A discriminative framework for detecting remote protein homologies

Tommi Jaakkola[†] Mark Diekhans[‡] David Haussler[‡]
tommi@ai.mit.edu *markd@cse.ucsc.edu* *haussler@cse.ucsc.edu*

[†] MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

[‡] Department of Computer Science
University of California
Santa Cruz, CA 95064

October 16, 1999

Abstract

A new method for detecting remote protein homologies is introduced and shown to perform well in classifying protein domains by SCOP superfamily. The method is a variant of support vector machines using a new kernel function. The kernel function is derived from a generative statistical model for a protein family, in this case a hidden Markov model. This general approach of combining generative models like HMMs with discriminative methods such as support vector machines may have applications in other areas of biosequence analysis as well.

1 Introduction

A core problem in statistical biosequence analysis is the annotation of new protein sequences with structural and functional features. To a degree, this can be achieved by relating the new sequences to proteins for which such structural properties are already known, i.e. by detection of protein homologies. Many statistical, sequence-based tools have been developed for detecting protein homologies. These include BLAST [3, 2], Fasta [41], PROBE [39], templates [43], profiles [19], position-specific weight matrices [21], and Hidden Markov Models (HMMs) [34]. Recent experiments [7, 40] have used the SCOP classification of protein structures [22] to test many of these methods to see how well they detect remote protein homologies that exist between protein domains that are in the same structural superfamily, but not necessarily in the same family. This work has shown that methods such as PSI-BLAST and HMMs, which build a statistical model from multiple sequences, perform better than simple pairwise comparison methods, but all sequence-based methods miss many important remote homologies.

We present and evaluate a new methodology for detecting remote protein homologies. In this approach we use generative statistical models built from multiple sequences, in this case HMMs, as a way of extracting features from protein sequences. This maps all protein sequences to points in a Euclidean feature space of fixed dimension. (See [36] for a different method of mapping protein sequences into Euclidean space). We then use a general discriminative statistical method to classify the points representing protein sequences by domain superfamily. This is quite distinct from methods that train the parameters of the HMM itself to give a more discriminative model [16, 38]. Other discriminative methods, using neural nets, are described in [10, 11]. Using our method, we obtain a substantial improvement in identifying remote homologies over what is achieved by HMMs alone, as they are currently employed. This new method also compares favorably to what have been called family pairwise search homology methods, in which the scores from all pairwise comparisons between a query protein and the members of a known protein family are combined to improve performance [20].

2 Methods

The statistical modeling approach to protein sequence analysis involves constructing a *generative* probability model, such as an HMM, for a protein family or superfamily [12]. Sequences known to be members of the protein family are used as *(positive) training examples.* The parameters of a statistical model representing the family are estimated using these training examples, in conjunction with general *a priori* information about properties of proteins. The model assigns a probability to any given protein sequence. If it is a good model for the family it is trained on, then sequences from that family, including sequences that were not used as training examples, yield a higher probability score than those outside the family. The probability score can thus be interpreted as a measure of the extent to which a new protein sequence is homologous to the protein family of interest. Considerable recent work has been done in refining HMMs for the purpose of identifying weak protein homologies in this way [34, 5, 14, 24, 33].

Let $X = [x_1, ..., x_n]$ denote a protein sequence, where each x_i is an amino acid residue. Suppose that we are interested in a particular protein family such as immunoglobulins and have estimated an HMM, H_1, for this family (for details of the estimation process see, e.g., [12]). We use $P(X|H_1)$ to denote the corresponding probability model. In a database search, a likelihood ratio score is often used in place of a simple probability $P(X|H_1)$:

$$\begin{aligned}\mathcal{L}(X) &= \log \frac{P(X|H_1)P(H_1)}{P(X|H_0)P(H_0)} \qquad (1)\\ &= \log \frac{P(X|H_1)}{P(X|H_0)} + \log \frac{P(H_1)}{P(H_0)}\end{aligned}$$

where the *null model* $P(X|H_0)$ is the probability model under the hypothesis that the sequence X does not belong to the family of interest; different types of null models are discussed in [6, 40]. A positive value of the likelihood ratio $\mathcal{L}(X)$ is taken as an indication that the new sequence X is indeed a member of the family. The constant factor

$\log P(H_1)/P(H_0)$, the log prior odds, provides an *a priori* means for biasing the decision and does not affect the ranking of sequences being scored.

2.1 Discriminative approaches

The parameters of a generative model are estimated in such a way as to make the positive training examples, proteins in the family being modeled, very likely under the probability model. In contrast, the parameters of a discriminative model are estimated using both positive training examples and *negative training examples*, which are proteins that are not members of the family being modeled. The goal in estimating the parameters of a discriminative model is to find parameters such that the score derived from the model can be used to discriminate members of the family from non-members, e.g. such that members of the family receive a high score and non-members receive a low score.

As discussed above, generative models can be used for discrimination when a null model is provided. Consider the log-likelihood ratio formulation above, eq. (2). By Bayes rule, we may rewrite this as follows:

$$\mathcal{L}(X) = \log \frac{P(X|H_1)P(H_1)}{P(X|H_0)P(H_0)} = \log \frac{P(H_1|X)P(X)}{P(H_0|X)P(X)} = \log \frac{P(H_1|X)}{P(H_0|X)} \tag{2}$$

where $P(X) = P(X|H_1)P(H_1) + P(X|H_0)P(H_0)$ is the overall probability model for sequences both in the family and not in the family. $P(H_1|X)$ is referred to as the *posterior probability* of the model; in our case it is the posterior probability that the sequence X belongs to the protein family being modeled. The score function $\mathcal{L}(X)$ is called the *log posterior odds* score. We should classify X as a member of the protein family if this score is positive, else as a non-member. While this decision rule is perfectly reasonable, and in fact optimal, when both the model H_1 and the null model H_0 are completely accurate, it can perform poorly when these models are not accurate. This can easily happen with limited training sets [6, 40] or when the inherent structural assumptions made during the model construction are inaccurate. This problem is made worse by the objective function typically used in estimating the parameters for the generative model, $P(X|H_1)$, and occasionally also used to estimate the parameters of the null model, $P(X|H_0)$. More specifically, $P(X|H_1)$ is refined *only* on the basis of the positive training examples, and the parameters for the null model $P(X|H_0)$, if modified at all, are adjusted *only* based on negative training examples. An improvement in the accuracy of either of these models does not directly translate into a better discriminative performance, which is governed by the score function $\mathcal{L}(X)$. In a discriminative approach the objective is to refine the discriminant function $\mathcal{L}(X)$ directly using both positive and negative training examples.

2.2 Kernel methods

Here we provide a brief introduction to a particular class of discriminative techniques known as kernel methods. Our introduction is geared towards our goals in biosequence analysis. For more details on the general class of kernel methods see e.g. [44, 37].

Suppose we have a training set of examples (protein sequences) $\{X_i\}, i = 1, \ldots, n$ for which we know the correct hypothesis class, H_1 or H_0. In other words, we have a set of protein sequences that are known to be either homologous to the family of interest or not. We model the discriminant function $\mathcal{L}(X)$ directly via the following expansion in terms of the training examples

$$\begin{aligned} \mathcal{L}(X) &= \log P(H_1|X) - \log P(H_0|X) \\ &= \sum_{i:X_i \in H_1} \lambda_i K(X, X_i) - \sum_{i:X_i \in H_0} \lambda_i K(X, X_i) \end{aligned} \tag{3}$$

This expansion could have been specified without any reference to the posterior class probabilities $P(H_1|X)$ and $P(H_0|X)$; the latter quantities are included here only for the purpose of illustration. The sign of the discriminant function determines the assignment of the sequences into hypothesis classes. The contribution, either positive or negative, of each training example (sequence) to the decision rule consists of two parts: 1) the overall importance of the example X_i as summarized with the non-negative coefficient λ_i and 2) a measure of pairwise "similarity" between the training example X_i and the new example X, expressed in terms of a *kernel* function $K(X_i, X)$. So, to restate the expansion more generally, the classification decisions are made on the basis of weighted pairwise similarities to the training examples.

The free parameters in the above decision rule are the coefficients λ_i and to some degree also the kernel function K. To pin down a particular kernel method, two things need to be clarified. First, we must specify how to set the values for the coefficients λ_i; this is explained in the next section. We subsequently discuss the second and deeper issue of how to choose an appropriate kernel function that defines the pairwise comparison between protein sequences.

2.2.1 Optimization of the discriminative coefficients

Since the sign of the discriminant function $\mathcal{L}(X)$ determines the predicted class for any sequence X, we would certainly like to have this sign correct and the value separated from zero by a large margin for as many of the training examples as possible. In other words, if we chose a margin of 1, our objective should be to find, if possible, coefficients λ_i so that

$$\mathcal{L}(X_i) \geq 1, X_i \in H_1 \tag{4}$$

$$\mathcal{L}(X_i) \leq -1, X_i \in H_0. \tag{5}$$

Since any margin, once achieved, can be increased by scaling the λ_i, to convert this into a constrained maximization problem, we must impose additional constraints on these coefficients, e.g. $0 \leq \lambda_i \leq 1$. Even with the additional constraints, the solution, if it exists, is not unique. Following Vapnik's work on *support vector machines* (SVMs)[44], we define a quadratic objective function for the coefficients that is then used to maximize the margins in a geometrically meaningful way[1]. The following objective function also generalizes to the setting where not all the margin constraints can be satisfied:

$$J(\lambda) = \sum_{i:X_i \in H_1} \lambda_i (2 - \mathcal{L}(X_i)) + \sum_{i:X_i \in H_0} \lambda_i (2 + \mathcal{L}(X_i)) \tag{6}$$

[1]Our formulation differs slightly from that of [44] but the geometric motivation remains the same. For details see [44] or [8].

