Automated Classification of DNA Structure from Sequence Information

David M. Loewenstern*
Department of Computer Science
Rutgers University
Piscataway, NJ 08855
and
Bell Laboratories
Whippany, NJ 07981

Helen M. Berman†
Department of Chemistry
and
Waksman Institute
Rutgers University
Piscataway, NJ 08855

Haym Hirsh‡
Department of Computer Science
Rutgers University
Piscataway, NJ 08855

June 26, 1997

Keywords: *DNA Backbone Conformation, Entropy Estimation, DNA Classification, Three-Dimensional DNA Structure.*

Abstract

We introduce an algorithm, LLLAMA, which combines simple pattern recognizers into a general method for estimating the entropy of a sequence. Each pattern recognizer exploits a partial match between subsequences to build a model of the sequence. Since the primary features of interest in biological sequence domains are subsequences with small variations in exact composition, LLLAMA is particularly suited to such domains. We describe two methods, LLLAMA-length and LLLAMA-alone, which use this entropy estimate to perform maximum *a posteriori* classification. We apply these methods to several problems in three-dimensional structure classification of short DNA sequences. The results include a

*Email: loewenst@paul.rutgers.edu. Phone: 201-761-5949. Fax: 908-445-0537.

†Email: berman@adenine.rutgers.edu.

‡Email: hirsh@cs.rutgers.edu.

surprisingly low 3.6% error rate in predicting helical conformation of oligonucleotides. We compare our results to those obtained using more traditional methods for automated generation of classifiers.

1 Introduction

Although it is often convenient to think of DNA as a sequence of characters drawn from an alphabet $\{A, C, G, T\}$, it is of course a chemically active molecule with a complex three-dimensional structure. It would be of biological interest to be able to predict three-dimensional structural characteristics of a sequence of DNA without deriving it using x-ray crystallography or NMR spectroscopy.

This paper examines several methods for predicting structural characteristics of a test sequence of DNA given only the sequence of nucleotides, and no other information about the sequence. In particular, conformational geometry, crystallographic unit cell, and space group information for the test sequence is not made available, and is in fact predicted by the methods.

There are three structural characteristics, or tasks, of interest. The first task is to predict the helical conformational class, *i.e.*, whether the DNA sequence forms an A-, B-, or Z-helix. The second task is to predict the crystal type, which is to say the crystallographic unit cell and space group. The third task is prediction of packing motif: a group of crystal types belong to the same motif if the molecular interactions within the crystal are similar [2]. We would like to solve these tasks for short DNA sequences (fewer than 13 nucleotides). For our purposes, it is sufficient to label the sequence with exactly one helical conformational class, one crystal type, and one packing motif.

The problem we are faced with is illustrated for the helical conformation task in Figure 1. The goal is to predict one of the three structural characteristics of short DNA sequences of test sequences. Furthermore, we wish to use general machine learning techniques which can be easily applied to a range of DNA classification tasks. To this end, we extracted a training corpus from the Nucleic Acid Database (NDB) [3] composed of all nucleotide sequences with exactly one known helical conformational class, crystal type, and packing motif. The corpus (unlike the NDB) contains only symbols representing the sequence itself, and does not contain three-dimensional coordinate information. To train the classifier for helical class, the corpus is labeled with helical classes from the NDB; similarly, the corpus is labeled with crystal types to train for crystal type classification and packing motifs to train for packing motif classification. Rather than engineering a specific classifier for each task by hand, we explored machine learning methods that extract classification information from the labeled training corpus alone, without using other biological information.

To rephrase the problem as a machine learning task, for each of the three tasks, we construct from the NDB a corpus of sequences drawn from one of two fixed alphabets. Each sequence in the corpus is labeled with a class which has also been extracted from the NDB, and ultimately was associated with each sequence by one or more teams of X-ray crystallographers. The corpora are used to train and test, and so compare, several different classification methods. The methods represent three different views of corpus-based DNA

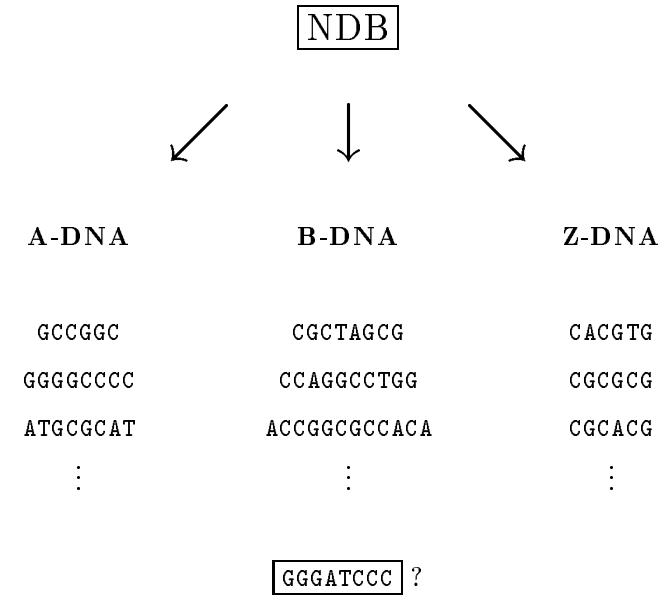

Figure 1: Sample data, in helical conformation classes, A-DNA, B-DNA, Z-DNA, as derived from the Nucleic Acid Database (NDB). The boxed sequence is to be classified.

structure classification. To illustrate these views, refer to Figures 2-6.

The first of these methods, represented by C4.5 [15], treats individual features separately, as a *set* of attribute/value pairs, also known as features. It then tries to build a classifier from training examples that examines groups of features to make a structure prediction. When there is little prior knowledge available to build the attributes, a naive feature representation (such as the sequence of nucleotides itself) must be used, which is generally not very effective [8] (Figure 2). This method works well when attributes represent more complex, independent features such as length, %CG, or charge (Figure 3). However, this representation requires a substantial amount of biological knowledge to create, and must be created anew for each new classification problem.

A second view, represented by a k-nearest-neighbor method [18], looks for similarities between a sequence and each individual training example, seen as a whole (Figure 4). Here, some biological knowledge must be used to judge similarity.

