

Journal of Bioinformatics and Computational Biology
© Imperial College Press

**AUTOMATING RECOGNITION OF REGIONS OF
INTRINSICALLY POOR MULTIPLE ALIGNMENT FOR
PHYLOGENETIC ANALYSIS USING MACHINE LEARNING**

YUNFENG SHAN AND EVANGELOS E. MILIOS

*Faculty of Computer Science, Dalhousie University,
6050 University Avenue, Halifax, NS Canada B3H 1W5
shan@cs.dal.ca, eem@cs.dal.ca*

ANDREW J. ROGER AND CHRISTIAN BLOUIN

*Dept. of Biochemistry and Molecular Biology, Genome Atlantic/Genome Canada,
Dalhousie University, Halifax, Nova-Scotia, Canada, B3H 1X5 roger@hades.biochem.dal.ca,
bongo@hades.biochem.dal.ca*

EDWARD SUSKO

*Department of Mathematics and Statistics, Dalhousie University,
Halifax, Nova-Scotia, Canada, B3H 4H7 susko@mathstat.dal.ca*

Received (Day Month Year)

Revised (Day Month Year)

Accepted (Day Month Year)

Phylogenetic analysis requires alignment of gene sequences. Automatic alignment programs produced regions of intrinsically poor alignment that are currently detected and deleted manually. We present the results of a machine learning approach to detection of these regions of the alignment. We compare naive Bayes, standard decision trees, and support vector machines.

The results show three algorithms can accurately identify the bad sites of multiple sequence alignment based on three attributes such as the gap ratio, the site likelihood and the degree of homoplasy (consistency index). Among three algorithms, naive Bayes and SVM provide the best performance for bad site prediction, but C4.5 decision trees provide the best performance for the ambiguous and good site predictions. Among three classes, it is the most difficult to distinguish the ambiguous sites and the good sites, but easiest to distinguish the bad sites among these three classes. No evident difference among three parsimony count index as an attribute for reducing classification error is observed. Generally, the classifiers of naive Bayes and C4.5 decision tree learnt from the subset of a balanced class distribution generally come up with optimal performance compared with natural class distributions: the random subset and the entire data set.

Keywords: Phylogenetic analysis; machine learning; multiple sequence alignment; gene; protein.

1. Introduction

The first step in phylogenetic analysis of single genes is to edit alignments by eye to remove regions of intrinsically poor alignment (i.e. where there is little information to discriminate between alternative alignments). Unless these ambiguously aligned regions are removed, subsequent phylogenetic analyses can be influenced by the noise thus introduced. However, when scaling up phylogenetic analysis to the 'phylogenomic level' (i.e. analysis of hundreds to thousands of genes), manual alignment editing is simply impossible. Simple alignment processing approaches, such as to remove all regions that contain gaps, ignore many of the cues that the 'expert' phylogeneticist might use to include or discard a site from subsequent analysis. In this paper, we attempt to capture these cues with a variety of simple and sophisticated measures of alignment 'quality'. These measures include: (1) the proportion of sequences in the alignment with gaps at the site, (2) the degree of homoplasy (consistency index) in gap-to-no-gap transitions at gap containing sites, and (3) the likelihood ratio for the data at a site that compares the likelihood of the data under a phylogenetic model to an independence model.

Manual alignment editing approach is time-consuming, or even impossible for long sequences and large size of multiple sequence alignments. Automatic approach is evidently necessary and even partial automatic approach also benefit for phylogenetic analysis. An intelligent and efficient approach is required to solve this problem. Machine learning is an automatic and intelligent learning technique. It may help to complete this task. At times, these aspects are equivalent to pattern recognition problem. The reliability of machine learning in pattern recognition problems in many fields has been well demonstrated.^{1,2,3,4,5} In this study, we present the results of a study on the feasibility of using machine learning as a detector for regions of intrinsically poor alignment for phylogenetic analysis after multiple alignment using the above three measures of alignment 'quality'. Furthermore, we compare naive Bayes, decision tree, and support vector machine so that we can select a better machine learning method for this task.

