# Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 01-033

## Evaluation of Techniques for Classifying Biological Sequences*

Mukund Deshpande and George Karypis

October 18, 2001

# Evaluation of Techniques for Classifying Biological Sequences*

Mukund Deshpande and George Karypis

University of Minnesota, Department of Computer Science/Army HPC Research Center
Minneapolis, MN 55455

{deshpand,karypis}@cs.umn.edu

**Abstract**

In recent years we have witnessed an exponential increase in the amount of biological information, either DNA or protein sequences, that has become available in public databases. This has been followed by an increased interest in developing computational techniques to automatically classify these large volumes of sequence data into various categories corresponding to either their role in the chromosomes, their structure, and/or their function. In this paper we evaluate some of the widely-used sequence classification algorithms and develop a framework for modeling sequences in a fashion so that traditional machine learning algorithms, such as support vector machines, can be applied easily. Our detailed experimental evaluation shows that the SVM-based approaches are able to achieve higher classification accuracy compared to the more traditional sequence classification algorithms such as Markov model based techniques and $K$-nearest neighbor based approaches.

**Keywords:** Classification, Sequences, Markov chains, bio-technology, SVM

## 1 Introduction

The emergence of automated high-throughput sequencing technologies has resulted in an exponential increase in the amount of DNA and protein sequences that is available in public databases. At the time of writing, GenBank [BBL+99] had over 16 billion DNA base-pairs, SWISS-PROT [BA99] contained around 100,000 protein sequences, and PIR [BGH+99] contained over 230,000 protein sequences. As the amount of sequence information continues to increase genome researchers are shifting their biological investigation to understand the function of the different genes and proteins, determine their regulatory mechanisms, and discover how the different proteins interact to form the genetic network [(ed00].

The sheer volume of existing and anticipated data makes data mining techniques the only viable approach to analyze this data within the genome of a single specie and across genomes of different species. Classification algorithms applied on sequence data can be used to gain valuable insights on the function of genes and proteins and their relations, since throughout evolution nature has reused similar building blocks, and it is well-known that strong sequence similarity often translates to functional and structural relations [Gus97]. In particular, sequence classification algorithms

*This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Related papers are available via WWW at URL: **http://www.cs.umn.edu/~karypis**

can be used to determine whether a DNA sequence is part of a gene-coding or a non-coding region, identify the introns or exons of a eukaryotic gene sequence, predict the secondary structure of a protein, or assign a protein sequence to a specific protein family.

Over the years, algorithms based on $K$-nearest neighbor, Markov models, and Hidden Markov models have been used extensively for solving various sequence-based classification problems in computational biology [Gus97, DHK+98, OHN+99, DEKM98, Mou01]. The popularity of these approaches is primarily due to the fact that they can directly (or easily modified to) take into account the sequential nature present in these data sets, which is critical to the overall classification accuracy. However, there have been few attempts in trying to use some of the more traditional machine learning classification algorithms (*e.g.*, decision trees, rule-based systems, support vector machines) [KH98, LZO99, ZLM00, WZH00, KDK+01], primarily because these algorithms were thought of not being able to model the sequential nature of the data sets.

The focus of this paper is to evaluate some of the widely used sequence classification algorithms in computational biology and develop a framework to model the sequences in such a fashion so that traditional machine learning algorithms can be easily used. To this end, we focus on three classes of classifiers. The first class is based on the $K$-nearest neighbor algorithm that computes the similarity between sequences using optimal pairwise sequence alignments. The second class is based on Markov models. In particular, we evaluate the performance of simple Markov chains of various orders, as well as the performance of sequence classifiers that were derived from Interpolated Markov Models (IMM) [SDKW98] and Selective Markov Models (SMM) [DK01]. The third class is based on representing each sequence as a vector in a derived feature space and then using support vector machines (SVM) [Vap95, Vap98] to build a sequence classifier. The key property of the derived feature space is that it is able to capture exactly the same sequential relations as those exploited by the different Markov model based schemes. In fact, in this derived space, the various Markov model based schemes are nothing more than linear classifiers, whose decision hyperplane was derived from the maximum likelihood estimates.

We experimentally evaluate the various classification algorithms on five different datasets that cover a wide-range of biologically relevant classification problems. Our results show that the SVM-based approaches are able to achieve higher classification accuracy than that achieved by the different Markov model based techniques as well as the much more expensive $K$-nearest based approaches. For most of the datasets, the improvements in classification accuracy is significant ranging from 5% to 10%.

# 2 Sequence Classification Techniques

In this section we discuss various techniques used for classifying sequence datasets. We start off by defining some terminology that will be used throughout this paper followed by a detailed description of three classes of algorithms.

**Definitions** A sequence $S_r$ is an ordered collection of symbols and is represented as $S_r = \{x_1, x_2, x_3, \ldots x_l\}$. The alphabet $\sigma$ for symbols ($x$) is known in advance and of fixed size $N$. Each sequence $S_r$ has a class label $C_r$ associated with it. The set of class labels ($C$) is also fixed and known in advance. To keep the discussion of this paper simple we will assume $C$ contains only two class labels ($C_+$, $C_-$); however, the classification techniques discussed here are not restricted to two class problems but can handle multi-class classification problems as well.

## 2.1 $K$ Nearest Neighbor Classifier

$K$-nearest neighbor (KNN) classification techniques are very popular in the biological domain because of their simplicity and their ability to capture sequential constraints present in the sequences. To classify a test sequence KNN first locates $K$ training sequences which are most similar to the test sequence. It then assigns the class label that occurs the most in those $K$ sequences (majority function) to the test sequence. The key part of the KNN classifier is the method used for computing the similarity between two sequences.

We discuss two similarity function specifically targeted towards capturing the sequential constraints of the dataset. The idea behind these techniques is to first compute an optimal alignment between two sequences and then score this alignment to determine the similarity between the sequences. The alignment score is nothing but a function of the

number of matched and unmatched symbols in the alignment. The two similarity functions used in the KNN classifier differ in the way the two sequences are aligned. The first similarity score, referred as the ***global alignment score***, computes the alignment score by aligning sequences across their entire length (hence the name global). Since global alignment aligns complete sequences, it is able to capture position specific patterns, *i.e*, the symbols which occur at the same relative position in the two sequences. However, this score is biased by the length of the sequence and hence needs to be normalized. The second similarity measure uses ***local alignment score***, in which only portions of the two sequences are aligned. Local alignment tries to locate the best aligning subsequences of the two sequences (hence the name local). Local alignment score can effectively capture small subsequences of symbols which are present in two sequences but may not necessarily be at the same position.