(recall that $\mathcal{L}(X_i)$ is a linear function of the λ_i coefficients). Subject to standard constraints on the kernel function[2], the solution to the constrained maximization of $J(\lambda)$ is unique and can be achieved iteratively. For λ_i corresponding to $X_i \in H_1$ we proceed as follows. If this coefficient were unconstrained, we would update it so that the discriminant function evaluated at X_i (i.e. $\mathcal{L}(X_i)$) after the update would be exactly 1. This would give the update rule

$$\lambda_i \leftarrow \frac{1 - \mathcal{L}(X_i) + \lambda_i K(X_i, X_i)}{K(X_i, X_i)} \tag{7}$$

It is clear that $\mathcal{L}(X_i) = 1$ holds after this update. However, this ignores the constraints $\lambda_i \in [0, 1]$. Taking these into account, we get the modified update

$$\lambda_i \leftarrow f\left(\frac{1 - \mathcal{L}(X_i) + \lambda_i K(X_i, X_i)}{K(X_i, X_i)}\right) \tag{8}$$

where the function f maintains the constraints: $f(z) = 0$ for $z \leq 0$, $f(z) = z$ for $0 \leq z \leq 1$ and $f(z) = 1$ otherwise. This gives the best approximation to the above update that is allowed given the constraints on λ_i. Proceeding analogously for $X_i \in H_0$ the update rule becomes:

$$\lambda_i \leftarrow f\left(\frac{1 + \mathcal{L}(X_i) + \lambda_i K(X_i, X_i)}{K(X_i, X_i)}\right) \tag{9}$$

By repeatedly updating the coefficients λ_i corresponding to the positive and negative training examples, it can be shown that the coefficients will converge to those that maximize the quadratic objective function $J(\lambda)$ subject to the given constraints. Although this simple procedure can be slow in pathological cases[3], we have found it to be fast enough in practice.

A useful property of the quadratic objective function $J(\lambda)$ is that it forces most of the λ_i coefficients to converge to zero. The training examples corresponding to these zero coefficients will not contribute to the classification rule as defined by the score function $\mathcal{L}(X)$ in eq. (4). Such training examples can be therefore safely ignored and this leads to fewer evaluations of the kernel function for each query sequence.

2.3 The Fisher kernel

Finding an appropriate kernel function for a particular application area can be difficult and remains largely an unresolved issue. We have, however, developed a general formalism for deriving kernel functions from generative probability models [27]. This formalism carries several advantages, including the ability to handle complex objects such as variable length protein sequences within the kernel function. Furthermore, the formalism facilitates the encoding of prior knowledge about protein sequences, via the probability models, into the

[2]More precisely, for any finite set of examples $X_i, i = 1, \ldots, n$ the kernel function $K_{ij} = K(X_i, X_j)$ must define a positive definite matrix.

[3]By changing at random the order in which each coefficient is updated, one can increase the rate of convergence significantly.

kernel function. We emphasize that this is an important consideration since the kernel function mediates all the pairwise comparisons between the protein sequences.

Our approach here is to derive the kernel function from HMMs corresponding to the protein family of interest. We are thus able to build on the work of others towards adapting HMMs for protein homology detection[34, 24, 33]. Our use of protein models in the kernel function, however, deviates from the standard use of such models in biosequence analysis. More precisely, the kernel function specifies a similarity score for any pair of sequences, whereas the likelihood score from an HMM only measures the closeness of the sequence to the model itself. Indeed, the HMM can assign the same likelihood to two widely different protein sequences. We must therefore be able to extract and entertain richer representations than the HMM score in order to carry out meaningful model-based pairwise comparisons.

Suppose now that we have estimated an HMM for a particular family of proteins such as the immunoglobulins. Let the corresponding probability model be $P(X|H_1, \theta)$, where the parameters θ include the output and the transition probabilities of an HMM trained to model immunoglobulins (see e.g. [34]). To compute the likelihood score for a query sequence X or, equivalently, to evaluate $P(X|H_1, \theta)$, we employ the standard forward-backward algorithm [42]. In addition to obtaining the generative likelihood for the query sequence, the forward-backward algorithm extracts what are known as *sufficient statistics* for the parameters. For HMMs, the sufficient statistics are the posterior frequencies of having taken a particular transition or having generated one of the residues of the query sequence X from a particular state. We can arrange these statistics in a vector of fixed dimension that contains a value (sufficient statistic) for each independent parameter in the model. The resulting vector of sufficient statistics reflects the generation process of the sequence from the HMM in the sense that it captures how each parameter is involved. More generally, it provides a complete summary of the sequence in the parameter space of the model.

We use the vector of sufficient statistics as an intermediate representation of the query sequence. This representation naturally respects the assumptions that went into building the HMM. Also, unlike the simple likelihood score, it provides the means for comparing two sequences relative to a single model. In other words, the closeness of the sufficient statistics provides an appropriate measure of similarity between the sequences.

This idea can be generalized considerably[29]. In the more general treatment, one works not with the vector of sufficient statistics directly but with an analogous quantity known as the *Fisher score*

$$U_X = \nabla_\theta \log P(X|H_1, \theta) \tag{10}$$

Each component of U_X is a derivative of the log-likelihood score for the query sequence X with respect to a particular parameter. The magnitude of the components specify the extent to which each parameter contributes to generating the query sequence. The computation of these gradients in the context of HMMs along with their relation to sufficient statistics is described in more detail in Appendix A.

Finding an appropriate kernel function in this new gradient representation is easier than in the original space of variable length protein sequences. We only need to quantify the similarity between two fixed length gradient vectors U_X and $U_{X'}$ corresponding to two sequences X and X', respectively. A number of appropriate kernel functions can be derived once we quantify a distance function between such vectors. A natural (squared) distance between the gradient vectors is given by (see, e.g., [26])

$$D^2(X, X') = \frac{1}{2}(U_X - U_{X'})^T \, F^{-1} \, (U_X - U_{X'}) \tag{11}$$

where F is the Fisher information matrix or, equivalently, the covariance matrix of the score vectors U_X when the examples (sequences) are sampled from the same probability model ($P(X|H_1, \theta)$ in our case). In the experiments reported in this paper, we used the following *Gaussian kernel*

$$K(X, X') = e^{\,-D^2(X,X')} \tag{12}$$

where, in addition, the Fisher information matrix F appearing in the distance function was approximated by $F \approx \sigma^2 \, I$, where I is the identity matrix and σ a scaling parameter. This approximation was made for efficiency reasons since the score vectors corresponding to the protein HMMs tend to have several thousand components. The scaling parameter was set equal to the median Euclidean distance between the gradient vectors corresponding to the training sequences in the protein family of interest and the closest gradient vector from a protein belonging to another protein fold. Roughly speaking, the scaling guarantees that, on average, the (approximate) distance from a protein sequence in the family of interest to its nearest non-member is one.

To summarize, we begin with an HMM trained from positive examples to model a given protein family. We use this HMM to map each new protein sequence X we want to classify into a fixed length vector, its Fisher score, and compute the kernel function on the basis of the Euclidean distance between the score vector for X and the score vectors for known positive and negative examples X_i of the protein family. The resulting discriminant function is given by

$$\mathcal{L}(X) = \sum_{i:X_i \in H_1} \lambda_i K(X, X_i) - \sum_{i:X_i \in H_0} \lambda_i K(X, X_i), \tag{13}$$

where K is the kernel function defined above and the λ_i are estimated from the positive and negative training examples X_i as described in Section 2.2.1. We refer to this as the *SVM-Fisher* method.

2.4 Combination of scores

In many cases we can construct more than one HMM model for the family or superfamily of interest. It is advantageous in such cases to combine the scores from the multiple models rather than selecting just one. Let $\mathcal{L}_i(X)$ denote the score for the query sequence X based on the i^{th} model. This might be the score derived from the SVM-Fisher method, Equation (13), or the log likelihood ratio for the generative HMM model,

$$\log \frac{P(X|H_1)}{P(X|H_0)},$$

or even a negative log E-value derived from a BLAST comparison, as described in section 3. We would like to combine the SVM-Fisher scores for X from all the models of the given family, and similarly with HMM and BLAST scores. Unfortunately, there is no clear optimal way to combine these scores that is practical to implement. How one should combine them depends on the joint distribution of all the score functions from the different models, and also on the exact performance measure one wants to optimize by performing this combination of scores [35, 4]. Since this is not the focus of this paper, here we only explore two simple heuristic means of combining scores. These are *average score*

$$\mathcal{L}_{ave}(X) = \frac{1}{N} \sum_i \mathcal{L}_i(X) \tag{14}$$

and *maximum score*

$$\mathcal{L}_{max}(X) = \max_i \mathcal{L}_i(X), \tag{15}$$

where in each case the index i ranges over all models for a single family of interest. These combination methods have also been explored in other protein homology experiments [20]. The average score method works best if the scores for the individual models are fairly consistent, and the maximum score method is more appropriate when we expect a larger value of some individual model score to be a more reliable indicator of a positive example. We have found that the maximum score method works better in our experiments with generative HMM models and BLAST scores, so this approach is used there. However, the average score method works better for combining scores from our discriminative models, so it is used in these experiments.