The training for support vector machine (SVM) is slow when training size is great. Manual annotation for target class is a time-consuming work for collecting training data set. Other costs caused by today's huge size of data sets include preprocessing, storage, clearance, and transformation of the data set into a form suitable for a particular learning method or computer software. For example, a common question at the beginning of a data mining project is: how many data patterns and in what proportion for each target class? Moreover, most existing learning algorithms or computer softwares cannot handle huge data sets at all. Therefore, selecting a proper small subset from an entire training set is always an issue to run quickly the training in reality. In order to minimize the impact of limiting the training set size, the training data has to be chosen carefully. In this study, we compare the performance of three classifiers: naive Bayes, decision tree and SVM learnt from entire training data set, random subset and balanced subset

with much smaller size than the entire data set in order to determine what data set size and what ratio of each target class are proper.

2. Attribute Selection

The following five attributes were selected.

A. Gap ratio (g):

$$g = \frac{c}{t},$$

where c = number of gap characters, and t = number of taxa. This is the proportion of alignment columns that have gap characters.

B. Site likelihood (h):

$$h = \frac{\log(l) - \log(r)}{(1 - g) * t},$$

where l = log likelihood at the site, r = the log likelihood at random, g = gap ratio, and t = number of taxa. This is the log likelihood at the site considering a data, a model and a tree minus the log likelihood if the base states were picked at random from a set of base frequencies in the evolutionary model.

C. Consistency index (CI):

$$CI = \frac{1}{PC},$$

where PC = parsimony count of gap of insertion or deletion.

D. Retention index (RI):

$$RI = \frac{PC_{max} - PC}{PC_{max} - PC_{min}},$$

where PC_{max} = maximum parsimony count of gap of insertion or deletion, PC_{min} = minimum parsimony count of gap of insertion or deletion.

E. $RC = RI * CI$.

Phylogenetic trees for the site likelihood and the consistency index were calculated using distance matrices estimated by the TREE-PUZZLE 5.0 program under a Jones Taylor Thornton (JTT) plus gamma model followed by the neighbor-joining algorithm. Parsimony scores for gaps (CI , RI , RC) were calculated using the PAUP*4b10 program. We use only one of 3 possible parsimony scores (i.e., one of CI , RI , RC for an attribute).

3. Training Data Collections

17821 sites of multiple sequence alignment were annotated by three phylogenetics experts here for the three classes: bad site, ambiguous site and good site manually for collecting a training data set.

A random subset was randomly sampled by a block of 100 samples once from the entire training data set without adjustment of class balance. A balance subset

4 Yunfeng Shan et al.

was randomly sampled by a block of 100 samples once from the entire training data set with adjustment of class balance. Natural class distribution data sets include random subset and entire data set. The size and composition of three subsets of data for training are shown in *Table 1*.

Table 1. Size and composite of three types of training data.

Types of training data	Size	Percentage
Whole data set:		
Bad site	4707	26.0
Ambiguous site	2825	16.0
Good site	10239	58.0
Total	17821	100.0
Random subset:		
Bad site	197	27.0
Ambiguous site	210	29.0
Good site	313	44.0
Total	720	100.0
Balance subset:		
Bad site	240	33.3
Ambiguous site	240	33.3
Good site	240	33.3
Total	720	100.0

4. Machine Learning Methods

We choose a naive Bayes, C4.5 decision tree and support vector machine because they all have shown good results on pattern recognition problems in previous studies.^{2,3,4,5,6}

4.1. Naive Bayes classifier

The naive Bayes classifier is applicable to the task of concept learning. The idea is to use a probabilistic model for classification. It inputs a set of examples in attribute-values and uses Bayes theorem to estimate the posterior probabilities of all classifications. For each instance of the example, a classification with maximum posterior probability is assigned as the prediction.

For the naive Bayes method the following assumption must be satisfactory: Let $x = (x_1, \dots, x_n)$ be an instance of the example and c from C a possible classification. Then $P(x | c) = \prod_{i \in \{1, \dots, n\}} P(x_i | c)$.

