BIOINFORMATICS ORIGINAL PAPER Vol. 23 no. 14 2007, pages 1728–1736
doi:10.1093/bioinformatics/btm247

Sequence analysis

Fast model-based protein homology detection without alignment

Sepp Hochreiter[1,*], Martin Heusel[2] and Klaus Obermayer[2]

[1]Institute of Bioinformatics, Johannes Kepler Universität Linz 4040 Linz, Austria and [2]Department of Electrical Engineering and Computer Science, Technische Universität Berlin and Bernstein Center for Computational Neuroscience, 10587 Berlin, Germany

Received on December 22, 2006; revised on April 18, 2007; accepted on May 1, 2007

Advance Access publication May 11, 2007

Associate Editor: Limsoon Wong

ABSTRACT

Motivation: As more genomes are sequenced, the demand for fast gene classification techniques is increasing. To analyze a newly sequenced genome, first the genes are identified and translated into amino acid sequences which are then classified into structural or functional classes. The best-performing protein classification methods are based on protein homology detection using sequence alignment methods. Alignment methods have recently been enhanced by discriminative methods like support vector machines (SVMs) as well as by position-specific scoring matrices (PSSM) as obtained from PSI-BLAST.

However, alignment methods are time consuming if a new sequence must be compared to many known sequences—the same holds for SVMs. Even more time consuming is to construct a PSSM for the new sequence. The best-performing methods would take about 25 days on present-day computers to classify the sequences of a new genome (20 000 genes) as belonging to just one specific class—however, there are hundreds of classes.

Another shortcoming of alignment algorithms is that they do not build a model of the positive class but measure the mutual distance between sequences or profiles. Only multiple alignments and hidden Markov models are popular classification methods which build a model of the positive class but they show low classification performance. The advantage of a model is that it can be analyzed for chemical properties common to the class members to obtain new insights into protein function and structure.

We propose a fast model-based recurrent neural network for protein homology detection, the 'Long Short-Term Memory' (LSTM). LSTM automatically extracts indicative patterns for the positive class, but in contrast to profile methods it also extracts negative patterns and uses correlations between all detected patterns for classification. LSTM is capable to automatically extract useful local and global sequence statistics like hydrophobicity, polarity, volume, polarizability and combine them with a pattern. These properties make LSTM complementary to alignment-based approaches as it does not use predefined similarity measures like BLOSUM or PAM matrices.

Results: We have applied LSTM to a well known benchmark for remote protein homology detection, where a protein must be classified as belonging to a SCOP superfamily. LSTM reaches state-of-the-art classification performance but is considerably faster for classification than other approaches with comparable classification performance. LSTM is five orders of magnitude faster than methods which perform slightly better in classification and two orders of magnitude faster than the fastest SVM-based approaches (which, however, have lower classification performance than LSTM). Only PSI-BLAST and HMM-based methods show comparable time complexity as LSTM, but they cannot compete with LSTM in classification performance.

To test the modeling capabilities of LSTM, we applied LSTM to PROSITE classes and interpreted the extracted patterns. In 8 out of 15 classes, LSTM automatically extracted the PROSITE motif. In the remaining 7 cases alternative motifs are generated which give better classification results on average than the PROSITE motifs.

Availability: The LSTM algorithm is available from http://www.bioinf.jku.at/software/LSTM_protein/

Contact: hochreit@bioinf.jku.at

1 INTRODUCTION

In our post-genomic era, there is increasing need to analyze the amino acid sequences obtained from whole genome sequencing. To deduce the function or the 3D structure of a protein from the amino acid sequence, the most successful approach is to detect its homology to other proteins. Therefore, pairwise alignment methods like the Smith–Waterman algorithm (Smith and Waterman, 1981) or its approximations like FASTA (Pearson and Lipman, 1988) or BLAST/PSI-BLAST (Altschul *et al.*, 1990) are important to measure the similarity, i.e. the homology, of proteins.

These alignment-based methods were recently enhanced by discriminative methods like the support vector machine (SVM), (Vapnik, 2000). SVM-based protein homology detection methods are based on kernels which use alignment methods or sequence identities to compute similarities between sequences. These methods include 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

*To whom correspondence should be addressed.

kernels (Dong *et al.*, 2006; Lingner and Meinicke, 2006) the Smith–Waterman and the local alignment kernel (Vert *et al.*, 2004). Recently, these methods have been improved by using profiles and position-specific scoring matrices (PSSM) instead of the original sequences (Rangwala and Karypis, 2005).

However, sequence similarity-based (i.e. alignment-based) methods still have a number of shortcomings. Despite their good performance, they require too much computation time for broad use (Vinga and Almedia, 2003). For example, to classify the protein sequences identified in a newly sequenced genome, the best-performing methods would take about one month to classify the genes only belonging to a single class. For hundreds of classes, their time complexity makes these approaches infeasible for practical use.

Alignment methods still have problems with genetic recombination and genetic shuffling (Vinga and Almeida, 2003).

