Fast Model-based Protein Homology Discovery without Alignment

Mani Manavalan

Technical Project Manager, Larsen & Toubro Infotech (LTI), Mumbai, **INDIA**

ABSTRACT

The need for quick gene categorization tools is growing as more genomes are sequenced. To evaluate a newly sequenced genome, the genes must first be identified and translated into amino acid sequences, which are then categorized into structural or functional classes. Protein homology detection using sequence alignment algorithms is the most effective way for protein categorization. Discriminative approaches such as support vector machines (SVMs) and position-specific scoring matrices (PSSM) derived from PSI-BLAST have recently been used to improve alignment algorithms. However, if a fresh sequence is being aligned, alignment algorithms take time. must be compared to a large number of previously published sequences — the same is true for SVMs. Building a PSSM for the PSSM is even more time-consuming than a fresh order It would take roughly 25 hours to implement the best-performing approaches to classify the sequences on today's computers. Describing a novel genome (20, 000 genes) as belonging to one single organism. There are hundreds of classes to choose from, though. Another flaw with alignment algorithms is that they do not construct a model of the positive class, instead of measuring the mutual distance between sequences or profiles. Only multiple alignments and hidden Markov models are common classification approaches for creating a positive class model, but they have poor classification performance. A model's advantage is that it may be evaluated for chemical features that are shared by all members of the class to get fresh insights into protein function and structure. We used LSTM to solve a well-known remote protein homology detection benchmark, in which a protein must be categorized as a member of the SCOP superfamily. LSTM achieves state-of-the-art classification performance while being significantly faster than other algorithms with similar classification performance. LSTM is five orders of magnitude quicker than the quickest SVM-based approaches and two orders of magnitude faster than methods that perform somewhat better in classification (which, however, have lower classification performance than LSTM). We applied

Source of Support: Nil
Conflict of Interest: **Non d**eclared

LSTM to PROSITE classes and analyzed the derived patterns to test the modeling capabilities of the algorithm. Because it does not require established similarity metrics like BLOSUM or PAM matrices, LSTM is complementary to alignment-based techniques. The PROSITE motif was retrieved by LSTM in 8 out of 15 classes. In the remaining seven examples, alternative motifs are developed that, on average, outperform the PROSITE motifs in categorization.

Key words: Protein homology discovery, Support vector machines (SVMs), Homology detection, LSTM Network

INTRODUCTION

The amino acid sequences derived from whole-genome sequencing are becoming increasingly important in our post-genomic era. The most successful strategy for determining a protein's function or 3D structure from its amino acid sequence is to look for similarities with other proteins. To measure the similarity, or homology, of proteins, pairwise alignment methods such as the Smith-Waterman algorithm (Smith and Waterman, 1981) or its approximations such as FASTA (Pearson and Lipman, 1988) or BLAST/PSI-BLAST (Altschul et al., 1990) are critical.

Discriminative algorithms such as the support vector machine (SVM) have recently improved these alignment-based methods (Vapnik, 2000). Protein homology identification approaches based on SVMs are based on kernels that compute similarities between sequences using alignment methods or sequence identities. The pairwise SVM method (Liao and Noble, 2002), the Fisher-kernel (Jaakkola et al., 1999), the mismatch kernel (Leslie et al., 2004a, b), and related kernels (Dong et al., 2006; Lingner and Meinicke, 2006), the Smith-Waterman, and the local alignment kernel (Vert et al., 2004) are all examples of these methods. Instead of the original sequences, these methods have recently been improved by employing profiles and position-specific scoring matrices (PSSM) (Rangwala and Karypis, 2005). (Vert et al., 2004). Waterman and the local alignment kernel.

Sequence similarity (i.e. alignment-based) techniques, on the other hand, still have several flaws. Despite their high performance, they take far too long to compute for widespread application (Vinga and Almedia, 2003). For example, the best-performing techniques would take around one month to identify the protein sequences identified in a newly sequenced genome if the genes only belonged to one class. These methods are infeasible for actual use for hundreds of classes due to their time complexity.

Genetic recombination and genetic shuffling still pose issues for alignment approaches (Vinga and Almeida, 2003). Another flaw with similarity-based techniques is that they aren't based on models. It's difficult to evaluate categorization results without a model in terms of relevant patterns or chemical properties (function, stability, folding). Model-based techniques, on the other hand, can discover structural biochemistry's key parts (Donepudi, 2014).

To alleviate the shortcomings of similarity-based techniques, we propose using recurrent neural networks (RNNs). RNNs have been used to predict protein secondary structure (Baldi et al., 1999) and sheet pairing (Cheng and Baldi, 2005) with great success.

RNNs (a) can extract dependencies between subsequences, whereas similarity-based approaches cannot. A subsequence AB, for example, may only be suggestive if it is followed by a subsequence CD later in the sequence; (b) can extract correlations within subsequences. Both AB and CD may be suggestive of the class (motif [AC]–[BD]), while AD may not (AD is a negative pattern). (c) has the ability to derive global sequence properties (hydrophobicity or atomic weight), and (d) can extract amino acid dependencies that span a large distance in the amino acid sequence.

Thus, RNNs can cope with interactions between distinct profiles, such as when a detected pattern inhibits or reinforces the storage of another pattern. They can calculate non-linear functions of suggestive patterns in the sequence, making them a potentially useful tool for analyzing amino acid sequences.

RNNs, on the other hand, have some drawbacks:

- they require a big enough training set for model selection;
- before employing them, an architecture must be chosen;
- the training phase may be computationally costly; and
- they are unable to discover similarities between negative cases.

Pointwise similarity metrics (identity, BLOSUM, or PAM matrices) are not a priori fixed when employing RNNs for homology identification, which is a novel feature. RNNs can train their similarity measure that is tailored to a specific classification task, combining patterns with sequence statistics such as hydrophobicity.

