Representation is Everything: Towards Efficient and Adaptable Similarity Measures for Biological Data

Charu C. Aggarwal
IBM T. J. Watson Research Center
charu@us.ibm.com

Abstract

Distance function computation is an important operation in biological data mining. This is because the notion of similarity is often required as a key subroutine in many data mining algorithms such as clustering, search or indexing. Furthermore, since such applications typically require repeated distance function computation, it is desirable to perform this task as efficiently as possible. A wide variety of alignment functions have been designed in the literature in order to determine similarity between biological strings. Typically, most of these methods such as the edit distance or the Smith-Waterman algorithm rely on either a local or global form of *alignment* between the two string representations of the biological data. Yet, many of these methods suffer from a variety of drawbacks both in terms of effectiveness and efficiency in measuring similarity. In this paper, we examine the key qualitative issues in measurement of similarity among distance functions, and question the fundamental assumptions in using alignment measures over string representations of the biological data. We argue that the alignment approach is rigid and brittle, ignores several relevant compositional characteristics of the strings, is not easily trainable from examples, and is inherently resistant to an efficient data mining process from a representational point of view. We propose a fundamental overhaul in the methods for similarity computation of biological data by changing the underlying representation of the data. Our proposed methodology (PROSAC) uses PRObabilistic SAmpling for similarity Computations, and results in a new representation of the strings at a higher level of abstraction; yet the abstraction process has the effect of removing the noise in the distance measurements of precise alignment techniques. In addition, some forms of string representation in the PROSAC approach map to well known representational and distance computation methods in the data mining field, and therefore existing data mining software can be used directly and efficiently on the new representation. We also show that the approach is amenable to the use of supervision for parametric determination of efficient distance functions for particular tasks in biological analysis.

Keywords: Biological data, distance functions

1 Introduction

One of the important and frequently encountered procedures in the mining of biological string data is that of distance function design. This is because distance function design remains one of the most important subroutines used in a variety of data mining applications. In many cases, the final quality of the mining results are dependent on the use of an effective distance function to represent the distances between the different records. For example, in a computational biology application, many classifiers are dependent on the use of distance functions in order to compute the distances between records. Similarly, many classical clustering algorithms such as k-means are inherently distance based, and may require millions of distance computations in order to create the segmentation of the data.

Biological data is often abstracted out into strings composed of four bases $\{A, C, T, G\}$. In some cases, the string representation may use the 20 amino-acids, which can themselves be expressed in terms of the four bases. Most known methods for measuring similarity in biological data rely on the use of alignment measures for similarity. Some examples of such measures are the Needleman-Wunsch or the Smith-Waterman Algorithm [8, 11], and their faster variants such as BLAST or FASTA [2, 9]. These methods try to construct an alignment of the strings by using global methods as in [2, 8] or by local methods such as [9, 11]. In global methods such as the Needleman-Wunch algorithm, we attempt to construct an alignment, so that the largest number of positions in the strings are matched. In the case of local methods such as the Smith-Waterman algorithm, we try to find two local regions in the strings

Data Set	Composition
HS	A 0.073 C 0.022 D 0.049 E 0.068 F 0.037 G 0.066 H 0.025 I 0.045 K 0.055 L 0.100 M 0.022 N 0.036 P 0.059 Q 0.045 R 0.055 S 0.078 T 0.054 V 0.064 W 0.014 Y 0.029
YST2	A 0.062 C 0.01 D 0.058 E 0.063 F 0.046 G 0.056 H 0.022 I 0.064 K 0.066 L 0.094 M 0.021 N 0.055 P 0.046 Q 0.038 R 0.043 S 0.088 T 0.059 V 0.061 W 0.012 Y 0.035

Table 1: Compositional Behavior of Two Data Sets

which are matched optimally.

The use of alignment functions for measuring similarity in biological data finds its basis in the fact that closely related biological entities often evolve from one another by a sequence of mutations or other changes in the underlying sequence structure of the data. While this may be true for very closely related strings, such measures become increasingly noise prone, when the strings are more distantly related. In more distantly related entities, the similarity may be defined by a variety of factors such as both the *sequence* and *compositional behavior* of the string. For example, two proteins with very different sequence order but with similar composition (in terms of the underlying amino acids) can often show similar behavior in a number of characteristics. Most alignment-based similarity functions do not take such characteristics into account. For this reason, a number of similarity functions have been developed which utilize the *compositional behavior* of the strings [7]. However, these methods do not use the sequencing of the string structures for computation of similarity. Ideally, it is desirable to design a similarity function in which both the sequencing and compositional behavior are incorporated directly or indirectly.

We note that the user of a string similarity function of a data mining application may have a variety of criteria in mind for the particular application at hand. The final quality of the results are dependent upon these particular criteria which aredefined by the end-user. Therefore, it is important to provide the ability to *supervise* the final creation of the similarity function. In addition, the distance function may be highly dependent upon the aggregate statistics of a particular data set. In order to understand this point better, let us consider the following examples:

- Let us consider the pair of data sets denoted by HS and YST2 which are described in better detail in the experimental section. These data sets were collected from the SWISS-PROT database and correspond to proteins drawn from homo-sapiens and yeast respectively. Each of the data sets contains 1000 sequences of amino acids. The details of the data sets and the collection process are discussed in detail in the empirical section. In Table 1, we have illustrated the relative compositions of the different amino acids in each of the two data sets. We note that the compositional characteristics of the two data sets are quite different. We further note that the similarity between two sequences for a given data set depends upon the *normalized* presence of the amino acids relative to the *entire* data set. For example, for the case of Table 1, if two sequences contain a very high level of the amino acid A, this should be treated differently from the situation in which the two sequences have a high level of presence of the amino acid W. This is because the former amino-acid is plentiful in both sequences, and a high presence in both is not very surprising from a statistical point of view. On the other hand, the latter amino-acid is rare in the underlying data set and a high level of presence in both sequences implies a greater level of similarity. Therefore, it is important to design similarity methods which take the *aggregate statistical behavior* of the *underlying data* into account. Since this aggregate statistical behavior could vary with the data set, the corresponding similarity function should vary as well. We note that alignment methods are typically not sensitive to the use of a *particular data set*. For example, the Smith Waterman algorithm uses the BLOSUM62 matrix to encode fixed domain specific knowledge to model intra-amino acid similarity. However, the importance of different amino acids between different sequences may vary with the data set, since the *contrasts* in distance calculations depend upon the overall statistical distributions of the amino-acids for that particular data set. In many domains such as text [10], such data-specific statistical normalizations have been extensively studied and validated in terms of their contributions to the quality of similarity search.

