regression [7].

3.2 Decision Tree Induction (DTI): DTI is a method for learning decision trees that best fit the data according to information-theoretic criteria and/or classification performance (including generalization performance) criteria. The decision tree representation is very expressive: it can model any classification function involving discrete variables. The representation consists of a tree in which each node corresponds to a predictor variable, each branch to a value (or range of values for continuous variables) and each leaf to a classification of the case (which is thus represented by the path from the root of the tree to the corresponding leaf). Because of the greedy search nature of tree construction, DTI may be trapped by local optima. Decision trees are convenient to use, are intuitively understood by humans, can be easily converted to sets of rules, and may suggest important variable interactions. In the reported experiments we used Quinlan's See5 commercial implementation of DTI (that uses extensions of the ID3 algorithm)[8, 9].

3.3 Support Vector Machines (SVMs): In their most commonly used form, SVMs are classifiers that search for hyperplanes that separate between data points in the data set. The separation seeks to maximize the distance between selected (boundary) data points of the different classes (the "support vectors"). Furthermore, non-linearly-separable data are transformed by projection to a higher-dimensional space in which separating hyperplanes can be found. The transformation is achieved through the use of functions ("kernels") such that optimization can be achieved by using only dot products of the original data vectors and not the whole data, which increases computational efficiency significantly. Widely-used kernels are polynomial kernels and Gaussian Radial Basis Function kernels. In contrast to most machine learning methods where optimization is based on steepest-ascent/descent hill climbing heuristic methods, in SVMs the optimization typically achieves a global maximum (for a chosen SVM class). Empirical evidence verifies that SVMs are in general, very robust to a low sample-to-feature ratio. In our experiments we used both code developed in our lab and the LIBSVM library in Matlab[10,11,12,13,14] with appropriate scripts to optimise the learning parameters.

3.4 Neural Networks (NNs): NNs are a method for learning continuous or discrete classification functions by representing variables and their associations as stochastic units interconnected by weighted links. In a typical feed-forward configuration, units are arranged in layers with the first layer corresponding to predictor variables, the last layer to the predicted variable(s) and the intermediate ("hidden") layers to complex associations among input variables. The units combine and transform the sum of inputs from other units so that an optimization procedure can optimize a fitness function over the target values at the output of the network. There exist several algorithms to train the networks, with back propagation (and several more gradient descent variants) being the most popular and widely tested ones. The most frequently-used fitness function is sum-of-squared-errors (SSE). Since the fitness function can be non-convex, ANNs trained by gradient-descent methods can be trapped in local minima. Under mild conditions a sufficiently large network can in the (sample) limit approximate any input-output function including highly non-linear classification functions. In our experiments we used the Mathworks Neural Network Toolbox library with appropriate scripts to optimise the learning parameters under the Matlab environment[9,14,15,16].

3.5 K-Nearest Neighbors (KNN). The KNN classifier does not build an explicit model of the data but classifies new instances by finding the *K* most similar train instances and assigning their most frequent class to the new instance[17]. The KNN classifier has good asymptotic properties and robust performance in a wide variety of classification tasks. In the reported experiments we used Matlab code developed in our laboratory.

4. Perfomance Metric and Evaluation

In all reported experiments we used accuracy (i.e., ratio of correct classifications over total number of classified cases) to evaluate the quality of the produced models. To make sure that we do not overestimate classification performance we employed a leave-one-out methodology[17] (i.e., build a separate model to classify each case in the data by excluding this case from the data, training in the remaining data and repeating the procedure for each case, averaging over all cases at the end).