Objectives of the Study

This study aimed at fast model-based protein homology discovery without alignment. To evaluate the objective, a newly sequenced genome, the genes must first be identified and translated into amino acid sequences, which are then categorized into structural or functional classes. Protein homology detection using sequence alignment algorithms is the most effective way for protein categorization.

LITERATURE REVIEW

The LSTM Network

Model architecture: In Figure 1, we propose the 'Long Short-Term Memory (LSTM) recurrent net architecture (Hochreiter and Schmidhuber, 1997) for protein homology detection. The network processes sequences element by element until they are eventually classified at the end. The network input is gathered from a window region surrounding the current position at each step.

LSTM includes memory cells, which are specially built memory sub-architectures that may store information such as the occurrence of a pattern from previously scanned regions without causing data loss (see Fig. 2 for more details on memory cells). The stored data is utilized to suppress or reinforce other patterns in the remaining sequence, and it is crucial for predicting the class at the end of the series (Bynagari, 2014).

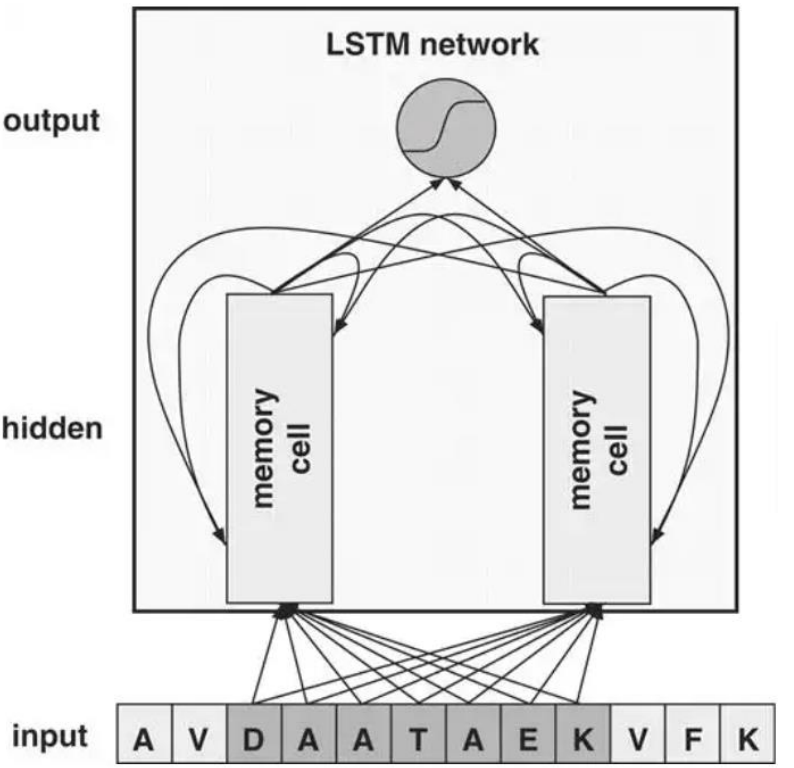

Figure 1: Recommended LSTM network with three layers: input layer

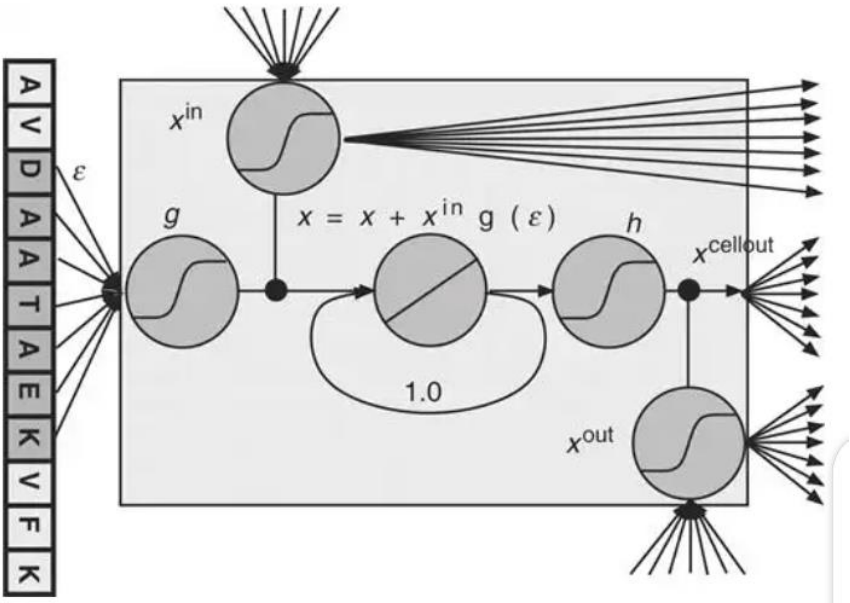

Figure 2: The LSTM memory cell

The importance of memory: Because information, such as a pattern, that is indicative of a protein class can be found at any point in the sequence, lengthy gaps between relevant sequence regions may exist. As a result, relevant data must be saved until it is required for classification. However, because of the exponential decay of previously viewed information with processing time, classical RNNs fail at this task, resulting in a 'vanishing gradient' problem (Hochreiter, 1991; Hochreiter et al., 2000).

RNNs require a properly constructed memory to store information until it is needed, to be a competitive approach for protein categorization. Note that the key information in the sequence was near to the prediction location in the Baldi et al., (1999) and Cheng and Baldi (2005) tasks, so such memories were not required.