This assumption is that the attributes are independent from each other. It is called class conditional independence. Using this assumption the classification c from C with maximum posterior probability $P(c | x) = \text{maximum}\{P(c) * \prod_{i \in \{1, \dots, n\}} P(x_i | c)\}$.

The learner estimates the probabilities by calculating the corresponding frequencies in the example set.

The assumptions such as class conditional independence is not always satisfactory. Another drawback is lack of available probability data. However, the naive Bayes has a much greater range of applicability than previously thought. For example, naive Bayes is optimal for learning conjunctions and disjunctions, even though they violate the assumption of independence.^{2,3} Naive Bayes was used to predict protein secondary structure and to sequence alignment algorithms for protein structure recognition.^{7,8}

4.2. *Decision tree classifier*

The C4.5 decision tree algorithm⁴ was used in this study with the default parameter settings.¹ It is one of the most popular algorithm and its performance is good for a variety of problem.

This algorithm uses an entropy-based measure known as information gain to select among the candidate attributes at each step when tree grows. Information gain measures how well a given attribute separate the training examples according to their target classification.

Given a set of examples, the learner uses a divide-and-conquer strategy to construct the tree.⁴ The sets of samples are accompanied by a set of attributes.

The decision tree was successfully used to classify membrane protein sequences based on functional classes,¹¹ to locate protein coding genes,¹² and to predict protein structure.¹³

Its advantage is that decision trees are easy to use, robust to noise and capable of learning disjunctive expressions. However, the decision trees are difficult to optimize.

4.3. *SVM classifier*

The SVM was developed by Vapnik based on the Structural Risk Minimization principle from statistical learning theory.¹⁴ The SVM is claimed to outperform most of other algorithms.⁵

The idea of an SVM is to separate classes with a separating surface that maximizes the margins between them. Non-linear input vectors are mapped through a high dimension feature space where the linear decision of the input vectors is computed in this feature space.

An SVM is a parameterized function whose functional form is defined before training. The SVM fits the function from a set of N examples. There are N free parameters in an SVM trained with N examples. To find these parameters, a quadratic programming (QP) problem must be solved. Sequential minimal optimization (SMO) is used to decompose the SVM QP problem in this experiment. The kernel is polynomial. We experiment here with these kernels for the degree $d = 1, 2, 3, 4, 5$. More detail descriptions can be found in Vapnik¹⁴ and Burges⁵.

The drawback of SVM is the time consuming for training and lack of expressive power. The SVM is a relatively new machine learning algorithm. However, the SVM has been popular in bioinformatics. It was used in classification of microarray data¹⁵, detection of the functions of genes for genomic annotation¹¹ and recognition of translation initiation sites¹⁶.

5. Experiment Design

We trained naive Bayes, C4.5 decision tree and SVM to recognize three classes: bad site, ambiguous site and good site. Performance was tested by using a two-way cross-validated experiment with default parameters.¹ This procedure was then repeated three more times, each time using a different random seed for 10-fold cross-validation. For SVM, the K one-against-other approach: one surface between class K and the K-1 other classes is constructed for each class (K SVM).

6. Performance Measures

To determine what class distribution and which algorithm will yield the best classifier, performance measures first must be chosen. In this study, three performance measures: precision, recall and correct rate are used. They are common metrics for information retrieval. Precision measures what proportion of items the classifier has been correct on for that class, while recall measure what proportion of relevant items were classified. Both precision and recall have an inverse relationship. Correct rate measures what proportion of items the classifier has been correct on for all the classes.

It is often useful to distinguish between the different types of errors that are made. To do this, we can construct a confusion matrix for a binary classification (*Table 2*). We use it for performance evaluation of SVM classifiers.

Table 2. A confusion matrix for a binary classification.

	Predicted class	
	Negative	Positive
Actual class:		
Negative	TN	FP
Positive	FN	TP

Note: Prediction of positive: TP = True Positive; FP = False Positive. Prediction of Negative: FN = False Negative; TN = True Negative.

