Scalable, Updatable Predictive Models for Sequence Data

Neeraj Koul, Ngot Bui, Vasant Honavar
Artificial Intelligence Research Laboratory
Dept. of Computer Science
Iowa State University
Ames, IA - 50014, USA.
Email: {neeraj, bpngot,honavar}@iastate.edu

***Abstract*—The emergence of data rich domains has led to an exponential growth in the size and number of data repositories, offering exciting opportunities to learn from the data using machine learning algorithms. In particular, sequence data is being made available at a rapid rate. In many applications, the learning algorithm may not have direct access to the entire dataset because of a variety of reasons such as massive data size or bandwidth limitation. In such settings, there is a need for techniques that can learn predictive models (e.g., classifiers) from large datasets without direct access to the data. We describe an approach to learn from massive sequence datasets using statistical queries. Specifically we show how Markov Models and Probabilistic Suffix Trees (PSTs) can be constructed from sequence databases that answer only a class of count queries. We analyze the *query complexity* (a measure of the number of queries needed) for constructing classifiers in such settings and outline some techniques to minimize the query complexity. We also show how some of the models can be updated in response to addition or deletion of subsets of sequences from the underlying sequence database.**

***Keywords*-sufficient statistics; PSTs; Markov Model;**

I. Introduction

Advances in high throughput sequencing and other data acquisition technologies have resulted in gigabytes of DNA, protein sequence data, and gene expression data being gathered at steadily increasing rates. These developments have resulted in unprecedented opportunities for learning from such data. Most machine learning techniques assume direct access to data. However, in many practical applications, the massive size of the data being made available coupled with memory and bandwidth constraints prohibit direct access to data. In addition, it is not difficult to envision settings in the near future, such as personalized medicine where privacy concerns may prohibit direct access to the data (e.g. DNA sequence of patients under treatment). Further, in settings where data is being made available at a rapid rate (e.g. sequence data), a local copy of the data may quickly become out of date. Hence, there is an urgent need for approaches to learning predictive models, from large datasets (that cannot fit in the memory available on the device where the learning algorithm is executed), that are scalable, able to cope with frequent data updates and do not require access to the underlying dataset.

Caragea et al. [1] have introduced a general strategy for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the remote data sources. This involves separating a learning algorithm into two components: (i) a statistical query[1] generation component that poses a set of statistical queries to be answered by a data source and (ii) a hypothesis construction component that uses the resulting statistics to modify a partially constructed hypothesis (and may further invoke the statistical query component as needed). Inspired by this work we extend this strategy to the setting of building predictive models from large sequence datasets by interacting with the data source that holds the dataset only through means of certain count queries. This approach allows us to cope with the challenges of massive data size (since in general the statistics of the data are much smaller than the size of the data), no access to underlying dataset (because it interacts with data source only through statistical queries) and in certain cases, data source updates (additions, deletions of large subsets of data).

We focus our attention on a class of *Markov Property* based class of predictive models for sequences: Markov Models, Probabilistic Suffix Trees, Interpolated Markov Models that are among some of the most widely used in sequence classification [2], text analysis [3] and related applications. We describe the specific type of queries that the data source should answer in order to build the predictive model, and precisely calculate the number of queries that are posed to a data source to build the predictor. The number of queries posed (called the *query complexity* in our model) is a measure of steps required to build the model and may be important in the cases where the data source associates a cost with answering a query or in the setting where bandwidth is at a premium. We describe certain optimization techniques that can be used to minimize the query complexity and in particular describe a lazy approach to classifying a test dataset that can be used to ameliorate

[1]A statistic is simply a function of a dataset; A statistical query returns a statistic (e.g., the number of instances in the dataset that have a specified value for a specified attribute.)

the exponential query complexity associated with a Markov Model. The rest of the paper is organized as follows. Section II covers the preliminaries. Section III describes a statistical query based approach to constructing Markov Model based sequence classifier and outlines some optimization techniques to minimize the query complexity. Section IV describes an extension of this approach to Probabilistic Suffix Trees. Section V describes an approach to updating Markov model based predictors using statistical queries. Section VI concludes with a brief summary and description of related work.