The LSTM (Hochreiter and Schmidthuber, 1997) is an RNN with a specially built memory sub-architecture called a "memory cell" for storing information, making it ideal for protein classification. Volume-conserving mappings built through a linear unit with a self-recurrent connection to weight one are used to create memory units with non-decaying information. In an LSTM network, Figure 2 depicts such a "memory cell." The volume-conserving mapping is carried out by the unit at the box's center.

An 'input gating' or attention unit (Figure 2, unit labeled 'xin') controls the input to the memory cell, blocking class-irrelevant information and storing only class-relevant information in memory. The activation of attention units is limited to the numbers 0 and 2, implying that the incoming data (t) is squished by a sigmoid function g. The activation function of the memory cell is determined by

$$x(t+1) = x(t) + x^{in}(t).g\big(\uparrow\in(t)\big) \quad \ldots\ldots\ldots 1$$

Where $\in(t)$ denotes local sequence information (see Equation (3)). The sigmoid function h (Fig. 2, unit designated as 'h') limits the output of the memory cell (Fig. 2, center) to a value between -1 and 1. Memory readout is controlled by an 'output gate' (Figure 2, unit labeled 'x^{out}'). The output of the cell, '$x^{cellout}$' is then calculated as follows:

$$x^{cellout}(t+1) = x^{out}(t).h\big(x(t+1)\big) = x^{out}(t).h(x(t) + x^{in}(t).g(\in(t))) \ldots\ldots\ldots 2$$

In theory, memory cells can be incorporated into any neural network architecture. The LSTM recurrent network structure, as shown in Figure 1, is used here.

Learning of profiles: To extract class-indicative information, we use profiles (Henikoff and Henikoff, 1994) as inputs to the LSTM network, which allows the LSTM to learn the profiles automatically by error propagation. Because they offer a weighted total of the amino acids inside the window, the input weights to a memory cell build a profile. We encode the amino acids in the input using a local encoding, which means that each amino acid is represented as a 20D vector with zeros except for one location, which has one. The input is a 20 x l matrix Y with components Y_{rj} if the profile has length l. The LSTM network is shown in Figure 3 with a profile as an input. Each memory cell has its input profile and the ith profile input $\in(t)$ to the LSTM network at sequence point t is computed as follows:

$$\in_i(t) = \sum_{(t,j)=(1,1)}^{(l,20)} w_{rj}^{i}, Y_{rj},\ Y_{rj} = \begin{cases} 1 & s_{i+r} = AA_j \\ 0 & otherwise \end{cases} \ldots\ldots\ldots 3$$

where s_t is the input sequence's ith element and AAj is the jth amino acid. w_{rj}^{i} is an element of the ith profile matrix, but it's also an LSTM network's input weight, allowing it to learn the profile automatically.

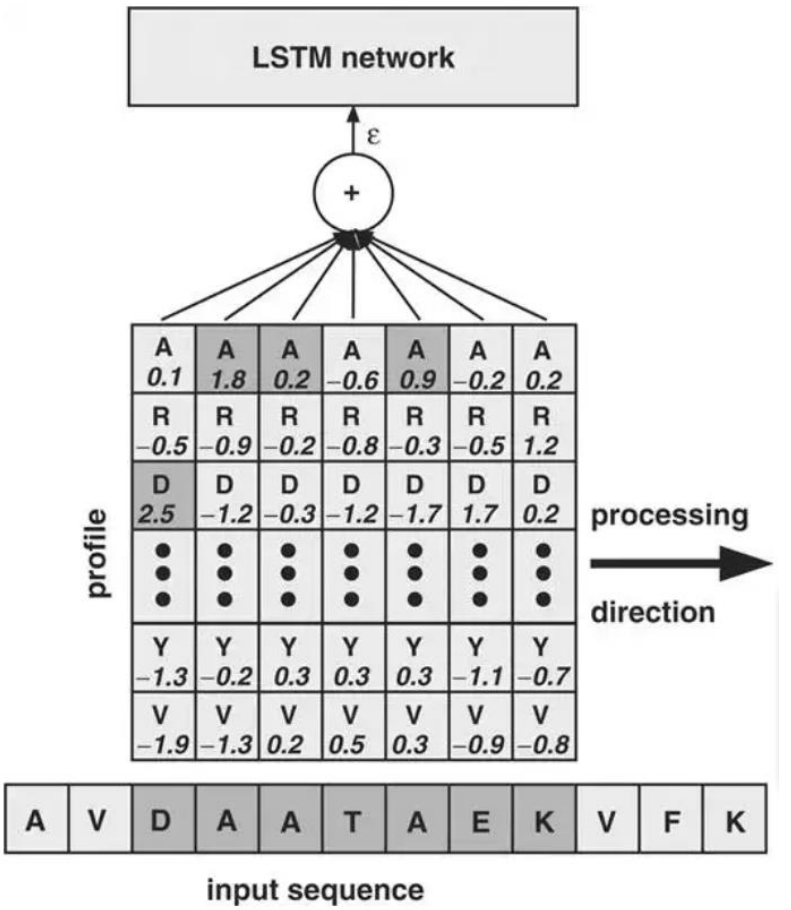

Figure 3: A profile as input to the LSTM network

Model characteristics: The original LSTM design is modified (for example, with two memory cells) for protein classification and learning profiles. The profiles, which are produced by all weights from the input to the memory cell, are the memory cells' only source of input. There are no weights from the input to any other units, as opposed to the original LSTM architecture. Other attention units, output gates, and memory cells provide input to the attention unit and output gate, however, the output unit only receives input from memory cells (see Figure 2). The LSTM architecture is learned using Hochreiter and Schmidhuber's (1997) LSTM learning algorithm, which is a gradient-descent method.