## Results

1. Data Cleaning and Imputation

We derived descriptive statistics and examined variable distributions, validity of values and proportions of missing values. As explained previously, array CGH is a technology in formative stages of development. As a result a high percentage of missing values was observed in most gene measurements. We decided to create a protocol for gene inclusion/exclusion for analysis on the basis of three criteria: (a) percentage of missing values, (b) a priori importance of a gene (based on known functional role in pathways that are implicated in carcinogenesis such as the P13-kinase pathway[2]), and (c) whether the existence of missing values was statistically significantly associated with the class to be predicted (at the 0.05 level and determined by a $G^2$ test[6]). This last criterion reflects the concern that if the values of a gene are not missing at random, but due to some selection process, then use of the gene in a predictive model may bias the model). The gene selection protocol followed can be formalized as follows:

```
1. For each gene $G_i$ compute an indicator variable $MG_i$ s.t. $MG_i$ is 1 in cases where $G_i$ is missing , and 0 in cases where $G_i$ was observed
2. Compute the association of $MG_i$ to the class variable $C$, $assoc(MG_i, C)$ for every $i$. ($C$ takes values in {SqCa, AdCa})
3. Accept a set of important genes $I$
4. if $assoc(MG_i, C)$ is statistically significant
     then reject gene $G_i$
   else
      if $G_i \in I$
      then accept $G_i$
      else
           if fraction of missing values of $G_i$ is
              >15% then reject $G_i$
           else accept $G_i$
```

388 variables were selected according to this protocol and were imputed before analysis.

## 2. Univariate Predictors

To obtain an initial estimate of how difficult is this classification problem, we run nominal regression of the diagnostic categories as a function of each one of the 388 variables separately for each variable on the full dataset. Since this is an exploratory step and for convenience, we did not cross-validate accuracy and considered these estimates as upper bounds on the true value. Variables were then ranked according to (a) statistical significance and (b) according to predictive accuracy of the diagnostic category (we used predictive accuracy and not the regression coefficient because the later is not comparable from model to model). As expected from the nature of this model, the two lists were almost identical. The best gene had an accuracy of 72%. Each of the 50 highest-ranking genes had an accuracy greater than 60%.

## 3. Machine Learning Models

Table 1 shows the various Machine Learning models developed. As can be seen from the table, using the default settings of the See5 program, and the leave-one-out approach explained in the methods section we derived an accuracy for DTI of 56.8%. K-Nearest Neighbors was applied with leave-one-out for K=1,2,3. The corresponding accuracies were: 86.49%, 75.68%, and 51.35%.

We experimented with several Support Vector Machines. The simplest SVM model (Linear Support Vector Machines) had a leave-one-out accuracy of 83.7%. We also built polynomial-kernel and Gaussian Radial Basis Function-kernel SVM models. Since SVM models in any given class can be thought as forming a strict ordering of increasing capacity (trading monotonically generalization performance for fit to the training data as capacity increases)[10], we followed the following procedure for optimizing polynomial and RBF-kernel SVMs: we kept increasing the complexity of the model until leave-one-out performance started to drop. For each degree value tested for the polynomial kernel case, we tried several misclassification costs (1, 10, 100, 1000). We obtained the results shown in Table 1. The simplest polynomial-kernel model with best leave-one-out accuracy among the ones tested had cost 10, degree 2, and accuracy 83.8%. Similarly we developed Gaussian RBF-kernel SVMs with gamma 1, 0.1, 0.05., 0.01, and 0.001. The corresponding leave-one-out accuracies were: 62.2, 83.8, 81.1, 81.1, and 56.8%.

Finally we explored Feed-Forward Neural Networks as follows: We optimised parameters by using 24 randomly-chosen cases out of our total of 37 cases (subject to the constraint that the two classes distribute approximately the same in the 24 cases as in the 37 cases). We used this reduced set to develop models with all possible combinations of number of hidden layer nodes (from the set {5,10,20,30}), and of number of training epochs (from the set {100, 500, 1000, 1500, 2000}). Networks were trained using variable-rate gradient descent[16]. The best configuration was found to be 500 epochs and 5 hidden units. Using this configuration we developed models and obtained a leave-one-out accuracy of 83.3%.

## 4. Additional Machine Learning Experiments

We discuss here additional experiments for improving the classifier models using feature selection and boosting: We attribute the poor performance of DTI to a combination of a large feature set and small sample size. When individual variables contribute little to the classification of some target variable, DTI may not perform well since it builds classifiers that examine variables sequentially so that the available sample gets fragmented into exponentially (to the number of variables in any path of the tree) small segments. The latter segments become so small that they do not suffice for reliable estimation of within-segment frequencies (and related statistics used in evaluating a tree). In our task, effective sample