The optimal alignment between two sequence (both global and local) is computed using a dynamic programming algorithm. The dynamic programming algorithm works by progressively aligning subsequences of the two sequences [Gus93]. The complexity of the dynamic programming algorithm is $O(n \times m)$, where $n$ and $m$ are the length of the sequences. Therefore, the task of locating $K$ most similar sequences in the database of training sequences is computationally expensive.

The overall alignment of the sequences is driven by a symbol *scoring matrix* that determines how similar the different symbols are when aligned together. The simplest scoring matrix has a score of one when the aligned symbols are identical, and zero otherwise. However, in the case of protein sequences, various scoring matrices have been developed that also take into account similarities between the different amino-acids. The idea behind these scoring matrices is that there is a varying degree of similarity between different amino-acids, and the alignment should take them into account. Examples of such scoring matrices include PAM [DSO78] and BLOSUM [HH92]. Furthermore, more advanced techniques also include scoring matrices that are position dependent, those for example used in PSI-BLAST [AK98].

## 2.2 Markov Chain based Classifiers

Markov chain based techniques are extremely popular for modeling sequences because of their inherent ability to capture sequential constraints present in the data. In this section we discuss different classification techniques based on Markov chains and their variants.

### 2.2.1 Simple Markov chain classifier

To build a Markov chain based classification model, we first partition the training sequences according to the class label associated with each sequence and then a simple Markov chain ($M$) is built for each of these smaller datasets. A test sequence $S_r$ is classified by first computing the likelihood/probability of that sequence being generated by each of those Markov chains *i.e*, computing the conditional probability $P(S_r|M)$. The Markov chain which gives the highest likelihood is used and its associated class label is assigned to the test sequence. For a two class problem this is usually done by computing the log-likelihood ratio given by

$$L(S_r) \quad = \quad \log \frac{P(S_r|M_+)}{P(S_r|M_-)}, \tag{1}$$

where $M_+$ and $M_-$ are the Markov chains corresponding to the positive and negative class, respectively. Using Equation 1 the value of $C_r$ is $C^+$ for $L(S_r) \geq 0$ and $C^-$ if $L(S_r) < 0$.

The conditional probability, $P(S_r|M)$, that the sequence is generated by the Markov chain $M$ is given by

$$\begin{aligned} P(S_r|M) &= P(x_l, x_{l-1}, x_{l-2}, \ldots, x_1|M) \\ &= P(x_l|x_{l-1}, x_{l-2}, \ldots, x_1, M) P(x_{l-1}|x_{l-2}, \ldots, x_1, M) \cdots P(x_1|M). \end{aligned}$$

The different conditional probabilities in the above equation can be calculated using the *Markov principle* that states that the symbol in a sequence depends only on its preceding symbol *i.e.*, $P(x_l|x_{l-1}, x_{l-2}, \ldots, x_1) = P(x_l|x_{l-1})$. In

AATCAGGAC -
CCAATGAGG +
TTGCCTTTA -
ACTAGCCCG +
GGTTCCGGC +
GTAACGAAC -

(a) Training Sequences

S(t) = AGGCCC

$L(S(t)) = \log(P(S(t)|M^+) - \log(P(S(t)|M^-)$

$= -7.0 - (-11.64)$

$= 4.64$

(c)Log Likelihood for Test Sequence

TPM for Class +

| Sym | Sup | A | C | G | T |
|---|---|---|---|---|---|
| A | 2 | **1**, -2.0 | **1**, -2.0 | **1**, -2.0 | **1**, -2.0 |
| C | 1 | **1**, -3.0 | **4**, -1.0 | **2**, -2.0 | **1**, -3.0 |
| G | 5 | **1**, -3.0 | **2**, -2.0 | **4**, -1.0 | **1**, -3.0 |
| T | 6 | **1**, -2.0 | **1**, -2.0 | **1**, -2.0 | **1**, -2.0 |

TPM for Class -

| Sym. | Sup | A | C | G | T |
|---|---|---|---|---|---|
| A | 7 | **3**, -1.41 | **3**, -1.41 | **1**, -3.00 | **1**, -3.00 |
| C | 6 | **1**, -2.0 | **1**, -2.0 | **1**, -2.0 | **1**, -2.0 |
| G | 3 | **2**, -1.32 | **1**, -2.32 | **1**, -2.32 | **1**, -2.32 |
| T | 5 | **2**, -1.80 | **1**, -2.80 | **1**, -2.80 | **3**, -1.22 |

(b) Transition Probabilty Matricies

**Figure 1**: Sequence classification methodology using Markov chains Figure (a) displays the training sequences with their associated class labels (b) displays the Transition Probability Matrices for two classes (c) shows the log likelihood computation using Equation 3

this case, $P(S_r|M)$ is given by

$$P(S_r|M) = P(x_l|x_{l-1}, M)P(x_{l-1}|x_{l-2}, M)\cdots P(x_1, M) = P(x_1|M)\prod_{i=2}^{l} P(x_i|x_{i-1}, M). \quad (2)$$

The Markov chain $M$ estimates the probability of $P(x_i|x_{i-1})$ by taking the ratio of the frequency of the sequence $(x_{i-1}, x_i)$ in the training dataset to the frequency of $x_{i-1}$. This conditional probability is usually referred as the *transition probability* $(\alpha_{x_{i-1},x_i})$; that is, the probability of making the transition from symbol $x_{i-1}$ to symbol $x_i$. Furthermore in Markov chain terminology every symbol in the alphabet is associated with a state and the transition probability $(\alpha_{x_{i-1},x_i})$ gives the probability of making a transition from state with symbol $x_{i-1}$ to state with symbol $x_i$. These transition probabilities for all pairs of symbols are usually stored in a matrix referred to as the Transition Probability Matrix (TPM).

Substituting Equation 2 in Equation 1 and using the new notation for transition probabilities we get,

$$L(S_r) = \log\frac{\prod_{i=1}^{l}\alpha^+_{x_{i-1},x_i}}{\prod_{i=1}^{l}\alpha^-_{x_{i-1},x_i}} = \sum_{i=1}^{l}\log\frac{\alpha^+_{x_{i-1},x_i}}{\alpha^-_{x_{i-1},x_i}}, \quad (3)$$

where $a^+_{x_{i-1},x_i}$ and $a^-_{x_{i-1},x_i}$ are the transition probabilities of the positive and negative class, respectively. Note that to avoid the inhomogeneity in the Equation 2 due of the probability $P(x_1|M)$, it is common to add an extra symbol, *begin*, to the alphabet, which is assumed to be present at the beginning of each sequences [DEKG98]. Equation 3 contains these extra states.