A third view builds models which attempt to predict the *nucleotides* within each sequence of a particular class, using an entropy estimation algorithm. These models may then be used to classify new sequences by judging to which class new sequences are most similar (Figure 5). They determine this similarity by determining the probability that the nucleotides of the test sequence could have been "generated" by the same model which "generated" the training corpus for the given class. These methods require good entropy estimation algorithms which can exploit the structure and redundancy in the training corpus for each class to predict the nucleotides of the test sequences.

```
pos10 = A: B-DNA (5.0)
pos10 = C: A-DNA (5.0)
pos10 = T: A-DNA (4.0)
pos10 = G:
|   pos8 = A: B-DNA (5.0)
|   pos8 = C: A-DNA (2.0/1.0)
              ⋮
pos10 = _:
|   pos8 = A: A-DNA (1.0)
|   pos8 = C: A-DNA (28.0)
|   pos8 = G: A-DNA (4.0/1.0)
|   pos8 = T: A-DNA (3.0)
|   pos8 = _:
|   |   pos2 = A: Z-DNA (1.0)
|   |   pos2 = C: A-DNA (2.0)
                  ⋮
```

For `GGGATCCC` :

pos1	G
pos2	G
pos3	G
pos4	A
pos5	T
pos6	C
pos7	C
pos8	C
pos9	-
pos10	_
pos11	_
pos12	_

Figure 2: Sample data as handled by C4.5, naive feature representation.

```
#A's = 0: A-DNA (5.0)
4 <= #A's <= 12: B-DNA (5.0)
1 <= #A's <= 3:
|   has_GCG = true: B-DNA (4.0)
|   has_GCG = false:
|   |   is_palindrome = true: A-DNA (4.0/1.0)
                    ⋮
```

For `GGGATCCC`:

#A's	1
#C's	3
#G's	3
#T's	2
has_GCG	false
has_AT	true
is_palindrome	true
%CG	75
length	8

Figure 3: Sample data as handled by C4.5, using features designed for the task.

distance(`GGGATCCC`,`GCCGGC`)=6.0
distance(`GGGATCCC`,`GGGGCCCC`)=2.0
distance(`GGGATCCC`,`ATGCGCAT`)=6.0
distance(`GGGATCCC`,`CGCTAGCG`)=6.0
⋮
nearest-neighbor(`GGGATCCC`)=`GGGGCCCC`

So, classify `GGGATCCC` as Class(`GGGGCCC`)=A-DNA

Figure 4: Sample data as handled by 1-nearest-neighbor using Smith-Waterman edit distance.

$Pr(\boxed{\texttt{GGGATCCC}}\|A-DNA) = 0.015$	$Pr(\boxed{\texttt{GGGATCCC}}\|B-DNA) = 0.002$	$Pr(\boxed{\texttt{GGGATCCC}}\|Z-DNA) = 0.005$
$Pr(A-DNA) = 0.38$	$Pr(B-DNA) = 0.38$	$Pr(Z-DNA) = 0.24$
$Pr(A-DNA\|\boxed{\texttt{GGGATCCC}}) = 0.74$	$Pr(B-DNA\|\boxed{\texttt{GGGATCCC}}) = 0.10$	$Pr(Z-DNA\|\boxed{\texttt{GGGATCCC}}) = 0.16$

So, classify $\boxed{\texttt{GGGATCCC}}$ as A-DNA

Figure 5: Sample data as handled by LLLAMA-alone.

$Pr(\boxed{\texttt{GGGATCCC}}\|A-DNA,8) = 0.15$	$Pr(\boxed{\texttt{GGGATCCC}}\|B-DNA,8) = 0.02$	$Pr(\boxed{\texttt{GGGATCCC}}\|Z-DNA,8) = 0.001$
$Pr(A-DNA,8) = 0.97$	$Pr(B-DNA,8) = 0.02$	$Pr(Z-DNA,8) = 0.01$
$Pr(A-DNA\|\boxed{\texttt{GGGATCCC}}) = 0.99$	$Pr(B-DNA\|\boxed{\texttt{GGGATCCC}}) = 0.01$	$Pr(Z-DNA\|\boxed{\texttt{GGGATCCC}}) = 0.00$

So, classify $\boxed{\texttt{GGGATCCC}}$ as A-DNA

Figure 6: Sample data as handled by LLLAMA-length, where the length of $\boxed{\texttt{GGGATCCC}}$ is 8.

We describe an entropy estimation algorithm, LLLAMA,[1] that is well-suited to entropy estimation of biological sequences, because it exploits inexact repeats of subsequences to make its nucleotide predictions. We then make use of this algorithm in two ways. One way classifies a test sequence by predicting for the test sequence the training class most likely to have generated the test sequence according to the class's LLLAMA-model (Figure 5). The second way classifies by predicting the training class most likely to have generated the test sequence according to a LLLAMA-model constructed from a reduced training corpus of sequences of the same length as the test sequence (Figure 6). We demonstrate that these entropy-estimation methods perform better in general on the given tasks than any of the others. We also demonstrate that all of the above methods perform surprisingly well on the three tasks, and so give hope that at least small-scale DNA structure prediction is computationally tractable.

The key result of our work is that all three types of methods perform adequately on all three tasks. The best method, LLLAMA-length, has 96.4% accuracy at predicting helical conformational class, 82.1% accuracy at predicting crystal type, and 89.1% accuracy at predicting packing motif given only a nucleotide sequence. These results are made more impressive considering the very small size of the training corpus (138 sequences, 6–12 nucleotides each). It is expected that as more sequences are entered into the NDB, it will become possible to train more accurate classifiers.

Section 2 describes the data in more detail. Section 3.1 provides a brief introduction to C4.5 and k-nearest-neighbor methods. Section 3.2 then describes the LLLAMA-alone and LLLAMA-length methods, and Section 3.3 the underlying LLLAMA algorithm. Finally, this paper presents experimental results comparing

[1] LLLAMA Looks Like A Meaningful Acronym, with apologies to Ogden Nash [14].

the classification methods in Section 4, discusses related work in Section 5, and draws conclusions in Section 6.

2 Data

2.1 DNA sequences

The data used for these analyses were those contained in the Nucleic Acid Database (NDB). The NDB contains all three-dimensional structures determined using x-ray crystallography. The data are organized in a relational database which is queried using the program NDBQuery [19]. Constraints were applied so that the resulting reports contained sequences sorted according to conformation type, crystal type, and packing type. The conformation types for DNA helices are the two right-handed forms, A-DNA and B-DNA, and the left-handed form Z-DNA. For this study, structures containing modified residues were included; structures with mismatches were rejected. In the NDB, the structures are classified according to their crystal type and packing motif. Crystal types are defined according to the unit cell dimensions and space groups. Structures are considered to be isomorphous if they have the same crystal type. At the time of these analyses, there were 11 crystal types among the A-DNA structures. B-DNA structures had 16 crystal types, and Z-DNA structures had 5 crystal types. Packing motifs are defined according to the way in which molecules interact in the crystal. For B-DNA there were three motifs, for Z-DNA there were two and for A-DNA there was one [2].

2.2 Data Representation

For testing purposes, the data for each of the three tasks were considered separately. In addition, for each task, we constructed three different data sets which were labeled by the structure class of each sequence. For each of the three data sets, training and test sets were constructed repeatedly using a leaving-one-out method [18]. The training sets were then used to train each of the various classification methods, which were compared on their average accuracy on the test sets as discussed in Section 4.

Two different data representations were tried for the data. Both representations included only the sequence of nucleotides on one strand of the oligonucleotide, plus delimiter characters. In one, a 13-character *expanded* alphabet was used: $a, c, g, t, u, I, A, C, G, T, U, \wedge, \$$, where I=inosine and A, C, G, T, U are chemically modified a, c, g, t, u respectively, and where $\wedge$ denotes the 3' end of the sequence and \$ the 5' end. The other representation used a six-character *simplified* alphabet: $a, c, g, t, \wedge, \$$. In this representation, both modified and unmodified nucleotides were represented by their unmodified symbol, and uracil was represented by t. Other than the delimiters $\wedge$ and \$, no information outside the sequences themselves was encoded into the data. The size of this *full* data set is 138 sequences, each of length 6 to 12 nucleotides.