Given this below table, we can now compute various other metrics:

$$Correct - rate = \frac{(TP + TN)}{(TN + TP + FN + FP)}.$$

AUTOMATING RECOGNITION OF REGIONS OF INTRINSICALLY POOR MULTIPLE ALIGNMENT FOR PHYLOGENETICS

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN}.$$

$$NegPrecision = \frac{TN}{TF + FN}, NegRecall = \frac{TN}{TN + FP}.$$

We extend those metrics to mult-class based on the above definition for binary-class (Table 3). We use it for performance evaluation of naive Bayes and decision tree classifiers.

Table 3. A confusion matrix for three classes.

		Predicted class		
		a	b	c
Actual class	a	Taa	Fba	Fca
	b	Fab	Tbb	Fcb
	c	Fac	Fbc	Tcc

Note: a = bad site, b = ambiguous site, c = good site. Prediction of a: Taa = True bad site; Fab = False bad site, actual ambiguous site; Fac = False bad site, actual good site. Prediction of b: Fba = False ambiguous site, actual bad site; Tbb = True ambiguous site, Fbc = False ambiguous site, actual good site. Prediction of c: Fca = False good site, actual bad site; Fbc = False ambiguous site, actual good site; Tcc = True good site.

Given this above table, we can now compute various other metrics:

$$Correct - rate = \frac{(Taa + Tbb + Tcc)}{(Taa + Tbb + Tcc + Fab + Fac + Fba + Fbc + Fca + Fcb)}.$$

$$Precision_a = \frac{Taa}{Taa + Fab + Fac}, Recall_a = \frac{Taa}{Taa + Fba + Fca}.$$

$$Precision_b = \frac{Tbb}{Tbb + Fba + Fbc}, Recall_b = \frac{Tbb}{Taa + Fab + Fcb}.$$

$$Precision_c = \frac{Tcc}{Tcc + Fca + Fcb}, Recall_c = \frac{Tcc}{Tcc + Fac + Fbc}.$$

7. Results and Discussion

7.1. Natural vs. balanced class distributions

Here the performance of the classifiers was analyzed using the natural and balanced class distributions. The purpose is to understand the effects of class imbalance on learning using entire training data set, random subset and balance subset.

When the size of training data is same (720), the classifiers of naive Bayes learnt from a balanced class distribution generally come up with higher correct

rate, precision and recall compared with a random subset (*Table 4–6*). When the size of training data is different, the classifiers of naive Bayes learnt from a balanced class distribution of smaller size (720) generally come up with higher precision for bad or ambiguous site prediction and all the recall compared with the natural class distributions of much greater size (17821), but lower precision for good site prediction and fewer correct rate (*Table 4–6*).

Table 4. Precision percentage of natural vs. balanced class distributions for three kinds of classifiers.

Type of Data set	Predicted Class		
	Bad	Ambiguous	Good
a. NB:			
Entire data set	90.9	35.6	78.8
Random subset	89.6	NaN	59.3
Balance subset	96.6	84.4	55.4
b. C4.5:			
Entire data set	92.3	59.9	80.7
Random subset	90.3	64.2	63.1
Balance subset	93.8	64.7	87.2
c. SVM:			
Random subset	91.0	NaN	59.0
Balance subset	97.0	NaN	NaN

Note: NaN = Not A Number, when a number is divided by zero. NB = naive Bayes, C4.5 = C4.5 decision tree, SVM = support vector machine.

When the size of training data is same (720), the classifiers of C4.5 decision tree learnt from a balanced class distribution generally come up with more correct rate, precision and recall compared with random subset (*Table 4–6*). When the size of training data is different (720 vs. 17821), the classifiers of C4.5 decision tree learnt from a balanced class distribution generally come up with more precision and recall for bad or ambiguous sites prediction compared with whole data set, but lower recall for good site prediction and fewer correct rate (*Table 4–6*).