II. Preliminaries and Notation

Let Σ be the alphabet from which the sequences are constructed and C be the set of classes to which the sequences can be assigned. Given a sequence $s = \sigma_1\sigma_2\ldots\sigma_n$ and a symbol $\sigma \in \Sigma$, let $s\sigma$ represent the sequence $\sigma_1\sigma_2\ldots\sigma_n\sigma$, let σs represent the sequence $\sigma\sigma_1\sigma_2\ldots\sigma_n$ and $suffix(s)$ represent the sequence $\sigma_1\sigma_2\ldots\sigma_{n-1}$. We associate with each dataset D of sequences a descriptor $Desc^s(D) = \langle\Sigma, C\rangle$ where Σ is the alphabet from which the sequences in D are constructed and C is the classes (e.g. protein family classes) to which the sequences in D can be assigned. Let $P(s)$ be the probability of observing a sequence s and $P(\sigma|s)$ be the probability of observing the symbol σ right after the subsequence s. The empirical values for $P(s)$ and $P(\sigma|s)$ are represented by $\hat{P}(s)$ and $\hat{P}(\sigma|s)$ respectively.

Suppose the data source D supports a set of primitive queries Q_D expressed in a query language supported by the data source holding D (e.g. if D is a RDBMS such as Oracle the query language will be SQL). To build a predictive model, we assume that the system expresses statistical queries against D in its own statistical query language Λ. A *query planner* Π that transforms a query $q(s_D)$ expressed in Λ for a statistic s_D into a plan for answering s_D using some subset of the primitive statistical queries Q_D. We assume that the query planner Π has at its disposal, a set of operators O that can be used to combine the answers to queries in Q_D to obtain a statistic s_D. In the case where Q_D correspond to count queries, O may include $+, -$. A *query plan* for s_D, denoted by $plan(s_D)$, is simply an *expression tree* that successively combines the answers to the primitive queries to obtain the answer to query s_D (expressed in the query language that is understood by the query planner): Each leaf node corresponds to a *primitive query* in Q_D and each non-leaf nodes corresponds to an operator in O. We assume that the planner Π is guaranteed to produce a correct plan $plan(s_D)$ for every statistic s_D that is expressible in Λ.

The learning algorithm L (say PST), when executed against a dataset D, generates at each step i, a *set* of statistical queries $S_i(D) = \{s_D(i,1)\cdots s_D(i,n_i)\}$ where each query in S_i is expressed in Λ. Let $Plan(S_i(D)) = \{plan(s_D(i,1))\cdots plan(S_D(i,n_i))\}$ be the set of plans generated by the query planner for the set of queries $S_i(D)$. We denote by $Q(plan(s_D(i,j)))$, the set of the primitive queries used in the plan $plan(s_D(i,j))$. Note that $Q(Plan(S_i(D)))$ denotes the subset of *primitive* queries against D that to answer the set of queries $S_i(D)$. Let $Q(Plan(S_i(D))) = \sum_{j=1}^{n_i} Q(plan(s_D(i,j)))$. Let $Q^L = \sum_i Q(Plan(S_i(D)))$. Clearly, $\forall j\ Q(plan(s_D(i,j))) \subseteq Q(Plan(S_i(D))) \subseteq Q^L \subseteq Q_D$. Consider a sequence of sets of statistical queries $S_1(D)\cdots S_i(D)$ generated by L when it is executed against a dataset D. Let ϕ_i be the corresponding sequence of sets of query plans $Plan(S_1(D)), Plan(S_2(D))\cdots Plan(S_i(D))$ produced by the query planner. Let $\hat{Q}(\phi_i) = \cup_{l=1}^{i}\hat{Q}(Plan(S_l(D)))$ denotes the set of primitive queries retrieved as a result. Assuming that L generates a sequence of m query sets $S_1(D)\cdots S_m(D)$ prior to terminating with a learned hypothesis, we can define the *query complexity* of ϕ_m, denoted by $QC(\phi_m)$, as $\left|\hat{Q}(\phi_m)\right|$, that is the total number of primitive queries that are posed to the data source based on ϕ_m. The task of the query planner is to generate a sequence of sets of query plans ϕ_m so as to minimize the query complexity $QC(\phi_m)$ which can be important in settings where the data source imposes a cost for answering each primitive query .