Additionally, a *pruned* data set was formed for each task by removing all sequences containing modified nucleotides or uracil. The resulting data set was represented using the simplified alphabet. The size of the

pruned data set is 86 sequences, again of length 6 to 12 nucleotides each.

3 Methods

3.1 Baseline Classification Methods

To provide a standard of comparison for our method, we rated the classification performance of three other methods on the same data.

The first of these methods was C4.5 [15]. For this method, the data was encoded as a twelve-position feature vector with feature 1 corresponding to the 3'-most nucleotide in the sequence, feature 2 its 5' neighbor, and so forth. If the sequence was shorter than twelve nucleotides, the last several features were given the placeholder value _. There were two alphabets used for encoding nucleotides as features for the full data set, a simplified alphabet $a, c, g, t, \_$, and a full alphabet $a, c, g, t, u, I, A, C, G, T, U, \_$, corresponding to the two alphabets described in Section 2.2. In addition, this method was run on the pruned data set using the unmodified only alphabet.

C4.5 works by considering each feature (nucleotide) separately. Each feature is examined to find the feature which best splits the training data into separate classes. In Figure 2, this is feature pos10. C4.5 then repeats this procedure for each subset formed by splitting using the feature, until each subset contains sequences which are nearly (within an external parameter) all from the same class. A test sequence such as `GGGATCCC` is classified by using its features to determine which final subset it would have joined if it had been part of the training set, and classifying it with the dominant class of that subset. In this example, `GGGATCCC` has features pos10=_ and pos8=C, which alone place it into a class labeled A-DNA. It should be noted that the representation making each feature correspond to a single nucleotide is sensitive to the alignment of the sequences.

The second method was a k-nearest-neighbor algorithm [18], using the Smith-Waterman edit distance function [17] to estimate the distance between sequences. Again, two different representations of the alphabet were used, an simplified alphabet a, c, g, t and an alphabet including modified and unmodified variants $a, c, g, t, I, A, C, G, T, U$. As discussed in Section 4, we used the external parameter value $k = 1$. k-nearest-neighbor classifies `GGGATCCC` by finding the nearest match, `GGGGCCCC`, and classifying `GGGATCCC` as the class to which `GGGGCCCC` belongs, A-DNA.

The third method was a simple baseline method which classifies each test sequence as the most frequently occuring class among all sequences of the same length in the training corpus. If there is no sequence of the same length in the training corpus, the most frequently occuring class of all sequences in the training corpus is used. This method is labeled "MFC" in the tables. MFC classifies `GGGATCCC` as A-DNA because there are more A-DNA sequences in the training set of length 8 than any other class of sequences of the same length.[2]

[2] We also evaluated a simpler form of MFC, which did not take into account sequence length. This method performed

3.2 Classification Method

Our overall goal is to classify unidentified DNA fragments according to class. We have three tasks: to classify by DNA conformation (3 classes: A-, B-, or Z- DNA), by crystal type (20 classes), or by packing motif (11 classes).

Our classification method is Maximum A Posteriori (MAP). We assume that there is an underlying probabilistic model for each class, and each DNA sequence in the class can be understood as being generated stochastically from the model. Therefore, there is a probability $\Pr(s|m)$ that a given sequence s could have been generated by the model m. We will discuss estimating $\Pr(s|m)$ in Section 3.3.

We make use of our models in two different classification methods. In the first, called LLLAMA-alone, we build a model m_C for each class C, and we estimate $\Pr(m_C)$ simply by counting the number of sequences in class C and dividing by the total number of sequences: that is, we assume $\Pr(m_C) = \Pr(C)$ and in general $\Pr(\cdot|m_C) = \Pr(\cdot|C)$. In Figure 5, as an example, we build models m_{A-DNA}, m_{B-DNA}, and m_{Z-DNA} using LLLAMA and the corresponding class of data in the training set. We are interested in classifying GGGATCCC, which is to say that we wish to choose the class C in { A-DNA, B-DNA, Z-DNA } which maximizes $\Pr(C|\texttt{GGGATCCC})$. By Bayes' rule:

$$\Pr(C|\texttt{GGGATCCC}) = \frac{\Pr(\texttt{GGGATCCC}|C)\Pr(C)}{\Pr(\texttt{GGGATCCC})}$$

Since $\Pr(\texttt{GGGATCCC})$ is the same for all classes C, we can ignore this term; we use m_C to estimate $\Pr(\texttt{GGGATCCC}|m_C)$, which we assume is the same as $\Pr(\texttt{GGGATCCC}|C)$. Finally, we classify sequence GGGATCCC as belonging to the class C whose model m_C maximizes $\Pr(\texttt{GGGATCCC}|m_C)\Pr(m_C)$, which in this case is A-DNA.

In the second classification method, called LLLAMA-length, we build different models $m_{C,L}$ for each class C and sequence length L. We can then estimate $\Pr(m_{C,L})$ by counting the number of sequences of length L in class C and dividing by the number of sequences of length L: that is, $\Pr(m_{C,L}) = \Pr(C|L)$. For example, in Figure 6, since GGGATCCC has length 8, we generate $m_{C,8}$ for each of the three classes, using LLLAMA and those sequences in the training set whose class is C and length is 8. We calculate $\Pr(C, 8)$ using the same training sequences, and classify sequence GGGATCCC as belonging to the class C whose model $m_{C,8}$ maximizes $\Pr(\texttt{GGGATCCC}|m_{C,8})\Pr(m_{C,8})$, again A-DNA.

It was not obvious before we ran our experiments whether LLLAMA-alone would outperform LLLAMA-length or *vice-versa*. On one hand, LLLAMA-length is able to make use of length information to make its classification. On the other, LLLAMA-alone is able to train the LLLAMA model on a larger pool of training data, composed of all lengths of sequences, and so is less likely to be over-trained.

extremely badly (error rates in excess of 50% on all tasks) and so is not discussed further.

Hamming distance	Expected # matches	Observed # matches	% hit at least once	% correct given 1+ hits
0	0.00183	0.831	71.36%	76.69%
1	0.0549	1.86	85.43	45.20
2	0.686	7.00	99.00	37.67
3	4.47	15.5	100.0	31.80
4	17.1	24.0	100.0	29.29
5	34.3	22.3	100.0	24.28
6	28.6	14.7	100.0	17.77

Table 1: Predicted vs. Observed Matches (A-DNA, simplified representation, window size 6).

3.3 Model generation: LLLAMA

3.3.1 Motivation

As described above, to maximize the *a posteriori* probability $\Pr(s|m)\Pr(m)$ for a test sequence of nucleotides $s = (s_1, s_2, ...s_n)$ and model m, we must estimate $\Pr(s|m)$. A reasonable way to do this is to view the sequence as a time series, and estimate each of the nucleotides incrementally, scanning from 3' to 5': $\Pr(s|m) = \Pr(s_1|m)\Pr(s_2|s_1, m)...\Pr(s_n|s_1, s_2, ..., s_{n-1}, m)$. For example, for GGGATCCC, we would estimate $\Pr(\texttt{GGGATCCC}|m)$ as $\Pr(\texttt{G}|m)\Pr(\texttt{G}|\texttt{G}, m)\Pr(\texttt{G}|\texttt{GG}, m)\Pr(\texttt{A}|\texttt{GGG}, m)\Pr(\texttt{T}|\texttt{GGGA}, m)...\Pr(\texttt{C}|\texttt{GGGATCC}, m)$. Each of the probability estimates for each nucleotide depends on a *context* of "previous" nucleotides. Each of the nucleotide probability estimates can be learned from a training set of sequences in the same class.