Computational complexity

For classifying a new sequence of length L, the LSTM method has a temporal complexity of $O(L^2)$. This complexity must be weighed against the complexity of the most efficient profile-based approaches available in the literature. The time required to compute the

profile of a new sequence is a major factor in Rangwala and Karypis' (2005) technique. To compute the profile, the NR database, which contains over 3 million entries, must be read, and multiple alignments followed by a profile-profile alignment must be constructed.

Alignment approaches, such as profile-profile alignments, have the complexity of OL^2. As a result, alignment-based kernel approaches have an $O(N_{SV}L^2)$complexity, where NSV is the number of support vectors. In Lingner and Meinicke, the time complexity has been decreased to OL^2 due to an explicit representation of the linear classifier (the weight vector) in feature space (2006). For L>100, LSTM is potentially more than two orders of magnitude quicker than the quickest SVM-based technique, as demonstrated experimentally in the following experiments.

METHODS

Three sets of experiments were carried out. On a benchmark dataset from the SCOP database, we evaluate the performance and time complexity of LSTM for remote homology identification and compare LSTM to several state-of-the-art techniques (Murzin et al., 1995). On a SCOP fold prediction challenge from Ding and Dubchak, we compare LSTM to several machine-learning algorithms that extract features from sequences in the second batch of experiments (2001).

On a SCOP fold prediction challenge from Ding and Dubchak, we compare LSTM to several machine-learning algorithms that extract features from sequences in the second batch of experiments (2001).

In the final series of tests, we evaluate LSTM's modeling capabilities and see if it can automatically extract relevant motifs for protein classification. The PROSITE database (Bairoch, 1995) is used to apply LSTM, and the motifs extracted by LSTM are compared to the PROSITE motifs (Sigrist et al., 2002).

SCOP superfamilies are used to detect remote homology

Data: We used the widely used benchmark dataset for remote homology identification from Liao and Noble (Liao and Noble, 2002), which can be found at http://www.cs.columbia.edu/compbio/svm-pairwise. There are 54 superfamily recognition jobs in the dataset. The training set's positive examples are drawn from one superfamily for each job, with one family being withheld. The objective is to find examples from the withheld family. Negative training examples are selected from beyond the family's fold.

Experimental Approach: The following approaches are used to benchmark: (a) PSI-BLAST (Altschul et al., 1997), (b) family pairwise search (FPS, Grundy, 1998), and (c) SAM-T98 (Karplus et al., 1998; Park et al., 1998). SVMs that take into account negative examples can improve these alignment-based approaches. We make the following comparisons: SVMs using the mismatch-kernel (d) the Fisher kernel SVM (Jaakkola et al., 1999), (e) SVMs using the Fisher kernel SVM (Jaakkola et al., 1999). (Leslie et al., 2004a, b); (f) the SVMpairwise (Liao and Noble, 2002)—feature vector is the Swith-Waterman alignment score to all other training sequences, (g) SW-kernel (Vert et al., 2004)–SW-pairwise scores are considered as kernel matrix, (h) the local alignment (LA) kernel (Vert et al., 2004) based on the BLOSUM matrix, I the oligomer-based distance SVM (Lingner and Meinicke, 2006). The comparison approaches based on PSSMs or profiles, which we summarize under (j),

include 'HMMSTR' from (Hou et al., 2004), 'Mismatch-PSSM', the mismatch kernel with PSSM (Kuang et al., 2005), and 'SW-PSSM', the SW-kernel with PSSM (Kuang et al., 2005). (Rangwala et al., 2005). We incorporated the results of the best parameters as reported in the related articles (notice that because hyperparameter selection was ignored, the results may be exaggerated). (The comparison approaches based on PSSMs or profiles, which we summarize under (j), include 'HMMSTR' from (Hou et al., 2004), 'Mismatch-PSSM', the mismatch kernel with PSSM (Kuang et al., 2005), and 'SW-PSSM', the SW-kernel with PSSM (Kuang et al., 2005). (Rangwala et al., 2005). We incorporated the results of the best parameters as reported in the related articles (notice that because hyperparameter selection was ignored, the results may be exaggerated). Finally, (k) we present the outcomes of our new LSTM approach.

Details on how to use LSTM

We chose the model's hyperparameters (number of memory cells, window size, learning rate, initialization, and so on) on a separate dataset from Gille et al. (2003), which is available through the program package STRAP http://www.3d-alignment.eu. The collection contains 500 proteasome sequences and 7400 negatives from the Protein Data Bank (PDB).

Architecture: There are thirteen memory cells in the architecture, and the profile length is eleven. All units are biased and have sigmoid activation functions in [0, 1], except for g and h which are sigmoid in [0, 2] and [-1, 1], respectively.

Initialization: Memory cell input bias: from –2.0 to –5.0 (descending every second cell by 0.5), memory cell output bias: –1.0, memory cell output to output unit weight: 1.0 and –1.0 alternating, yielding 7 positive and 6 negative memory cells. The rest of the weights are set to zero.

Output coding: 0.8 (positive class) and 0.2 (negative class) output coding (negative class). Learning parameters: 500 epochs of the learning period, 14 0:01

Learning rate: Positive examples are cloned until their number equals or exceeds the number of negative examples by at least a factor of five.

Training set: Running PSI-BLAST with five iterations and default parameters against the NR database for each positive example and picking instances with e-values smaller than 10.0 in the last iteration expands the positive training set. It's worth noting that the PSI-BLAST run was just used to expand the training set [see disadvantage (1) of RNNs at the end of the Introduction], and it wasn't required for classifying fresh samples.

Evaluation: The area under the receiver operating characteristics curve was used to assess the quality of a test set example ranking (ROC). The approaches were assessed using 54 ROC scores ranging from 0.5 (random guessing) to 1.0 (perfect prediction) (perfect prediction). The area under the ROC50, which is the area under the ROC up to 50 false positives, was also utilized as a more precise quality metric. The ROC's false positive rate is essentially rescaled by ROC50.

