

Improved access to sequential motifs: A note on the architectural bias of recurrent networks

Mikael Bodén

School of Information Technology and Electrical Engineering
University of Queensland, QLD 4072, Australia.

John Hawkins

School of Information Technology and Electrical Engineering
University of Queensland, QLD 4072, Australia.

Abstract

For many biological sequence problems the available data occupies only sparse regions of the problem space. To use machine learning effectively for the analysis of sparse data we must employ architectures with an appropriate bias. By experimentation we show that the bias of recurrent neural networks – recently analysed by Tino, Cernansky and Benuskova [8], and Hammer and Tino [9, 3] – offers superior access to motifs (sequential patterns) compared to the, in bioinformatics, standardly used feed forward neural networks.

1 Introduction

Recurrent neural networks have a history of being successfully utilised for sequence recognition tasks. Furthermore, there have been observations indicating that they are biased to this end [5, 2]. In spite of this, there has been a noticeable absence in the literature of a theoretical foundation by which to qualify the limits and bounds of their bias. A burst of papers recently shed light on the generalization capabilities of recurrent networks, suggesting that the networks exhibit intrinsic properties that naturally lend themselves to sequential prediction tasks [9, 3, 8]. With small weights, the recurrent feedback realizes a contractive state function inherently sensitive to past states (and therefore previous inputs). When states are grouped, transitions between groups over time abstractly resemble those of a finite markovian model with a variable memory length. Viewed as such, a recurrent neural network cannot help but be biased toward sequence recognition tasks.

In this study we elaborate on what it means in practice for an architecture to be biased towards sequential processing. Following Christiansen and Chater [2] and the recent work of Tino and colleagues [9, 3, 8] we use the phrase *architectural bias* to describe a form of discrimination that occurs in the network dynamics prior to training. This form of analysis allows a straightforward comparison between the representational abilities of recurrent neural networks against feed forward networks.

The synthetic data chosen has a deliberate bioinformatics flavour; consisting of sequential symbolic data at the level of nucleotides (4 symbols) and the level of amino acids (20 symbols). To date, biological sequence problems have predominantly been approached using feed forward networks. However, living in a combinatorial space, biological sequences (ranging hundreds or even thousands of elements) exhibit extreme sparseness. Sparseness presents an obstacle for automated algorithms that attempt to find *general* patterns in data sets. The sequence spaces in which the algorithms search are simply too vast for a largely unbiased classifier (like a feed forward network) to operate in. Tino, Hammer, Cernansky and Benuskova’s work [8] justifies a closer look at what the bias of recurrent networks can do for sequence recognition and bioinformatics in particular.

From the results in our study, we argue that before any training takes place, the recurrent network is positioned so that sequential patterns are more accessible for evaluation compared to its feed forward cousin. The results provide an explanation of why recurrent networks are sometimes more successful at sequence analysis tasks (e.g. [1]).

2 Method

We use the term motif to describe the patterns sought within sequences because it is the essential currency of much of the work done in bioinformatics. A motif is a relatively short (compared to the whole sequence) pattern within a sequence of discrete symbols. Its structure varies depending on the particular biological polymer, but often consists of a moderately fixed length subsequence with some degree of polymorphism, within 10 to 50 percent.

We use P sequences $\mathcal{S} = S_1, S_2, \dots, S_P$, each of a pre-specified length L . $S_j = [s_{j,1}, s_{j,2}, \dots, s_{j,L}]$ denotes the sequence consisting of elements $s_{j,i} \in \Lambda^U$. Λ^U is the set of U symbols, for example $\Lambda^2 = \{\text{A}, \text{B}\}$.

A motif is defined as a sequence $M = [m_1, m_2, \dots, m_N]$ (where N is the length, $N \leq L$). Each

Position dependent motif	Shift invariant motif
<pre> A A B B A A B A A B A A B B A A A A A A A B B B B A B A B B A A A B A A A A A A </pre>	<pre> A B A A A B A B B A A A B A A A A B A B A A B A B B B A B A A A A A B A B A A A </pre>

Figure 1: Motifs are specified on two forms: Position dependent (examples of matching sequences above for $M = [A\epsilon\epsilon B\epsilon A\epsilon A\epsilon\epsilon]$) and shift invariant (examples of matching sequences below for $M = [AABAB]$).

$m_i \in \Lambda^{U*}$, where Λ^{U*} is the set of symbols Λ^U plus the symbol ϵ . ϵ is used as a wild card element, matching all symbols in Λ^U . We are concerned with studying the accessibility of motifs on two specific forms (exemplified in Figure 1).

- Position dependent motifs are specified as a pattern of symbols relative to a fixed position of the sequence.

$$match(S_j, [m_1, \dots, m_N]) = \begin{cases} 1 & \text{if } \forall i \ m_i = s_{j,i} \vee m_i = \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

- Shift invariant motifs are specified as a pattern occurring over any positions in the sequence. The relative position between components is, however, fixed.

$$match(S_j, [m_1, \dots, m_N]) = \begin{cases} 1 & \text{if } \exists d \ \forall i \ m_i = s_{j,i+d} \vee m_i = \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Shift-invariant patterns are not explicitly addressed by Tino and colleagues. Recognition of such patterns tests not only the network's ability to organize the state space so that sequential patterns are visible but also the ability to retain information over a variable number of steps.