The problem is that it is entirely possible that a particular context in the test sequence has never been seen in the training set. In that case, we may either relax our matching criterion, thereby permitting near matches to our context when estimating a nucleotide probability, or we may use a shorter context for matching. We then have the problem of choosing which of several possible estimates to make.

Consider Table 1. Here, randomly chosen 6-nucleotide subsequences of A-DNA are compared to all other 6-nucleotide subsequences of our A-DNA database. If we assume that A-DNA is composed of random sequences of bases of approximately equal probabilities, we expect to see, averaged over the entire A-DNA database, 0.00183 exact matches, 0.0549 one-base mismatches, 0.686 two-base mismatches, and so forth. In fact, we see 0.831 exact matches, and in general there are more near matches than expected. When such near matches exist, they are good predictors of the following base: nearly 77% prediction accuracy for exact matches, 45% for one-base mismatches, compared with an expected 25% for completely random sequences. Unfortunately, exact matches only occur for about 71% of the subsequences. To have exact matches for every sample, we would need to restrict our context to a much smaller size (3) where the predictive accuracy is much lower (28%). In general, longer contexts permit better predictive accuracy but fewer exact or near

matches.

Our objective, therefore, is to combine matches from many different context sizes and many different match distances, placing greater weight with matches which are more likely to have greater predictive accuracy.

3.3.2 Description

In this section we describe our model in formal terms.[3] One may view each row of Table 1 as corresponding to a predictive expert. The prediction of the 2-mismatch expert is formed by examining all past matches to our trailing context window with exactly 2 mismatches, and capturing the distribution of the following character by maintaining a simple table of counters. The simplest way to combine these experts is by a fixed set of weights that sum to one.

But suppose that while trying to predict a particular character position with context size $w = 7$, our past experience includes no perfect matches (*i.e.*, no 0-mismatches, or matches of Hamming distance 0), and no 1-mismatches, or matches of Hamming distance 1. For example, if we are trying to predict the character after `GGGATCC`, the past window in the training set with the fewest mismatches is distance 2: `GGGGCCC` from sequence `GGGCCCC`. In particular it makes no sense to give any weight to the opinions of the experts for Hamming distances 0 or 1 – in fact their opinion is not even well-defined in this case. So only the 6 experts corresponding to Hamming distance $2-7$ are relevant. In what follows, we will refer to this value as *first Hamming*. Finally, since we don't know *a priori* how window size will influence the prediction, our model is formed at the uppermost level by a mixture of models, each considering a fixed window size from some fixed prior set.

We denote by b the discrete random variable representing the character of the test sequence to be predicted. By w we denote the positive integer random variable corresponding to the length of our trailing context window; it ranges from 1 to the length of the longest sequence in the training set, L. Next, f denotes the first Hamming distance to be considered. It may assume values $0, \ldots, min(w, h_{max})$, where h_{max} is an external parameter not set by the LLLAMA algorithm, which may assume values $0, \ldots, L$. By h we denote the Hamming distance associated with each expert, so h lies in the range $0, \ldots, min(w, h_{max})$. Finally, we use *past* to represent our modeling past, *i.e.* the DNA that we have already predicted, or have set aside as reference information. Therefore, to estimate $\Pr(s_n | s_{n-1}, s_{n-2}, ..., s_1, m)$, we will estimate $\Pr(b|past)$ for $b = s_n$ and $past = (s_{n-1}, s_{n-2}, ..., s_1, m)$.

Given a fixed window size $w = k$, and distance $h = i$, there is a natural prediction $\Pr(b|h = i, w = k, past)$ formed by locating all distance i matches in any training sequence to the trailing context of length k, and then using the distribution of characters that follow them. This is a single *expert* as described above. This prediction is independent of f so $\Pr(b|h = i, f = j, w = k, past) = \Pr(b|h = i, w = k, past)$ for all legal

[3] See [11] for a more complete treatment.

values of j. Our prediction $\Pr(b|past)$ arises from the joint probability $\Pr(b, h, f, w|past)$ by summing over the hidden variables h, f, w as follows:

$$\Pr(b|past) = \sum_{i,j,k} \Pr(b|h = i, f = j, w = k, past) \cdot \Pr(h = i, f = j, w = k, past)$$

The final term is then expressed as a product of conditionals:

$$\Pr(h = i, f = j, w = k, past) =$$
$$\Pr(h = i|f = j, w = k, past) \cdot \Pr(f = j|w = k, past) \cdot \Pr(w = k|past) \cdot \Pr(past)$$

In this expression $\Pr(past) = 1$ and $\Pr(f = j|w = k, past) = 1$ for j equal to the distance of the closest match to our trailing context window of length k, in the *past*. At all other values $f = 0$. That is, f is a Boolean selector function $f(j, k, past)$. We assume w is independent of the past in our model, and so $\Pr(w = k)$ consists of a fixed vector of L mixing coefficients that select a window size. Finally, in our model, h is also independent of the past, and so $\Pr(h = i|f = j, w = k)$ consists of a fixed vector of $k - j + 1$ mixing coefficients that select a Hamming distance given the earlier choice of a window size k, and observation that the nearest past match is at distance j. We then have:

$$\Pr(b|past) = \sum_{i,j,k} \Pr(b|h = i, w = k, past) \cdot \Pr(h = i|f = j, w = k) \cdot f(j, k, past) \cdot \Pr(w = k)$$

The first term in the summation is recognized as a single expert, the second selects an expert based on f and w, the third deterministically selects a single f value, which receives probability 1, and the final term selects a window size.

The learning task before us is to estimate the parameters $\Pr(h = i|f = j, w = k)$ and $\Pr(w = k)$ by examining the training set T. Our algorithm is an application of the Baum-Welch algorithm for Hidden Markov Models [1], and may also be viewed as an instance of Expectation Maximization (EM), a later rediscovery [5] of essentially the same algorithm and underlying information theoretic inequality. This method iteratively updates the above parameters, with the guaranteed result that the probability of the training set $\prod_{b \in T} \Pr(b|past_b)$ increases or remains the same with each iteration. The number of iterations used by LLLAMA is an external parameter, n_{iter}.

4 Experimental Results

The key result is reported in Table 2. More detailed results are reported in Figures 7 and 8. For each task, representation, and data set the accuracy of LLLAMA-length is listed. LLLAMA-length can be trained to perform each of the three tasks well, with accuracy far greater than chance. Specifically, the best accuracy achieved on the helical conformational classification task was 96.4%; on the crystal type task, 82.1%; and on the packing motif task, 89.1%.