Test durations: On an Opteron 165 1.8 GHz processor, computing times were measured. Using Vert et al (2004) software, the time for the LA-kernel was calculated. We used the LA-kernel to measure a 165-base-pair test sequence (the average length of proteins in the dataset). Because Lingner and Meinicke (2006) claim that the oligomer technique is 1000

times faster than the LA-kernel, the time for the oligomer method can be calculated from the LA-kernel time. Because BLAST is faster than the exact Smith-Waterman algorithm, the time for the SW kernel was constrained by BLAST (NCBI bl2seq 2.2.14 from http://www.ncbi.nlm.nih.gov/Ftp/) for pairings of proteins. According to Leslie et al., we utilized software from http://www1.cs.columbia.edu/compbio/string-kernels/ to measure the time of the mismatch kernel (2004a). The PSI-BLAST, SAM-T98, and Fisher-kernel test timings were calculated using Tarnas and Hughey's CPU values (1998). Rangwala and Karypis' software was used to calculate the test times for 'SW-PSSM' (2005). The time it took to compute a profile for one sequence was 90 seconds, giving a lower bound of 500 hours for 20 000 test instances for profile approaches. SVM methods with a profile and LSTM take 110 and 117 hours to train, respectively. PSI-BLAST runs (105 h) generate the profiles SVM or expand the positives LSTM, resulting in very long training times.

RESULTS AND DISCUSSION

For all methods utilized in our benchmark, Table 1 presents the average values of the area under ROC, the area under ROC50, and the time complexity. Except for LSTM, the classification findings are from Hou et al. (2004), Kung et al. (2005), Liao and Noble, (2002), Lingner and Meinicke, (2006), Rangwala and Karypis, (2005), and Vert et al (2004), The time measurements were taken from Linger and Meinicker (2006), Madera and Gough (2002), and Tarnas and Hughey (1998), and were done with the string kernel software from http://www1.cs.columbia.edu/compbio/string-kernels/ and the BLAST algorithms from http://www.ncbi.nlm.nih.gov/ Ftp/. Figures 4 and 5 illustrate how many classification tasks each technique achieved a given ROC value.

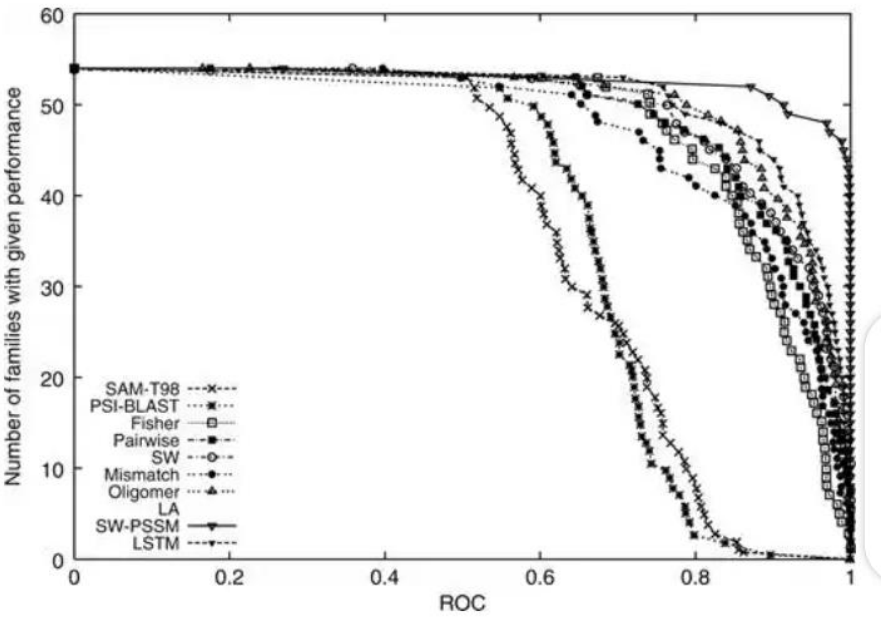

Figure 4: Comparison of homology detection methods for the SCOP 1.53 benchmark dataset.

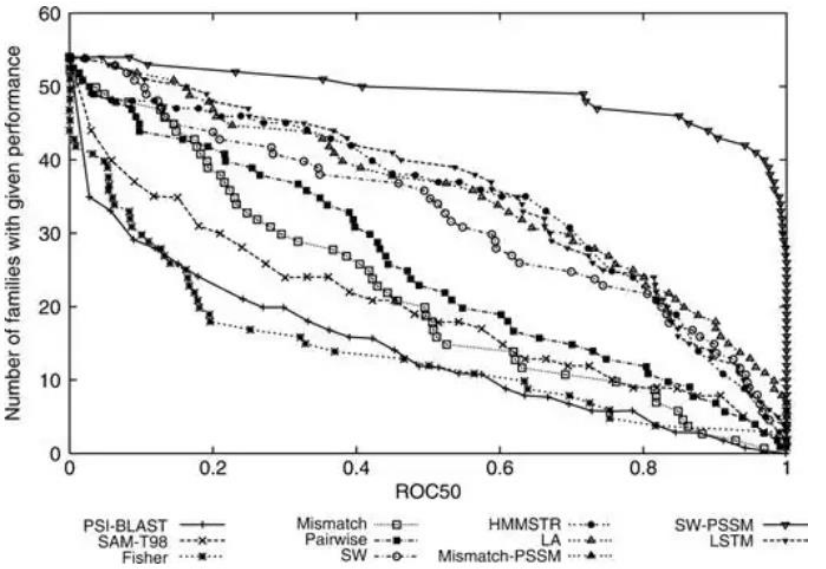

Figure 5: Comparison of homology detection methods for the SCOP 1.53 benchmark dataset

The table and figures reveal that only a few of the similarity-based approaches using profiles perform better than LSTM in terms of ROC values. We examined the following LSTM misclassifications that were corrected using similarity-based methods: When employing similarity-based approaches, LSTM false positives had a high resemblance to positive training instances, but this similarity was outvoted by similarity to a single negative training example (cf. RNN's disadvantage (4) after the Introduction).