III. MM(k-1): Markov Model of order k -1

Markov Models have been used successfully in literature to address sequence based tasks (see [4], [5], [6]). In general, it is assumed that access to the training dataset D is available. In contrast, in our setting, we assume local access to D is unavailable due to a variety of reasons. However, we assume that the learner has access to the descriptor of the data (i.e. $Desc^s(D) = \langle\Sigma, C\rangle$) and the data source holding D answers certain count queries over the dataset D. In particular, we assume the data source answers the following three types of queries: (1) the query to compute the count of sequences in D that have the subsequence s (including overlaps), denoted by $S(D, s)$; (2) the query to compute the count of the sequences in D that belong to the class c_k and have the subsequence s (including overlaps), denoted by $S(D, s, C = c_k)$ and (3) the query to compute the count of sequences in D that belong to the class c_k and have subsequences of length $|s|$ (including overlaps), denoted by $S(D, |s|, C = c_k)$.

In the MM(k-1), the estimate of the probability that a given sequence (unlabeled) $s = \sigma_1\sigma_2\ldots\sigma_n$ belongs to the class c_j is given by

$$\hat{P}_{MM(k-1)}(s, c_j) = \prod_{i=k}^{n} \hat{P}(\sigma_i|\sigma_{i-1}\ldots\sigma_{i-k+1}, C = c_j) \tag{1}$$

In the sufficient statistics model, the required terms in equation (1) can be computed using the supported queries

as

$$\hat{P}(\sigma_i|\sigma_{i-1}\ldots\sigma_{i-k+1},c_j)=\frac{S(D,\sigma_i\sigma_{i-1}\ldots\sigma_{i-k+1},C=c_j)}{S(D,|s|,C=c_j)} \quad (2)$$

The Naive Bayes for sequence classification is special case of the MM(k-1) with $k=1$ and as such can be implemented in the sufficient statistics model in a straightforward way. A straightforward approach to classify any given sequences using $MM(k-1)$ would be to precompute the results for all possible queries that may be needed to classify the set of sequences in Σ^*. It is clear from equations (1) and (2) that this involves all queries of the form $S(D,s,C=c_j)$ where $|s|=k$ and $S(D,|s|,C=c_j)$ that can be posed over $Desc^s(D)$. Since the total number of unique subsequences of length (k) is $|\Sigma|^k$, the Query Complexity of this approach is $|C|(|\Sigma|^k+1)$. As, the number of queries posed in this approach is exponential in k, it is often not feasible for large k. Hence, it is of interest to explore optimization techniques that minimize the number of queries posed to the data source D.