3 Experimental Methods

We designed a set of experiments to determine the ability of SVM-Fisher kernel discriminative models to recognize remote protein homologs. The SVM-Fisher kernel methods were compared to *BLAST* [3, 18] and the generative HMMs built using the *SAM-T98* methodology [40, 32, 23, 24]. The experiments measured the recognition rate for members of superfamilies of the SCOP protein structure classification scheme [22]. We simulate the remote homology detection problem by withholding all members of a SCOP family from the training set and training with the remaining members of the SCOP superfamily. We then test sequences from the withheld family to see if they are recognized by the model built from the training sequences. Since the withheld sequences are known remote homologs, we are able to demonstrate the relative effectiveness of the techniques in classifying new sequences as remote members of a superfamily. In a sense, we are asking, “Could the method discover a new family of a known superfamily?”

3.1 Overview of experiments

The SCOP version 1.37 PDB90 domain database, consisting of protein domains, no two of which have 90% or more residue identity, was used as the source of both training and test sequences. PDB90 eliminates a large number of essentially redundant sequences from the

SCOP database. The use of the domain database allows for accurate determination of a sequence's class, eliminating the ambiguity associated with searching whole-chain protein databases.

The generative models were obtained from an existing library of *SAM-T98* HMMs. The *SAM-T98* algorithm, described more fully in [32], builds an HMM for a SCOP domain sequence by searching the non-redundant protein database *NRP* for a set of potential homologs of the sequence and then iteratively selecting positive training sequences from among these potential homologs and refining a model. The resulting model is stored as an alignment of the domain sequence and final set of homologs.

All SCOP families that contain at least 5 PDB90 sequences and have at least 10 PDB90 sequences in the other families in their superfamily were selected for our test, resulting in 33 test families from 16 superfamilies (table 1). When testing the recognition of each of these families, the training and test sets were constructed as follows. The positive training examples were selected from the remaining families in the same superfamily and the negative training examples from outside the fold that the family belongs to. The positive test examples consisted of all the PDB90 sequences in the family of interest. The negative test examples were chosen from outside of the fold and in such a way that the negative examples in the training set and those in the test set never came from the same fold. Figure 1 shows an example of the division. All of the training and test sequences, as well as the *SAM-T98* models and alignments are available from our web site [28].

For each of the 33 test families, all the test examples, both positive and negative, were scored, based on a discriminant function constructed from the training examples. We used various performance criteria, described in the following section, to evaluate the quality of the discriminant function. The main purpose of the discriminant function is to give better scores to the positive rather than the negative test examples. Using this setup, the performance of the SVM-kernel method was compared to the performance of the generative HMM alone, and to BLAST scoring methods.

As a gauge of the distance of the test sequences from the positive training examples, the percent residue identity, when available, was obtained from the SCOP domain structural alignment database created by Gerstein and Levitt [17]. The alignment database was created for SCOP release 1.35, and thus did not fully cover the test set from release 1.37. Additionally, the majority (98%) of the pairs of the test and positive example sequences where not determined to have sufficient structural similarity for alignment by the database. Nevertheless, the structural alignments in this database give some rough idea of how remote the homologies are that we are trying to find.

To use the Gerstein and Levitt database, for each sequence in a test family, we define its similarity to the positive training examples for this family as the maximum percent residue identity between the test sequence and any of these positive training examples, computed from the Gerstein-Levitt alignments. If no such alignments can be found for a given test sequence, its similarity is taken to be zero. Separately for each of our 33 test families, we compute the similarity values for all sequences in the given test family, sort these values in descending order, and report the quartiles of this distribution in Table 1. The column marked "50" is the 50th percentile of this distribution, i.e. the median. The columns marked "25" and "75" give the first and third quartile, respectively. A dash represents a value of zero that arises from the lack of any appropriate structural alignments in the database.

3.2 Multiple models used

After selecting a test family, we must construct a model for its superfamily using available sequences from the other families in that superfamily. The *SAM-T98* method starts with a single sequence (the guide sequence for the domain) and builds a model. In general, there are too many sequences in the other families of the superfamily to consider building a model around each one of them, so we used a subset of PDB90 superfamily sequences that were present in a diverse library of existing HMMs. The SVM-Fisher method was subsequently trained using each of these models in turn. The scores for the test sequences, given each HMM model, were computed from Equation (13), and the scores obtained based on multiple models were combined according to Equation (14).

3.3 Details on the training and test sets

In each experiment, all PDB90 sequences outside the fold of the test family were used as either negative training or negative test examples. All experiments were repeated with the test/training allocation of negative examples reversed. This resulted in approximately 2400 negative test sequences for most test families. The split of negative examples into test and training was done on a fold-by-fold basis, in such a way that folds were never split between test and train. This insured that a negative training example was never similar to a negative test example, which might give a significant advantage to discriminative methods. In actual applications, this requirement could be relaxed, and further improvements might be realized by using discriminative methods.

For positive training examples, in addition to the PDB90 sequences in the superfamily of the test family (but not in the test family itself), we also had available the final set of domain homologs found by the *SAM-T98* method for the HMM models built from the subset of SCOP sequences in the superfamily. We tried two types of experiments, in which the positive training examples consisted of

1. only PDB90 sequences in the superfamily of the test family but excluding those in the test family itself and

2. in addition to the above sequences, all the domain homologs found by each individual *SAM-T98* model built for the selected guide sequences that belong to the superfamily but not to the test family itself.

The results for the latter method were slightly superior for the SVM-Fisher method, and we report them here.

3.4 BLAST methods

Two *BLAST* methodologies were used for comparison, each using WU-BLAST version 2.0a16 [45, 1]. These are family pairwise search homology methods, as explored in [20].

In both methods, the PDB90 database was queried with each positive training sequence, and E-values were recorded. One method, referred to as *BLAST:SCOP-only* in the results section, used positive training examples as defined by (1) above. The other, which we call *BLAST:SCOP+SAM-T98-homologs*, included the *SAM-T98* domain homologs as positive examples, as in (2) above. In both cases, the scores were combined by the maximum method, so the final score of a test sequence in the PDB90 database was taken to be the maximum $-\log$ E-value for any of the positive training example query sequences. This score measures the BLAST-detectable similarity of the test sequence to the closest sequence in the set of positive training sequences. In [20], a related combination rule, which instead used the average of the BLAST bit scores, was suggested. We tried a similar average method, taking the average of the $-\log$ E-values, which should in theory be more accurate than averaging the bit scores. However, the maximum method performed best, so we report results for that combination method only.

3.5 Generative HMM scores

Finally, we also report results using the *SAM-T98* method as a purely generative model. The null model used here is the reverse sequence model from [40, 32]. We used the same data and the same set of models as in the SVM-Fisher score experiments; we just replaced the SVM-Fisher score with the *SAM-T98* score. However, the scores were combined with the maximum method, since that performed slightly better in this case.

In these experiments we also tried two different types of positive training examples. In the first set of experiments, we used only the domain homologs found by the *SAM-T98* method itself as a training set for each HMM. Thus, we simply used the *SAM-T98* models as they were given in the existing library of models. In the second experiment, we retrained each of these models using all of the data in (2) above. That is, using all of the SCOP sequences in the superfamily being modeled (but not in the family itself), and all of the domain homologs found by the given *SAM-T98* model and by the models built from other guide sequences from this superfamily (but not in the family itself). Thus in this latter case, each HMM was trained on same set of positive training examples used by the SVM-Fisher and BLAST:SCOP+SAMT98-homologs methods. Performance was somewhat better in the latter case at higher rates of false positives (RFP, see below), but was worse at lower RFP, making the method of less practical value. Therefore, we report the results of the first experiment here.

4 Results

Here we provide a comparison of the results of the best performing approaches for each of the methods. Since the numeric scores produced by each method are not directly comparable, we use the rate of false positives (*RFP*) as our performance measure, as in [40]. The *RFP* for a positive test sequence is defined as the fraction of negative test sequences that score as high or better than the positive sequence.