- We note that the nature of the distance calculations are often deeply dependent upon the compositional behavior of the individual amino-acids in the sequence. For example, in a given application such as classification, the overall classification behavior may depend upon a combination of compositional similarity, and local or global alignment [7]. Therefore, many data mining algorithms have to rely on a *combination* of different ad-hoc methods

for similarity measurement. We note that this is because while the string representation is effective in maintaining *sequencing* information, it is poor in maintaining compositional behavior over different parts of the strings. It is a definite advantage to choose a *representation* for the string data which is such that it encodes both sequencing and compositional information together with their relative importance with the help of a few parameters. Furthermore, the representation should be such that important data mining operations (such as representing the centroid of a set of strings) should be implementable in a form which is similar to the original data. In many cases, this can allow the use of existing data mining algorithms and software with very few modifications.

- One of the drawbacks of alignment approaches is that it requires the use of computations whose complexity increases quadratically with string size. Since biological strings are often extremely long, this results in limitations in applying the approach to computationally intensive applications on very large data sets. Furthermore, since these distance functions are defined *algorithmically* rather than in closed form, it is intuitively more difficult to incorporate user-defined criteria through supervised parametric choices.

One important issue about similarity function design is that the nature of the features which need to be used for the purpose of the data analysis are often dependent upon the particular task at hand. Some tasks may be more amenable to global similarity, others to local similarity, and yet others to compositional similarity. For example, while it may be known that certain amino-acids are more important for a given task, the relative importance of their composition and sequencing may vary with the task at hand. This task may be defined only in terms of the *known similarity* of user-defined examples. This known similarity can then be used in a *supervised setting* in which the relative importance of different amino acids, or the relative importance of the global or local compositional features can be controlled with the use of appropriate parameters. One of the difficulties with sequencing based approaches is that they try to aim for either exact local matches [2, 9, 11] or they aim for a global match [8] which does not preserve local compositions very well. In reality, we are often looking for matches which use a combination of the *sequence* and *composition* of the string. In many cases, the nature of the similarity function may only be defined in a *supervised setting*, in which a combination of the local and global behavior may be used in order to define the similarity function.

In this paper, we will discuss a new approach to similarity computations in string data. The idea here is to use probabilistic sampling at equidistant positions along an aligned string at varying levels of granularity. Each sample point is no longer a deterministic alphabet of the string, but is a probabilistic alphabet depending upon the *local* distributions of the alphabets from that point. We will show that different patterns of sampling (such as equi-depth or equi-width sampling) may be used in order to construct representations which are more prone to global or local similarity in strings of varying lengths. We will also show that by varying the parameters of the sampling process, it is possible to obtain different levels of importance of the global, local or compositional characteristics of the strings. Furthermore, some variants of the approach allow the construction of distance functions in closed form with *parametric decomposability* in terms of the contributions of the local, global as well as the individual amino-acid compositions. This is very useful in a setting in which it is desirable to supervise the construction of the distance function with specific user-defined criteria or user-derived examples. This is required in practical settings where the user may have specific data mining tasks at hand which require different definitions of similarity (eg. cluster segmentation based on structure or function). We do not know of any other similarity function in the biological domain which can encode such problem-specific information. The resulting similarity computation method is referred to as PROSAC (PRObabilistic SAmpling for Similarity Computations) and turns out to be a flexible, effective, and efficient similarity search method in practice.

This paper is organized as follows. In section 2, we will discuss the desiderata for effective distance function design and introduce the PROSAC framework. We will discuss both the PROSAC representation and the similarity function constructed using this representation. In section 4, we discuss the use of supervision for development of similarity functions which are specific to a given task. In section 5, we present the effectiveness and efficiency of the similarity function over a wide variety of data sets. Finally, the conclusions and summary are contained in section 6.

2 The PROSAC Framework for Similarity Computations

In this section, we will discuss the PROSAC approach for similarity computations on string based data. Before discussing details of the similarity computations, we will discuss the desiderata for the design of similarity functions in biological data. The desiderata for the effective design of similarity functions are as follows:

(1) Data Set Specific: The key characteristic of a similarity function is to distinguish between the different records in the data set. For this purpose, the aggregate statistics of the records are useful for distinguishing between the different features. For example, in the text domain, the use of aggregate statistical characteristics of the words are quite common for the measurement of similarity. For example, in the text domain, the cosine coefficient [10] does not treat the attributes uniformly but normalizes uses the inverse document frequency across the particular data set. In many practical data domains such as text, categorical data and time series data [4, 10], the use of *data set-specific* aggregate statistics is considered a natural choice for effective definition of similarity. However, in the case of the biological domain, this is constrained by the computational difficulty of string-based alignment approaches to similarity. In a biological application, the relative presence of the different amino-acids for a *given data set* is useful information in the computation of similarity.

(2) Interpretability, Decomposability and Closed Form Representation: We note that a variety of factors can influence the similarity for a given data set in the biological domain. This can include the global, local or compositional characteristics of the string. It is desirable to construct a *closed form representation* of the similarity function which is both intuitively interpretable from a probabilistic point of view, and for which the relative importance of different factors can be controlled with the use of a few parameters. In this sense, alignment functions such as the Smith-Waterman function or the Needleman-Wunsch method [8, 11] are defined *algorithmically* rather than in closed form. This makes it more difficult to control the relative importance of different factors in the similarity computation process.