A. Optimization Techniques for MM(k-1)

In what follows, we give some examples of optimizations that can help reduce the query complexity of sequence classification. Let $\Sigma^k \subset \Sigma^*$ be the set of all possible sequences of length k over the alphabet Σ. It follows that $S(D,|s|,C=c_j)=\sum_{s^i\in\Sigma^{|s|}}S(D,s^i,C=c_j)$. Hence, the queries of form $S(D,|s|,C=c_j)$(where$|s|=k$) needed in equation (2) need not be posed and can be computed from the queries of the form $S(D,s,C=c_j)$. With this optimization $QC(MM(k-1)=|C||\Sigma|^k$. However, the query complexity is still exponential in k. An approach to ameliorate this exponential explosion in the number of queries is to use the *lazy approach* to classify sequences, where instead of precomputing the results for all the possible queries ahead of time, only the answers to the queries needed to classify a given dataset of sequences are retrieved. Consider a test dataset $\mathcal{T}=\{s^1,s^2,\ldots s^t\}$ of t sequences that need to be classified. Given a sequence s, let $\Lambda(s,k)$ be the set of unique subsequences of length k in s. From equation (1) it follows that the required queries to classify a sequence s is $|C|(1+|\Lambda(s,k)|)$. Hence, the query complexity of the lazy approach to classify the dataset $\mathcal{T}$ is $|C|(1+\sum_{i=1}^{|\mathcal{T}|}|\Lambda(s^i,k)|)$. Since $\Lambda(s,k)$ is atmost $|s|-k+1$ (i.e. when all subsequences of length k in s are unique), the query complexity $QC(MM(k-1))\leq |C|\left(1+\sum_{i=1}^{|\mathcal{T}|}(|s^i|-k+1)\right)$. The query complexity can be further reduced through the use of caching. Suppose that the system maintains a cache of answers to primitive queries answered during the execution of L against D. Before querying the data source D we can check if the answer to the query is available in the cache. Assuming that the sequences to be classified arrive in the order $s^1,s^2\ldots s^t$, let $cache_i$ contain the answers to queries answered by D in the course of classifying sequences s^1 through s^{i-1}. Because the cache is initially empty, $cache_1=\phi$. Let $\Lambda(s^i,k,D)$ denote the results to the queries for the counts, as obtained from D, of corresponding sequences in $\Lambda(s^i,k)$. Hence, $cache_{i+1}=\{\Lambda(s^1,k,D)\cup\Lambda(s^2,k,D)\ldots\cup\Lambda(s^i,k),D\}$. The additional queries needed to be posed to the datasource D to classify sequence s^i given that the sequences $s^1\ldots s^{i-1}$ have been already classified correspond to obtaining the set of counts $\Delta(s^i,k,D)=\Lambda(s^i,k,D)-\{\Lambda(s^1,k,D)\cup\Lambda(s^2,k,D)\ldots\cup\Lambda(s^{i-1},k,D)\}$. Hence, the query complexity of the resulting approach is $QC(MM(k-1))=|C|(1+\Sigma_{i=1}^{|\mathcal{T}|}|\Lambda(s^i,k)|)-\{|\Lambda(s^{i-1},k)\cup\Lambda(s^{i-2},k)\ldots\Lambda(s^0,k)|\}$ with $\Lambda(s^0,k)=\phi$.

B. Interpolated Markov Models

Higher order Markov Models have a greater expressive power than their lower order counterparts. However, the higher the order of the Markov model, the less reliable are the estimates of the model parameters. The Interpolated Markov Models provide a means of dealing with this problem using a weighted combination of Markov models with several different choices of k (see [7], [8]). Given a sequence $s=\sigma_1\sigma_2\ldots\sigma_n$ let $s_i=\sigma_1\sigma_2\ldots\sigma_i$ be the subsequence ending at position i and $s_{i,j}=\sigma_{i-j}\sigma_{i-j+1}\ldots\sigma_{i-1}$ be sequence composed of the j positions that precede σ_i. Then the estimate of the probability of a sequence s belonging to the class c_j using an Interpolated Markov Model of order k is denoted by $\hat{P}_{IMM(k)}(s,c_j)$ and

$$\hat{P}_{IMM(k)}(s,c_j)=\Sigma_{i=1}^{n}IMM_k(s_i,c_j)$$

where $IMM_k(s_i,c_j)=\lambda_k(s_{i-1})\hat{P}_{MM(k)}(s_i,c_j)+(1-\lambda_k(s_{i-1}))IMM_{k-1}(s_i,c_j)$ and $\lambda_k(s_{i-1})$ is the numeric weight associated with the k-mer ending at position $i-1$ in sequence s (i.e. $s_{i,k}$) and $\hat{P}_{MM(k)}(s_i,c_j)$ is the estimate obtained from training data with the k^{th} order Markov model (see [8], [9] for details). The estimate $\hat{P}_{MM(k)}(s_i,c_j)$ required to build the Interpolated Markov Models (IMMs) can be computed, in the sufficient statistics model, as described earlier in section III. Hence we need a way to compute the numeric weight $\lambda_k(s_{i-1})$ using only statistical queries. Consider for example, the computation of $\lambda_k(s_{i-1})$ in Glimmer [9]. These weights can be computed in our setting using statistical queries of the form $S(D,s_{i,k},C=c_j)$ and $S(D,s_{i,k}\sigma,C=c_j)$ where $\sigma\in\Sigma$. Specifically, $\lambda_k(s_{i-1})=1$ when $S(D,s_{i,k},C=c_j)$ is greater than some threshold (for Glimmer the threshold is 400). When the count is less than the threshold, we compare the observed frequencies of $S(D,s_{i,k}\sigma,C=c_j)$ $(\sigma\in\Sigma)$ with those predicted by IMM of order $k-1$. Using a statistical test we compute the confidence (say d) that the observed frequencies are not consistent with those predicted by $\hat{P}_{IMM(k-1)}(s_{i,k}\sigma,c_j)$. When $d<0.5$, $\lambda_k(s_{i-1})=0$ and for $d\geq 0.5$, $\lambda_k(s_{i-1})=d/400\times S(D,s_{i,k},C=c_j)$. Thus Interpolated Markov