Another shortcoming of similarity-based methods is that they are not model based. Without a model it is difficult to interpret classification results in terms of relevant patterns or chemical properties which are relevant for the protein class (function, stability, folding). Whereas, model-based methods are able to identify relevant elements for structural biochemistry.

Here, we suggest to use recurrent neural networks (RNNs) in order to overcome the earlier mentioned disadvantages of similarity-based methods. RNNs have already successfully been applied to protein secondary structure prediction (Baldi, *et al.*, 1999) and to sheet pairing prediction (Cheng and Baldi, 2005).

In contrast to similarity-based methods, RNNs

(a) can extract dependencies between subsequences. A subsequence AB, e.g. may only be indicative if it is followed later by the subsequence CD;

(b) can extract correlations within subsequences. Both subsequences AB and CD may be indicative for the class (motif [AC]–[BD]), however, AD may not be indicative for the class (AD is a *negative pattern*);

(c) can extract global sequence characteristics (hydrophobicity or atomic weight), and

(d) can extract dependencies between amino acids which range over a long interval in the amino acid sequence.

RNNs can thus deal with interactions between different profiles, e.g. a detected pattern can inhibit or reinforce the storage of another pattern. They can compute non-linear functions of indicative patterns in the sequence, therefore, they are in principle an interesting tool for amino acid sequence processing.

However, RNNs also have disadvantages: (1) they need a large enough training set for model selection; (2) an architecture must be chosen before using them; (3) the training phase may be computational intensive and (4) they cannot detect similarities between negative examples.

The new aspect when using RNNs for homology detection is that pointwise similarity measures (identity, BLOSUM or PAM matrices) are not a priori fixed. RNNs can learn their own similarity measure suited for a specific classification task, where they may combine patterns with sequence statistics like hydrophobicity.

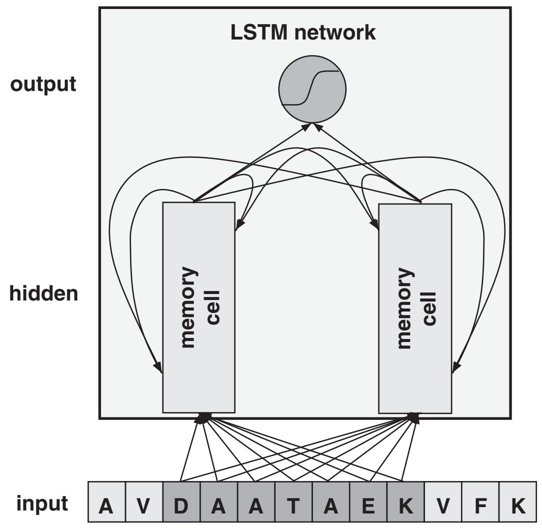

Fig. 1. Proposed LSTM network with three layers: input layer (window of the amino acid sequence – shaded elements); hidden layer (with memory cells – see Fig. 2) and output layer. Arrows represent weighted connections; the hidden layer is fully connected. The subnetworks denoted by 'memory cell' can store information and receive three kinds of inputs, which act as pattern detectors (input → hidden), as storage control (recurrent connections) and as retrieval control (recurrent connections). The stored information is combined at the output. The amino acid sequence is processed by scanning the sequence step by step from the beginning to the end.

2 THE LSTM NETWORK

2.1 Model architecture

We suggest the 'Long Short-Term Memory' (LSTM), (Hochreiter and Schmidhuber, 1997) recurrent net architecture in Figure 1 for protein homology detection. Sequences are processed element by element by the network and are finally classified at sequence end. At each step, the network input is taken from a window region around the current position.

LSTM contains specially designed memory sub-architectures called 'memory cells' which are able to store information like the occurrence of a pattern from previously scanned regions without loss (see Fig. 2 for more details on memory cells). The stored information is used to inhibit or reinforce other patterns in the remaining sequence and is important at sequence end for predicting the class.

2.2 The importance of memory

Information, e.g. a pattern, which is indicative for a protein class can be located at any position in the sequence, hence, long intervals may occur between relevant sequence locations. Therefore, the relevant information must be stored until it is needed for classification. But classical RNNs fail at this task because of the exponential decay of previously seen information with processing time leading to a 'vanishing gradient' problem (Hochreiter, 1991; Hochreiter: *et al.*, 2000).

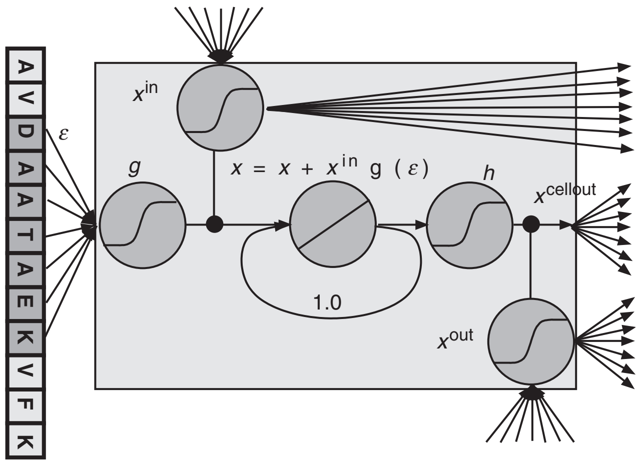

Fig. 2. The LSTM memory cell. Arrows represent weighted connections of the neural network. Circles represent units (neurons), where the activation function (linear or sigmoid) is indicated in the circle.