(3) Computational Efficiency: In many biological applications, the strings can have thousands of characters over databases containing millions of records. Furthermore, a given data mining application such as clustering may require distance computations which may grow faster than linearly with the number of records. In spite of recent improvements such as FASTA or BLAST [2, 9], the process of computing alignments between long strings continues to be a computationally challenging problem. This is because of the similarity computation methods in strings which utilize dynamic programming methods such as the edit distance. In order to make the approach truly scalable to complex and computationally intensive data mining problems with large data sets, it is desirable to design similarity computation methods which are extremely efficient in practice.

(4) Trainable: We note that the use of a particular similarity function may depend upon the particular task at hand. For example, if a given set of proteins need to be clustered on the basis of function, this is a different situation than one in which the proteins need to be clustered on the basis of homology. In such cases, *training examples* are often available in order to measure similarity. While classification on the basis of particular criteria such as structure or function has been extensively studied using domain-specific criteria, arbitrary applications in data mining may require the explicit design of efficiently implementable similarity *functions* in closed form, which work with pre-defined user-defined criteria. Typically, alignment based functions are more difficult to train than parametric closed form distance functions because of their iterative and algorithmic approach of calculating similarity in the former. This paper will aim to design similarity functions which are amenable to such training.

2.1 The Density Based Sampling Approach

In this section, we will discuss the density-based sampling approach for string similarity search. While our density based approach derives its origin in an alignment of the two strings, it provides a higher level of abstraction, which improves the quality of the similarity function. We assume that the strings are drawn from the alphabet of size l which is defined by $\Sigma = \{\sigma_1 \ldots \sigma_l\}$. In order to explain the density based sampling approach, let us consider two sequences S_1 and S_2 which are currently aligned using a hypothetical alignment based approach $\mathcal{H}$. We note that the alignment approach could either be local or global. In the case of local alignment, only subsequences of the original strings may be aligned. Let us define two aligned sub-segments of S_1 and S_2 by S_1' and S_2' respectively. In the case of global alignment, the string S_1 is the same as S_1', and the string S_2 is the same as S_2'. On the other hand, S_1' and S_2' are smaller (but contiguous) segments in the case of local alignment. Let us assume that the alphabets in the sub-strings S_1' and S_2' are denoted by $a_1 a_2 \ldots a_p$, and $b_1 b_2 \ldots b_q$, where $a_i, b_i \in \Sigma$.

Let us now sample k positions from the aligned strings. We note that each of these aligned positions could correspond to a gap, or it could correspond to one of the alphabets from the two strings. For example, consider the two strings MGSDKERDT and MPASREDT. The optimal alignment for the two strings using the edit distance is shown below:

MG-SDKERDT
MPASR-E-DT

We note that this alignment could be different depend-

ing upon the particular algorithm which is used for alignment purposes. For example, the Smith-Waterman algorithm or the Needleman-Wunsch Algorithm could provide very different alignments. We will try to compute the effectiveness of a particular kind of alignment between two strings using a sampling approach on the positions in the individual strings. The measure of alignment between the two strings depends upon whether a global or local approach is used for the computation. For this purpose, we assume that certain positions in the string are sampled at random. Let us consider the situation in which we have an alignment of length N between S_1' and S_2', and we sample the kth position in the alignment. While the alignment itself has a (k/N)th fraction on the left of it (including itself), the exact alphabet drawn from the two strings may not correspond to the (k/N)th alphabet from the two strings. This is because of the presence of gaps throughout the alignment. Therefore, each sampled position needs to be represented probabilistically in terms of the alphabets in the locality of the sampled position. For this purpose, we use a sampling approach which constructs a probabilistic profile of the sampled position of the alignment. We make the weaker assumption that the alphabets for the (k/N)th position of the alignment are drawn from the locality of the (k/N)th position in the corresponding aligned *substrings* S_1' and S_2'. We note that we are using the aligned *substrings* rather than the original strings in order to account for the fact that the alignment may be local rather than global. First, we assume that the probability distribution of the (k/N)th position is computed using a Gaussian model. For this purpose, we define the *indicator function* $\mathcal{I}(\sigma, m, S)$ for a particular symbol $\sigma \in \Sigma$ and position m of string $S = g_1 \ldots g_r$ as follows:

$$\mathcal{I}(\sigma, m, S) = 0 \text{ if } g_m \neq \sigma$$
$$1 \text{ if } g_m = \sigma$$

For the string S_1', we define the density $f(\sigma, m, S_1')$ for alphabet σ at position $m < p$ as follows:

$$f(\sigma, m, S_1') = \sum_{j=1}^{q} \frac{\mathcal{I}(\sigma, m, S_1')}{\sqrt{2 \cdot \pi} \cdot h} e^{-||(j-m)||^2/h^2} \quad (2.1)$$

Here h is the bandwidth of the density estimation process. We note that the density estimation process computes the relative density of each symbol at the sampled position m using a gaussian kernel smoothing criterion. The parameter h can be used to control the level of smoothing. We note that a higher value of the bandwidth creates a greater level of smoothing for the density estimation process. We note that this approach

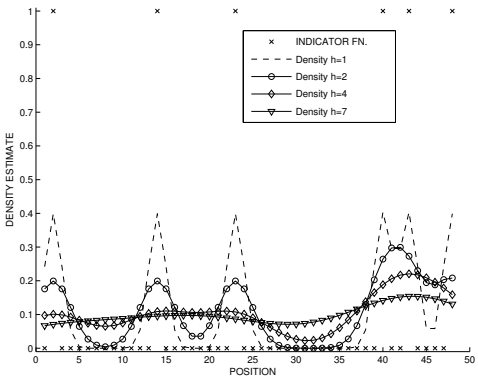

Figure 1: Density Estimation with Indicator Function for different Bandwidths

is essentially a string version of non-parametric density estimation [12] often used for quantitative data sets. Note that the density function for a particular alphabet is essentially a smoothed version of the indicator function. For example, consider the string and the corresponding indicator function for the alphabet A below:

String: MANMQGLVERLERAVSRLESLSAESHR
PPGNCGEVNGVIAGVAPSVEA
I.Fun.: 01000000000001000000001000000000000
0000100100001

In Figure 1, we have illustrated the density estimates for the corresponding string along with the indicator function (for alphabet A). It is clear that different values of h result in a different level of smoothness in the estimation of the density of a given symbol in the string. We also note that the bandwidth h defines a locality in which the compositional behavior of the string is more relevant than the exact sequencing of the alphabets. The exact value of the bandwidth h will be defined by the frequency of sampling along different points in the string. We further note that while this smoothing loses some sequencing information within a locality, it is more noise-free in terms of representing the compositional behavior within the locality. Furthermore, the overall combination of the density characteristics over the different alphabets provides a comprehensive overview of the behavior of that locality. Once the composition for a given locality has been determined, we can use it to construct the sampling based *relative probability estimation* of a given symbol relative to the other symbols. We note that the density estimate $f(\cdot, \cdot, \cdot)$ can be used in order

to construct the relative probability of the different symbols at a given position. The relative probability $p(\sigma, m, S_1')$ of the symbol σ at position m in string S_1' is defined as follows:

$$p(\sigma, m, S_1') = \frac{f(\sigma, m, S_1')}{\sum_{\sigma \in \Sigma} f(\sigma, m, S_1')} \tag{2.2}$$

Thus, the relative probability is computed in terms of the fractional density of a given symbol in relationship with the other symbols. Next, we will discuss how sampling is used in order to construct the probability estimation over different data localities. In this process, we pick certain positions in the string and calculate the density estimates at these sample points. The probability estimates of the different symbols at these sampled points are used in order to construct the similarity function between the two strings. We note that the exact properties of the similarity function depend upon the nature of the underlying sampling policy and bandwidth h. The bandwidth h depends upon the underlying sampling policy as well. The sampling policy is defined in a way so as to generate either a global or local match between the two strings. The two major sampling policies which are defined are as follows:

- **Equi-depth Sampling:** We note that the different strings may have widely varying lengths. In equi-depth sampling, the positions from the string are sampled in such a way that an equal number of positions are sampled from each. Therefore, the distances between individual sample points may vary with the string. The primary parameter for the equi-depth sampling process is the number of positions sampled n_s. The equi-depth sampling process is designed to compute the global match between the two strings.

- **Equi-width Sampling:** In equi-width sampling, we sample positions from the string at equal distances from one another. We note that a different number of positions may be sampled from strings of widely varying lengths. The primary parameter for the equi-width sampling process is the width between successively sampled positions in the strings. This is referred to as the sampling width s_w. The equi-width sampling process is designed to compute a local match between the two strings since it tries to compare the local composition of segments of equal lengths.

We note that the process of density estimation at the sampled positions creates a new representation of the string which encodes the adjacency information among the symbols in an indirect way. This is a key factor in the creation of a similarity function which does not require the alignment method of the string representation. An important factor in the success of the PROSAC approach is that the *representation* of the string is transformed to a new form.

2.2 Choice of bandwidth The choice of bandwidth is dictated by the rate of sampling. In general, the bandwidth should be chosen in such a way that the a the density at a sampled position is affected by all symbols between it and the next sampled position, but the positions at or near the next sampled position do not affect the corresponding density values substantially. Therefore, a natural approach is to choose the bandwidth equal to half the distance between two sampled positions. This choice can be used for both the equi-depth and equi-width sampling methods.

2.3 Hierarchical Enhancements We can improve the algorithms further by using hierarchical enhancements. In these enhancements, we perform the sampling in a hierarchical way in which the granularity of the higher level sample reduces over the lower level representation by a factor of 2. The repeated process of hierarchical reduction of granularity can be used in conjunction with both the global and local approach. Thus, in the case of equi-depth sampling, the number of positions sampled are reduced by a factor of 2. In the case of equi-width sampling, the width of the positions sampled is increased by a factor of 2.

The sampling process is used to construct new representations of the strings which reflects the relative presence of the different symbols in the localities of the sampled positions. We note that the construction of the new string representations is a pre-processing phase which is completed at the very beginning of the algorithm. The new sampled representations are used in order to design the corresponding matching algorithms (global algorithms for the equi-depth case, and local algorithms for the equi-width case). We will discuss these algorithms in the next section. In order to facilitate the discussion of these algorithms, we will introduce some additional notations and definitions. For the global case, since the number of positions sampled are n_s, and there are a total of l symbols, the number of probability values stored is equal to $l * n_s$. We assume that the probability of the ith sampled position taking on the symbol $\sigma_j \in \Sigma$ in string S_1' is denoted by $P(i, j, S_1)$. The notation for the local case is exactly similar, except that in this case, the number of sampled positions will depend upon the length of the string (and the sampling width).

3 Matching Algorithms

In this section, we will discuss the matching algorithms which are used for the similarity computations. We will discuss the matching algorithms for both the global and local case.

3.1 Global Matching Algorithm The global matching algorithm is extremely straightforward because of the nature of the equi-depth sampling. We note that the probability of the two ith sampled positions in the strings S_1 and S_2 containing the symbol σ_j is given by $P(i, j, S_1) * P(i, j, S_2)$. Therefore, the probability of a match at the ith sampled position is given by $\sum_{j=1}^{l} P(i, j) * P(i, j)$. Therefore, the expected number of matches $M(S_1', S_2')$ over all the n_s positions is given by the following:

$$\text{Exp. Num. of Matches} = E[M(S_1, S_2)] = \sum_{i=1}^{n_s} \sum_{j=1}^{l} P(i, j, S_1) * P(i, j, S_2)$$

As mentioned in an earlier section, the similarity function should be tailored to the aggregate characteristics of the data set. While, the above expression may provide the expected number of matches, it does take into account the fact that a match on a rare amino-acid may be more indicative of similarity than a match on a more frequent amino-acid. A typical example of such a normalization is often done in the Information-Retrieval domain in which we use the inverse-document frequency to weight the importance of a given word. Therefore, we associate the normalization factor s_j with the amino-acid j, which indicates the importance of amino-acid j. The corresponding *normalized* expected number of matches $E_N[M(S_1, S_2)]$ is given by the following expression:

$$E_N[M(S_1, S_2)] = \sum_{i=1}^{n_s} \sum_{j=1}^{l} s_j * P(i, j, S_1') * P(i, j, S_2)$$

We note that the normalization factor may be constructed using a variety of heuristics such as a direct generalization of the idf-normalization used in the text domain. In this case, the value of s_j is chosen as the inverse frequency of the relative presence of the different symbols. In addition, it is also possible to pick the values of the normalization factor s_j in a supervised way. Such supervised techniques for picking the normalization factor can be useful in tailoring the similarity function to a specific application. We note that this is a specific advantage of using this kind of decomposable technique to retain the training ability of the method.