Figure 1 illustrates an example of sequence classification on a dataset of DNA sequences. Figure 1(a) shows the training sequences with their respective class labels, these sequences are first split into two parts for computing the TPM associated with each class label. Figure 1(b) displays these TPM, each cell in the TPM shows the frequency of observing the pair of symbols corresponding to the row and the column of that cell, each cell also contains the log of the corresponding transition probability. Figure 1(c) shows the log-likelihood ratio for a test sequence using Equation 3.

### 2.2.2 Higher Order Markov Chains

The Markov chain $(M)$ we have discussed so far is referred to as the first order Markov chain, since we only look one symbol in the past to compute the transition probability. A generalization of the first order Markov chain is the $k^{th}$ order Markov chain in which the transition probability for a symbol $x_l$ is computed by looking at the $k$ preceding symbols

*i.e.*, $P(x_l|x_{l-1}, x_{l-2}, \ldots, x_1) = P(x_l|x_{l-1}, x_{l-2}, \ldots, x_{l-k})$. The $k^{th}$ order Markov chain will have $N^k$ states each associated with a sequence of $k$ symbols. For example, for a second order Markov chain the transition probability of a symbol $x_i$ , $(\alpha_{x_{i-2}x_{i-1},x_i})$, will depend on two preceding symbols $x_{i-2}$ and $x_{i-1}$. Moreover, the second order Markov chain will have $N^2$ states and a TPM of size $N^2 \times N$.

In general, these higher order models have better classification accuracy as compared to lower order models because they are able to capture longer ordering constraints present in the dataset. However, the number of states in higher order Markov chains grows exponentially with the order of the chain. Consequently, it is harder to accurately estimate all the transition probabilities from the training set. That is, there are many high-order states that are not frequently observed in the training set, leading to inaccurate probability estimates. In principle, this problem can be solved if the size of the training set is increased at the same rate as the state-size of the higher order Markov chain. However, this cannot be easily done for many applications areas. To overcome these problems many variants of the Markov chains have been developed. In the next section we discuss two such variants: the Interpolated Markov Models [SDKW98] and the Selective Markov Models [DK01]. The basic idea of these approaches is to combine portions of different order Markov chains so that to improve the overall classification accuracy.

### 2.2.3 Interpolated Markov models

Interpolated Markov models (IMM) [SDKW98] address the problem of accuracy in the higher order models due to the presence of a large number of states with low accuracies. The solution that was proposed by IMMs is to build a series of Markov chains starting for the $0^{th}$ order Markov chain up to the $k^{th}$ order, where $k$ is user defined and fixed before hand. IMM computes the transition probability for a symbol as a linear combination of the transition probabilities of the different order Markov chains starting from $0^{th}$ order to $k^{th}$ order Markov chain. That is, $P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, IMM_k)$ is given by

$$P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, IMM_k) = \sum_{j=0}^{k} \gamma_{j,i} P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, M_j),$$

where $P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, M_j)$ is the transition probability for the $j$th-order Markov chain and $\gamma_{j,i}$ is a constant indicating how the particular transition probability of the $j$th-order model will be weighted in the overall calculation. The motivation behind this approach is that even though higher order Markov states capture longer context they have lower support. Hence, using probabilities from lower order states which have higher support helps in correcting some of the errors in estimating the transition probabilities for higher order models. Furthermore, by using the various $\gamma$ parameters, IMMs can control how the various states are used while computing the transition probabilities. A number of methods for computing the various $\gamma$ factors where presented in [SDKW98] that are based on the distribution of the different states in the various order models. However, the right method appears to be dataset dependent.

The IMM techniques were originally developed for the problem of predicting the next symbol in the sequence [SDKW98] and not for sequence classification. However, since an IMM is nothing more than an alternate method for estimating the various conditional probabilities, it can be easily used for sequence classification, as well. In particular, we can build an IMM model for the positive and negative class and then classify a sequence by taking the log-likelihood ratio of their conditional probabilities. That is,

$$L(S_r) = \sum_{i=1}^{l} \log \frac{P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, IMM_k^+)}{P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, IMM_k^-)},$$

where $IMM_k^+$ and $IMM_k^-$ are the interpolated Markov models for the positive and negative class, respectively.

### 2.2.4 Selective Markov Models

Selective Markov models (SMM) [DK01] is another set of techniques which address the problems associated with higher order Markov chains and like IMMs they were developed to predict the next symbol in the sequence. These techniques first build varying order Markov chains and then prune the non-discriminatory states from the higher

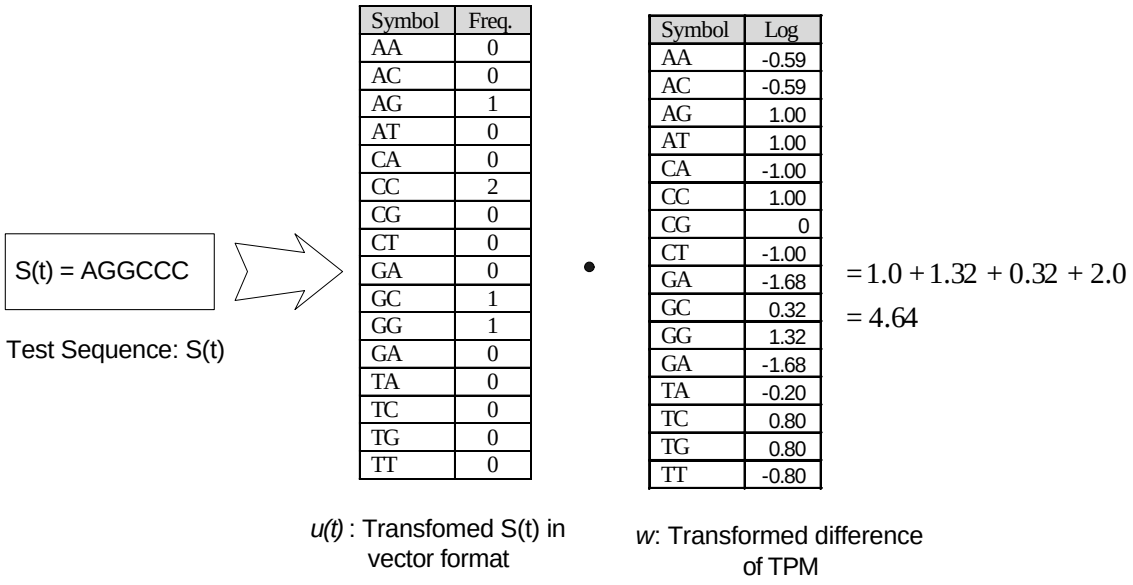


**Figure 2**: Sequence classification as a dot product training sequence vector $S(t)$ and the weight vector $w$ obtained from the Transition probability matrices.

order Markov chains. The key task in building a SMM is to decide which states are non-discriminatory so they can be pruned. A number of techniques have been proposed for deciding which states to prune [DK01]. The simplest method uses a *frequency threshold* and prunes all the states which occur less than that frequency threshold. The most advanced method uses a validation set to estimate the error rates of the states of the different order Markov chains and uses these estimates to prune the higher-error states for each type of prediction. Once the non-discriminatory states have been pruned, the condition probability $P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1, SMM_k)$ is equal to the conditional probability that corresponds to the highest order Markov chain present amongst the remaining states. Note that unlike IMMs that combine different order Markov chains, SMMs select the appropriate order Markov chain for each prediction. SMM-based techniques can be easily extended to solve the sequence classification problem, as was done for the IMM in Section 2.2.3. In our experiments we used the simple frequency-threshold based method to prune the infrequent occurring states of the various Markov chains.