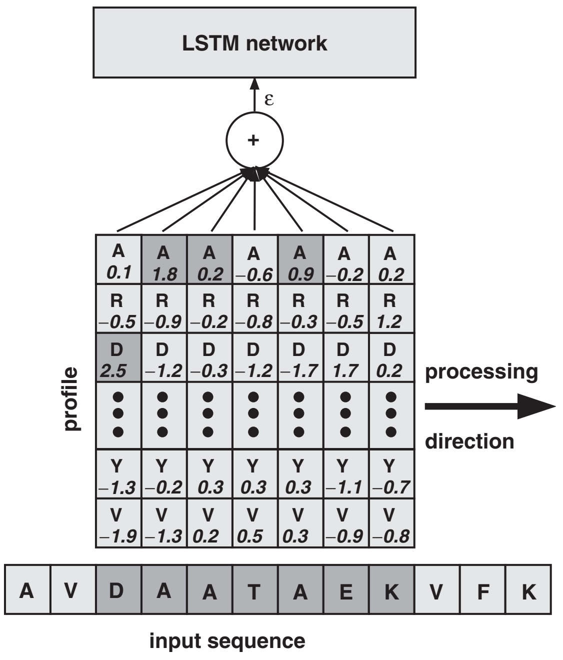
LSTM network
ε
+
A 0.1 | A 1.8 | A 0.2 | A −0.6 | A 0.9 | A −0.2 | A 0.2
R −0.5 | R −0.9 | R −0.2 | R −0.8 | R −0.3 | R −0.5 | R 1.2
D 2.5 | D −1.2 | D −0.3 | D −1.2 | D −1.7 | D 1.7 | D 0.2
Y −1.3 | Y −0.2 | Y 0.3 | Y 0.3 | Y 0.3 | Y −1.1 | Y −0.7
V −1.9 | V −1.3 | V 0.2 | V 0.5 | V 0.3 | V −0.9 | V −0.8
profile
processing
direction
A V D A A T A E K V F K
input sequence

Fig. 3. A profile as input to the LSTM network which scans the input from left to right. Amino acids are represented by the one letter code. Shaded squares in the matrix match the input and contribute to the sum ϵ.

In order to be a competitive method for protein classification, RNNs require a specially designed memory to store information until it is being used. Note, that for the tasks in Baldi *et al*., (1999) and Cheng and Baldi (2005) the important information in the sequence was close to the prediction position and such memories were not necessary.

LSTM, (Hochreiter andSchmidhuber, 1997) is an RNN with a designed memory sub-architecture called 'memory cell' to store information; therefore it is suited for protein classification. Memory units with non-decaying information are realized by volume-conserving mappings constructed through a linear unit with a self-recurrent connection with weight one. Figure 2 shows such a 'memory cell' within an LSTM network. The unit in the center of the box implements the volume-conserving mapping.

The input to the memory cell is controlled by an 'input gating' or attention unit (Fig. 2, unit marked 'x^{in}') which blocks class-irrelevant information, so that only class-relevant information is stored in memory. The activation of attention units is bounded by 0 and 2, i.e. the incoming information $\epsilon(t)$ is squashed by a sigmoid function g. The memory cell's activation function is given by

$$x(t+1) \ = \ x(t) \ + \ x^{in}(t) \cdot g(\epsilon(t)) \ , \tag{1}$$

where $\epsilon(t)$ is the local sequence information (cf. Equation (3)). The output of the memory cell (Fig. 2, center) is bounded between −1 and 1 by the sigmoid function h (Fig. 2, unit labeled as 'h'). Memory readout is controlled by an 'output gate' (Fig. 2, unit labeled 'x^{out}'). The cell's output $x^{cellout}$ is then computed as follows:

$$\begin{aligned} x^{cellout}(t+1) \ &= \ x^{out}(t) \cdot h(x(t+1)) \ = \\ x^{out}(t) \cdot h\big(x(t) \ &+ \ x^{in}(t) \cdot g(\epsilon(t))\big) \ . \end{aligned} \tag{2}$$

Memory cells can in principle be integrated into any neural network architecture. Here, we use the LSTM recurrent network structure as depicted in Figure 1.

2.3 Learning of profiles

In order to extract class-indicative information, we adopt profiles (Gribskov *et al*., 1987 or Henikoff and Henikoff, 1994) as inputs for the LSTM network which enable LSTM to *automatically learn* the profiles by error propagation. The input weights to a memory cell form a profile because they provide a weighted sum of the amino acids within the window.