P(1, *, S1) P(2, *, S1) P(3, *, S1) P(4, *, S1) P(5, *, S1) P(6, *, S1)
P(1, *, S2) P(2, *, S2) P(3, *,S2) P(4, *, S2)

P(1, *, S1) P(2, *, S1) P(3, *, S1) P(4, *, S1) P(5, *, S1) P(6, *, S1)
P(1, *, S2) P(2, *, S2) P(3, *,S2) P(4, *, S2)

P(1, *, S1) P(2, *, S1) P(3, *, S1) P(4, *, S1) P(5, *, S1) P(6, *, S1)
P(1, *, S2) P(2, *, S2) P(3, *,S2) P(4, *, S2)

Figure 2: Local Matching Alignment

In the experimental section, we will show the advantages of using such an approach, in which we tailor the distance function based on arbitrary criteria.

In the above description, we have used the dot product as a natural method for similarity computations. However, it is also possible to use other metrics such as the sum of the differences in the density distributions of the two strings over the different symbols. This effectively maps to the (weighted) Manhattan metric. The distance using this Manhattan metric $L_1(S_1, S_2)$ is defined as follows:

$$L_1(S_1, S_2) = \sum_{i=1}^{n_s} \sum_{j=1}^{l} s_j * |P(i, j, S_1') - P(i, j, S_2)|$$

In this case, a lower value of the distance function implies a greater level of similarity.

3.2 Local Matching Algorithm The local matching algorithms are much more difficult because we need to account for the varying number of positions at which the matches could take place between different pairs of strings. Therefore, we need to normalize for the number of positions. Let us assume that the two strings which are used for matching purposes have N_1 and N_2 symbols in them respectively. Then, the number of sampled positions in the two strings S_1 and S_2 are given by $n_1 = [N_1/s_w]$ and $n_2 = [N_2/s_w]$ respectively for a sampling width s_w. Now, we need to deal with the tricky issue of comparing the distributions of the two strings.

In order to compare the distributions of the two strings, we need to use an iterative matching algorithm between the two segments of length n_1 and n_2. Let us assume without loss of generality that $n_1 > n_2$. In order to find the optimal local match, we use an algorithm in which we try to find all possible *contiguous matches* between S_1 and S_2. In a *contiguous match* we do not account for a gap between sampled positions. This is a reasonable assumption, if the gaps between successively sampled positions (and the magnitude of the smoothing bandwidth) is larger than the typical gaps between alignments found by traditional algorithms such as the Smith-Waterman or Needleman-

Wunsch algorithms. The smoothing effect of the density estimation process accounts for the gaps within a corresponding data locality. In Figure 2, we have illustrated an example of the three possible matches between the two strings of lengths 4 and 6 respectively. In general, for two strings of length n_1 and n_2 which satisfy $n_1 > n_2$, we can have $n_1 - n_2 + 1$ possible alignments. However, we first need to define the corresponding matching function between the contiguous segments of the strings. First, we need to define the similarity between a pair of positions in the strings. First, we define the *default* probabilities for the different positions and symbols. Let us assume that the default probabilities of the symbol σ_j at the ith position for string S is denoted by $P^0(i, j, S)$. We note that the default probabilities are defined using the aggregate behavior over the entire data set. Then, for the ith position in string S_1, and kth position in string S_2, the corresponding similarity value denoted by $Sim(i, k, S_1, S_2)$ is defined as follows:

$$Sim(i,k,S_1,S_2) = \sum_{j=1}^{l} [P(i,j,S_1) * P(k,j,S_2)] - P^0(i,j,S_1) * P^0(k,j,S_2)$$

We note that this definition for similarity is quite similar to the global case, except that we are also subtracting out the matching probability using the default rates of presence of the different symbols. Such a computational process ensures that two randomly generated strings from the data-specific composition of symbols will have average matching probability of zero. Furthermore, the contribution of a given position can be positive or negative depending upon its deviation from expected values. This is useful in comparing the relative similarity of different pairs of strings of differing lengths. For each $i \in \{0 \ldots n_1 - n_2\}$, the overall similarity for a given alignment from position i to position $i + n_2$ of string S_1 to the string S_2 is given by the sum of the corresponding similarity values over all positions:

$$Align(i, S_1, S_2) = \sum_{j=0}^{n_2} Sim(j + i, j, S_1, S_2)$$

The local similarity is the maximum similarity over all possible such alignments, and is given by $\max_i Align(i, S_1, S_2)$. The aim of finding the maximum similarity over all possible alignments is to ensure that the best possible local matching is chosen for the purposes of alignment.

We note that a number of variations are possible on this basic approach with the use of density differentials as opposed to the probabilistic estimate. The only difference is in how $Sim(i, k, S_1, S_2)$ is calculated. In this case, we compute this differential as follows:

$$Sim(i,k,S_1,S_2) = \sum_{j=1}^{l} [P(i,j,S_1) - P(k,j,S_2)] - \sum_{j=1}^{l} \sqrt{P^0(i,j,S_1) * (1 - P^0(k,j,S_2))}$$

We note that the second term in the expression represents the standard deviation of a bernoulli distribution using the default probabilities of occurrence of the different symbols. This second term is included in order to normalize between alignments of varying lengths. In this case, smaller values of the function imply greater similarity. Correspondingly, the optimal alignment is computed by minimizing the objective function over all possible alignments. One observation about the local alignment method is that it replicates a number of features of the gap based alignment method by substituting with density-based smoothing. In other words, the smoothing process in density estimation substitutes for the process of finding gaps in subsets of the data streams. As a result, it is able to compute the similarity values in a fairly straightforward way by using contiguous segments. As will be evident from our results in the empirical section, such an approach is extremely efficient and does not lead to a reduction in accuracy.