Table 1: Results of remote homology detection on the SCOP benchmark database

Techniques	M	P	V	S	ROC	ROC50	Duration (s)
PSI- BLAST	-	-	-	-	0.692	0.263	5.4
PS	-	-	-	-	0.595	-	6790
AM-T98	+	-	-	-	0.673	0.373	199
Fisher	-	-	-	+	0.886	0.249	>199
Mismatch	-	-	-	+	0.871	0.390	370
Pairwise	-	-	-	+	0.895	0.463	>690
SW	-	-	-	+	0.915	0.584	>460
LA	-	-	-	+	0.922	0.660	33000
Oligomer	-	-	-	+	0.918	0.507	2000
HMMSTR	-	+	+	+	-	0.639	>30000
Mismatch-PSSM	-	+	+	+	0.970	0.793	>30000
SW-PSSM	-	+	+	+	0.981	0.903	>37200
LSTM	+	-	+	-	0.931	0.651	19

The first column gives the method. The columns 2–5 denote whether the method belongs to a class ('+') or not ('-'), where the classes are 'M' for model-based, 'P' for profile input, 'V' for semi-supervised, and 'S' for SVM.

However, Lingner and Meinicke (2006) discovered that the best performing techniques (Dong et al., 2006, Rangwala and Karypis, 2005) may have been optimized for the test data, resulting in overoptimistic findings. Because LSTM performs similarly to the LA-kernel approach (Vert et al., 2004), it is one of the better approaches for scanning the primary sequence. When it comes to classification accuracy, all model-based approaches fall short of the LSTM approach.

The results for time complexity show that only PSI-BLAST is faster than LSTM in terms of computational time. LSTM outperforms SAMT98 by an order of magnitude. It's worth noting that LSTM just scans the sequence and doesn't have to align it to a profile. The fastest SVM-based technique is 2 orders of magnitude slower than LSTM. To process a new sequence, methods that perform better than LSTM require 5 orders of magnitude more CPU time.

Test for PFAM (Polyfluoroalkylamine)

Because SCOP only covers 15% of the PFAM families, we also tried LSTM, PSI-BLAST, and SAM 3.5 on the PFAM database. PSI-BLAST and SAM were improved by utilizing the best score from all positive training sequences as the positive class score (like the FPS method). To develop a profile for PSI-BLAST, we used the NR database twice. SAM built an alignment with the help of the NR database. W0.5 was used to construct the HMM, which was then calibrated. Both SAM and PSI-BLAST construct a model based on multiple alignments (a profile and an HMM, respectively), which is also the foundation of the PFAM classification.

These procedures are likely exaggerated because the dataset's underlying model is the same as the model created by the methods we're comparing. To label the PFAM sequences, we looked for PFAM families in the SCOP database that had a distinct superfamily. The SCOP superfamily to which these PFAM families belong is used to name them. Only PFAM families in the SCOP test set are labeled positive for remote homology detection. Table 2 shows the results for the PFAM dataset. When it comes to testing novel sequences, LSTM has the greatest results and is the fastest.

Table 2: Results on the PFAM database 'ROC all' denotes the area under the curve for all test examples

Technique	ROC all	ROC remote	Duration (s)
PSI-BLAST	0.70	0.68	3000
SAM 3.5	0.84	0.75	72000
LSTM	0.87	0.78	1620

SCOP fold prediction

We ran another benchmark using the Ding and Dubchak dataset to compare our method to other machine-learning algorithms that describe amino acid sequences using extracted features (2001). We consider the 'one-versus-other' job, in which a binary classifier is built for each class, with the positive class members in the positive class and the negative class members in the negative class. The results of the Ding and Dubchak (2001) dataset are presented in Table 3. In terms of Q-value, LSTM surpasses the other approaches.

Table 3: Results on the dataset from Ding and Dubchak (2001) for different machine-learning approaches

Technique	Q
Neural network	41.7
LSTM	51.6
Support vector machine	45.1

Motif extraction: PROSITE database (see Figure 6)

Table 4 shows the average test results from the Swiss Prot database for 10 classes. The table demonstrates that LSTM has the best sensitivity with just a minor disadvantage in specificity, resulting in a significant reduction in the balancing error in favor of LSTM. When LSTM motifs are employed instead of PROSITE motifs, classification performance is still greater (in the sense that it was before) in 8 of the 10 classes. Even when LSTM discovers distinct patterns, performance improves in 5 of the 7 classes. This demonstrates that by employing LSTM as a technique for 'explorative data analysis,' new suggestive patterns can be extracted.