We use a local encoding of the amino acids in the input, i.e. each amino acid is represented as a 20D vector with zeros, except for one position, which contains a one. If the profile has length l, then the input is a $20 \times l$ matrix $\boldsymbol{Y}$ with components $Y_{\tau j}$. Figure. 3 depicts a profile as input to the LSTM network. Each memory cell uses its own input profile, and the ith profile input $\epsilon_i(t)$ at sequence position t to the LSTM network is computed as

$$\epsilon_i(t) \ = \sum_{(\tau,j)=(1,1)}^{(l,20)} w^i_{\tau j} \ Y_{\tau j}, \quad Y_{\tau j} = \begin{cases} 1 & s_{t+\tau} = \mathrm{AA}_j \\ 0 & \text{otherwise} \end{cases} , \tag{3}$$

where s_t is the tth element of the input sequence and AA_j represents the jth amino acid. $w^i_{\tau j}$ is an element of the ith profile matrix but is simultaneously an input weight of the LSTM network which makes LSTM to automatically learn the profile.

2.4 Model characteristics

For protein classification and learning profiles, the original LSTM architecture is modified (cf. Fig. 1, e.g. with two memory cells). The memory cells receive their only input through the profiles, where a profile is formed by all weights from the input to the memory cell. In contrast to the original

LSTM architecture there are no weights from the input to any other units. The attention unit and the output gate receives inputs from other attention units, other output gates and other memory cells, while the output unit only obtains input from the memory cells (cf. Fig. 2).

The LSTM architecture is trained with the LSTM learning procedure as described in Hochreiter and Schmidhuber (1997) which is a gradient-descent method.

2.5 Computational complexity

The LSTM approach has time complexity of $O(L)$ for classifying a new sequence of length L. This complexity has to be compared with the complexity of the most efficient profile-based methods from the literature. The method of Rangwala and Karypis (2005) is dominated by the time needed to compute the profile of a new sequence. In order to compute the profile, the NR database with more than 3 million entries must be scanned and a multiple alignment must be generated followed by a profile–profile alignment. Alignment methods have complexity of $O(L^2)$ including profile–profile alignments. Thus alignment-based kernel methods possess complexity of $O(N_{\mathrm{SV}}L^2)$, where N_{SV} is the number of support vectors. Because of an explicit representation of the linear classifier (the weight vector) in feature space, the time complexity has been reduced to $O(L^2)$ in Lingner and Meinicke (2006). For $L > 100$ LSTM is theoretically more than two orders of magnitude faster than the fastest SVM-based method which has been confirmed experimentally in the experiments described subsequently.

3 NUMERICAL EXPERIMENTS

We perform three sets of experiments. In the first set, we assess the performance and the time complexity of LSTM for remote homology detection and compare LSTM to various state-of-the-art approaches on a benchmark dataset from the SCOP database (Murzin *et al.*,1995).

In the second set of experiments, we compare LSTM to different machine-learning methods which extract features from the sequences on a SCOP fold prediction task from Ding and Dubchak (2001).

In the third set of experiments, we assess LSTM's modeling performance and investigate whether LSTM automatically extracts indicative motifs for a protein classification. LSTM is applied to the PROSITE database (Bairoch, 1995), and the motifs extracted by LSTM are compared to the PROSITE motifs (Sigrist *et al.*,2002).

3.1 Remote homology detection: SCOP superfamilies

3.1.1 Data We used the widely used benchmark dataset for remote homology detection from (Liao and Noble, 2002) which is available under http://www.cs.columbia.edu/compbio/svm-pairwise. The dataset defines 54 superfamily recognition tasks. For each task, the positive examples of the training set are taken from one superfamily from which one family is withhold. The task is to detect the examples from the withhold family. Negative training examples are chosen from outside the fold the family belongs to.

3.1.2 Methods We perform benchmark with the following methods: (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). These alignment-based methods can be enhanced by SVMs which also take negative examples into account. We compare: (d) the Fisher-kernel SVM (Jaakkola *et al.*, 1999, (e) SVMs using the mismatch-kernel (Leslie *et al.*, 2004a, b)—the mismatch-kernel is similar to the BLAT alignment (Kent, 2002), (f) the SVM-pairwise (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) local alignment (LA) kernel (Vert *et al.*, 2004) based on the BLOSUM matrix, (i) the oligomer-based distance SVM approach (Lingner and Meinicke, 2006). We then include into the comparison methods which are based on PSSMs or profiles and which we summarize under (j) which includes '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 (Rangwala *et al.*, 2005). We included the result of the best parameters as given in the corresponding publications (note, that we may overestimate the results because hyperparameter selection was avoided). Finally, (k) we report the results obtained with our new LSTM method.

3.1.3 LSTM implementation detatils Hyperparameter selection: we selected the parameters of the model (number of memory cells, window size, learning rate, initialization, etc.) on a separate dataset from Gille *et al.* (2003) which is available via the program package STRAP http://www.3d-alignment.eu. The dataset consists of 500 proteasome sequences and 7400 negatives from PDB.