3.3 Supervision for Distance Functions In many applications, the concept of distances may not be concretely defined, but may vary with the application criteria. One advantage of our approach is that it allows the use of supervision in distance function design at least in the case of global alignments. This is because the distance function can be represented in closed form as illustrated in the aforementioned expressions for $E_N[M(S_1, S_2)]$ and $L_1(S_1, S_2)$. In the previous section, we have discussed that the normalization factor s_j is chosen using the inverse frequency method as in the text domain [10]. While the normalization approach is a natural choice in the absence of additional information about the relationship of the different amino-acids to the similarity values, this may not be the case in particular applications involving protein structure and function. In the supervised case, it is possible to train for the values of the different parameters s_j. We note that in an arbitrary application, the aims of the user may influence the importance of a given symbol, and therefore the value of the parameter s_j. For example, this is useful in a clustering application in which we are processing proteins based on particular kinds of structure or function. In such cases, it may be desirable to use a supervision process (as discussed in [1]) in order to determine the final solution. In this case, it is

Data Set Name	Keywords
YST1	Yeast
YST2	Yeast
MS	Mouse
HS	Homo Sapiens

Table 2: Keywords used in SWISS-PROT database

assumed that we have a *training data set* containing pairs of records together with the corresponding similarity value. We construct a solution by fitting the model parameters $s_1 \ldots s_l$ to these user defined similarity values. A standard least squares error minimization technique can be used for the regression, as discussed in [1]. We note that such a technique is not possible in the case of alignment techniques such as the Smith-Waterman method, or the Needleman-Wunsch method. This is because such methods cannot directly represent the similarity function in closed form in terms of a symbol-specific set of parameters. Furthermore, the computational intensiveness of these methods precludes the development of practical approaches for these methods. We will illustrate a number of advantages of using the supervised approach in the next section.

4 Empirical Results

In this section, we will discuss some experimental results illustrating the effectiveness and efficiency of the similarity search method. We note that the problem of measuring effectiveness of a similarity function is a difficult one, since the qualitative notion of similarity depends upon the application at hand. This has generally been an issue with many classical similarity functions [11, 8], which are designed to encode mutation-based distances (as the primary qualitative notion) but their effectiveness with arbitrary notions of similarity remains unknown. In this paper, we will study the behavior of these objective functions with soft notions of similarity which are generated from the meta-information in the SWISS-PROT database. We use a combination of the taxonomy and the keyword fields in the meta-information to create a textual representation of the salient characteristics of the protein. We note that the combination of the taxonomy and keyword fields create a textual description of the protein which reflects many of its characteristics, but it does not define a "hard" objective function witha pre-defined meaning. This is particularly useful for testing the behavior of different similarity search methods. The cosine similarity between two such textual representations was defined as the similarity between two strings. We note that while the notion of similarity may vary depending upon the application, such a soft definition of similarity is useful for determining how widely known methods such as the Smith-Waterman and Needleman-Wunsch algorithm would perform for arbitrary criteria in a variety of applications. Furthermore, since most alignment based methods are not easily amenable to training, such soft testing methods expose the advantages of supervision which is natural to our approach. For the testing process, we will use three variants of our algorithm. These three variants correspond to the following:

- **PROSAC-G**: This is the global version of the PROSAC algorithm.In each case, the strings were compressed to 32 positions. The smoothing bandwidth was chosen using the relationship described earlier. The actual similarity function was computed using the variation on the Manhattan distance metric.
- **PROSAC-L**: This is the local version of the PROSAC algorithm. In each case, the strings were compressed by a factor of 10 (with rounding). As in the previous case, the similarity function was computed using the variation on the Manhattan distance metric which was discussed in this paper.
- **PROSAC-S:** This is the supervised version of the PROSAC algorithm. In this case, the global version of the algorithm was used in conjunction with training for the weights of the different amino-acids. This approach is particularly useful in tailoring the similarity function to a particular kind of application. In this case, we used a separate data set of 1000 records to perform the training. This data set was not used for testing the similarity function.

We used the Smith-Waterman and Needleman-Wunsch algorithms as baselines in order to evaluate the effectiveness of the PROSAC algorithm. For the case of the Smith-Waterman algorithm, we used a much higher open gap penalty than the gap extension penalty in line with observations made by other researchers [13]. Specifically, we used the BLOSUM62 matrix in conjunction with an open gap penalty of 10 and a gap extension penalty of 0.5 as suggested in [13]. For the case of the Needleman-Wunsch algorithm, we used an open gap penalty of 10.

4.1 Data Sets In order to generate the data sets for testing the algorithm we used repeated queries on the SWISS-PROT database. A number of data sets were generated using the keywords discussed in Table 2. Specifically, we queried the web interface of the

SWISS-PROT database using the above keywords. The corresponding data set names are illustrated in the same table. In each case, we only used the first 1000 entries from the data set to test the effectiveness of the similarity computations. The only exception was the Yeast database in which two sets of 1000 entries were used to create the two data sets YST1 and YST2 respectively. For the case of the supervised approach, we used the last set of 1000 entries for training purposes. The aim of using a disjoint set of training entries was to avoid data set-specific overtraining for the choices of parameters in the similarity function.