Models can be implemented using statistical queries against the data source D.

IV. Probabilistic Suffix Trees

The Probabilistic Suffix Trees (PSTs) originally introduced by Ron et al. [10] have been successfully used to model and predict protein families [2] [11]. The PSTs exploit the so called *short memory* feature of natural sequences wherein the probability distribution of the next symbol given the preceding sequence can be approximated by observing at most L preceding symbols of the sequence (L being the memory length of the PST). To use PSTs for sequence classification, we need to train a PST for each class; To classify an unlabeled sequence, we compute the probability of the sequence given the class (i.e., the corresponding PST) and assign it to the class with the largest probability. We first describe the algorithm to build a PST (say for class label $C = c_j$) using an available training dataset. The specific construction algorithm, **Build-PST**, is adapted from [2] and is described below. The procedure uses five external parameters: L the memory length, P_{min} the minimum probability which subsequences are required to occur and three parameters α, γ_{min} and r with values between zero and one (refer [2] for details). The procedure uses $\bar{T}$ to denote the PST and $\bar{T}$ is constructed iteratively starting with the root node. Each node (say labeled with s) maintains a vector $\bar{\gamma}_s$ which encodes the probability distribution (over the next symbol) associated with the node s (we use $\bar{\gamma}_s(\sigma)$ to denote the probability of the symbol σ in the distribution $\bar{\gamma}_s$).

Algorithm: **Build-PST**$(P_{min}, \alpha, \gamma_{min}, r, L)$

(1) *Initialization*: let $\bar{T}$ consists of a single root node (with an empty label), and let $\bar{S} \leftarrow \{\sigma | \sigma \in \Sigma \text{ and } \hat{P(\sigma)} \leq P_{min}\}$

(2) *Building the PST skeleton*: while $\bar{S} \neq \phi$, pick any $s \in \bar{S}$ and do:

(a) Remove s from $\bar{S}$

(b) If there exists a symbol $\sigma \in \Sigma$ such that

$$\hat{P}(\sigma|s) \geq (1+\alpha)\gamma_{min}$$

and

$$\frac{\hat{P}(\sigma|s)}{\hat{P}(\sigma|suffix(s))} \begin{cases} \geq r \\ or \\ \leq 1/r \end{cases}$$

then add to $\bar{T}$ the node corresponding to s and all the nodes on the path to s from the deepest node in $\bar{T}$ that is a suffix of s.

(c) If $|s| \leq L$ then add the strings $\{\acute{\sigma}s | \acute{\sigma} \in \Sigma$ and $\hat{P}(\acute{\sigma}s) \geq P_{min}\}$ (if any) to $\bar{S}$.

(3) *Smoothing the prediction probabilities* For each s labeling a node in $\bar{T}$, let

$$\bar{\gamma}_s(\sigma) = (1 - |\Sigma|\gamma_{min})\hat{P}(\sigma|s) + \gamma_{min}$$

Note the final step (step(3)) of the algorithm corresponds to a parameter smoothing step.