One way of specifying the minimum frequency-threshold for a particular state-transition pair, is to do so as either a fraction of the overall number of sequences or as a fixed constant. However, depending on the length of the sequences, the order of the model, and the alphabet size, such measures will tend to perform a pruning with dramatically different characteristics. For this reason, in our generalization of the SMMs for sequence classification, we specify the minimum frequency threshold as a parameter $\delta$ that indicates that a particular state-transition pair is kept only if it occurs $\delta$ times more frequently than its expected frequency, when uniform distribution is assumed. For example if a dataset contains $n$ sequences with average length $l$, and the size of the alphabet is $N$, then the average frequency of a subsequence of length $k$ is given by $\text{Freq}_k = (nl)/N^k$. Therefore, a subsequence is pruned if it occurs fewer than $\delta \times \text{Freq}_k$ times.

## 2.3 Feature based Sequence Classification

The various algorithms that we described so far were specifically designed to take advantage of the sequential nature of the data sets. However, traditional machine learning classification algorithms such as decision trees, rule-based classifiers, and support vector machines can also be used to classify sequence data sets, provided that the sequences are first modeled into a form that is suitable to these algorithms, that take into account their sequential nature. In the rest of this section we present one such class of transformations that was motivated by analyzing the classification model used by the various Markov chain-based techniques described in the previous section.

Recall from Equation 3, that the simple first order Markov chain will classify a particular sequence $S_i$ by looking

at the sign of

$$L(S_i) = \sum_{i=1}^{l} \log \frac{a^+_{x_{i-1},x_i}}{a^-_{x_{i-1},x_i}}.$$

If the number of distinct symbols in the different sequences is $N$ (*e.g.*, four for nucleotides and 20 for amino-acids), then the above equation can be re-written as the dot-product of two vectors, *i.e.*,

$$L(S_i) = u^t w, \tag{4}$$

where $u$ and $w$ are of length $N^2$. Each one of the dimensions of these vectors corresponds to a unique pair of symbols. If $\alpha\beta$ is the symbol-pair corresponding to the $j$th dimension of these vectors, then $u(j)$ is equal to the number of times $\alpha\beta$ appears in $S_i$, and $w(j)$ is equal to $\log(a^+_{\alpha,\beta}/a^-_{\alpha,\beta})$. This re-write of Equation 3 is illustrated in Figure 2 for the example of Figure 1.

A similar transformation can be performed for all different variants of the Markov chains that were described in Section 2.2. In the case of higher order Markov chains, the dimensionality of the new space will become equal to $N^{k+1}$, where $k$ is the order of the model. In the case of interpolated Markov models, the new space will have a dimensionality that is equal to $N + N^2 + \cdots N^{k+1}$, whereas in the case of selective Markov models, the dimensionality of the space will be equal to the number of states that were retained from the different order Markov chains after pruning. For higher order Markov chains and interpolated Markov models, each sequence will be represented as a frequency vector, whereas in the case of selective Markov models, the frequency along the various dimensions will be computed following the rules described in Section 2.2.4 for selecting the various states and transitions of the different order Markov chains.

There are two important aspects of these transformations. First, it allows us to view each sequence as a vector in a new space whose dimensions are made of all possible pairs, triplets, quintuplets, *etc*, of symbols, and second it shows that each one of the Markov chain based techniques is nothing more than a linear classifier [Mit97], in which the decision hyperplane is given by the vector $w$ that corresponds to the log-odds ratios of the maximum likelihood estimates for the various transition probabilities.

The vector-space view of each sequence allow us to use any of the traditional classification techniques that operate on objects represented by multidimensional vectors. In our study we chose to use support vector machines that have been recently shown to be well-suited for high dimensional data sets. Support vector machines (SVM) is a relatively new learning algorithm proposed by Vapnik [Vap95, Vap98]. This algorithm is introduced to solve two-class pattern recognition problems using the Structural Risk Minimization principle [Vap95, CV95]. Given a training set in a vector space, this method finds the *best* decision hyperplane that separates two classes. The quality of a decision hyperplane is determined by the distance (referred as margin) between two hyperplanes that are parallel to the decision hyperplane and touch the closest data points of each class. The *best* decision hyperplane is the one with the maximum margin. By defining the hyperplane in this fashion, SVM is able to generalize to unseen instances quite effectively. The maximum margin hyperplane can be found using quadratic programming techniques. SVM extends its applicability on the linearly non-separable data sets by either using soft margin hyperplanes, or by mapping the original data vectors into a higher dimensional space in which the data points are linearly separable. The mapping to higher dimensional spaces is done using appropriate kernel functions, resulting in efficient algorithms.

The use of a linear SVM classifier in the feature space corresponding to that used by the interpolated Markov model techniques can also be viewed as a method to automatically learn the various $\gamma$ parameters required to combine the conditional probabilities of the different order Markov chains. In fact, the linear SVM model can be used to learn the exact $\gamma$ parameters of IMM by representing each sequence as a weighted frequency vector, in which the $i$th dimension of the vector corresponds to the frequency of a particular sequence multiplied by the log-ratio of the conditional probabilities for the positive and negative class. In this modeling, the linear model learned by SVM is nothing more than the vector of the $\gamma$ parameters.

# 3 Experimental Evaluation

We experimentally evaluate the different sequence classification schemes presented in Sections 2 and 2.3 on different sequence datasets. In this section we describe the various datasets, our experimental methodology, and the performance of the different classification algorithms.