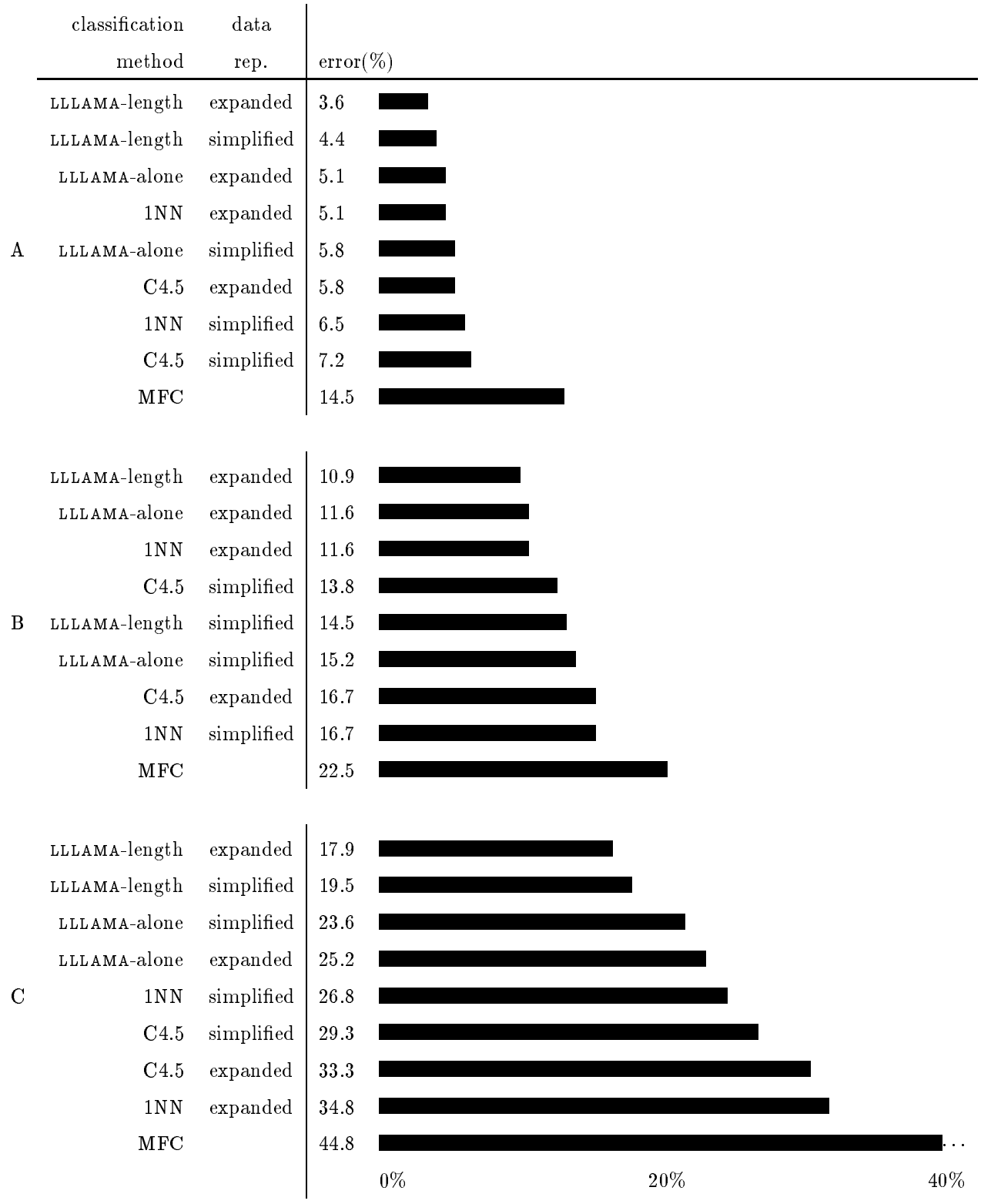

Figure 7: Error rates on the full data set, (A) helical conformation task, (B) packing motif task, (C) crystal type task. LLLAMA-length and LLLAMA-alone are new methods proposed in this paper. C4.5 is a standard machine learning algorithm. 1NN is 1-nearest-neighbor, also a standard machine learning algorithm. MFC is most-frequent-class, the error rate associated with choosing the most frequent class of all sequences in the training set with the same length as the test sequence. *Expanded* and *simplified* are alternative data representations.