Architecture: thirteen memory cells, profile length: 11. 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: output unit bias: –1.0 (to account for more negative examples), 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, this yields to 7 positive and 6 negative memory cells. All other weights are set to 0.

Output coding: 0.8 (positive class) and 0.2 (negative class).

Learning parameters: 500 epochs learning time, learning rate: $\alpha = 0.01$. Positive examples are cloned until their number is at least 1/5 of the number of negative examples.

Training set: the positive training set is extended by running PSI-BLAST with five iterations and default parameters against the NR database for each positive example and selecting examples with e-values smaller than 10.0 in the last iteration. Note, that the PSI-BLAST run was only used to extend the training set [cf. disadvantage (1) of RNNs at end of Introduction] and is not necessary for classifying new examples.

3.1.4 Evaluation The quality of a ranking of the test set examples was evaluated by the area under the receiver operating characteristics curve (ROC). The methods were evaluated through 54 ROC values, where the ROC value is between 0.5 (random guessing) and 1.0 (perfect prediction). As a more precise quality measure, we also used the area under the ROC50 which is the area under the ROC up to 50 false positives. ROC50 essentially rescales the false positive rate of the ROC.

3.1.5 Test times Computing times were measured on an Opteron 165 1.8 GHz machine. The time for the LA-kernel was evaluated by using the software from Vert *et al*. (2004). With the LA-kernel, we measured a test sequence with length 165 (the average length of proteins in the dataset). The time for the oligomer method can be computed from the LA-kernel time because in Lingner and Meinicke (2006) the authors report that the oligomer method is 1000 times faster than the LA-kernel. The time for the SW kernel was lower bounded by BLAST (NCBI bl2seq 2.2.14 from http://www.ncbi.nlm.nih.gov/Ftp/) for pairs of proteins because BLAST is faster than the exact Smith–Waterman algorithm. For measuring the time of the mismatch kernel, we used the software from http://www1.cs.columbia.edu/compbio/string-kernels/ according to Leslie *et al*. (2004a). The PSI-BLAST, SAM-T98 and Fisher-kernel test times were computed from the CPU values given in Tarnas and Hughey (1998). The test times for 'SW-PSSM' were computed with the software from Rangwala and Karypis (2005). The time for computing a profile for one sequence was 90 s which gives 500 h for 20 000 test examples as an lower bound for profile methods. The training times of SVM-methods using a profile and LSTM are 110 h and 117 h, respectively. These high training times result from PSI-BLAST runs (105 h) which construct the profiles SVM or extend the positives LSTM.

3.1.6 RESULTS Table 1 summarizes the average values of the area under ROC, the area under ROC50 and the time complexity for all methods used in our benchmark. The classification results except for LSTM are taken from Hou *et al*. (2004), Kung *et al*. (2005), Liao and Noble, (2002), Lingner and Meinicke, (2006), Rangwala and Karypis, (2005), Vert *et al*. (2004), The time measurements are from Linger and Meinicker, (2006), Madera and Gough, (2002), Tarnas and Hughey, (1998), and measured using the string kernel software from http://www1.cs.columbia.edu/compbio/string-kernels/ and BLAST algorithms from http://www.ncbi.nlm.nih.gov/Ftp/. Figures 4 and 5 show for each method how many classification tasks reached a certain ROC value.

Results for the ROC values: the table and the figures show that only some of the similarity-based methods using profiles show a better performance than LSTM. We looked at those LSTM misclassifications which were made correctly by similarity-based methods: LSTM false positives possessed high similarity to positive training examples however using similarity-based methods this similarity was outvoted by a similarity to a specific negative training example (cf. RNN's disadvantage (4) at the end of the Introduction).

However, Lingner and Meinicke (2006) noticed that the best performing approaches (Dong *et al*., 2006, Rangwala and Karypis, 2005) might be optimized on the test data, i.e. the results may be overoptimistic. LSTM has similar performance as the LA-kernel approach (Vert *et al*., 2004) and, therefore, is among the best methods which are solely based on scanning the primary sequence. All model-based approaches perform worse than the LSTM approach with respect to classification performance.

Results for the time complexity: Concerning the computational time of the methods, only PSI-BLAST is faster than LSTM. LSTM is even 1 order of magnitude faster than SAM-T98. Note, that LSTM only scans the sequence and has not to align it to a profile. LSTM is 2 orders of magnitude faster

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

Method	M	P	V	S	ROC	ROC50	Time
(a) PSI-BLAST	–	–	–	–	0.693	0.264	5.5 s
(b) FPS	–	–	–	–	0.596	–	6800 s
(c) SAM-T98	+	–	–	–	0.674	0.374	200 s
(d) Fisher	–	–	–	+	0.887	0.250	>200 s
(e) Mismatc h	–	–	–	+	0.872	0.400	380 s
(f) Pairwise	–	–	–	+	0.896	0.464	>700 s
(g) SW	–	–	–	+	0.916	0.585	>470 s
(h) LA	–	–	–	+	0.923	0.661	550 h
(i) Oligomer	–	–	–	+	0.919	0.508	2000 s
(j) HMMSTR	–	+	+	+	–	0.640	>500 h
(j) Mismatch-PSSM	–	+	+	+	0.980	0.794	>500 h
(j) SW-PSSM	–	+	+	+	0.982	0.904	>620 h
(k) LSTM	+	–	+	–	0.932	0.652	20 s