## 3.1 Dataset Description

The performance of the various sequence classification algorithms was evaluated on five different datasets. Each of the datasets tackles a well identified sequence classification problem in computational biology. The sequences can be divided in two broad types: nucleotide sequences which have an alphabet of size four, and amino-acid sequences which have an alphabet of size twenty. Table 1 shows some general statistics on the various datasets. As can be seen, the datasets vary a lot in both the number of sequences as well as the length of the sequences present in the dataset.

| *Nucleotide Sequences* | | | *Amino Acid Sequences* | | |
|---|---|---|---|---|---|
| *Dataset* | *# Sessions* | *Avg. Length* | *Dataset* | *# Sessions* | *Avg. Length* |
| **Splice** (S-EI) | 1, 527 | 60.0 | **Peptidias** (P-) | 1,584 | 511.3 |
| Exon | 762 | 60.0 | cysteine | 416 | 854.3 |
| Intron | 765 | 60.0 | metallo | 580 | 512.6 |
| **Mouse Genome** (MG-GN) | 10,918 | 361.6 | serine | 775 | 500.5 |
| exon | 6,593 | 215.6 | **Protein Structure** (PS-) | 16,154 | 5.2 |
| intron | 4,325 | 584.4 | Helix | 4,636 | 9.0 |
| **Ecoli Genome** (EC-CN) | 3,370 | 74.9 | Turn | 6,079 | 2.3 |
| coding region | 1,700 | 75 | Strand | 5,439 | 5.2 |
| non-coding region | 1,670 | 74.9 | | | |

**Table 1**: Preliminary dataset statistics.

The ***splice dataset*** is distributed as a part of UCI KDD Archive [BM98]. The dataset contains nucleotide sequences of fixed length of 60 bases and each sequence is assigned a class label of either intron or exon. The splice dataset contains total of 1,527 sequences and will be referred as **S-EI** in the experimental evaluation section.

The ***Mouse Genome dataset*** is another dataset which is made up of nucleotide sequences. This dataset was created by querying NCBI's Entrez database [BBL$^+$99] for intron and and exon sequences present in the Mouse genome. Note that unlike the splice dataset this dataset contains complete intron/exon sequences. This is one of our larger dataset and contains 10,918 sequences of average length 361.6 symbols, it will be referred as **M-EI**.

The ***E-Coli Genome dataset*** contains sequences from the genome of prokaryote organism *Escherichia coli* (e-coli) [NTPP01]. To build this dataset the e-coli genome was first split into two sets of sequences, one containing sequences making up the coding region whereas other containing sequences making up the non-coding region. Next a fixed length sequence of (75 base pairs) was obtained from each of those coding and non-coding region sequences. This dataset contains a total of 3,370 sequences, which is equal to the number of coding and non-coding regions in the e-coli genome.

The ***Protein Structure dataset*** is made of amino acid sequences and addresses the problem of assigning a secondary structure to a protein sequence. The dataset was created by processing the SWISS-PROT [BA99] database to obtain proteins for which secondary structure was known in advance. These protein sequences were then split into smaller sequences based on their secondary structure. This data set has three class labels *helix*, *strand* and *turn*. The dataset contains a total of 16,154 sequences of average length 5.2.

The ***Peptidias dataset*** also contains amino-acid sequences and tackles the problem of assigning a family (class label) to protease sequences [RB93], which are protein sequences. The protease sequences are obtained form the MERPOS [mer] database where they are classified based on their similarity in evolutionary relationship [RdHD$^+$87]. As can be expected the evolutionary information is available for few sequences and many protease sequences are not assigned to any family[1]. The dataset contains sequences from four families: *cysteine type*, *metallo-type*, *serine-type*,

[1]Proteases (scientifically referred as Peptidase) are necessary for the survival of all living creatures, and are and encoded by about 2% of genes in all kinds of organisms.

and *aspartic-type*. For our evaluation we will be concentrating only on the first three families as they are very few sequences belonging to the aspartic-type family. The total number of sequences in this data is 1,584 with an average length of 511.3.

## 3.2 Experimental Methodology

The performance of the various classification algorithms was measured using the classification accuracy. The accuracy for each scheme is estimated using ten-way cross validation. To make the evaluation of different schemes easier we restricted our experiments to two class problems; however, the proposed schemes can be easily extended to handle multiple classes. Since the datasets which are used for sequence classification have different characteristics the performance of each scheme is also not uniform across the datasets. To ease comparison of different schemes we have displayed the maximum accuracy attained for every dataset (across all the classification schemes) in bold font.

## 3.3 Performance of the KNN Techniques

Our first set of experiments was focused on evaluating the performance of the $K$-nearest-neighbor algorithm. Table 2 shows the average classification accuracy achieved by the KNN algorithm for $K = 1$, $K = 5$, and $K = 20$. For each value of $K$, the table shows three sets of experiments each corresponding to a different similarity function. The columns labeled "Global" determine the similarity between sequences by using the score of the global sequence alignment, whereas the columns labeled "Local" use the local alignment score instead. Also, for comparison purposes, the results shown in the columns labeled "Cosine" correspond to the accuracy achieved by the KNN algorithm when each sequence is represented as a frequency vector of the different symbols that it contains, and the similarity between sequences was computed as the cosine of the two vectors. Note that this representation is similar to the feature-space of the 0th order Markov chain and does not take into account any sequential constraints. Also, in our experiments we only used the simple binary scoring matrices between the different nucleotides or amino-acids. This was primarily done to make it easier to compare the various KNN and Markov chains based methods, as both methods can take advantage of relations between different amino-acids.

| *Dataset* | *K = 1* | | | *K = 5* | | | *K = 20* | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Cosine* | *Global* | *Local* | *Cosine* | *Global* | *Local* | *Cosine* | *Global* | *Local* |
| S-EI | 0.7294 | 0.9220 | 0.9115 | 0.7360 | **0.9390** | 0.9200 | 0.7497 | 0.9155 | 0.8866 |
| PS-HT | 0.8609 | 0.8890 | 0.4503 | 0.8010 | 0.8861 | 0.6876 | 0.6841 | 0.8559 | 0.4326 |
| PS-TS | 0.8142 | 0.8214 | 0.6000 | 0.7122 | 0.8162 | 0.7123 | 0.7648 | 0.6174 | 0.8298 |
| PS-HS | 0.7355 | 0.8151 | 0.5676 | 0.7322 | 0.8040 | 0.5125 | 0.7378 | 0.7532 | 0.4807 |
| EC-CN | 0.6166 | 0.6765 | 0.6908 | 0.6240 | 0.6721 | 0.6756 | 0.6623 | 0.5275 | 0.6264 |
| MG-GN | 0.6736 | 0.8451 | 0.3611 | 0.6920 | 0.8661 | 0.4811 | 0.7048 | 0.8634 | 0.5068 |
| P-CM | 0.8985 | 0.9618 | 0.6044 | 0.8754 | 0.9477 | 0.4719 | 0.8062 | 0.9015 | 0.4176 |
| P-CS | 0.9202 | 0.9722 | 0.7649 | 0.8891 | 0.9580 | 0.6683 | 0.8530 | 0.8858 | 0.4408 |
| P-MS | 0.8774 | 0.9719 | 0.6693 | 0.8553 | 0.9505 | 0.5897 | 0.8324 | 0.9062 | 0.5719 |