Table 4: Results of PROSITE protein classification tested on the Swiss Prot database

Technique/motif	Sensitivity	Specificity	Balanced Error
PROSITE	85.90 (15.61)	99.93 (0.14)	7.07 (7.78)
LSTM	98.23 (3.54)	99.78 (0.18)	0.98 (1.81)
Motif	86.81 (9.1)	99.92 (0.15)	6.62 (4.58)

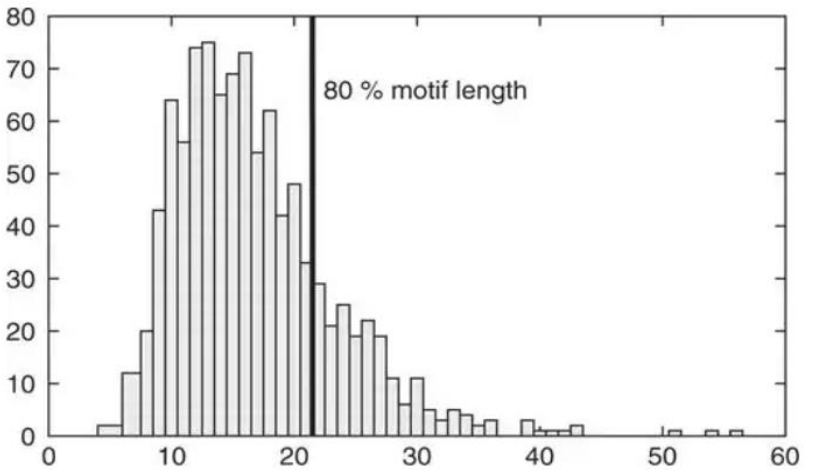

Figure 6: Histogram of motif length in the PROSITE database

Table 5 compares PROSITE and LSTM motifs, revealing that LSTM automatically recovers a motif that is comparable to PROSITE in 8 out of 15 occurrences. Alternative and, given the classification performance, even more, suggestive motifs are identified in seven situations. These findings show that LSTM models can extract additional information, which could lead to new insights into protein function or structure.

Table 5: The motifs found by LSTM compared to the PROSITE motifs

PROSITE	4FE4S_ FERREDOXIN (385)
	PROSITE C-x(2)-C-x(2)-C-x(3)-C-[PEG]
LSTM (≈)	C-x(2)-C-x(2)-C-x(2)-{C}-[AC]-[PEG]
	AA_ TRNA_ LIGASE_ I (913)
PROSITE	P-x(0,2)-[GSTAN]-[DENQGAPK]-x-[LIVMFP]-
	[HT]-[LIVMYAC]-G-[HNTG]-[LIVMFYSTAGPC]
LSTM (≈)	[ACFILMPV]-H-[ILMVFY]-G-[HGNT]-{DEHNPQR}-
	{DEP}-{CHKRY}-{DER}-[AILMSTVY]-{EGHPW}
	ATPASE_ ALPHA_ BETA (376)
PROSITE	P-[SAP]-[LIV]-[DNH]-x(3)-S-x-S
LSTM (≈)	[ILV]-G-[CELR]-x(0,2)-[DGNV]-x-[ILRSV]-[AGS]-
	[DEKNQRV]-[AEGPV]-[DILMV]-[ADRT]-[DEGLNV]
	CITRATE_ SYNTHASE (76)
PROSITE	G-[FYA]-[GA]-H-x-[IV]-x(1,2)-[RKT]-x(2)-D-[PS]-R
LSTM (≈)	[ASG]-R-x(2)-G-W-x-A-H-x(2)-E OR
	[ASG]-[QK]-x-P-x-[LIVM]-[AV]-A-x(2)-Y
	CYTOCHROME_ C (388)
PROSITE	C-{CPWHF}-{CPWR}-C-H-{CFYW}
LSTM (≈)	C-{CFP}-{CRWY}-C-H-{CFHWY}
	DEHYDROQUINASE_ I (44)
PROSITE	D-[LIVM]-[DE]-[LIVMN]-x(18,20)-[LIVM](2)-x-
	[SC]-[NHY]-H-[DN]
LSTM (≈)	D-[LIVA]-[LIVAY]-E-[LIVFW]-R-[LIVA]-D
	HISTONE_ H3_ 1 (44)
PROSITE	K-A-P-R-K-Q-L
LSTM (≈)	T-G-x-K-A-P-R
	INSULIN (194)
PROSITE	C-C-{P}-x(2)-C-[STDNEKPI]-x(3)-[LIVMFS]-x(3)-C
LSTM (≈)	C-C-{CDW}-x(2)-C-[DEIKNPSTB]-x(3)-[FILMV]-x(3)-C
	INVOLUCRIN (14)
PROSITE	M-S-[QH]-Q-x-T-[LV]-P-V-T-[LV]
LSTM (≈)	L-E-L-P-E-Q-Q OR Q-Q-E-S-x-E-x-E-L
	PHOSPHOFRUCTOKINASE (97)
PROSITE	[RK]-x(4)-G-H-x-Q-[QR]-G-G-x(5)-D-R
LSTM (≈)	[ILV]-E-V-M-G-[HR]-x(2)-[GS]
	PHOSPHOPANTETHEINE (198)

CONCLUSION

We've introduced a new method for protein sequence categorization and motif extraction dubbed LSTM. LSTM achieves state-of-the-art results on a benchmark homology detection dataset while being five orders of magnitude quicker than the best-performing approaches and two orders of magnitude faster than the fastest SVM-based method. In a reasonable amount of time, LSTM can categorize a full genome into structural or functional classifications while ensuring state-of-the-art performance. The modeling strength of LSTM was proved on PROSITE datasets: new and even more suggestive motifs were

discovered. As a result, LSTM models of structural classes can be used to identify regions that are crucial for the 3D structure, such as for the folding process, or 3D stability. Because it does not employ the BLOSUM or PAM matrices to measure similarities, LSTM is a good complement to alignment-based techniques. Rather, it automatically derives new similarity metrics and can deal with genetic recombination and shuffling (Vinga and Almeida, 2003). LSTM can calculate local and global sequence statistics such as hydrophobicity, build negative class patterns, and use a recognized pattern to inhibit or strengthen another pattern. As a result, LSTM may be able to uncover novel regularities in protein structure that would otherwise be missed by traditional alignment techniques. As a result, LSTM could be beneficial for detecting alternative splice sites or extracting nucleosome positions from DNA data, among other things.