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. The sixth and seventh column report the average area under the receiver operating curve ('ROC') and the same value for maximal 50 false positives ('ROC50'). The last column reports the average time needed to classify 20 000 new sequences into one superfamily. For calculation of computation time see text. The classification results except for LSTM are taken from Hou *et al*. (2004), Kuang *et al*. (2005), Liao and Noble, (2002), Lingner and Meinicke, (2006), Vert *et al*. (2004), Rangwala and Karypis, (2005). Only some of the profile-based methods show a better classification performance than LSTM but they are three orders of magnitude slower.

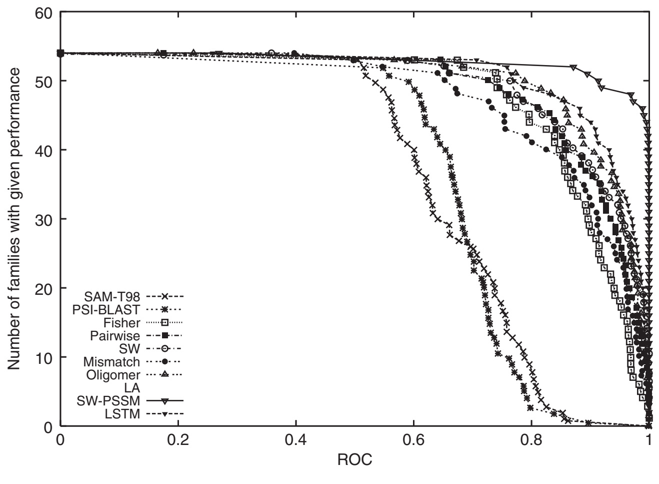

Fig. 4. Comparison of homology detection methods for the SCOP 1.53 benchmark dataset. The total number of families for which a given method exceeds a ROC threshold is plotted.

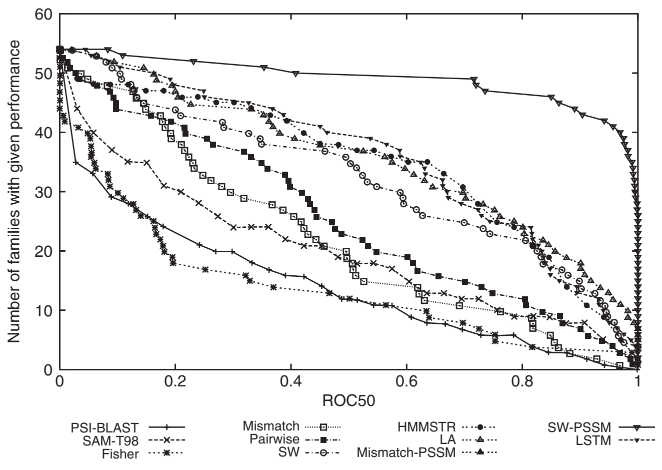

Fig. 5. Comparison of homology detection methods for the SCOP 1.53 benchmark dataset. The total number of families for which a given method exceeds a ROC50 threshold is plotted.

than the fastest SVM-based method. Methods which perform better than LSTM need 5 orders of magnitude more CPU time to process a new sequence.

3.1.7 PFAM test We additionally tested LSTM, PSI-BLAST and SAM 3.5 on the PFAM database, because SCOP covers only 15% of the PFAM families, hence PFAM allows to test for more variants of negatives. We optimized the performance of PSI-BLAST and SAM by using as score for the positive class the best score from all positive training sequences (like the FPS method). For PSI-BLAST, we performed two iterations on the NR database to build a profile. The NR database was also used by SAM to build an alignment. The HMM was build with w0.5 and calibrated. Note, that both methods SAM and PSI-BLAST build a model based on multiple alignment (a profile and an HMM, respectively) which is also the basis of the PFAM classification.

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

Method	ROC all	ROC remote	Time
PSI-BLAST	0.80	0.69	50h
SAM 3.5	0.85	0.76	1200h
LSTM	0.88	0.79	27h

'ROC remote' gives the area under the curve if only the remote homologous test sequences are considered. The last column gives the computation time on the PFAM database.

It might be possible that these methods are overestimated because the model underlying the dataset is the same as the model which was build by the methods we compare. To label the PFAM sequences we identified the PFAM families which had an unique superfamily in the SCOP database. These PFAM families are labeled according to the SCOP superfamily they belong to. For remote homology detection, only PFAM families in the SCOP test set are labeled positive. The results on the PFAM dataset are given in Table 2. LSTM shows the best performance and is fastest in testing new sequences.