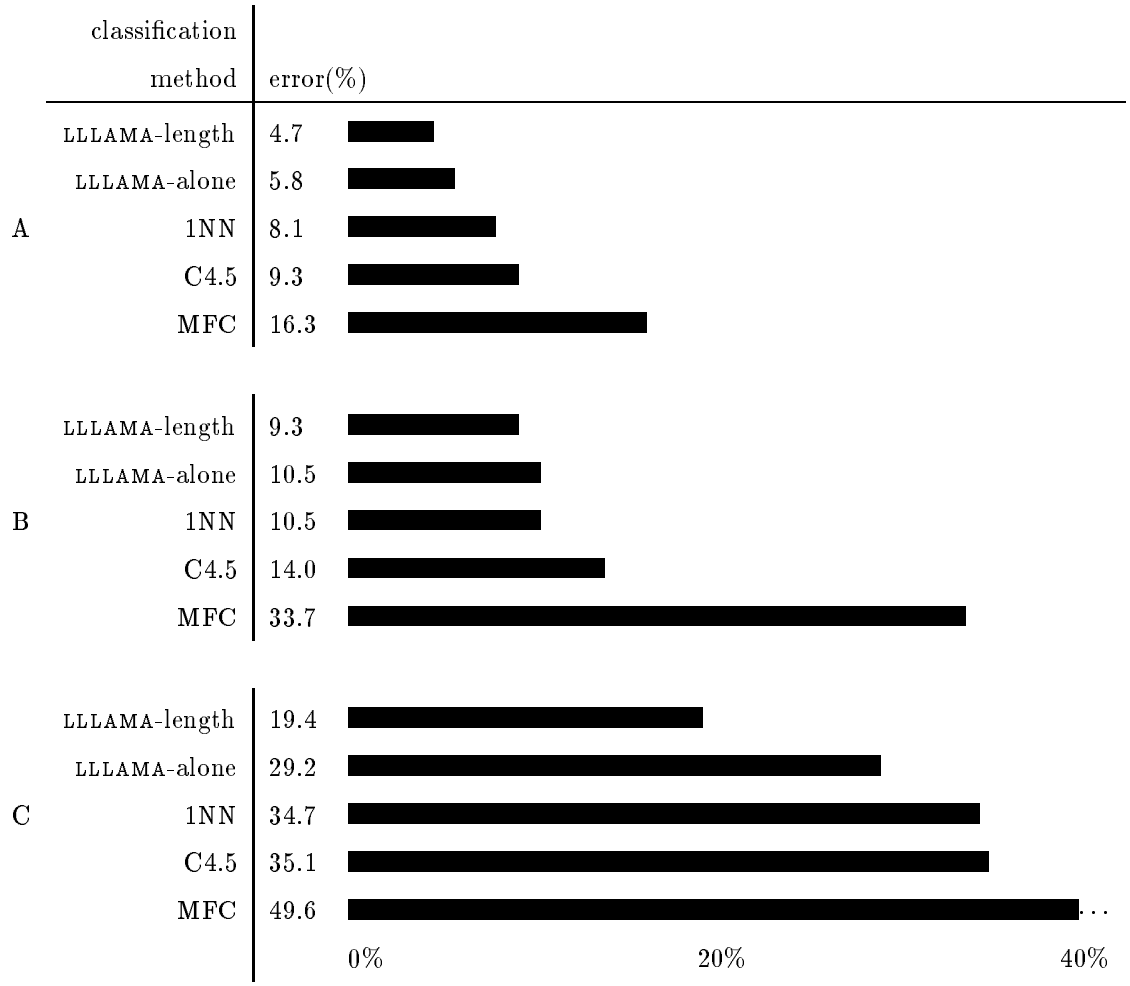

Figure 8: Error rates on the pruned data set, (A) helical conformation task, (B) packing motif task, (C) crystal type task. LLLAMA-length and LLLAMA-alone are new methods proposed in this paper. C4.5 is a standard machine learning algorithm. 1NN is 1-nearest-neighbor, also a standard machine learning algorithm. MFC is most-frequent-class, the error rate associated with choosing the most frequent class of all sequences in the training set with the same length as the test sequence.

data set and representation	task	accuracy (%)
full, expanded	conformation	96.4
	motif	89.1
	crystal	82.1
full, simplified	conformation	95.6
	motif	85.5
	crystal	80.5
pruned	conformation	95.3
	motif	90.7
	crystal	80.6

Table 2: Accuracy of LLLAMA-length on predicting helical conformational class, packing motif type, and crystal type, for different training data sets and representations.

The performance of LLLAMA-alone and LLLAMA-length are shown graphically in Figures 7 and 8. To provide a point of reference, we compare our model directly against the methods described in Section 3.1. Each method and representation is shown in order of increasing error rate, as calculated by the leaving-one-out method [18].

In these figures, k-nearest-neighbor is only reported for $k = 1$, and labeled "1NN". To make the bar graphs readable, the bars representing the error for the most frequent class method were sometimes clipped.

Figure 7 depicts the error rates of the various methods on the conformation, motif, and crystal type classification tasks on the full data set. Both data representations are shown for each method for this data set. Figure 8 depicts the error rates of the methods on the corresponding tasks on the pruned data set. There is only one data representation for this data set.

There were several possible ways to address the issue of tuning LLLAMA's external parameters, h_{max} and n_{iter}. The ideal method would have been to divide the data set into separate training, parameter tuning, and test sets, but this was not feasible with such a small data set. Nor did the data set size support a cross-validation suite in which the training partition was subdivided into training and parameter tuning sections. We decided instead to estimate lower and upper bounds on error on unseen data by tuning the parameters on the test task itself and on a related task respectively. That is, to estimate an upper bound for the error on a given "test" task, we chose values for h_{max} and n_{iter} which minimized the error of the same method on either of the other two tasks. To estimate a lower bound, we chose values which minimized error on the test task itself. With sufficient data, we believe this lower bound would converge to the actual error. Figures 7 and 8 report these lower bounds. Figure 9 reports both the lower and upper bounds for LLLAMA-length for all tasks, data sets, and data representations.

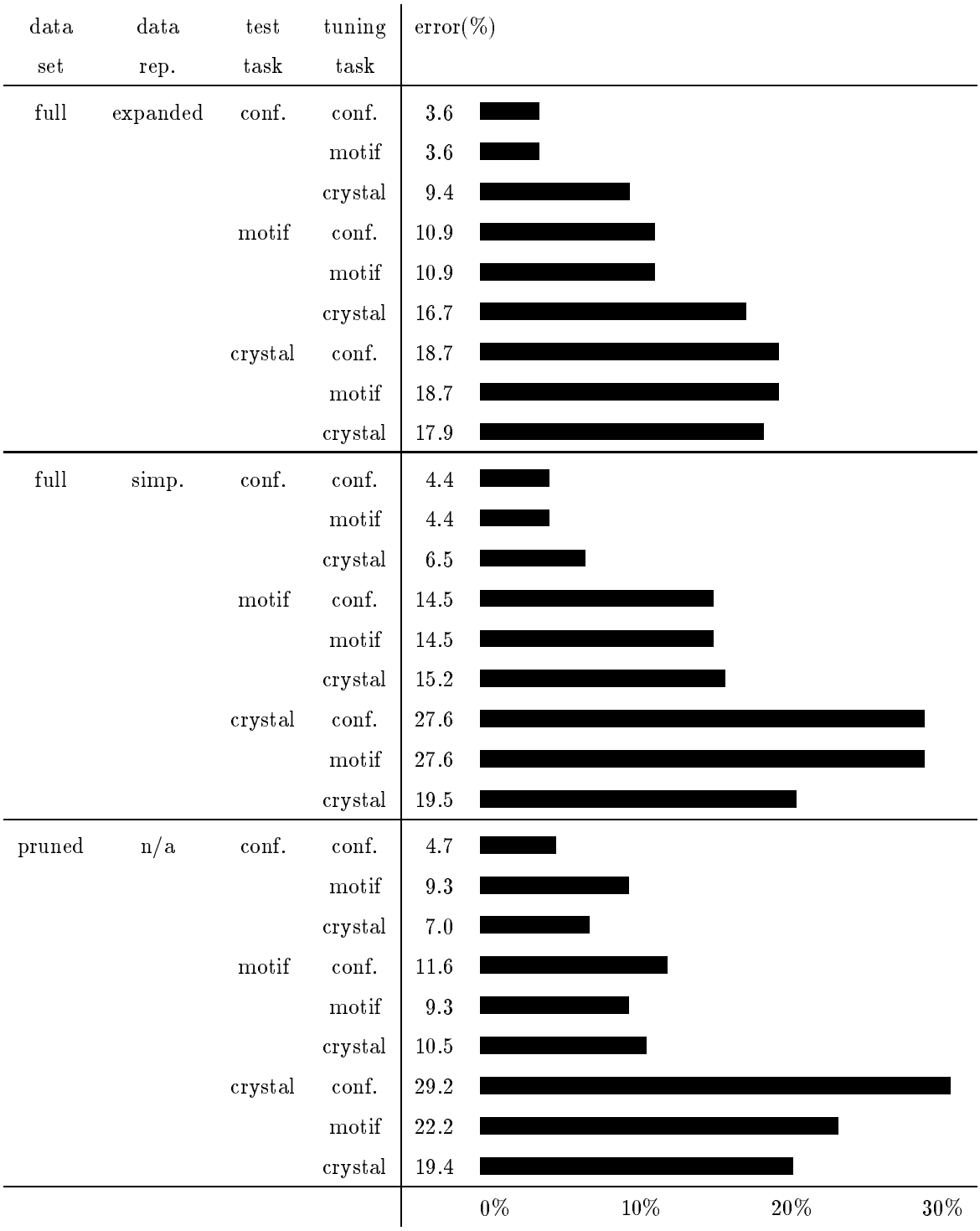

Figure 9: LLAMA-length classification error on all testing tasks and data representations using different tuning tasks.