Build-PST iteratively adds nodes (step (2)) to obtain a PST. The terms calculated in step (2) are $\hat{P}(\sigma|s)$, $\hat{P}(\acute{\sigma}s)$ and $\hat{P}(\sigma|suffix(s))$. These terms can be calculated using statistical queries as follows:

- $\hat{P}(\sigma|s) = \frac{S(D, s\sigma, C=c_j)}{S(D, |s\sigma|, C=c_j)}$
- $\hat{P}(\acute{\sigma}|s) = \frac{S(D, \acute{\sigma}s, C=c_j)}{S(D, |\acute{\sigma}s|, C=c_j)}$
- $\hat{P}(\sigma|suffix(s)) = \frac{S(D, suffix(s)\sigma, C=c_j)}{S(D, |suffix(s)\sigma|, C=c_j)}$

Since $|s\sigma| = |\acute{\sigma}s|$, it follows that the query $S(D, |s\sigma|, C = c_j)$ is the same as $S(D, |\acute{\sigma}s|, C = c_j)$. Hence, in each iteration of step (2), requires five different queries to be answered by D. If $r(D, c_i)$ is the number of times the step (2) is executed during the construction of the PST for class $c_i \in C$, then $QC(PST) = 5\sum_{i=1}^{|C|} r(D, c_i)$. In practice $r(D, c_i)$ and hence the Query Complexity depends on the dataset D as well as choice of P_{min}. However, in the worst case the number of queries submitted is bounded by the number of queries needed to to build Markov Models of length through 1 and L. Hence, the query complexity $QC(PST) \leq |C| \sum_{k=1}^{L} |\Sigma|^k$.

V. Updatable Predictive Models

The advent of automated high throughput sequencing techniques has resulted in an exponential increase in the rate at which genomic sequence data is being generated. Many practical applications call for techniques that allow the predictive models to be updated without the need to regenerate the model from scratch. The update can either be *additive* wherein new data needs to be incorporated into the model or *subtractive* wherein the contributions of some of the old data need to be discarded from the model.

Given a dataset D and a learning algorithm ψ, let $\psi(D)$ be a predictive model (e.g., a Markov model) built from the data set D using a learning algorithm ψ. In the sufficient statistics model, let $\theta^{\psi}(D)$ be the set of primitive queries required over dataset D to build $\psi(D)$.

Updatable Model Given datasets D_1 and D_2 such that $D_1 \subseteq D_2$, we say that the predictive model constructed using ψ is updatable iff we can specify functions f and g such that

1) $\theta^{\psi}(D_2) = f(\theta^{\psi}(D_2 - D_1), \theta^{\psi}(D_1))$
2) $\theta^{\psi}(D_1) = g(\theta^{\psi}(D_2), \theta^{\psi}(D_2 - D_1))$

Theorem 1: Markov Models are *updatable* by a statistical query based learning algorithm.

Proof: For a Markov Model queries of the form $S(D, s, C = c_j)$ and $S(D, |s|, C = c_j)$ over $Desc^s(D)$ form the set $\theta^{\psi}(D)$ (see equations (1) and (2)). Given datasets D_2 and D_1 such that $D_1 \subseteq D_2$, it is easy to see that

$S(D_2, s, C = c_j) = S(D_2 - D_1, s, C = c_j) + S(D_1, s, C = c_j)$. Similarly, $S(D_2, |s|, C = c_j) = S(D_2 - D_1, |s|, C = c_j) + S(D_1, |s|, C = c_j)$. As a result the set $\theta^{\psi}(D_2)$ can be constructed from $\theta^{\psi}(D_2 - D_1)$ and $\theta^{\psi}(D_1)$. Similarly, $S(D_1, s, C = c_k) = S(D_2, s, C = c_k) - S(D_2 - D_1, s, C = c_k)$ and $S(D_1, |s|, C = c_k) = S(D_2, |s|, C = c_k) - S(D_2 - D_1, |s|, C = c_k)$. Consequently the set $\theta^{\psi}(D_1)$ can be constructed from $\theta^{\psi}(D_2)$ and $\theta^{\psi}(D_2 - D_1)$. ■

Observation The PST built using the **Build-PST** procedure is not updatable. This is due to the fact that in step 2(c) a string is added to set $\bar{S}$ only if it probability is greater than P_{min}. It is possible that this condition is satisfied for a string (say x) in $D_2 - D_1$ but not in D_1 (say when x never occurs in D_1 but occurs in $D_2 - D_1$). As a result the queries to estimate $P(\sigma|x)$ from dataset D_1 are never posed while being posed for the dataset $D_2 - D_1$. Hence, the PST is not updatable since it is not possible to compute $\hat{P}(\sigma|x)$ for the dataset D_2 only from queries posed to compute $\hat{P}(\sigma|x)$ for dataset $D_2 - D_1$

VI. Summary and Related Work

Summary: Due to the exponential increase in the rate at which sequence data are being generated, there is an urgent need for efficient algorithms for learning predictive models of sequence data from large sequence databases and for updating the learned models to accommodate additions or deletions of data in settings where the sequence database can answer only a certain class of statistical queries.