| **METHOD** | **GENES** | **PARAMETERS & OBSERVATIONS** | **LEAVE-ONE-OUT ACCURACY (%)** |
|---|---|---|---|
| Decision Tree Induction | 388 | default | **56.8** |
| KNN | 388 | k=1,2, 3 | **86.5**, 75.7, 83.7 |
| Linear SVM | 388 | cost=1, 10, 100, 1000 | **83.8**, 83.8, 83.8, 83.8 |
| Polynomial-kernel SVM | 388 | degree=2 , cost=1, 10, 100, 1000 | 78.4, **83.8**, 83.8, 83.8 |
| | | degree=3 , cost=1, 10, 100, 1000 | 78.4, 83.8, 83.8, 83.8 |
| | | degree=4,5 , cost=1, 10, 100, 1000 | 83.8 |
| | | degree=6, cost=1, 10, 100, 1000 | 81.1 |
| RBF-kernel SVM | 388 | gamma=1, 0.1, 0.05, 0.01, 0.001 | 62.2, **83.8**, 81.1, 81.1, 56.8 |
| NNs | 388 | 500 epochs, 5 hidden units (optimised separately –see text), variable learning rate | **83.8** |

**Table 1. Machine Learning Models (In bold, best accuracies per model class)**

was exhausted so quickly that the resulting trees had only a depth of up to two or three variables.

When the 8 best univariate predictors were given for training to the DTI algorithm, leave-one-out accuracy was improved, slightly, to 70.3%. We also experimented with boosting Decision Tree learning. Boosting[17,18] is a method for creating a set of classifiers (a "classifier ensemble") that is used for classification by voting among the ensemble members. The ensemble typically consists of weak classifiers but exhibits high classification performance as a whole. Duda et al. [17] provide an introduction on the theory of Boosting. In our dataset the boosted leave-one-out performance of DTI (with the 8 best univariate predictor genes) was improved to 78.4% (Table 2).

We also tried SVM-based feature selection. Guyon et al.[19] suggest a method for selecting features on the basis of the magnitudes of weight vectors developed after training a linear SVM. The method uses the fact that the linear support vector classifier assigns weights to each predictor such that the absolute value of a weight for a particular feature, determines the influence of that predictor on the decision surface. A second experiment with feature selection was conducted using this feature selection criterion to extract (in a nested cross-validation manner) the 80 most important genes. Then linear SVMs were trained and tested with leave-one-out on the full data. The resulting accuracy was found to be 89.2%.

Next, we trained Neural Networks and linear SVMs using Principal Component Analysis[21] for feature selection. The results were similar for the two classifiers as expected (Table 2 summarizes ranges for leave-one-out accuracy). The best accuracies were achieved when using the first 13, 14, 15, 16, 35, or 36 components, while the worst when using the first 9 ones.

## Study Limitations and Implications

The experiments presented here support the hypothesis that gene copy numbers as measured by array CGH are, *collectively*, an excellent indicator of the histological subtype. In this report we did not address at all the feasibility of discovering possible new cancer classes on the basis of molecular information. Such classes may carry more important information, clinically, than histopathology.

Gene copy number is a more stable property of cells than gene expression levels or protein concentrations. As such, array CGH has the potential to offer valuable complementary

| **METHOD** | **GENES** | **PARAMETERS & OBSERVATIONS** | **LEAVE-ONE-OUT ACCURACY (%)** |
|---|---|---|---|
| DTI | 8, 15, 50 | 8, 15, 50 best univariate predictors | 70.3 |
| | 8, 15, 50 | 8, 15, 50 best univariate predictors + boosting | 78.4 |
| Linear SVM | 80 | 80 best genes according to weights in linear SVM trained with all genes | **89.2** |
| NN/ Linear SVM | 388 genes→ 1 to36 first Principal Components (27.7-100 % of variance explained) | | 64.8(min)-83.8(max), |

**Table 2. Additional (Exploratory Optimization) Experiments**

information to, for instance, cDNA array assays or MALDI mass-spectrometry measurements. An interesting next research direction is, therefore, to combine all three types of data together with clinical and traditional histopathology information and investigate its association to important clinical outcomes (such as response to treatment and prognosis).