**Table 2**: Results of $K$ Nearest Neighbor scheme for different similarity schemes at different values of k.

Looking at these results we can see that for most datasets the scheme that uses global alignment scores as the similarity measure outperforms the other two schemes on almost all the datasets and for all the three values of $K$. Cosine similarity based techniques tend to perform significantly worse than the global alignment method for most datasets. The only exception are the PS-HT and PS-TS datasets for which the cosine-based similarity approaches achieve comparable classification accuracy to that achieved by global alignment. This can be attributed to the fact that the length of sequences for these datasets is extremely short (shown in Table 1). As a result, these datasets have limited sequential information which can be exploited by alignment based schemes. Also, the local alignment scheme performs very poorly especially on the protein sequences, indicating that basing the classification only on a single subsequence is not a good strategy for these datasets. Finally, note that unlike the behavior usually observed in other datasets, the classification accuracy does not improve as $K$ increases but actually decreases. In fact, when $K = 1$, the KNN approach tends to achieve the best results in seven out of the nine experiments.

## 3.4 Performance of the Simple Markov Chains and their Feature Spaces

Our second set of experiments was focused on evaluating the performance achieved by simple Markov chains as well as by SVM when operating on the same feature space used by the simple Markov chains. Table 3 shows the classification accuracy achieved by the zeroth-, first-, second-, and third-order Markov chains, and by the SVM algorithm in the corresponding feature spaces. Note that all the SVM results were obtained using linear kernel functions. The columns labeled "Markov" correspond to the results obtained by the Markov chains, and the columns labeled "SVM" correspond to the SVM results.

| *Dataset* | *Order = 0* | | *Order = 1* | | *Order = 2* | | *Order = 3* | |
|---|---|---|---|---|---|---|---|---|
| | *SVM* | *Markov* | *SVM* | *Markov* | *SVM* | *Markov* | *SVM* | *Markov* |
| S-EI | 0.7439 | 0.7399 | 0.7812 | 0.7700 | 0.8342 | 0.8041 | 0.8768 | 0.8454 |
| PS-HT | 0.8735 | 0.7455 | 0.8704 | 0.7487 | 0.9510 | 0.9277 | 0.8587 | 0.8196 |
| PS-TS | 0.8481 | 0.8045 | 0.8574 | 0.7888 | 0.8384 | 0.8070 | 0.7842 | 0.7378 |
| PS-HS | 0.7596 | 0.7217 | 0.8042 | 0.7409 | 0.8462 | 0.8043 | 0.6873 | 0.6526 |
| EC-CN | 0.6664 | 0.6676 | 0.7139 | 0.6931 | 0.7323 | 0.7077 | 0.7320 | 0.7338 |
| MG-GN | 0.6912 | 0.6455 | 0.8188 | 0.7219 | 0.8977 | 0.7921 | 0.9186 | 0.8178 |
| P-CM | 0.7328 | 0.7219 | 0.8914 | 0.8342 | 0.9497 | 0.9246 | 0.9427 | 0.9497 |
| P-CS | 0.7380 | 0.6767 | 0.9160 | 0.7943 | 0.9639 | 0.9420 | 0.9689 | 0.9672 |
| P-MS | 0.7225 | 0.6797 | 0.8833 | 0.7785 | 0.9431 | 0.9003 | **0.9645** | 0.9505 |

**Table 3**: Results of SVM & Markov chain based classifier at different order values.

Two key observations can be made by looking at the results in this table. First, for most datasets, the classification accuracy obtained by either the Markov chains or SVM improves with the order of the model. The only exception are the experiments performed with the datasets corresponding to secondary structure prediction (PS-HT, PS-TS, and PS-HS), in which the classification accuracy for Markov chains peaks at the first-order model, and for SVM peaks at the second-order model. The reason for that is that these datasets are made of very short sequences, that depending on the particular class, range on the average from 2.3 to 9.0 nucleotides. Consequently, higher order models and their associated feature spaces contain very few examples for the accurate calculation of the transition probabilities. Second, comparing the performance of the Markov chains against the performance achieved by SVM, we can see that almost across-the-board SVM is able to achieve higher classification accuracies. In fact, for many experiments, this improvement is in the range of 5% to 10%. This should not be surprising because as discussed in Section 2.3, both Markov chains and SVM with a linear kernel function learn a linear model. In the case of Markov chains this model is learned by using a simple maximum likelihood approach, whereas in the case of SVM the model is learned so that it best separates the two classes, which is inherently more powerful.

## 3.5 Performance of the Interpolated Markov Models and their Feature Spaces

The third set of experiments is focused on evaluating the performance of the interpolated Markov models and the performance of SVM when the various sequences are represented in the feature space used by these models. The classification accuracy achieved by these techniques is shown in Table 4. This table contain three sets of experiments. The experiments labeled *"Order = 1"* correspond to the IMM that combines the zeroth- and the first-order Markov chains, the ones labeled *"Order = 2"* correspond to the IMM that combines the zeroth-, first-, and second-order model, and so on. In our IMM experiments, we weighted the different order Markov chains equally. Better results can potentially be obtained, by weighting the conditional probabilities of the various order Markov chains differently. We did not pursue this any further, since as our discussion in Section 2.3 shows, this is done automatically by SVM.

From the results shown in Table 4 we can see that the SVM classifier outperforms the IMM based techniques for the majority of the datasets. The only class of datasets that IMM does better than SVM are the ones derived from peptidias (P-CM, P-CS, P-MS), in which the higher order IMM models do considerably better than the corresponding SVM models. We are correctly investigating the reason for this reduction in the accuracy of the SVM models. Second, we can see that the traditional Markov chain based classifiers presented in the previous section tend to outperform IMM based techniques on most of the datasets. The only exception to that are the protein structure datasets for which IMM derived classifiers achieve the highest accuracy amongst all the classifiers. This can be attributed to the fact that

| *Dataset* | *Order = 1* | | *Order = 2* | | *Order = 3* | |
|---|---|---|---|---|---|---|
| | *SVM* | *Markov* | *SVM* | *Markov* | *SVM* | *Markov* |
| S-EI | 0.7727 | 0.7530 | 0.8022 | 0.7864 | 0.8277 | 0.8153 |
| PS-HT | 0.8858 | 0.7529 | **0.9571** | 0.9392 | 0.8610 | 0.8481 |
| PS-TS | **0.8603** | 0.8116 | 0.8492 | 0.8359 | 0.7878 | 0.7770 |
| PS-HS | 0.8022 | 0.7454 | **0.8677** | 0.8441 | 0.7215 | 0.7202 |
| EC-CN | 0.7047 | 0.6830 | 0.7195 | 0.7044 | 0.7317 | 0.7207 |
| MG-GN | 0.8132 | 0.6922 | 0.8801 | 0.7355 | 0.8938 | 0.7630 |
| P-CM | 0.7689 | 0.8082 | 0.7840 | 0.9236 | 0.7941 | 0.9557 |
| P-CS | 0.8144 | 0.7565 | 0.8471 | 0.9135 | 0.8622 | **0.9722** |
| P-MS | 0.7844 | 0.7372 | 0.8161 | 0.8531 | 0.8265 | 0.9387 |

**Table 4**: Results of IMM based classifiers on different orders of Markov chain.

these datasets have sequences which are comparatively short hence there is a greater benefit in using different order Markov states for classification.