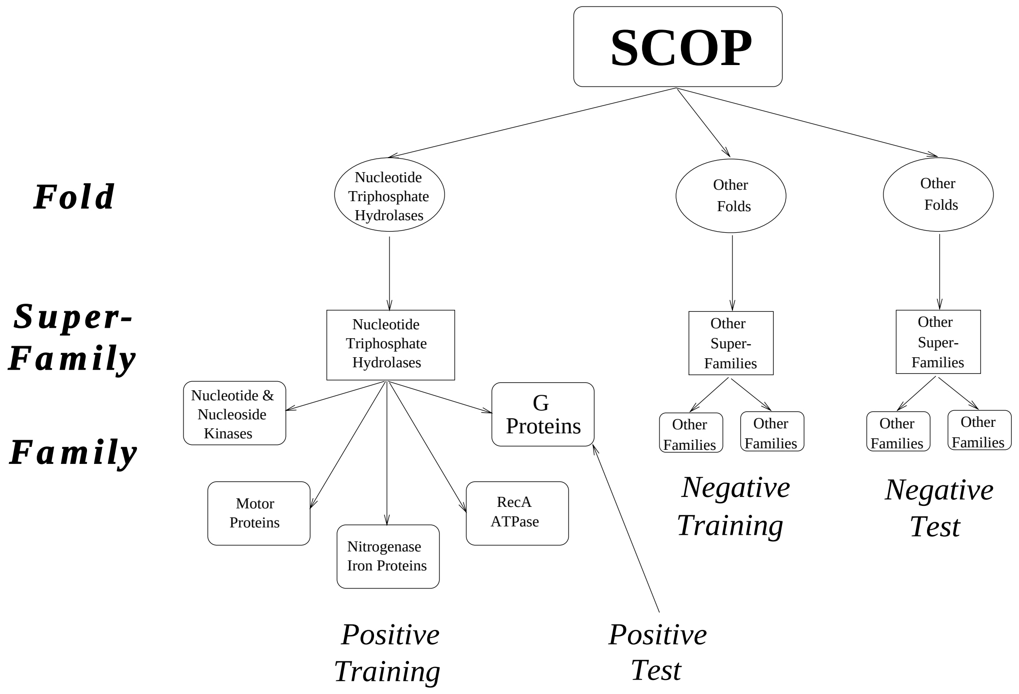

Figure 1: Separation of the SCOP PDB90 database into training and test sequences, shown for the *G proteins* test family.

	SCOP	SCOP	SAM-T98		IDE Percentile		
#	Superfamily	Family	HMM	Hom	25	50	75
1	Globin-like(46)	Phycocyanins(8)	13	840	21.5	-	-
2	4-helical cytokines(18)	Long-chain cytokines(7)	9	57	-	-	-
3		Short-chain cytokines(6)	7	104	19.4	18.5	16.6
4		Interferons/interleukin-10(5)	8	68	20.1	17.9	-
5	EF-hand(24)	Parvalbumin(6)	13	761	23.4	23.3	22.4
6		Calmodulin-like(12)	7	807	18.8	16.8	-
7	Immunoglobulin(252)	V set domains(120)	7	1827	25.5	-	-
8		C1 set domains(103)	15	4701	-	-	-
9		C2 set domains(5)	15	4701	21.4	19.8	-
10		I set domains(8)	10	5791	24.3	20.2	-
11		E set domains(16)	13	5410	19.8	14.3	-
12	Cupredoxins(35)	Plastocyanin/azurin-like(18)	1	737	-	-	-
13		Multidomain cupredoxins(15)	8	152	19.7	18.3	-
14	Viral coat and capsid	Plant virus proteins(8)	16	152	18.4	16.6	-
15	proteins(50)	Animal virus proteins(37)	10	19	14.8	-	-
16	ConA-like lectins/ glucanases(25)	Legume lectins(9)	7	104	-	-	-
17	Trypsin-like serine	Prokaryotic proteases(8)	9	793	24.0	-	-
18	proteases(37)	Eukaryotic proteases(26)	5	52	29.6	26.2	-
19	Acid proteases(20)	Retroviral protease(6)	8	280	-	-	-
20	Lipocalins(18)	Retinol binding protein-like(6)	6	178	-	-	-
21	Glycosyl- transferases(35)	alpha-Amylases, N-terminal domain(12)	15	92	-	-	-
22		beta-glycanases(11)	9	89	18.3	17.8	-
23		type II chitinase(6)	11	115	19.1	18.8	-
24	NAD(P)-binding Rossmann-fold	Alcohol/glucose dehydrogenases, C-terminal domain(7)	8	650	-	-	-
25	domains(54)	Glyceraldehyde-3-phosphate dehydrogenase-like, N-terminal domain(12)	8	650	-	-	-
26		Formate/glycerate dehydrogenases, NAD-domain(5)	8	650	20.1	19.1	18.2
27		Lactate and malate dehydrogenases, N-terminal domain(15)	8	650	19.9	-	-
28	P-loop containing	Nucleotide and nucleoside kinases(11)	5	510	-	-	-
29	nucleotide triphosphate hydrolases(27)	G proteins(8)	9	112	18.5	17.1	-
30	Thioredoxin-like(20)	Thioltransferase(7)	2	276	-	-	-
31		Glutathione S-transferases, N-terminal domain(9)	9	269	18.3	-	-
32	alpha/beta- Hydrolases(25)	Fungal lipases(8)	9	105	29.4	18.4	-
33	Periplasmic binding protein-like II(16)	Transferrin(6)	8	89	-	-	-

Table 1: SCOP 1.37 test families used in these experiments. The first column is a family numeric identifier used in other graphs. The number of PDB90 superfamily or family domains is in parenthesis following the name. *SAM-T98* has the number of HMMs used as models for the family and the average number of homologs used in training each of the HMMs. The last three columns are the highest percent sequence identity with the SCOP training families, as described in the text.

4.1 G-proteins

Here, as an example, we look at the results for the *G proteins* family of the *nucleotide triphosphate hydrolases* SCOP superfamily.

The HMMs used in the recognition of members of the *G proteins* family were taken from two other families in the superfamily: *nucleotide and nucleoside kinases*, and *nitrogenase iron protein-like* (table 2). The positive training examples were the SCOP PDB90 sequences from all families in the superfamily except the G protein family (table 3) along with the HMM domain homologs found by the models corresponding to the guide sequences, as listed in table 2.

This experiment tested the ability of the methods to distinguish the 8 PDB90 *G proteins* from 2439 sequences in other SCOP folds. The results are given in table 4. It is seen that the *SVM-Fisher* method scores 5 of the 8 *G proteins* better than all 2439 negative test sequences, and gets a lower rate of false positives than the other methods on the other 3 sequences, with the exception of 1eft-3.

We summarize the performance of the four methods in recognizing this family by looking at two overall figures of merit. The first is the RFP obtained when it is demanded that all of the sequences in the family are recognized. This is the same as the maximum RFP of any sequence in the family. Under this measure of performance, we get 0.867 for BLAST:SCOP-only, 0.568 for BLAST:SCOP+SAMT98-homologs, 0.428 for *SAM-T98*, and 0.051 for SVM-Fisher for the G-proteins family.

Since the maximum RFP can be dominated by a few outliers, which for some reason may be particularly hard for a method to recognize, we also consider the median RFP for the sequences in the family. This is obtained by sorting the RFPs for the test sequences from smallest to largest, and taking the value that occurs in the middle of this list. To get a good median RFP requires only that at least half of the sequences in the family be easy for the method to recognize. For each method, different sequences may be included in this “easier half” of the family. Under this measure of performance, we get 0.378 for BLAST:SCOP-only, 0.330 for BLAST:SCOP+SAMT98-homologs, 0.007 for *SAM-T98*, and 0.0 for SVM-Fisher for the G-proteins family.

4.2 Overall results

In Table 5 we give the performance of all four methods on each of the 33 protein families we tested, as measured by the maximum and median RFP. We also computed these statistics for the first and third quartile, and the relative performance of the four methods was similar (data not shown).

A graphical comparison of the overall results for the 33 test families is given in figures 2 through 7.

4.3 Further experiments

We did further experiments to verify that the *SVM-Fisher* method was not relying too heavily on length and compositional bias in discriminating one protein domain family from another, as suggested by a referee of this paper. Such information would not be derivable

Family	HMM	# Homologs
Nucleotide and nucleoside kinases	1dekA	8
	1aky	131
	1ukz	136
	3adk	128
	1gky	131
	1ukd	126
Nitrogenase iron protein-like	1dts	46
	1nipA	268
	1adeA	41

Table 2: Models used for detection of members of the SCOP *G proteins* family. The *HMM* column gives the PDB id of the guide sequence used to create the model followed by the number of homologs in the final *SAM-T98* alignment.

Family	SCOP Domains
Nucleotide and nucleoside kinases	1ak2 1akeA 1aky 1dekA 1gky 1ukd
	1ukz 1vtk 1zin(1) 2ak3A 3adk
G proteins	1dar(2) 1eft(3) 1etu 1gia(2) 1guaA
	1hurA 1tadA(2) 5p21
Motor proteins	1mmd(2) 2mysA(2)
Nitrogenase iron protein-like	1adeA 1dts 1nipA
RecA protein-like (ATPase-domain)	1bmfA(3) 1bmfD(3) 2reb(1)

Table 3: SCOP PDB90 members of the *nucleotide triphosphate hydrolases* superfamily. The domains are identified by their PDB chain identifier. For partial-chain domains, the SCOP domain number within the chain is included.