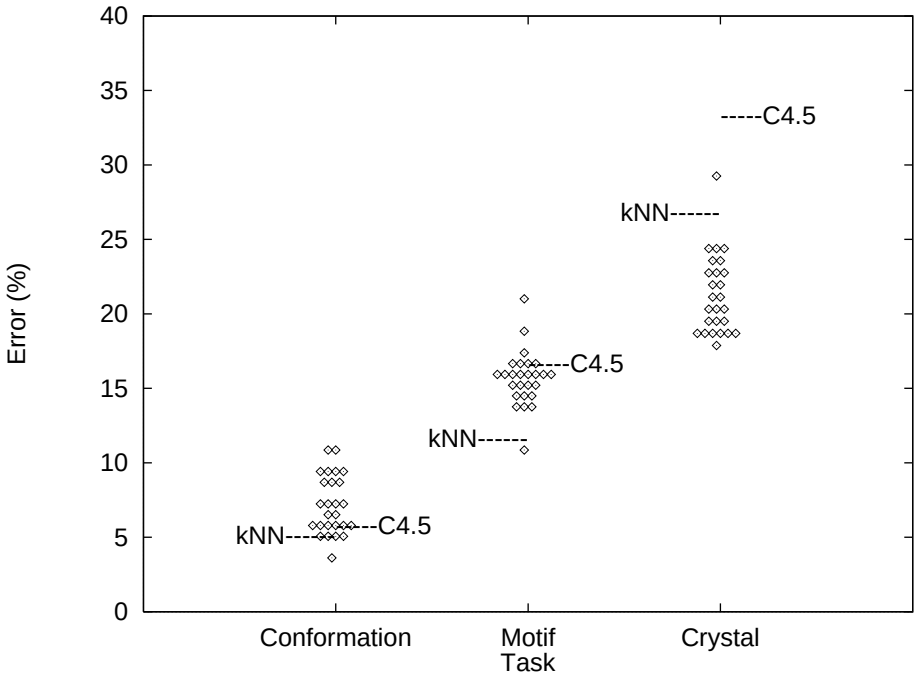

Figure 10: LLLAMA-length classification error (%) (expanded representation, full data set). Each dot represents the classification error of LLLAMA-length using a single choice of external parameters. C4.5 and kNN are standard machine learning techniques provided as benchmarks for measuring the performance of LLLAMA-length.

A reasonable way to use LLLAMA on a small data set would be to tune it on a set of related problems, and choose the parameters which minimized error. That is, since we do not have enough data to form a tuning set, we can use a different problem as the tuning set. Figure 9 shows this method of tuning. Error rates on each of the tuning tasks, data sets, and representations are shown for LLLAMA-length. For each test task, we report the error rate on test data for tuning on each of the three tasks. It is clear that there is always at least one task which generates relatively low error rates when used to tune the external parameters, other than the test task itself.

For k-nearest-neighbor, we chose the default value $k = 1$ for the number of neighbors to compare before making a classification decision. In fact, classification accuracy for k-nearest-neighbor using edit distance fell monotonically with increasing values of k in the range tested ($\{1, 3, 5, 7\}$) for all tasks, representations, and data sets. For similar reasons, C4.5 is reported only for unpruned trees using the default parameters.

LLLAMA-length outperforms all other methods for all tasks, representations, and data sets for at least two of the three possible choices of tuning sets. In fact, it turns out that tuning the parameters on the motif task is a very good way to find good parameters for the conformation task and *vice versa*. The k-nearest-neighbor method performs comparably to LLLAMA-alone in most cases.

Figure 10 is a scatter plot showing the error rates associated with all external parameter values tested on LLLAMA-length, using the expanded data representation and full data set. This plot is representative of similar scatter plots for all LLLAMA methods, data representations, and data sets. From this plot one can see that the spread of LLLAMA's performance is not very great, and that for some problems (*e.g.*, the crystal task) it outperforms other methods for many different parameter settings.

5 Related Work

Several papers have explored entropy estimation techniques to estimate the probability that a sequence of nucleotides could have been generated from a model. In general, these methods do not permit the use of mismatches in either the building or application of the models, which is one of the key ideas presented in this paper. [13, 12] use such methods to compare the probability that the sequence was generated from the model to a null hypothesis that the sequence was randomly generated from a flat distribution. [6] estimates the entropy near splice junctions in eukaryotic genes, and proposes using this method to detect splice junctions.

Several methods apply Bayesian techniques and Expectation Maximization to the problem of modeling DNA sequences, but do so in a task-specific way which does not permit broader application of the methods. [9], for example, is not intended to be a general modeler, but a specific model for a specific task, finding genes in *E. coli*. [4] applies Expectation Maximization to a Markov Chain modeling *E. coli* promoters to find the most likely promoter starting position in a training set.

[7] is a method for modeling DNA sequences, in particular prokaryotic promoters, by scanning for mismatches in a manner somewhat like LLLAMA. Unlike LLLAMA, the method the authors describe does not generate a Bayesian model for predicting the next character, but a score for detecting near-repeats. This score is used to find regions of similarity among different sequences.

6 Discussion and Future Work

There are several conclusions which the work presented in this paper appear to support. The first conclusion, perhaps most interesting from a biological perspective, is that DNA helical conformation, packing motif, and crystal type can be predicted from sequence information alone, at least for short sequences. No additional biological information was encoded into the representations used. Our results our especially promising using LLLAMA-length, but even standard machine learning methods such as C4.5 and k-nearest-neighbor perform credibly well. The 96.4% classification accuracy on the helical conformation prediction task is especially noteworthy. It should be clear to the biologist that oligonucleotide length is generally a useful feature for classification but not sufficient for classification by itself, as seen with the MFC method.

The entropy-estimation/MAP methodology used by LLLAMA-length and LLLAMA-alone provide broad applicability that would be difficult to capture using other methods. If new crystal type or packing motif

classes are added to the NDB, their models may be learned and added to LLLAMA-length or LLLAMA-alone without requiring retraining of the models for the other classes.

In addition, the LLLAMA models may be used directly, without retraining, for more than classification. For example, to find the degree of local A-helical conformational propensity (or propensity for any other class) of each nucleotide of a long sequence such as a gene or chromosome (see [10], Figure 11 for an example). LLLAMA has been applied to large biological sequence problems as well, such as comparing coding and non-coding regions in an entire chromosome [11]. Since there is nothing in the LLLAMA algorithm or the LLLAMA-alone classification method which requires it to be applied to nucleotides, they may be expected to be applicable in many situations in which modeling or classification of sequences is desired, such as protein secondary structure prediction.

One surprising ramification of our work concerns the identification of the principal factors determining the helical conformation of oligonucleotides. Common belief and intuition is that the length of an oligonucleotide and the specific environmental conditions of its crystallization play an important role in determining its conformation. However, our results indicate that the nucleotide sequence itself is every bit as important — if not more important — in determining a compound's conformation. From sequence alone (without explicitly taking into account either oligonucleotide length or environmental conditions) LLLAMA-alone predicts helical conformation with 94.9% accuracy. Even the standard classification techniques C4.5 and k-nearest-neighbor, which were used here as baselines, also predict helical conformation well without explicitly considering length or environmental conditions. In contrast, MFC, which does take length into account but does not consider nucleotide sequence or environmental conditions, performs more poorly than the other methods presented, with a prediction accuracy of only 85.5%. Similar observations can be made concerning the importance of sequence in determining packing motifs and crystal types. Although they are less well-understood and thus the factors that impact upon them still not fully clear, our results indicate that nucleotide sequence is at least as important as environmental conditions and length for predicting these two finer-grained structural characteristics as well.