Generally, the classifiers of C4.5 decision tree and naive Bayes learnt from subset of a balanced class distribution generally come up with optimal performance compared with natural class distributions of training subset (*Table 4–6*). These results are consistent with previous report.¹⁰

For bad site prediction, the classifiers of SVM learnt from a balanced class distribution generally come up with more correct rate, precision and recall compared with random subset, but not for ambiguous and good site prediction (*Table 4–6*).

We repeated the experiment three more times with different random splits of the data. The results showed that the variance introduced by the random splitting of the data is small and varies by less than $\pm 1\%$.

Table 5. Recall percentage of natural vs. balanced class distributions for three kinds of classifiers.

Type of Data set	Predicted Class		
	Bad	Ambiguous	Good
a. NB:			
Entire data set	88.4	7.0	97.1
Random subset	91.4	0	98.3
Balance subset	95.0	20.4	98.3
b. C4.5:			
Entire data set	91.2	15.3	98.0
Random subset	89.3	16.2	95.2
Balance subset	94.2	87.1	56.7
c. SVM:			
Random subset	85.0	0	98.0
Balance subset	95.0	0	0

Note: NaN = Not A Number, when a number is divided by zero. NB = naive Bayes, C4.5 = C4.5 decision tree, SVM = support vector machine.

Table 6. Correct rate percentage of natural vs. balanced class distributions for three kinds of classifiers.

Type of Data set	a. Naive Bayes	b. C4.5 decision tree	c. SVM
Entire data set	80.4	91.2	NA
Random subset	67.8	89.3	77.0
Balance subset	71.1	94.2	78.2

Note: NA = Not Available.

7.2. Performances of three kinds of classifiers: naive Bayes vs. C4.5 decision tree vs. SVM

The precisions of C4.5 decision tree classifiers learnt from all three types of training data are higher than those of naive Bayes classifiers except for the bad and ambiguous site prediction by the classifiers learnt from the balance training data (Table 4).

The recalls of C4.5 decision tree classifiers learnt from all three types of training data vary compared with those of naive Bayes classifiers. The recalls of C4.5 decision tree classifiers learnt from entire data set are higher than those of naive Bayes classifiers. The recalls of C4.5 decision tree classifiers learnt from random and balance training data are lower than those of naive Bayes classifiers for bad and good site predictions, but higher for ambiguous site predictions (Table 5).

The correct rates of C4.5 decision tree classifiers learnt from all three types of training data are higher than those of naive Bayes classifiers (Table 6).

For bad site prediction, the classifiers of SVM learnt from a balance subset and

random subset generally come up with higher precision compared with those of C4.5 decision tree classifiers (*Table 4–6*). The SVM classifiers have similar very high precision (97.0%) and recall (95.0%) to naive Bayes classifiers. Because the training for SVM took much time, longer than 3 weeks for a run, we did not train the SVM classifiers using entire data set.

However, for the ambiguous and good site predictions, the classifiers of SVM learnt from a balanced and a random class distribution training data subset do not produce higher correct rate, precision and recall compared with those of C4.5 decision tree and naive Bayes classifiers (*Table 4–6*). The recall of SVM classifiers were not higher than those of C4.5 decision tree and naive Bayes classifiers for bad site prediction, either (*Table 4–6*).

The SVM classifiers are often claimed to outperform the conventional learners.⁵ When we construct a binary SVM classifier by bad site class against others, our results for bad site prediction is consistent with that report. However, the performance of SVM fails for ambiguous site and good site predictions. For some data sets, the SVM may not be cable of finding a separating hyperplane in feature space, either because the kernel function is not applicable for the training data or because the data sets contain mislabelled target class examples.⁹ For the former problems, different kernel function or other learning should be examined. The latter problems can be solved by using a soft margin that allows some examples to fall on the wrong side of the separating hyperplane.⁹ Completely to specify a SVM, two parameters: the kernel function and the magnitude of the penalty for violating the soft margin must be specified. The setting of both parameters depends on the particular data sets. Experiment is only approach to obtain the best combination for both parameters. We experimented with default of the magnitude of the penalty and with changes in exponent of the kernel E from 1 to 5.¹ More studies are still necessary for a better application of SVM to distinguish the ambiguous site and good site such as adding more attributes and iterating feedbacks for target class annotation.