Sequence	BLAST	B-Hom	S-T98	SVM-F
5p21	0.043	0.010	0.001	0.000
1guaA	0.179	0.031	0.000	0.000
1etu	0.307	0.404	0.428	0.038
1hurA	0.378	0.007	0.007	0.000
1eft(3)	0.431	0.568	0.041	0.051
1dar(2)	0.565	0.391	0.289	0.019
1tadA(2)	0.797	0.330	0.004	0.000
1gia(2)	0.867	0.421	0.017	0.000

Table 4: Rate of false positives for *G proteins* family. BLAST = BLAST:SCOP-only, B-Hom = BLAST:SCOP+SAMT-98-homologs, S-T98 = SAMT-98, and SVM-F = SVM-Fisher method.

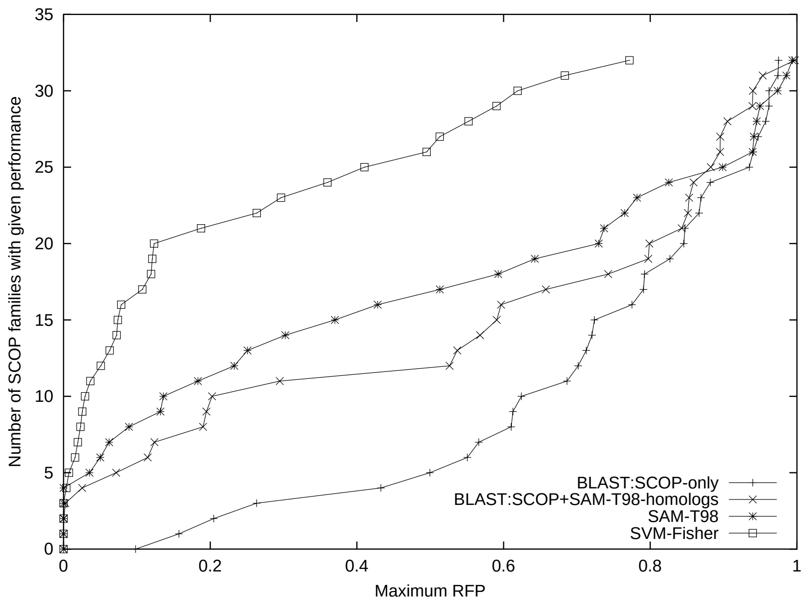

Figure 2: Here we compare the overall performance for the four methods on the 33 test families. For each family we computed the maximum RFP for that family, as shown in Table 5. Possible values for this RFP are shown on the X-axis. On the Y-axis we plot the number of SCOP families, out of the 33 families that we tested, for which the given method achieves that RFP performance or better.

#	Family	Maximum RFP				Median RFP			
		BLT	BLH	ST98	SVM	BLT	BLH	ST98	SVM
1	Phycocyanins	0.882	0.743	0.950	0.619	0.391	0.342	0.450	0.364
2	Long-chain cytokines	0.847	0.526	0.994	0.123	0.721	0.397	0.446	0.035
3	Short-chain cytokines	0.686	0.658	0.513	0.023	0.407	0.114	0.109	0.002
4	Interferons/interleukin-10	0.613	0.799	0.765	0.119	0.324	0.440	0.289	0.004
5	Parvalbumin	0.098	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	Calmodulin-like	0.433	0.002	0.000	0.000	0.023	0.000	0.000	0.000
7	Immunoglobulin V dom	0.720	0.115	0.974	0.016	0.135	0.000	0.000	0.000
8	Immunoglobulin C1 dom	0.624	0.000	0.000	0.063	0.033	0.000	0.000	0.000
9	Immunoglobulin C2 dom	0.263	0.124	0.136	0.019	0.119	0.006	0.000	0.000
10	Immunoglobulin I dom	0.157	0.190	0.251	0.495	0.007	0.004	0.000	0.000
11	Immunoglobulin E dom	0.792	0.797	0.899	0.683	0.168	0.329	0.178	0.073
12	Plastocyanin/azurin-like	0.869	0.895	0.730	0.772	0.016	0.049	0.039	0.013
13	Multidomain cupredoxins	0.775	0.853	0.233	0.360	0.342	0.116	0.003	0.002
14	Plant virus proteins	0.975	0.940	0.782	0.410	0.641	0.391	0.088	0.133
15	Animal virus proteins	0.962	0.997	0.941	0.513	0.750	0.630	0.204	0.066
16	Legume lectins	0.551	0.895	0.643	0.552	0.278	0.298	0.278	0.083
17	Prokaryotic proteases	0.962	0.025	0.000	0.000	0.080	0.002	0.000	0.000
18	Eukaryotic proteases	0.846	0.000	0.000	0.000	0.000	0.000	0.000	0.000
19	Retroviral protease	0.500	0.195	0.183	0.187	0.238	0.108	0.012	0.003
20	Retinol binding	0.827	0.843	0.940	0.121	0.475	0.293	0.165	0.051
21	alpha-Amylases, N-term	0.935	0.953	0.737	0.037	0.630	0.851	0.007	0.000
22	beta-glycanases	0.974	0.939	0.370	0.079	0.517	0.338	0.009	0.008
23	type II chitinase	0.724	0.905	0.945	0.263	0.350	0.426	0.110	0.031
24	Alcohol/glucose dehydro	0.610	0.203	0.050	0.025	0.041	0.004	0.019	0.008
25	Rossmann-fold C-term	0.713	0.883	0.593	0.107	0.121	0.299	0.015	0.005
26	Glyceraldehyde-3-phosphate	0.791	0.537	0.062	0.004	0.315	0.102	0.009	0.002
27	Formate/glycerate	0.702	0.295	0.302	0.074	0.022	0.049	0.001	0.002
28	Lactate&malate dehydro	0.947	0.851	0.132	0.297	0.530	0.330	0.024	0.002
29	G proteins	0.867	0.568	0.428	0.051	0.378	0.330	0.007	0.000
30	Thioltransferase	0.205	0.072	0.986	0.029	0.000	0.000	0.000	0.000
31	Glutathione S-transfer	0.566	0.597	0.825	0.590	0.311	0.201	0.273	0.238
32	Fungal lipases	0.957	0.591	0.089	0.007	0.044	0.053	0.000	0.000
33	Transferrin	0.940	0.859	0.035	0.072	0.875	0.433	0.007	0.026

Table 5: Maximum and median rates of false positives for all 33 families. BLT = BLAST:SCOP-only, BLH = BLAST:SCOP+SAMT-98-homologs, ST98 = SAMT-98, and SVM = SVM-Fisher method.

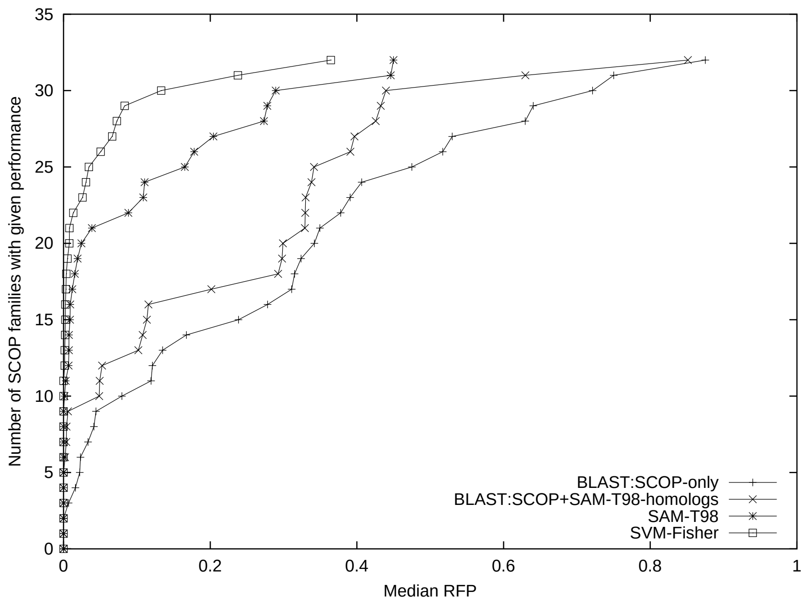

Figure 3: This plot is similar to that of Figure 2, however the performance is measured as the median RFP.

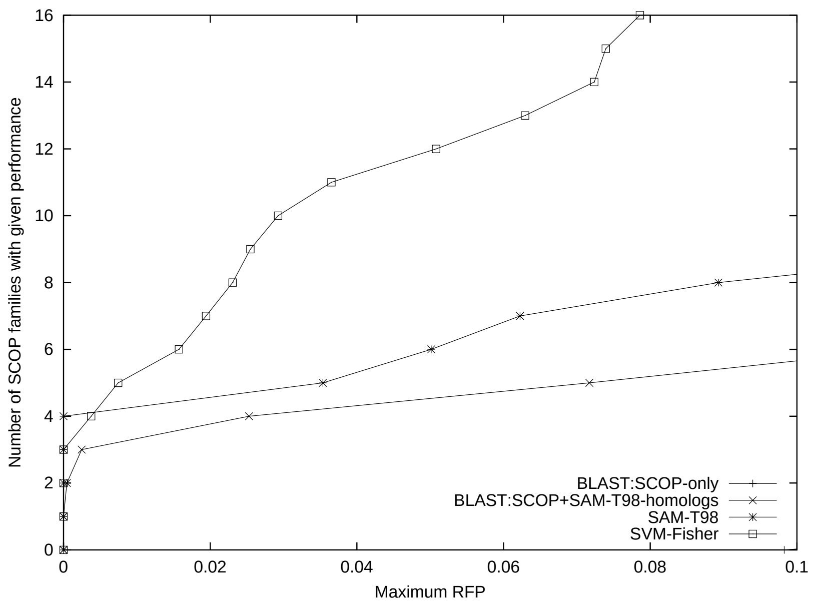

Figure 4: Detail plot of the low RFP region of Figure 2.