While our results indicate that nucleotide sequence is a key determinant of the structural characteristics examined in this paper, we cannot argue that oligonucleotide length or environmental considerations play no role. In the first place, LLLAMA-length, which takes into account both length and sequence, always outperforms LLLAMA-alone on the same data, indicating a role for oligonucleotide length in determining structural characteristics. Second, although some of our learning methods, including LLLAMA-alone, C4.5, and k-nearest-neighbor, do not *explicitly* examine the effect of length on predicting structural characteristics, they are all *implicitly* affected by length. LLLAMA-alone's model is affected by length in that it builds a mixture of models for different context window sizes. To determine the structure of a new oligonucleotide all models that are based on context windows that exceed the oligonucleotide's length are discarded, resulting in an implicit, if diffuse, effect of length on predictions. A different argument applies to C4.5, but to the same effect: the representation used with C4.5 uses special markers for positions that do not exist

for short sequences, and since the resulting tree can test for the presence of these markers it is able to indirectly take length into account. Finally, the k-nearest-neighbor method, by using the Smith-Waterman edit distance function, has a bias towards matching oligonucleotides of similar lengths. In summary, we cannot rule out length as a contributing, if perhaps more diffuse, factor in oligonucleotide structure, and further experiments would be necessary to ascertain what role it plays in determining a compound's structural characteristics. Similarly, further experiments would be necessary to assess the contribution of environment on an oligonucleotide's structure.

To address the issue of oligonucleotide length directly, we need to consider applying our classification techniques to polynucleotides. This leads us to consider the admittedly harder task of predicting regions of different structural character in naturally occuring polynucleotides, as for example genomic DNA. This task would be worth studying for its own biological significance, apart from its value in addressing the length issue. As an example of how the methods described in this paper could be brought to bear on this task, LLLAMA-alone could be applied to a sliding window acting upon a large nucleotide sequence, such as a chromosome, to find regions of different helical conformation.

From a computer science perspective, it is noteworthy that LLLAMA displayed significant advantages over more standard methods. In all cases, the best classification method was LLLAMA-length. In a domain in which data is relatively plentiful, building a set of LLLAMA-based classifiers with different parameters and choosing the one which performs best on a separate tuning set may be expected to classify new data even better than the other methods presented here.

The primary goal behind the creation of LLLAMA was to build a method that works reasonably well across a large range of biological sequence problems with minimal background knowledge. The use of multiple representations and data pruning allowed us to examine the effect of adding additional constraints imposed by our knowledge about the underlying biology of the problem. We found that adding this additional knowledge was straightforward, in the form of new data representations. It will be interesting to assess the effect of still more biological knowledge, such as preferred conformational angles or local polarity.

In future work, it will also be worthwhile to examine alternatives to the length-partitioning used in LLLAMA-length. This method worked well on our data, but it does discard a great deal of training data. The methodology used in LLLAMA-length and MFC was applied to C4.5, for instance, but the results were substantially worse across the board than for C4.5 alone. A better method might be to incorporate length partitioning as another partition within the LLLAMA model, much as first Hamming distance is now handled. This would allow the method to examine mixtures of length-partitioned and length-ignoring models.

Nearest neighbor classification using Smith-Waterman edit distance worked surprisingly well. It also shares with LLLAMA the advantages of extensibility to long sequences and ability to recognize subsequence features. It also has the advantage of much shorter training time. Extending the method by tuning the Smith-Waterman matching parameters (for example, see [16]) could yield significant improvements in classification accuracy with only minimal cost in training time.

References

[1] L. E. Baum and J. E. Eagon, *An inequality with application to statistical estimation for probabalistic functions of a Markov process and to models for ecology*, Bull. AMS, 73 (1967), pp. 360–363.

[2] H. M. Berman, A. Gelbin, and J. Westbrook, *Nucleic acid crystallography: A view from the nucleic acid database*, Prog. Biophys. Mol. Biol., (1997). In Press.

[3] H. M. Berman, W. K. Olson, D. L. Beveridge, J. Westbrook, A. Gelbin, T. Demeny, S.-H. Hsieh, A. R. Srinivasan, and B. Schneider, *The nucleic acid database – a comprehensive relational database of three-dimensional structures of nucleic acids*, Biophys. J., 63 (1992), pp. 751–759.

[4] L. Cardon and G. Stormo, *Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments*, Journal of Molecular Biology, 223 (1992), pp. 159–170.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum-likelihood from incomplete data via the EM algorithm*, J. Royal Statistical Society Ser. B (methodological), 39 (1977), pp. 1–38.

[6] M. Farach, M. Noordewier, S. Savari, L. Shepp, A. Wyner, and J. Ziv, *On the entropy of DNA: Algorithms and measurements based on memory and rapid convergence*, in Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 1994.

[7] D. Galas, M. Eggert, and M. Waterman, *Rigorous pattern-recognition methods for DNA sequences*, Journal of Molecular Biology, (1985), pp. 117–128.

[8] H. Hirsh and M. O. Noordewier, *Using background knowledge to improve inductive learning of DNA sequences*, IEEE Expert, (1994).

[9] A. Krogh, I. Mian, and D. Haussler, *A hidden Markov model that finds genes in Escheria coli DNA*, Nucleic Acids Research supplement, 22 (1994), pp. 4768–4778.

[10] D. M. Loewenstern and P. N. Yianilos, *Significantly lower entropy estimates for natural DNA seqeuences*, Tech. Rep. 96-51, DIMACS, 1996. Extended version of [11].

[11] ——, *Significantly lower entropy estimates for natural DNA seqeuences*, in Proceedings of the Data Compression Conference, March 1997.

[12] A. Milosavljević, *Discovering sequence similarity by the algorithmic significance method*, in International Conference on Intelligent Systems for Molecular Biology, 1993, pp. 284–291.

[13] ——, *Sequence comparisons via algorithmic mutual information*, in International Conference on Intelligent Systems for Molecular Biology, 1994.

[14] O. Nash, *The Lama*, in Selected Poetry of Ogden Nash, Little, Brown, 1995, p. 310.

[15] J. R. Quinlan, *Induction of decision trees*, Machine Learning, 1 (1986), pp. 81–106.

[16] E. S. Ristad and P. N. Yianilos, *Learning string edit distance*, Tech. Rep. TR-532-96, Princeton University, 1996.

[17] T. F. Smith and M. Waterman, *Identification of common molecular subsequences*, Journal of Molecular Biology, 147 (1981), pp. 195–197.

[18] S. Weiss and C. Kulikowski, *Computer Systems that Learn*, Morgan Kaufmann, Palo Alto, CA, 1991.

[19] J. Westbrook, T. Demeny, and S.-H. Hsieh, *NDBQuery, v4.0, A simplified user interface to the Nucleic Acid Database*, Tech. Rep. NDB99, Rutgers University, 1996.