## 3.6 Performance of the Selective Markov Models and their Features Spaces

Last set of results evaluate the performance of selective Markov models and SVM using the pruned feature space of SMM. The classification accuracies of these two schemes is presented in Table 5. The experiments were conducted at three different order values. The result in column *"Order =1"* correspond to SMM built using zeroth and the first order Markov chain which are then pruned using frequency threshold. Similarly the column *"Order = 2"* contains the result for SMM which was built using zeroth, first and second order Markov chain and so on. For each SMM we let $\delta$ take the values of 0.0, 1.0, and 2.0. The value of $\delta = 0.0$ indicates no pruning and leads to models that are identical to the simple Markov chains presented in Table 3, $\delta = 1.0$ and $\delta = 2.0$ indicate that progressively more number of states in the Markov chain are pruned.

| *Dataset* | *Order = 1* | | | | | | *Order = 2* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *SVM* | | | *Markov* | | | *SVM* | | | *Markov* | | |
| | $\delta = 0.0$ | $\delta = 1.0$ | $\delta = 2.0$ | $\delta = 0.0$ | $\delta = 1.0$ | $\delta = 2.0$ | $\delta = 0.0$ | $\delta = 1.0$ | $\delta = 2.0$ | $\delta = 0.0$ | $\delta = 1.0$ | $\delta = 2.0$ |
| S-EI | 0.7812 | 0.7812 | 0.7865 | 0.7701 | 0.7721 | 0.7268 | 0.8343 | 0.8343 | 0.8074 | 0.8042 | 0.8146 | 0.7649 |
| PS-HT | 0.8705 | 0.8717 | 0.8727 | 0.7488 | 0.7446 | 0.7337 | 0.9510 | 0.9516 | 0.9519 | 0.9357 | 0.9420 | 0.9415 |
| PS-TS | 0.8574 | 0.8596 | 0.8545 | 0.7889 | 0.7830 | 0.7753 | 0.8385 | 0.8384 | 0.8385 | 0.8153 | 0.8199 | 0.8197 |
| PS-HS | 0.8043 | 0.7961 | 0.7864 | 0.7409 | 0.7328 | 0.7286 | 0.8463 | 0.8463 | 0.8422 | 0.8077 | 0.8138 | 0.8140 |
| EC-CN | 0.7139 | 0.7139 | 0.6964 | 0.6932 | 0.6932 | 0.6519 | 0.7323 | 0.7320 | 0.7187 | 0.7077 | 0.7119 | 0.7095 |
| MG-GN | 0.8188 | 0.8191 | 0.8099 | 0.7219 | 0.6662 | 0.6861 | 0.8978 | 0.8982 | 0.8883 | 0.7922 | 0.7801 | 0.7315 |
| P-CM | 0.8915 | 0.8835 | 0.8654 | 0.8343 | 0.8112 | 0.8093 | 0.9498 | 0.9488 | 0.9407 | 0.9247 | 0.9267 | 0.9127 |
| P-CS | 0.9160 | 0.9110 | 0.9051 | 0.7943 | 0.7792 | 0.7674 | 0.9639 | 0.9614 | 0.9563 | 0.9404 | 0.9412 | 0.9236 |
| P-MS | 0.8833 | 0.8841 | 0.8620 | 0.7786 | 0.7764 | 0.7424 | 0.9431 | 0.9468 | 0.9372 | 0.8997 | 0.8945 | 0.8797 |

| *Dataset* | *Order = 3* | | | | | |
|---|---|---|---|---|---|---|
| | *SVM* | | | *Markov* | | |
| | $\delta = 0.0$ | $\delta = 1.0$ | $\delta = 2.0$ | $\delta = 0.0$ | $\delta = 1.0$ | $\delta = 2.0$ |
| S-EI | 0.8769 | 0.8749 | 0.8769 | 0.8454 | 0.8467 | 0.8500 |
| PS-HT | 0.8587 | 0.8587 | 0.8587 | 0.8489 | 0.8535 | 0.8538 |
| PS-TS | 0.7842 | 0.7842 | 0.7842 | 0.7650 | 0.7675 | 0.7688 |
| PS-HS | 0.6873 | 0.6873 | 0.6873 | 0.6856 | 0.6876 | 0.6896 |
| EC-CN | 0.7320 | 0.7309 | 0.7264 | 0.7338 | **0.7344** | 0.7255 |
| MG-GN | 0.9186 | **0.9188** | 0.9083 | 0.8179 | 0.8039 | 0.8028 |
| P-CM | 0.9427 | 0.9427 | 0.7250 | 0.9538 | **0.9598** | 0.9467 |
| P-CS | 0.9689 | 0.9689 | 0.9715 | 0.9681 | 0.9681 | 0.9681 |
| P-MS | **0.9645** | 0.9645 | 0.7476 | 0.9527 | 0.9542 | 0.9468 |

**Table 5**: Results of SMM based classifiers on orders of Markov chain and varying values of $\delta$.

Looking at the results of Table 5 we can see that as it was the case with both the simple Markov chains and IMMs, using SVM on SMM's feature space leads to higher classification accuracy than that obtained by SMM. Also, even though for many problems the accuracy achieved by SMM does improve as $\delta$ increases, the gains in classification accuracy are rather small. One of the reasons for this behavior may be the relatively simple strategy that we followed

for pruning the various states of the different order Markov chains.

### 3.7 Discussion of the Results

Studying the results across the different classification schemes we can make three key observations. First, for most of the datasets, the SVM classifier used on the feature spaces of the different Markov chains and its variants achieves substantially better accuracies than the corresponding Markov chain classifier. This suggests that the linear classification modeled learned by SVM is better than the linear models learned by the Markov chain-based approaches. The classification accuracy of SVM can be improved even further if higher order kernels (*e.g.*, polynomial and Gaussian) are used. Our preliminary results with a second-order polynomial kernel suggest a considerable improvement in accuracy.