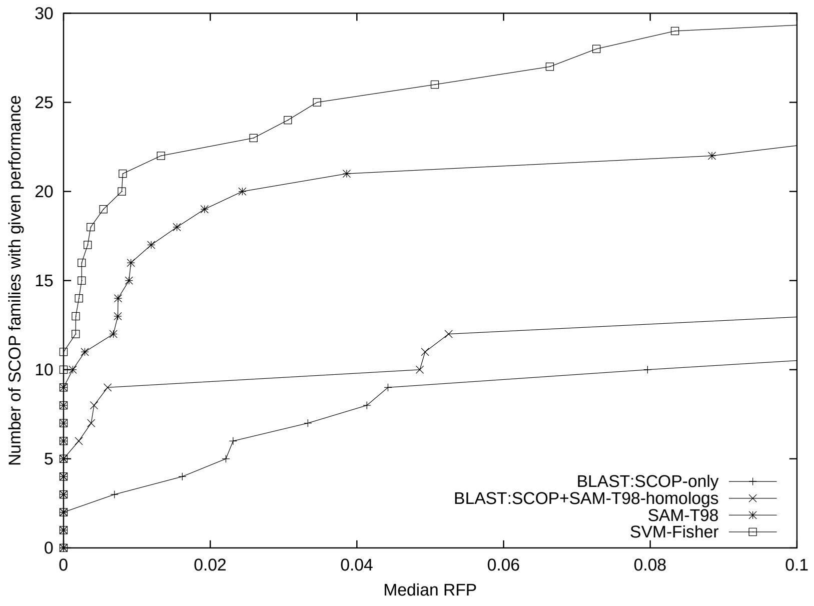

Figure 5: Detail plot of the low RFP region of Figure 3.

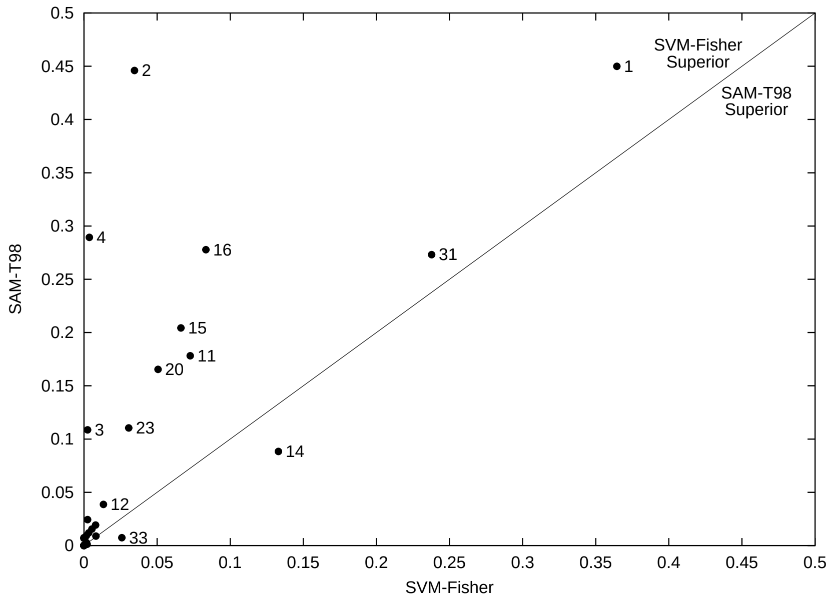

Figure 6: Family-by-family comparison of the performance of the *SAM-T98* and *SVM-Fisher* methodologies in terms of median RFP. Each family is plotted as a point (x, y) where x is median RFP for the SVM-Fisher method and y is the same thing for the *SAM-T98* method. The numbers refer to the families in table 2.

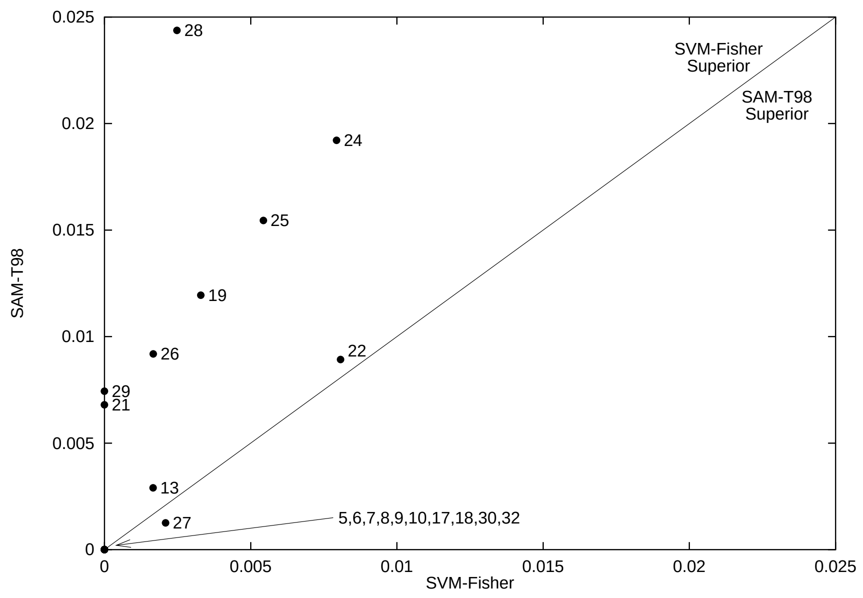

Figure 7: Detail of the better performing families in Figure 6.

from the test sequence in the case that the domain to be classified is contained in a larger test protein sequence. To simulate this situation, we appended randomly generated amino acids onto the ends of all the sequences in PDB90, creating a set of padded PDB90 sequences that all had length 1200 (the largest domain in PDB90 has length 905). The distribution of these random amino acids was determined from the overall amino acid frequencies in PDB90. The fraction of the padding that occured at the beginning of the sequence versus at the end of the sequence was determined uniformly at random as well.

We reran the experiments reported above with this padded PDB90 data set. In cases where homologs were used, these were randomly padded as well. Apart from a slight reduction of the gap between *SVM-Fisher* and the other methods, the results were on average qualitatively similar to those obtained without padding, as shown in Figures 8 and 9.

The datasets for all the experiments are available from our web site [25].

5 Discussion

We have developed a new approach to the recognition of remote protein homologies that uses a discriminative method built on top of a generative model such as an HMM. Our experiments show that this method significantly improves on previous methods for the classification of protein domains based on remote homologies.

All the methods considered in this paper combine multiple scores for each query sequence. The multiple scores arise either from several models that are available for a particular superfamily (HMM and SVM-Fisher) or because each known sequence can be scored

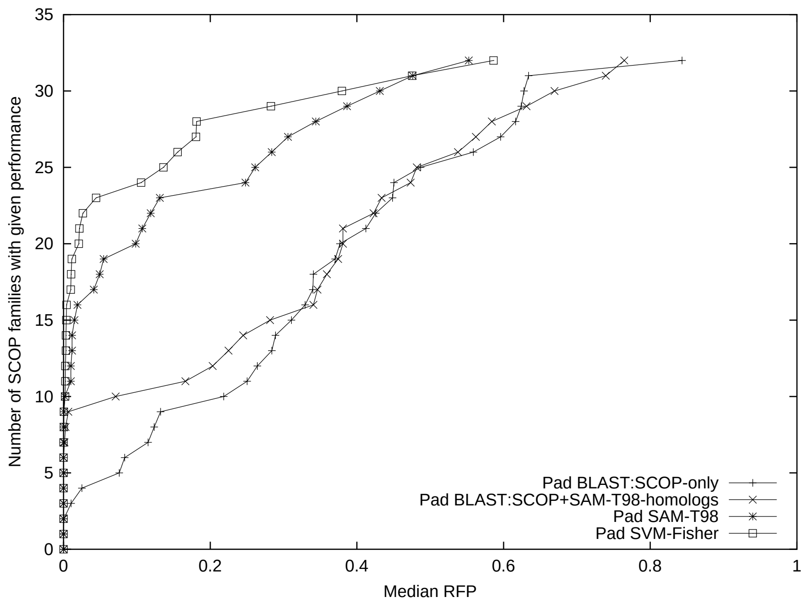

Figure 8: Comparison the overall performance for the four methods on the 33 test families at the median RFP using padded sequences.