7.3. *Effects of different parsimony count index on performance of classifiers*

We experimented with three kinds of parsimony count index: CI, RI, and RC using three kinds of classifiers: naive Bayes, C4.5 Decision tree and SVM classifiers here. No evident difference was observed. The results showed that the variance in precisions, recalls and correct rates of these three algorithms introduced by the different parsimony count index was small and varies by less than $\pm 0.5\%$.

8. Conclusions

We have demonstrated that these three algorithms of machine learning can accurately recognize a bad site of multiple sequence alignment based on three attributes such as the gap ratio, the degree of homoplasy (consistency index) and the site likelihood. Among the techniques examined, naive Bayes and SVM classifiers are the

AUTOMATING RECOGNITION OF REGIONS OF INTRINSICALLY POOR MULTIPLE ALIGNMENT FOR PHYLOGENETICS

best for the bad site predictions, but C4.5 decision trees provide the best performance for ambiguous and good site predictions - better than naive Bayes and SVM classifiers.

Among these three classes, it is the most difficult to distinguish the ambiguous sites from the good sites, but easiest to distinguish the bad sites among these three classes by means of three algorithms. There is no evident difference among these three possible parsimony count index as an attribute for reducing classification error.

Generally, the classifiers of naive Bayes and C4.5 decision tree learnt from the subset of a balanced class distribution generally come up with optimal performance compared with the natural class distributions: the random subset and the entire data set.

Acknowledgements

We appreciate some of code written by Davin J. Butt. This work was supported in part by Grant 227085-00 from the Natural Sciences and Engineering Research Council, Canada awarded to AJR and is also part of a Genome Canada/Genome Atlantic large-scale project.

References

1. I. H. Witten and E. Frank, *Data mining: Practical machine learning tools and techniques with Java implementations*. (Morgan Kaufmann, San Francisco, CA, 1987).
2. P. Domingos and M. Pazzani, In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, ed. L. Saitta. (Morgan Kaufmann, 1996), p. 105.
3. P. Domingos and M. Pazzani, In *Machine Learning*, **9**, (1997), p. 103.
4. J. R. Quilan, *C4.5: Programs for Machine Learning* (San Mateo, CA: Morgan Kaufmann, 1993).
5. C. J. C. Burges, *Data Mining and Knowledge Discovery*, **2**, 2, (1998).
6. T. Joachims, In *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, (1998), p. 137.
7. P.-L. Wang and D. Zhang, In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI02)*, (IEEE, 2002), p. 252.
8. R. A. Goldstein, Z. A. Luthey-Schulten, P. G. Wolynes, *System Sciences*, (1994), p. 306.
9. M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares Jr., and D. Haussler, In *Proc. Natl. Acad. Sci.*, **97**, 1, (2000), p. 262.
10. G. Weiss and F. Provost, Technical Report ML-TR 43, Department of Computer Science, Rutgers University, (2001).
11. S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara, and S. Arikawa, Technical Report RIFIS-TR-CS-60, (Research Institute of Fundamental Information Science, Kyushu University, 1992).
12. S. Salzberg, *J. Comp. Biol.*, **2**, (1995), p. 473.
13. J. Selbig, T. Mevissen, T. and T. Lengauer, *Bioinformatics*, **15**, (1999), p. 1039.
14. V. Vapnik, *Statistical Learning Theory*, (John Wiley Sons, Inc., 1998).
15. T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer and D. Haussler, *Bioinformatics*, **16**, (2000), p. 906.

12 *Yunfeng Shan et al.*

16. A. Zien, G. Rtsch, S. Mika, B. Schlkopf, T. Lengauer, and K.-R Mller, *Bioinformatics*, **16**, (2000), p. 799.

Photo and Bibliography

To be written.