3.2 SCOP fold prediction

To compare our method with other machine-learning methods which describe the amino acid sequence by extracted features, we performed another benchmark on the dataset from Ding and Dubchak, (2001). We consider the 'one-versus-other' task where for each class a binary classifier is constructed with the class members in the positive class and the rest in the negative class.

3.2.1 Data This data set was also derived from SCOP and consists of 622 positive training sequences which are classified into 27 folds. The test dataset was the PDB40D with 386 sequences also classified into the given 27 folds.

3.2.2 Methods For the different methods the authors in Ding and Dubchak (2001) extracted the optimal description parameters including amino acid composition, predicted secondary structure, hydrophobicity, normalized van der Waals volume, polarity, polarizability, etc. These features characterize the sequence through composition, percent in each class, transition frequencies and distribution. The neural network model was a three-layer feed-forward neural network trained by conjugate gradient. The authors in Ding and Dubchak (2001) found that one hidden unit and two output units gave good performance while being low complex to ensure generalization. The SVM was tested with a linear, a polynomial and an RBF kernel, where the authors in Ding and Dubchak (2001) found that the latter gave the best results.

3.2.3 Evaluation The accuracy Q (rate of correct classification) of fold prediction was measured on the independent PDB40D dataset. The task is a multiclass problem, therfore only classifications which are uniquely assigned to a class are judged as being correct. For LSTM, a protein is

Table 3. Results on the data set from Ding and Dubchak (2001) for different machine-learning methods

Method	*Q*	Method	*Q*
NN	41.8	SVM	45.2
LSTM	51.7		

where 'NN' means neural network and 'SVM' support vector machine. LSTM yields the highest accuracy.

Table 4. Results of PROSITE protein classification tested on the SwissProt database

Method/motif	Sensitivity	Specificity	Balanced Error
PROSITE	85.91 (15.62)	99.94 (0.15)	7.08 (7.79)
LSTM	98.24 (3.55)	99.79 (0.19)	0.99 (1.82)
Motif	86.82 (9.2)	99.93 (0.16)	6.63 (4.59)

All numbers are averaged over given 15 classes, the SD of the results is given in brackets. Results are reported for the PROSITE motif ('PROSITE'), for LSTM ('LSTM'), and for the motif extracted from LSTM ('motif'). The columns show (left to right): method, sensitivity (true positives divided by all positives in percent, 'sens.'), specificity (true negatives divided by all negatives in percent, 'spec.'), and the balanced error in percent ('bal. err.'). The balanced error is the mean of the class 1 and the class 2 error rate and is an appropriate measure for classification with unbalanced class sizes.

assigned to that class where the according classifier gives the largest output.

3.2.4 Results In Table 3, the results on the dataset from Ding and Dubchak, (2001) are reported. LSTM clearly outperforms the other methods in terms of the *Q*-value.

3.3 Motif extraction: PROSITE database

3.3.1 Data We randomly chose 15 PROSITE protein classes from version 18.20. The LSTM training data consisted of 90% of the positive examples plus 1300 randomly chosen negative examples. The remaining 10% positives are withhold as test examples.

3.3.2 Methods After training the LSTM network, we identified the sequence positions where the profiles are activated (highest activation) by examples from the positive training set. From these positions a motif was extracted by multiple alignment (we used ClustalW, Thompson *et al.*, 1994) of the subsequences in the input window at these positions. The multiple alignment can deal with the fact that more than one motif is encoded in the profile. We then compare the extracted motif from LSTM with the PROSITE motif.

In the PROSITE database 80% of the motifs are 21 amino acids or less long (Fig. 6), therefore a window length of 21 was used in this experiment. We only used two memory cells (one for patterns of the positive class and one for negative class) to avoid superimposition of motifs. The initialization of the cell input bias was increased to −6.0 to obtain large absolute input

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

	• 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}-[A**C**]-[**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**-{**CP**WH**F**}-{**CPWR**}-**C**-**H**-{**CFYW**}
LSTM (≈)	**C**-{**CFP**}-{**CR**WY}-**C**-**H**-{**CF**H**WY**}
	•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)-[**LIVMF**S]-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)
PROSITE	[DEQGSTALMKRH]- . . . -[DNEKHS]-**S**-[**LIVM**ST]-PCFY- . . . -[LIVMWSTA]-[LIVGSTACR]-x(2)-[LIVMFA]
LSTM (≈)	[LFT]-x(1,2)-[DEQSTAK]- . . . -[DEHQSN]-**S**-[**LIVM**A]-x(4)-[LIVMSTA]-x(3)-[LIVMAF]-[DEHQSTAR]
	•SERPIN (156)
PROSITE	[LIVMFY]-x-[LIVMFYAC]-[DNQ]-[RKHQS]-[PST]-F-[LIVMFY]-[LIVMFYC]-x-[LIVMFAH]
LSTM (≉)	F-{ADEGINP}-[IKLMNSV]-x(6,7)-V-x-M-M
	•UPF0011 (26)
PROSITE	S-D-A-G-x-P-x-[LIV]-[SN]-D-P-G
LSTM (≉)	R-x(4)-[LF]-x(5)-[LIVF]-x(2)-E-D-T-R
	•ZINC_ FINGER_ C2H2_ 1 (792)
PROSITE	C-x(2,4)-**C**-x(3)-[LIVMFYWC]-x(8)-**H**-x(3,5)-**H**