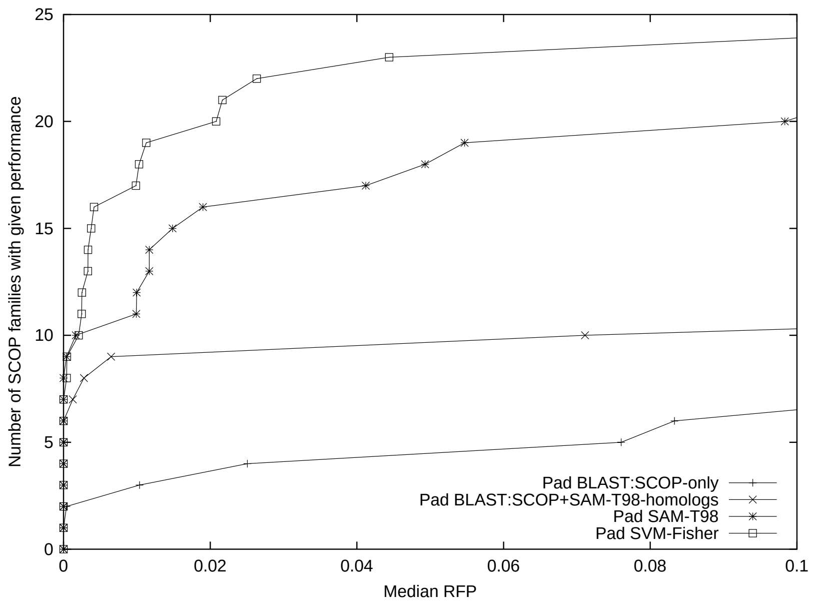

Figure 9: Detail plot of the low RFP region of Figure 8.

against the query sequence (BLAST, see [20]). The simple combination rules employed in this paper for each method are not necessarily optimal and further work needs to be done in this regard. It should be noted that while our methods for combining BLAST and HMM scores are essentially the same as those explored in [20], the relative performance of the simple generative HMM method versus the family pairwise search homology methods using BLAST is reversed in our experiments: here the HMM performs better. This is not surprising, since our tests consisted of finding very remote homologies for the most part, whereas the tests in [20] were for finding sequences that were mostly in the same family as the training sequences. Furthermore, the families in [20] were not defined by structure using SCOP, but rather by sequence similarity itself. There were also differences in the construction of the HMMs[4]. Our experimental results show, however, that it may be wise to build more powerful, discriminatively trained protein classification methods on top of HMM methods, rather than replace HMM methods with combinations of BLAST scores.

The discriminative SVM-Fisher method relies on the presence of multiple training examples from the superfamily of interest, and works best when these training sequences are not the same as those used to estimate the parameters of the underlying HMM. This presents us with an allocation problem, i.e., which sequences should be used for estimating the parameters of the HMM and which ones left for the discriminative method. This issue becomes especially important in cases where there are relatively few known sequences and homologs in the superfamily of interest. A possible solution to this problem and one that we have already successfully experimented with, concerns the use of generic protein models rather than those tuned to the particular family of interest. By generic models we mean HMMs constructed on the basis of statistical properties of short amino acid sequences that map on to structurally conserved regions in proteins [9]. Since the role of the HMM in our discriminative formalism is to provide features relevant for identifying structural relationships, the use of such generic models seems quite natural.

In the future, it will also be important to extend the method to identify multiple domains within large protein sequences. Since our experiments with artificially padded sequences were successful, we are confident that these methods can be adapted to the identification of multiple domains. However this work remains to be done. Finally, while this discriminative framework is specifically developed for identifying protein homologies, it naturally extends to other problems in biosequence analysis, such as the identification and classification of promoters, splice sites, and other features in genomic DNA.

Acknowledgments

Tommi Jaakkola acknowledges support from a DOE/Sloan Foundation postdoctoral grant 97-2-6-CMB. David Haussler and Mark Diekhans acknowledge the support of DOE grant DE-FG03-95ER62112 and NSF grant DBI-9808007. We thank Richard Hughey and Kevin Karplus for suggestions and for making their software availabile.

[4]Our tests used the *SAM-T98* method for constructing HMMs, whereas the tests in [20] used an earlier version of the HMMER system [13, 15], with the default parameters, which does not perform as well [30]; more recent and carefully tuned versions of HMMER would likely have performed better.

References

[1] S. Althschul and W. Gish. Local alignment statistics. *Methods Enzymol.*, 266:460–480, 1996.

[2] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.*, 25:3899–3402, 1997.

[3] Stephen F. Altshul, Warren Gish, Webb Miller, Myers Eugene W., and Lipman David J. Basic local alignment search tool. *JMB*, 215:403–410, 1990.

[4] TL Bailey and M. Gribskov. Methods and statistics for combining motif match scores. *Journal of Computational Biology*, 5(2):211–221, SUMMER 1998.

[5] P. Baldi, Y. Chauvin, T. Hunkapillar, and M. McClure. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the USA*, 91:1059–1063, 1994.

[6] Christian Barrett, Richard Hughey, and Kevin Karplus. Scoring hidden Markov models. *Comput. Applic. Biosci.*, 13(2):191–199, 1997.

[7] Steven Elliot Brenner. *Molecular propinquity: evolutionary and structural relationships of proteins.* PhD thesis, University of Cambridge, Cambridge, England, 1996.

[8] C. Burges. Geometry and invariance in kernel based methods. *Advances in kernel methods – Support vector learning*, 1998.

[9] C. Bystroff and D. Baker. Blind predictions of local protein structure in casp2 targets using the i-sites library. *Proteins: Structure, Function and Genetics, Suppl.*, 1:167–171, 1997.

[10] I. Dubchak, I. Muchnik, S.R. Holbrook, and S.H. Kim. Prediction of protein folding class using global description of amino acid sequence. *pnas*, 92:8700–8704, December 1995.

[11] Inna Dubchak, Ilya Muchnik, and Sung-Hou Kim. Protein folding class predictor for scop: Approach based on global descriptors. In *Proceedings, 5th International Conference on Intelligent Systems for Molecular Biology*, pages 104–108, Jun 1997.

[12] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[13] Sean Eddy. HMMER WWW site.
`http://hmmer.wustl.edu/`.

[14] Sean Eddy. Multiple alignment using hidden Markov models. In Christopher Rallings et al., editors, *ISMB-95*, pages 114–120, Menlo Park, CA, July 1995. AAAI/MIT Press.

[15] S.R. Eddy. Hidden markov models and large-scale genome analysis. *Transactions of the American Crystallographic Association*, 1997. From a talk on HMMER and Pfam given at the 1997 ACA annual meeting in St. Louis.

[16] S.R. Eddy, G. Mitchison, and R. Durbin. Maximum discrimination hidden Markov models of sequence consensus. *J. Comput. Biol.*, 2:9–23, 1995.

[17] Mark Gerstein and Michael Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the SCOP classification of proteins. *Protein Sci.*, 7:445–456, 1998.

[18] W. Gish and D. J. States. Identification of protein coding regions by database similarity search. *Nature Genetics*, 3, 1993.

[19] Michael Gribskov, Andrew D. McLachlan, and David Eisenberg. Profile analysis: Detection of distantly related proteins. *Proceedings of the National Academy of Sciences of the USA*, 84:4355–4358, July 1987.

[20] Willian N. Grundy. Family-based homology detection via pairwise sequence comparison. In *Int. Conf. Computational Molecular Biology (RECOMB-98)*, New York, 1998. ACM Press.

[21] Steven Henikoff and Jorja G. Henikoff. Position-based sequence weights. *JMB*, 243(4):574–578, November 1994.

[22] T. Hubbard, A. Murzin, S. Brenner, and C. Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Res.*, 25(1):236–9, January 1997.

[23] R. Hughey and A. Krogh. SAM: Sequence alignment and modeling software system. Technical Report UCSC-CRL-95-7, University of California, Santa Cruz, Computer Engineering, UC Santa Cruz, CA 95064, 1995.

[24] Richard Hughey and Anders Krogh. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *Comput. Applic. Biosci.*, 12(2):95–107, 1996. Information on obtaining SAM is available at `http://www.cse.ucsc.edu/research/compbio/sam.html`.

[25] T. Jaakkola, M. Diekhans, and D. Haussler. Data set for *a discriminative framework for detecting remote protein homologies*, 1998. `http://www.cse.ucsc.edu/research/compbio/discriminative/fisher-scop-data.tar.gz`.

[26] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies, 1998. Unpublished, available from `http://www.cse.ucsc.edu/research/compbio/research.html`.

[27] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, San Mateo, CA, 1998. Morgan Kauffmann Publishers.

[28] T. S. Jaakkola, M. Diekhans, and D. Haussler. UCSC discriminative models in computational biology home page.
http://www.cse.ucsc.edu/research/compbio/discriminative/.

[29] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. 1998. Available from
http://www.cse.ucsc.edu/research/ml/publications.html.

[30] Rachel Karchin and Richard Hughey. Weighting hidden Markov models for maximum discrimination. Bioinformatics to appear, 1998.

[31] Kevin Karplus. UCSC Dirichlet mixture WWW page.
http://www.cse.ucsc.edu/research/compbio/dirichlets/index.html.

[32] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[33] Kevin Karplus, Kimmen Sjölander, Christian Barrett, Melissa Cline, David Haussler, Richard Hughey, Liisa Holm, and Chris Sander. Predicting protein structure using hidden Markov models. *Proteins: Structure, Function, and Genetics*, Suppl. 1:134–139, 1997.

[34] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *JMB*, 235:1501–1531, February 1994.

[35] D. V. Lindley. Reconciliation of discrete probability distributions. *Bayesian Statistics*, 2:375–390, 1985.