Second, the performance of the SVM classifier is influenced by the nature of the feature space on which the SVM classifier is built. This is apparent by comparing the results achieved on the IMM's feature space with the results achieved on the simple Markov chain's feature space. The maximum accuracy attained by the SVM classifier on simple Markov chains is always higher than that obtained by SVM on the IMM's feature space (the protein structure dataset is an exception because of its peculiar sequence length characteristics). This shows that increasing the feature space and in the process increasing the information available for the classifier does not necessarily improve the accuracy. At the same time, we see that even if we do even a simple frequency based feature selection, as it is done in SMM, we can achieve some improvement in the overall accuracy. These results suggest that proper feature selection can lead to improved performance, even when a powerful classifier, such as SVM, is used.

Third, we observe that the KNN scheme for the splice dataset outperforms all other classifiers by a significant margin—the nearest classifier lagging behind by approximately 6%. Our analysis of that dataset showed that the reason for this performance difference is that the KNN algorithm by computing global alignments is able to take advantage of the relative position of the aligned sequences. In fact, we performed a simple experiment in which we represented each object as a binary vector with $60 \times 4$ dimensions, in which the $i$th location of the sequence was mapped to the $(4i \ldots 4i + 3)$ dimensions of that vector, and based on the particular nucleotide, one of the four entries was one and the remaining zero. Given this representation, we used SVM to learn a linear model that was able to achieve an accuracy of over 97%, which is the highest ever reported for this dataset. This illustrates the benefit of incorporating information about the position of the symbols in the sequences, something that cannot currently be done with the Markov chain-based techniques described in this paper.

## 4 Conclusion and Direction of Future Research

In this paper we presented different sequence classification techniques and evaluated their performance on variety of sequence datasets. We show that the traditional Markov chain based sequence classifier can be thought as a linear classifier operating on a feature space generated by the various state-transition pairs of the various Markov chains. This formulation of sequence classification allows use to use sophisticated linear classifiers in place of the Markov classifier leading to substantial improvements in accuracy.

Our experimental evaluation of the various schemes also identified two important directions for future research. First, feature selection does appear to play an important role even when sophisticated classification algorithms are used. Developing methods for intelligently selecting the right set of states from the various Markov models is very much an open research area. Second, our results also point to the well-known fact that besides sequence relations, effective classifiers for biological sequences must also take into account information about the relative position of the various shared subsequences or features. One of the future goals of our research is to develop features which can exploit this position specific information.

## References

[AK98] S.F. Altschul and E.V. Koonin. Iterated profile searches with psi-blast - a tool for discovery in protein databases. *Trends in Biochemistry Science*, 23, 1998.

[BA99] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Res.*, 27(1):49–54, 1999.

[BBL+99] Dennis .A. Benson, Mark . S. Boguski, David J. Lipman, James Ostell, B. F. Francis Ouellette, BArabra A. Rapp, and David L. Wheeler. GenBank. *Nucleic Acids Research*, 27(1):12–17, 1999.

[BGH+99] W. C. Barker, J. S. Garavelli, D. H. Haft, L. T. Hunt, C. R. Marzec, B. C. Orcutt, G. Y. Srinivasarao, L.S. L. Yeh, R. S. Ledley, H.W. Mewes, F. Pfeiffer, and A. Tsugita. The PIR-International protein sequence database. *Nucleic Acids Res.*, 27(1):27–32, 1999.

[BM98] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[CV95] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[DEKG98] R. Durbin, S. Eddy, A. Krogh, and Mitchinson G. *Biological sequence analysis*. Cambridge University Press, 1998.

[DEKM98] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchinson. *Biological sequence analysis*. Cambridge University Press, 1998.

[DHK+98] A. L. Delcher, D. Harmon, S. Kasif, O. White, and S. L. Salzberg. Improved microbial gene identification with glimmer. *Nucleic Acid Research*, 27(23):4436–4641, 1998.

[DK01] M. Deshpande and G. Karypis. Selective markov models for predicting web-page accesses. In *First International SIAM Conference on Data Mining*, 2001.

[DSO78] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5, 1978.

[(ed00] Ritu Dhand (editor). *Nature Insight: Functional Genomics*, volume 405. 2000.

[Gus93] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55:141–154, 1993.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[HH92] S. Henikoff and J. G. Henikoff. Amino acid substiution matrices from protein blocks. In *Proceedings of the National Academy of Sciences of the USA*, 1992.

[KDK+01] Michihiro Kuramochi, Mukund Deshpand, George Karypis, Qing Zhang, and Vivek Kapur. Promoter prediction for prokaryotes. In *Passific Symposium on Bioinformatics (submitted)*, 2001. Also available as a UMN-CS technical report, TR# 01-030.

[KH98] Daniel Kudenko and Haym Hirsh. Feature generation for sequence categorization. In *In proceedings of AAAI-98*, 1998.

[LZO99] Neal Lesh, Mohammed J. Zaki, and Mitsunari Ogihara. Mining features for sequence classification. In *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1999.

[mer] Merpos database.

[Mit97] T.M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[Mou01] David W. Mount. *Bioinformatics: Sequence and Genome Analysis*. CSHL Press, 2001.

[NTPP01] Nicole T. Nicole T. Perna and III et al Plunkett, Guy. Genome sequence of enterohemorrhagic escherichia coli. *Nature*, 2001.

[OHN+99] Uwe Ohler, Stefan Harbeck, Heinrich Niemann, Elmar Noth, and Martin G. Reese. Interpoalted markov chains for eukaryotic promoter recognition. *Bioinformatics*, 15(5):362–369, 1999.

[RB93] N. D. Rawlings and A. J. Barrett. Evolutionary families of peptidases. *Biochemistry. Journal*, 1993.

[RdHD+87] G.R. Reeck, C. de Haen, Teller D.C., R.F. Doolittle, and et al Fitch, W.M. ẗhomologyïn proteins and nucleic acids: a terminology muddle and a way. *Cell*, 1987.

[SDKW98] Steven L Salzberg, Arthur L. Delcher, Simon Kasif, and Owen White. Microbial gene identification using interpolated markov models. *Nucleic Acids Research*, 1998.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[Vap98] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[WZH00] K Wang, S. Zhou, and Y. He. Growing decision trees on support-less assoication rules. In *Proceedings of SIGKDD 2000*, 2000.

[ZLM00] Mohamed J. Zaki, Neal Lesh, and Ogihara Mitsunari. Planmine: Predicting plan failures using sequence mining. *Intelligence Review, special issue on the Application of Data Mining*, 2000.