(Continued)

Table 5. Continued

LSTM (≈)	[CFA]-x(2)-**C**-x(3)-[CFY]-x(5)-[LFQ]-x(2)-**H**-x(3)-**H** •ZINC_ PROTEASE (546)
PROSITE	[GSTALIVN]-x(2)-**H**-**E**-[LIVMFYW]-{DEHRKP}-**H**-x-[LIVMFYWGSPQ]
LSTM (≈)	[GILNSTV]-[AFILMTVY]-x-**H**-**E**-[AFILMTVY]-[AGILMSTV]-**H**

'-' separates positions, 'x(a)' means a arbitrary amino acids, 'x(*a*,*b*)' *a* to *b* arbitrary amino acids, '[...]' denotes alternatives and '{...}' exceptions. The brackets after LSTM indicate whether its motif is similar to the PROSITE motif ('≈') or not ('≉'). In bold are sub-patterns with match in both motifs.

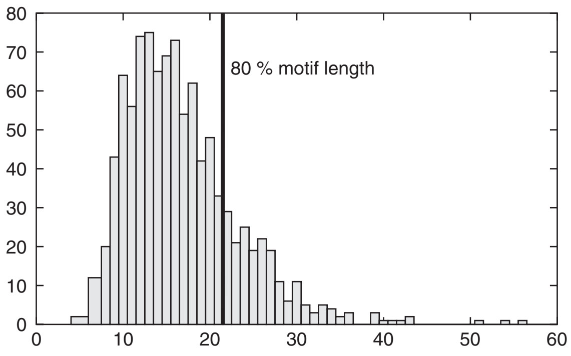

Fig. 6. Histogram of motif length in the PROSITE database. Eighty percent of the motifs have length 21 or less.

weights which support motif extraction. LSTM parameters are as in the SCOP experiment except that we sped up learning by a learning rate of 0.1 and a momentum factor of 0.9 and, therefore, stopped learning at 100 epochs.

3.3.3 Evaluation The classification performances are tested on the SwissProt database (version 42.9) with 143 790 sequences. The motifs are also compared with respect to their position in the amino acid sequence in order to determine whether both motifs identified the same conserved region.

3.3.4 Results The average test results on the SwissProt database are given in Table 4 for 15 classes. The table shows that LSTM performs best in terms of sensitivity with only a marginal deficit in terms of specificity, leading to drastic reduction of the balanced error in favor for LSTM. When motifs extracted by LSTM are used in a replacement of PROSITE motifs, classification performance is still superior (in the earlier sense) in 10 out of the 15 classes. Even when different patterns are found by LSTM, performance is better in 5 out of 7 classes. This shows that new indicative motifs can be extracted using LSTM as a tool for 'explorative data analysis'.

The motifs of PROSITE and LSTM are reported in Table 5, which shows that in 8 out of 15 cases LSTM automatically extracts a motif similar to that of PROSITE. However, in 7 cases alternative and—given the classification performance — even more indicative motifs are identified.

These results demonstrate that new information can be extracted from LSTM models which may give new insights into protein function or structure.

4 CONCLUSION

We have presented a novel method, called LSTM, for protein sequence classification and motif extraction. On a benchmark homology detection dataset, LSTM reaches state-of-the-art results while being five orders of magnitude faster than the best-performing methods and two orders of magnitude faster than the fastest SVM-based method. LSTM can be used to classify a whole genome into structural or functional classes in reasonable time while guaranteeing state-of-the-art performance.

On PROSITE datasets the modeling strength of LSTM was demonstrated: new and even more indicative motifs were found. LSTM models of structural classes may, therefore, be used to identify regions which are relevant for the 3D structure, i.e. which are important for the folding process or for the 3D stability.

LSTM is complementary to alignment-based approaches because it does not measure similarities based on the BLOSUM or PAM matrices. Rather it extracts new similarity measures automatically and may cope with genetic recombination and genetic shuffling (Vinga and Almeida, 2003).

LSTM may compute local and global sequence statistics like hydrophobicity, may construct patterns for the negative class, may use a detected pattern to inhibit or reinforce another pattern. Therefore, LSTM may detect new regularities in protein structure which cannot be discovered using standard alignment methods. LSTM may therefore be a useful method, e.g. to detect alternative splice sites or to extract nucleosome positions from DNA data.

ACKNOWLEDGEMENTS

We thank Rene Pfeifer for performing the PROSITE experiments. This work was founded in parts by the BMBF (Grant no. 01GQ0414).

Conflict of Interest: none declared.

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.

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.

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 für Informatik, Lehrstuhl Prof. Brauer, Tech. Univ. München*.

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 Schölkopf, 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*., 1**47**, 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 Schölkopf, 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.