[36] M. Linial, N. Linial, N. Tishby, and G. Yona. Global self-organization of all known protein sequences reveals inherent biological signatures. *jmb*, 268(2):539–556, 1997.

[37] D. J. C. MacKay. Introduction to gaussian processes. 1997. Available from http://wol.ra.phy.cam.ac.uk/mackay/.

[38] H. Mamitsuka. A learning method of hidden markov models for sequence discrimination. *Journal of Computational Biology*, 3(3):361–373, Fall 1996.

[39] A. Neuwald, J. Liu, D. Lipman, and C. E. Lawrence. Extracting protein alignment models from the sequence database. *Nucleic Acids Res.*, 25:1665–1677, 1997.

[40] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect twice as many remote homologues as pairwise methods. *JMB*, 284(4):1201–1210, 1998. Paper available at http://www.mrc-lmb.cam.ac.uk/genomes/jong/assess_paper/assess_paperNov.html.

[41] W. Pearson and D. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the USA*, 85:2444–2448, 1988.

[42] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[43] W. Taylor. Identification of protein sequence homology by consensus template alignment. *JMB*, 188:233–258, 1986.

[44] V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, 1995.

[45] WU-BLAST WWW archives.
`http://blast.wustl.edu/`.

A Computation of the Fisher scores

Here we specify in more detail the computation of the Fisher scores, the gradients with respect to the parameters of the HMM. We will only consider the gradients with respect to the emission probabilities since the Fisher score vectors used in the experiments in this paper correspond to only those gradients. For a tutorial on HMMs see [42].

We use $P(x|s,\theta) = \theta_{x|s}$ to denote the probability of emitting a residue x from the amino acid alphabet while in state $s \in \{1,\ldots,m\}$. Thus $\sum_x \theta_{x|s} = 1$ for all s. Furthermore, let $P(s'|s,\tau) = \tau_{s'|s}$ be the transition probability from the current state s to the next state s'. For simplicity we assume that there is a fixed, silent starting state s_0. The HMM defined in this way assigns a probability value to each sequence $X = [x_1,\ldots,x_n]$ given by

$$P(X|\theta,\tau) \quad = \quad \sum_{s_1,\ldots,s_n} \prod_i P(x_i|s_i,\theta)P(s_i|s_{i-1},\tau) = \sum_{s_1,\ldots,s_n} \prod_i \theta_{x_i|s_i}\tau_{s_i|s_{i-1}} \tag{16}$$

where $\{s_1,\ldots,s_n\}$ specifies a hidden state sequence and the summation is over all possible such sequences.

Our interest here is to compute the derivatives of $\log P(X|\theta,\tau)$ with respect to the emission probabilities $\theta_{x|s}$, as these are the components of the Fisher score vector U_X we use. Since the $\theta_{x|s}$ parameters are tied in the sense that they sum to one for each fixed state s, we first rewrite them in terms of a set of independent parameters

$$\theta_{x|s} = \frac{\theta_{x,s}}{\sum_{x'} \theta_{x',s}} \tag{17}$$

and assume that the current values of $\theta_{x,s}$ are set so that $\sum_{x'} \theta_{x',s} = 1$, implying that $\theta_{x,s} = \theta_{x|s}$. Consequently,

$$\frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \log P(X|\theta,\tau) = \frac{1}{P(X|\theta,\tau)} \sum_{s_1,\ldots,s_n} \frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \prod_i \frac{\theta_{x_i,s_i}}{\sum_{x'} \theta_{x',s_i}} \tau_{s_i|s_{i-1}} \tag{18}$$

Let us first consider

$$\frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \prod_i \frac{\theta_{x_i,s_i}}{\sum_{x'} \theta_{x',s_i}} \tau_{s_i|s_{i-1}} = \sum_k \left[\frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \left(\frac{\theta_{x_k,s_k}}{\sum_{x'} \theta_{x',s_i}} \right) \right] \tau_{s_k|s_{k-1}} \prod_{i \neq k} \frac{\theta_{x_i,s_i}}{\sum_{x'} \theta_{x',s_i}} \tau_{s_i|s_{i-1}}$$

$$
\begin{aligned}
&= \sum_k \left[\frac{\delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}}}{\sum_{x'} \theta_{x',s_i}} - \frac{\theta_{x_k,s_k} \sum_{x'} \delta_{s_k,\tilde{s}}\delta_{x',\tilde{x}}}{(\sum_{x'} \theta_{x',s_i})^2} \right] \tau_{s_k|s_{k-1}} \prod_{i \neq k} \frac{\theta_{x_i,s_i}}{\sum_{x'} \theta_{x',s_i}} \tau_{s_i|s_{i-1}} \\
&= \sum_k \left[\delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}} - \theta_{x_k|s_k}\delta_{s_k,\tilde{s}} \right] \tau_{s_k|s_{k-1}} \prod_{i \neq k} \theta_{x_i|s_i}\tau_{s_i|s_{i-1}} \\
&= \sum_k \left[\frac{\delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}}}{\theta_{x_k|s_k}} - \delta_{s_k,\tilde{s}} \right] \prod_i \theta_{x_i|s_i}\tau_{s_i|s_{i-1}} \\
&= \sum_k \left[\frac{\delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}}}{\theta_{\tilde{x}|\tilde{s}}} - \delta_{s_k,\tilde{s}} \right] \prod_i \theta_{x_i|s_i}\tau_{s_i|s_{i-1}}
\end{aligned}
$$

where the first two equalities follow from the chain rule and the third one from our previous setting: $\theta_{x,s} = \theta_{x|s}$ and so $\sum_{x'} \theta_{x',s} = 1$. Inserting this result back into eq (18) and flipping the order of the summations we get

$$
\begin{aligned}
\frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \log P(X|\theta,\tau) &= \sum_k \frac{1}{P(X|\theta,\tau)} \sum_{s_1,\ldots,s_n} \left[\frac{\delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}}}{\theta_{\tilde{x}|\tilde{s}}} - \delta_{s_k,\tilde{s}} \right] \prod_i \theta_{x_i|s_i}\tau_{s_i|s_{i-1}} \\
&= \frac{1}{\theta_{\tilde{x}|\tilde{s}}} \sum_k E\{\, \delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}} \,|X,\theta,\tau\,\} - \sum_k E\{\, \delta_{s_k,\tilde{s}} \,|X,\theta,\tau\,\} \\
&= \frac{\xi(\tilde{x},\tilde{s})}{\theta_{\tilde{x}|\tilde{s}}} - \xi(\tilde{s}) \qquad (19)
\end{aligned}
$$

where $\xi(\tilde{s}) = \sum_{x'} \xi(x',\tilde{s})$ and the posterior expectations

$$
\sum_k E\{\, \delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}} \,|X,\theta,\tau\,\} = \xi(\tilde{x},\tilde{s}) \qquad (20)
$$

are the sufficient statistics for the emission probabilities. They are obtained directly and efficiently via the standard forward-backward algorithm [42]. $\xi(\tilde{x},\tilde{s})$ can be also seen as the expected posterior frequency of visiting state $\tilde{s}$ and generating residue $\tilde{x}$ from that state.

To summarize so far, the Fisher score vector U_X relative to the emission probabilities is a vector whose components are indexed by (x,s) and the corresponding values given by eq (19). We emphasize that $\xi(\tilde{x},\tilde{s})$ come from the standard forward-backward algorithm and thus the cost of computing the Fisher score vector is of the same order as simply evaluating $P(X|\theta,\tau)$.

A.1 Fisher score from a mixture decomposition

The Fisher score vectors used in the experiments in this paper do not come directly from the emission probabilities but from a mixture decomposition of the emission probabilities. It is often advantageous to decompose the emission probabilities into a mixture

$$
\theta_{x|s} = \sum_l c_{l|s}\theta^{(l)}_{x|s} \qquad (21)
$$

where $\sum_l c_{l|s} = 1$ and each component $\theta^{(l)}_{x|s}$ concentrates on a particular subclass of residues such as the hydrophobic residues. Computing the Fisher score vector on the basis of the mixture coefficients $c_{l|s}$ allows us to abstract away from the residue identity[5]. The components $\theta_{x|s}^{(l)}$ in our case were extracted from a 9-component Dirichlet mixture developed by Kevin Karplus [31]. To find the components of this new Fisher score vector we can proceed analogously to the previous derivation. We get

$$\frac{\partial}{\partial c_{l,s}} \log P(X|\theta, c, \tau) = \sum_x \xi(x, s) \left[\frac{\theta_{x|s}^{(l)}}{\theta_{x|s}} - 1 \right] \tag{22}$$

for all values of $l \in \{1, ..., 9\}$ and $s \in \{1, \ldots, m\}$. A rather surprising and a very convenient property of these derivatives is that we do not need to know explicitly the values of the coefficients $c_{l|s}$, only the values $\theta_{x|s}$ and those of the component probabilities $\theta_{x|s}^{(l)}$. This holds whenever $\theta_{x|s}$ can be decomposed in terms of $\theta_{x|s}^{(l)}$ as in Equation (21).

[5]By adjusting the components $\theta_{x|s}^{(l)}$, the information about the residue identity can be preserved if necessary.