4.2 Evaluation Criteria A number of evaluation criteria were used to test the effectiveness of the different similarity functions: (1) The first criterion was the correlation between the true similarity values and the similarity values determined using the PROSAC approach. This correlation value ranged between 0 and 1 and was typically very noisy. Clearly, higher values of the correlation were more desirable. We will present the results for different variations of the PROSAC approach along with the corresponding results from alignment based methods. (2) The second criterion was the efficiency of the approach. In this case, we tested the running time of the approach for different methods. Clearly, in order for an approach to be effective in data mining algorithms which require repeated applications of the similarity function, it is desirable for the computational complexity of the approach to be as low as possible. One disadvantage of alignment based methods such as the Smith-Waterman algorithm or the Needleman-Wunsch algorithm are the extremely high running times because of the dynamic programming methodology in similarity computations. This makes it difficult to use alignment based approaches in computationally intensive subroutines of data mining algorithms.

4.3 Effectiveness Results In Figures 3, 4, 5 and 6, we have illustrated the qualitative behavior of the similarity functions for the data sets YST1, YS2, MS, and HS respectively. As discussed earlier, the qualitative behavior of a similarity function was defined as the correlation between the computed value (from the different similaryity functions) and true value (using our soft keyword based criterion). Since some of the similarity functions are minimization functions and others are maximization functions, we used the modulus of the correlation in each case. It is interesting to see that in each case, most variations of the PROSAC approach perform better than either the Smith-Waterman or the Needleman-Wunsch algorithms. In the case of the HS data set (Figure 6), the relative advantage of

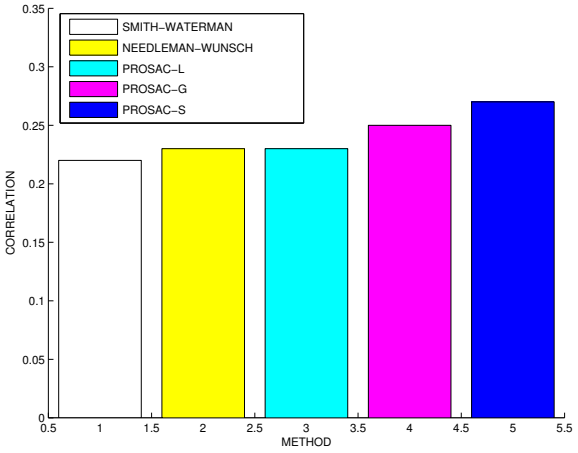

Figure 3: Accuracy of Different Methods (YST1)

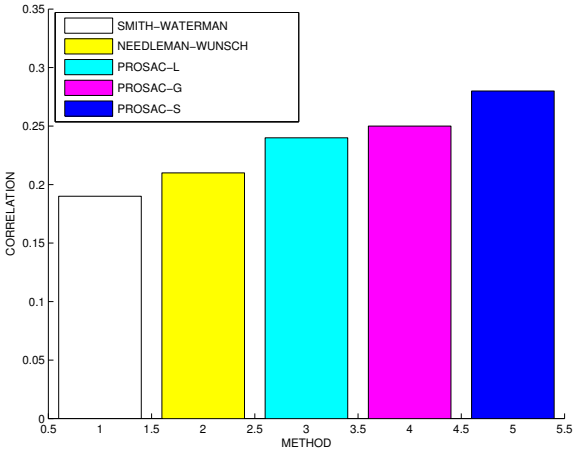

Figure 4: Accuracy of Different Methods (YST2)

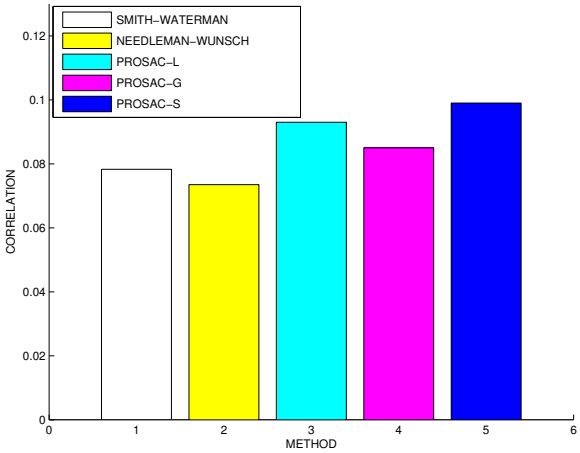

Figure 5: Accuracy of Different Methods (MS)

Data	Smith-Wat.	Need.-Wunsch	PROSAC-G	PROSAC-L	PROSAC-S	PROSAC-S (training)
YST1	723.5 s	536.3 s	3.3 s	14.7 s	3.2 s	135.2 s
YST2	741.6 s	561.7 s	3.2 s	15.1 s	3.1 s	141.7 s
MS	1013.5 s	805.4 s	3.7 s	17.3 s	3.9 s	163.6 s
HS	1234.7 s	871.7 s	4.2 s	22.5 s	4.4 s	205.2 s

Table 3: Efficiency Results

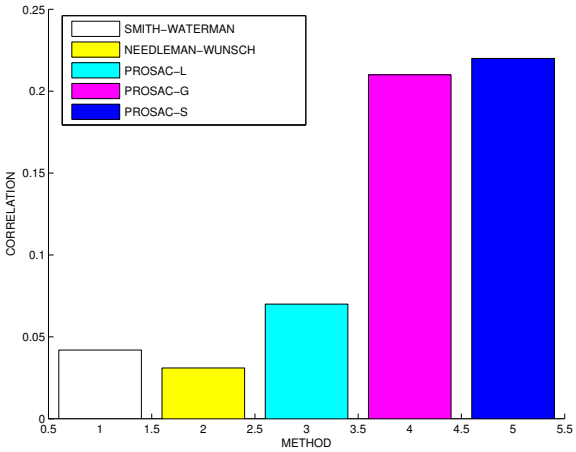

Figure 6: Accuracy of Different Methods (HS)

the PROSAC approach was particularly large. The relative quality of the two alignment approaches varied with the data set, but were almost always superseded by the different versions of the PROSAC method. We note that this data set contained proteins which could often be quite distant from one another, and in many cases, the alignments based similarity function was too noisy to provide robust results. On the other hand, the smoothing methodology of the PROSAC approach was able to abstract out the similarity even in distantly related strings.