REFERENCES

Altschul, S.F. et al. (1990) Basic local alignment search tool. J. Mol. Biol., 215, 403–410.

Altschul, S.F. et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res., 25, 3389–3402.

Bairoch, A (1999) The PROSITE database, its status in 1995. Nucleic Acids Res., 24, 189–196.

Baldi, P. et al. (1999) Exploiting the past and the future in protein secondary structure prediction. Bioinformatics, 15, 937–946.

Bynagari, N. B. (2014). Integrated Reasoning Engine for Code Clone Detection. *ABC Journal of Advanced Research*, *3*(2), 143-152. https://doi.org/10.18034/abcjar.v3i2.575

Cheng, J. and Baldi, P. (2005) Three-stage prediction of protein beta-sheets by neural networks, alignments, and graph algorithms. Bioinformatics, 21, i75–i84.

Ding, C. and Dubchak, I. (2001) Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics, 17, 349–358.

Donepudi, P. K. (2014). Voice Search Technology: An Overview. *Engineering International*, *2*(2), 91-102. https://doi.org/10.18034/ei.v2i2.502

Dong, Q.W et al. (2006) Application of latent semantic analysis to protein remote homology detection. Bioinformatics, 22, 285–290.

Gille, C. et al. (2003) A comprehensive view on proteasomal sequences: implications for the evolution of the proteasome. J. Mol. Biol., 326, 1437–1448.

Gribskov, M. et al. (1987) Profile analysis: detection of distantly related proteins. Proc. Natl Acad. Sci., 84, 4355–4358 .

Grundy, W.N. (1998) Family-based homology detection via pairwise sequence comparison. In Proceedings of 2nd Annual International Conference on Computational Molecular Biology, pp. 94–100. ACM Press, New York, USA.

Henikoff, S. and Henikoff, J.G. (1994) Position-based sequence weights. J. Mol. Biol., 243, 574–578.

Hochreiter, S. (1991) Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut fu¨r Informatik, Lehrstuhl Prof. Brauer, Tech. Univ. Mu¨nchen.

Hochreiter, S. and Schmidhuber. J. (1997) Long short-term memory. Neural Comput., 9, 1735–1780.

Hochreiter, S. et al. (2001) Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kolen, J. and Kremer, S. (eds), A Field Guide to Dynamical Recurrent Networks. Wiley-IEEE Press, Piscataway, NJ.

Hou, Y. et al. (2004) Remote homolog detection using local sequence-structure correlations. Proteins Struct., Funct. and Bioinformatics, 57, 518–530.

Jaakkola, T. et al. (1999) Using the fisher kernel method to detect remote protein homologies. In Proc. the Seventh International Conference on Intelligent Systems for Molecular Biology, 16, 149–158. AAAI Press, Menlo Park, CA.

Karplus, K. et al. (1998) Hidden markov models for detecting remote protein homologies. Bioinformatics, 14, 846–856.

Kent, W. J. (2002) BLAT - the BLAST like alignment tool. Genome Research, 12, 656–664.

Kuang, R. et al. (2005) Profile-based string kernels for remote homology detection and motif extraction. Journal of Bioinformatics and Computational Biology, 3, 527–550.

Leslie, C. et al. (2004a) Mismatch string kernels for discriminative protein classification. Bioinformatics, 20, 467–476.

Leslie, C. et al. (2004b) Inexact matching string kernels for protein classification. In Scho¨ lkopf, B. Tsuda, K. and Vert, J.P. (eds), Kernel Methods in Computational Biology, pp. 95–111. The MIT Press, Cambridge, Massachusetts, London, England.

Liao, L. and Noble, W.S. (2002) Combining pairwise squence similarity support vector machines for remote protein homology detection. In Proceedings of the Sixth International Conference on Computational Molecular Biology, pp. 225–232. ACM Press, New York, USA.

Lingner, T. and Meinicke, P. (2006) Remote homology detection based on oligomer distances. Bioinformatics, 22, 2224–2236.

Madera, M. and Gough. J. (2002) A comparision of profile hidden Markov model procedures for remote homology detection. Nucleic Acids Res., 30, 4321–4328.

Murzin, A.G. et al. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol.Biol., 247, 536–540.

Park, J. et al. (1998) Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. J. Mol. Biol., 284, 1201–1210.

Pearson, W. and Lipman, D. et al. (1988) Improved tools for biological sequence comparison. Proc. Natl Acad. Sci., 85, 2444–2448, .

Rangwala, H. and Karypis, G. (2005) Profile based direct kernels for remote homology detection and fold recognition. Bioinformatics, 21, 4239–4247 .

Sigrist, C.J.A. et al. (2002) PROSITE: A documented database using patterns and profiles as motif descriptors. Brief. Bioinform., 3, 265–274.

Smith, T. and Waterman, M. et al. (1981) Identification of common molecular subsequences. J. Mol. Biol., 147, 195–197.

Tarnas, C. and Hughey, R. (1998) Reduced space hidden Markov model training. Bioinformatics, 14, 401–406.

Thompson, J.D. et al. (1994) CLUSTAL W: improving the sensivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res., 22, 4673–4680.

Vapnik V.N. (2000) The Nature of Statistical Learning Theory. Statistics for Engineering and Information Science. 2nd edition, Springer Verlag. New York.

Vert, J.P. et al. (2004) Local alignment kernels for biological sequences. In Scho¨ lkopf, B. Tsuda, K. and Vert, J.-P. (eds.), Kernel Methods in Computational Biology, pp. 131–154. The MIT Press, Cambridge, Massachusetts, London, England.

Vinga, S. and Almeida, J. (2003) Alignment-free sequence comparision–a review. Bioinformatics, 19. 513–523.

--0--