In this paper we presented an approach to learning predictive models from sequence data using sufficient statistics by posing count queries against a sequence data source. This approach can be used to build the predictive model without access to the underlying data as long the data source is able to answer a class of count queries. In addition, this approach scales well to settings where the dataset is very large in size because it does not need to load the entire dataset in memory. We have also outlined some optimization techniques to minimize the number of queries submitted to the data source. In addition, we showed how the class of Markov model based predictors can be updated in response to addition or deletion of subsets of the data.

Related Work: The approach to learning Markov models and their variants presented in this paper builds on the statistical query based approach to learning from large datasets (including distributed data sets) introduced by Caragea et al. [1]. Markov Models have been successfully used in a broad range of applications in computational biology including gene finding (e.g. GeneMark [4] and GenScan [5]), protein classification [12] [13] , among others. Interpolated Markov Models have been used for gene finding by Salzberg et al. [8]. Bajerono et al. [2] and [11] have used Probabilistic Suffix Trees for protein classification. Variable order Markov Models are discussed in [6]. Abouelhoda et al. [14] have investigated approaches to reducing the memory requirements of suffix tree construction algorithms.

Acknowledgement

This research was supported in part by the grant IIS 0711356 to Vasant Honavar from the National Science Foundation.

References

[1] D. Caragea, A. Silvescu, and V. Honavar, "A framework for learning from distributed data using sufficient statistics and its application to learning decision trees," *International Journal of Hybrid Intelligent Systems*, vol. 1, p. 2004, 2004.

[2] G. Bejerano and G. Yona, "Variations on probabilistic suffix trees: statistical modeling and prediction of protein families," *Bioinformatics*, vol. 17, no. 1, pp. 23–43, 2001.

[3] A. McCallum, D. Freitag, and F. C. N. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *ICML*, 2000, pp. 591–598.

[4] M. Borodovsky and J. D. McIninch, "Genmark: Parallel gene recognition for both dna strands," *Computers & Chemistry*, vol. 17, no. 2, pp. 123–133, 1993.

[5] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic dna," *J. Mol. Biol*, vol. 268, pp. 78–94, 1997.

[6] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order markov models," *J. Artif. Intell. Res. (JAIR)*, vol. 22, pp. 385–421, 2004.

[7] H. Zhu, J. Wang, Z. Yang, and Y. Song, "Interpolated hidden markov models estimated using conditional ml for eukaryotic gene annotation," in *Computational Intelligence and Bioinformatics*, 2006, pp. 267–274.

[8] S. Salzberg, A. L. Delcher, S. Kasif, and O. White, "Microbial gene identification using interpolated markov models," *Nucleic Acids Research*, vol. 26, pp. 544–548, 1998.

[9] S. L. Salzberg, M. Pertea, A. L. Delcher, M. J. Gardner, and H. Tettelin, "Interpolated markov models for eukaryotic gene finding," *Genomics*, vol. 59, pp. 24–31, 1999.

[10] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia," in *Machine Learning*, vol. 6, 1996, pp. 176–183.

[11] Z. Sun and J. S. Deogun, "Local prediction approach for protein classification using probabilistic suffix trees," in *APBC*, 2004, pp. 357–362.

[12] Z. Yuan, "Prediction of protein subcellular locations using markov chain models," *FEBS Letters*, pp. 23–26, 1999.

[13] O. Yakhnenko, A. Silvescu, and V. Honavar, "Discriminatively trained markov model for sequence classification," in *ICDM*, 2005, pp. 498–505.

[14] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch, "Replacing suffix trees with enhanced suffix arrays," *J. of Discrete Algorithms*, vol. 2, no. 1, pp. 53–86, March 2004.