The other observation is that the global version of the PROSAC approach is very robust in most cases, whereas for the case of the alignment approaches, both the global and local approaches provides comparatively similar results. The reason for the effectiveness of the global PROSAC approach is that while it performed a global matching for the purposes of overall similarity measurement, the smoothing approach helped in reducing the noise in the segment-wise local alignments. In all cases, the supervised approach was always superior to the other methods in terms of the quality of similarity search. This is because the supervised approach was able to isolate and use the *data set specific* behavior for similarity search. The data set specific behavior was encoded in the varying importance of the different amino acids, as was learned during the training phase of the supervised approach. We note that in many data mining applications, the nature of similarity may be defined by the domain of the data. In such cases, the supervised approach can be very useful when training labels are available. We note that the supervised approach was made possible by the use of a representation which reduces the need for a dynamic programming representation.

4.4 Efficiency Results One of the key weakness of the different alignment methods are their high computational requirements. The high computational requirements of the alignment methodology can often make them impractical for very computationally intensive data mining applications. Our new representational scheme for similarity computations is very efficient since it uses simple distance computations akin to standard Euclidean computations. This is key to the potential use of this technique in applications which require large numbers of repeated distance computations.

All results here are presented on a 1.6 GHz laptop running the Windows XP system and with 1GB of main memory. In Table 3, we have illustrated the running times in seconds (over all similarity computations) for the different data sets and methods. For the case of the supervised approach, we have illustrated both the distance computation and the training times. We note that since the training process needs to be performed only once, these running times are less critical than those for the distance computations. The training process is applied only once on the data set, and once the parameters are learned, the distance computation process can be applied repeatedly.

We note that the (distance computation) running times for all versions of the PROSAC approach are at least *an order of magnitude* better than both the alignment based methods. Among the different variations of the PROSAC approach, the PROSAC-L method required the greatest amount of running time. This is

because the PROSAC-L approach required a comparison between different kinds of (simplified) alignments. These simplified alignments were significantly more efficient to compute than the dynamic programming methods of the Smith-Waterman and Needleman Wunsch method. In the case of the PROSAC approach, straightforward additive distance computations need to be performed on a compressed density representation of the string. This results in improved efficiency. The improved efficiency is critical for the use of the approach in data mining algorithms such as clustering which may require millions of such computations in one application of the method. In such cases, the alignment based methodologies become impractical because of the large running times. Furthermore, the global variation of the PROSAC approach provides a quantitative representation of fixed dimensionality on which most current data mining algorithms can be applied. This is because most operations of classical clustering algorithms such as averaging or centroid determination can be easily defined on the fixed dimensionality quantitative representation. This obviates the need to re-design different kinds of data mining applications on biological data. The good qualitative results of similarity functions defined the PROSAC approach make it likely that such similarity-driven applications can be used effectively while retaining substantial advantages in computational efficiency. In future work, we will investigate the use of such density based string representations on a variety of classical data mining algorithms.

5 Conclusions and Summary

The problem of distance function design has been extensively studied in the biological domain because of its applicability to a wide variety of problems. In spite of the large amount of research on the topic alignment-based methods rely on fixed definitions of similarity, and are often not very useful for arbitrary applications in which the similarity criterion may vary with the application at hand. In this paper, we discussed the importance of a *string representation* in the measurement of similarity. We showed that density based approaches can be used in order to create similarity functions which are effective for both local and global matching over arbitrary similarity functions. Furthermore, representational transformations can make the similarity computation process very efficient. This is particularly useful in many data mining applications such as clustering in which the similarity function may be computed millions of times over different pairs of strings. Furthermore, some of the similarity functions from this approach can be expressed in closed form. Such a closed form expression is naturally amenable to developing a *parameter based supervised approach* which is not easily possible with non-closed form similarity functions such as alignment based methods. Such supervision based approaches are particularly useful for a variety of arbitrary applications in which the particular structure or function being measured depend upon the application at hand. Our results show that the PROSAC approach can retain its effectiveness in both the supervised and unsupervised case, and is also significantly more efficient from a computational perspective.

References

[1] C. C. Aggarwal, *Towards Systematic Design of Distance Functions for Data Mining Applications*, ACM KDD Conference, (2003), pp. 9–19.

[2] S. F. Altschul, W. Gisha, W. Millerb, E. W. Meyersc and D. J. Lipman, *Basic Local Alignment Search Tool*, Journal of Molecular Biology, 215(3), (1990), pp. 403–410.

[3] S. F. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, *Gapped Blast and Psi-BLAST: A new generation of database search programs*, Nucleic Acids Research, 25(17), (1997), pp. 3389–3402.

[4] G. Das, and H. Mannila, *Context-Based Similarity Measures for Categorical Databases*, PDKK Conference, (2000), pp. 201–210.

[5] J. Foote, *A Similarity Measure for Automatic Audio Classification*, AAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora, (1997).

[6] D. Gunopulos, and G. Das, *Time Series Similarity Measures and Time Series Indexing*, ACM SIGMOD Conference, (2001).

[7] J. Gracy, and P. Argos, *Automated protein sequence database classification. I. Integration of compositional similarity search, local similarity search, and multiple sequence alignment*, Bioinformatics, 14(2), (1998), pp. 164–173.

[8] S. Needleman, and C. Wunsch, *A general method applicable to the search for similarities in the amino-acid sequences of two proteins*, Journal of Molecular Biology, 48 (1970), pp. 443–453.

[9] W. R. Pearson, and D. J. Lipman, *Improved Tools for Biological Sequence Comparison*, Proc. Natl. Acad. Sci., 85 (1988), pp. 2444-2448.

[10] G. Salton, and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw Hill, New York, (1983).

[11] T. F. Smith, and M. S. Waterman, *Identification of Common Molecular Subsequences*, Journal of Molecular Biology, 147, (1981), pp. 195–197.

[12] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, (1986).

[13] http://www.cbi.pku.edu.cn/tools/EMBOSS/water.html