From a computational perspective, bio-informatics datasets, such as the one studied here, challenge the limits of the state-of-the-art applied machine learning methods in several ways: small sample size increases the danger of overfitting model parameters to the data. Splitting the data set further to perform parameter-optimizing cross-validation creates the danger of ending up with sample subsets that are too small to be useful. High rates of missing values compromise the learning ability of machine learning algorithms, while very large variable-to-sample ratios impede even methods that are usually robust to very high dimensionality, as was demonstrated in our experiments. Moreover, small sample sizes make application of causal model induction methods[20] very difficult. With the exception of DTI, which was particularly vulnerable to the characteristics of the data, there were no clear "winners" among the algorithms tested. Finally, this study was not designed to investigate rigorously the effect of various feature selection approaches. The preliminary impression is that feature selection may have a small effect, on performance. A formal study to evaluate the benefits of applying additional established (e..g, heuristic wrappers[22, 23], the Koller-Sahami algorithm[24]) and newly developed methods (such as Markov Blanket-based approaches[25]) is warranted in this domain.

## References

1. Pinkel D., Segraves R., Sudar D., et al., High Resolution Analysis of DNA Copy Number Variation Using Comparative Genomic Hybridization To Microarrays. Nature Genetics, 1998, 20:207-211.
2. Massion P.P., Kuo W-L., Stokoe D., et al. Genomic copy number analysis of non-small cell lung cancer using array comparative genomic hybridization: implications of the PI3-kinase pathway. To appear in *Cancer Research.*
3. Kanehisa, M. Post-genome informatics. Oxford University Press; 2000.
4. Lin A.M., and Johnson K.F., editors. Methods of microarray data analysis. Kluwer Academic Publishers; 2002.
5. Cooper, G.F., Abraham V., Aliferis C.F., et al. Predicting dire outcomes of patients with community acquired pneumonia. Submitted
6. Agresti, A., Categorical data analysis. John Wiley and Sons; 1990.
7. SPSS 10 for Windows, SPSS Inc.
8. See5 version 5.1, RuleQuest Research 2001.
9. Mitchell, T.M., Machine learning. McGraw-Hill Co., Inc.; 1997.
10. Vapnik V.N., Statistical learning theory. John Wiley and Sons; 1992.
11. Burges C.J.C. A tutorial on support vector machines for pattern recognition. Data Minining and Knowledge Discovery. 1998, 2(2):1-47.
12. Scholkopf, B., C.J.C. Burges, and A.J. Smola, eds. Advances in kernel methods: support vector learning. The MIT Press; 1999.
13. Chang C.C., Lin, C.J, LIBSVM:a library for support vector machines (version 2.3). Department of computer science and information engineering, National Taiwan University, Taipei 106, Taiwan.
14. MATLAB 6.1, The MathWorks Inc.
15. Hagan, M.T., Demuth H.B., and. Beale M.H, Neural network design. PWS Publishing; 1996.
16. Demuth, H. and Beale M., Neural network toolbox user's guide. Matlab user's guide. 2001: The MathWorks.
17. Duda, R.O., Hart P.E., and Stork D.G., Pattern classification. 2nd ed. John Wiley and Sons; 2001.
18 Schapire R.E. The strength of weak learnability. Machine Learning, 1990, 5(2):197-227.
19. Guyon, I., J. Weston, S. Barnhill, et al., Gene selection for cancer classification using support vector machines. Machine Learning, 2002, 46: 389-422.
20. Glymour, C. and G.F. Cooper, editors. Computation, causation, and discovery. AAAI/MIT Press; 1999.
21. Tabachnick, B.G. and L.S. Fidell, Using multivariate statistics. 2nd ed. Harper Collins; 1989.
22. Caruana, R. and D. Freitag. Greedy attribute selection. In International Conference on Machine Learning. 1994, 26-36.
23. Kohavi, R. and John G.H., Wrappers for feature subset selection. Artificial Intelligence, 1997. 97(1-2):273-324.
24. Koller, D. and Sahami M.. Toward optimal feature selection. In Thirteenth International Conference in Machine Learning. 1996, 284-292.
25. Aliferis C.F., Tsamardinos I.T., Markov Blanket induction for feature selection. Technical Report DSL-02-02, Department of Biomedical Informatics, Discovery Systems Laboratory, Vanderbilt University.

## Acknowledgments

The first author thanks Tricia Thornton for her assistance with the data preparation phase, and Ioannis Tsamardinos for helpful comments. Dr. Massion was supported by a Parker B. Francis Fellowship and by a Research Grant from the American Lung Association.