Each element in a sequence is encoded for presentation to a network. We use unique one-hot codes (one bit on, rest is off) for all symbols in Λ^U (always resulting in codes that have U bits). For Λ^2 ,

$$\phi(s_{j,i}) = \begin{cases} [1, 0] & \text{if } s_{j,i} = A \\ [0, 1] & \text{if } s_{j,i} = B \end{cases} \quad (3)$$

In general we refer to the network as producing a state from the sequence.

$$\mathbf{w}(S_j) = \mathbf{x}_j \quad (4)$$

However, the networks operate on the sequence differently. Feed forward networks accept as input

the whole sequence.

$$\mathbf{x}_j = \mathbf{f}(\phi(s_{j,1})|\dots|\phi(s_{j,L})) \quad (5)$$

where $|$ signifies concatenation of vectors.

The recurrent network takes one element at a time, working its way recursively from 1 to L (second index of inputs).

$$\mathbf{x}_j = \mathbf{g}(\phi(s_{j,1}), \mathbf{g}(\phi(s_{j,2}), \mathbf{g}(\phi(s_{j,3}), \dots))) \quad (6)$$

A null vector terminates the recursion.

Moreover, we need a method and measure to determine the accessibility of patterns in the state space. Tino and colleagues [8] show how states in a recurrent network can be transformed into discrete states of a finite memory machine by running vector quantization on all states. Using K-means to quantize the continuous state space, we need to specify K , the number of codebook vectors. K-means is allowed to converge to stable codebook vectors denoted by $V = [\mathbf{v}_1, \dots, \mathbf{v}_K]$.

Running the network on a sequence $S_j = [s_{j,1}, \dots, s_{j,L}]$, the state is mapped to a group index, $C(\mathbf{x}_j) \in \{1, 2, \dots, K\}$, where C selects the closest codebook vector by Euclidean distance.

$$C(\mathbf{x}) = \arg \min_k \|\mathbf{x} - \mathbf{v}_k\| \quad (7)$$

The same procedure is applied to all sequences in $S_j \in \mathcal{S}$. Each group will contain a subset of all sequences of which some contain the motif ($match(S_j, M) = 1$) and some do not ($match(S_j, M) = 0$). For the set of sequences \mathcal{S} and a group k , the number of positives is denoted by p_k and the number of negatives is denoted by n_k . The entropy for each group is denoted by E_k .

$$E_k = -(p_k/(p_k + n_k)) \cdot \log_2(p_k/(p_k + n_k)) - (n_k/(p_k + n_k)) \cdot \log_2(n_k/(p_k + n_k)). \quad (8)$$

The entropy measures the homogeneity within the group, considering the proportion of sequences containing the motif and those that do not. Lower value indicates higher homogeneity, more transparent access to the motif. Below we report the average entropy over all groups $\sum_k E_k/K$.

The dynamics of the feed forward neural networks are standardly defined as:

$$\mathbf{f}(\mathbf{x}) = \sigma(W_F \cdot \mathbf{x} + \mathbf{b}_F) \quad (9)$$

Where W_F is the input-to-state weight matrix and \mathbf{b}_F is the set of state biases of the feed forward network. The output function is given by σ , a sigmoidal output function (we use the logistic function). Similarly, the recurrent neural networks the dynamics are described by the following equation:

$$\mathbf{g}(\mathbf{x}, \mathbf{z}) = \sigma(W_R \cdot \mathbf{x} + \Omega_R \cdot \mathbf{z} + \mathbf{b}_R) \quad (10)$$

Where W_R is the input-to-state weight matrix, Ω_R is the state-to-state weight matrix for the recurrent connections and the state biases are depicted by \mathbf{b}_R .

Both network architectures are initialized with weights randomly drawn from a uniform distribution $[-0.5, 0.5]$.

3 Results

Each simulation involved a configuration that varied the following parameters,

- The size of the alphabet (values: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
- The length of the sequence (values: 6, 10, 20, 30, 40, 50, 75, 100)
- The size of the motif in proportion to the sequence length (values: $\frac{1}{5}, \frac{1}{3}, \frac{1}{2}$)
- The proportion of the motif that contains wildcards (values: 0, $\frac{1}{5}, \frac{1}{3}, \frac{1}{2}$)
- The number of network hidden nodes - state dimensions (values: 5, 10, 20)
- The number of groups in the K-means clustering (values: 4, 6, 8, 10, 12)

For each configuration, we randomly generated 500 sequences of which half was positive (containing the motif) and half was negative. The motif pattern was also generated randomly for each configuration. Each and every configuration was tested using 10 differently initialized networks. For configurations that are neutral with respect to some parameters, outcomes for all values of such parameters were collected and the average result is reported.

First, a slight decrease in entropy was noted for both types of networks when the number of state dimensions was increased. Similar observations hold for the number of groups. As expected, the average entropy goes down with an increased number of groups (the extreme that each sequence gets its own group would result in perfect discrimination but poor generalization to novel sequences).

We tried a large number of configurations with the common observation that for recurrent networks most position dependent motifs are readily accessible without training. The result that recurrent networks outperform feed forward networks is consistent for the sequence lengths we tested (see Figure 2). The average entropy was around 0.1 for recurrent networks and close 0.8 for the feed forward network (an entropy of 1.0 would indicate random organization). When the motif was allowed to move across the sequence the problem was much more difficult for both architectures (the average entropy was 0.95 for most configurations). However, at least for short sequences and small number of symbols, the recurrent network was able to accommodate some of the invariance without being specifically trained to do so (see Figure 2).

To further elucidate the scalability of the results we made a detailed analysis of the entropy for various numbers of symbols (see Figure 3). With the number of symbols, the specificity of motifs in the data increases too. Generally, the entropy increases slightly with an increased number of symbols.

The introduction of gaps in motifs (wildcards) resulted in a minor increase of the entropy for both architectures (see Table 1).

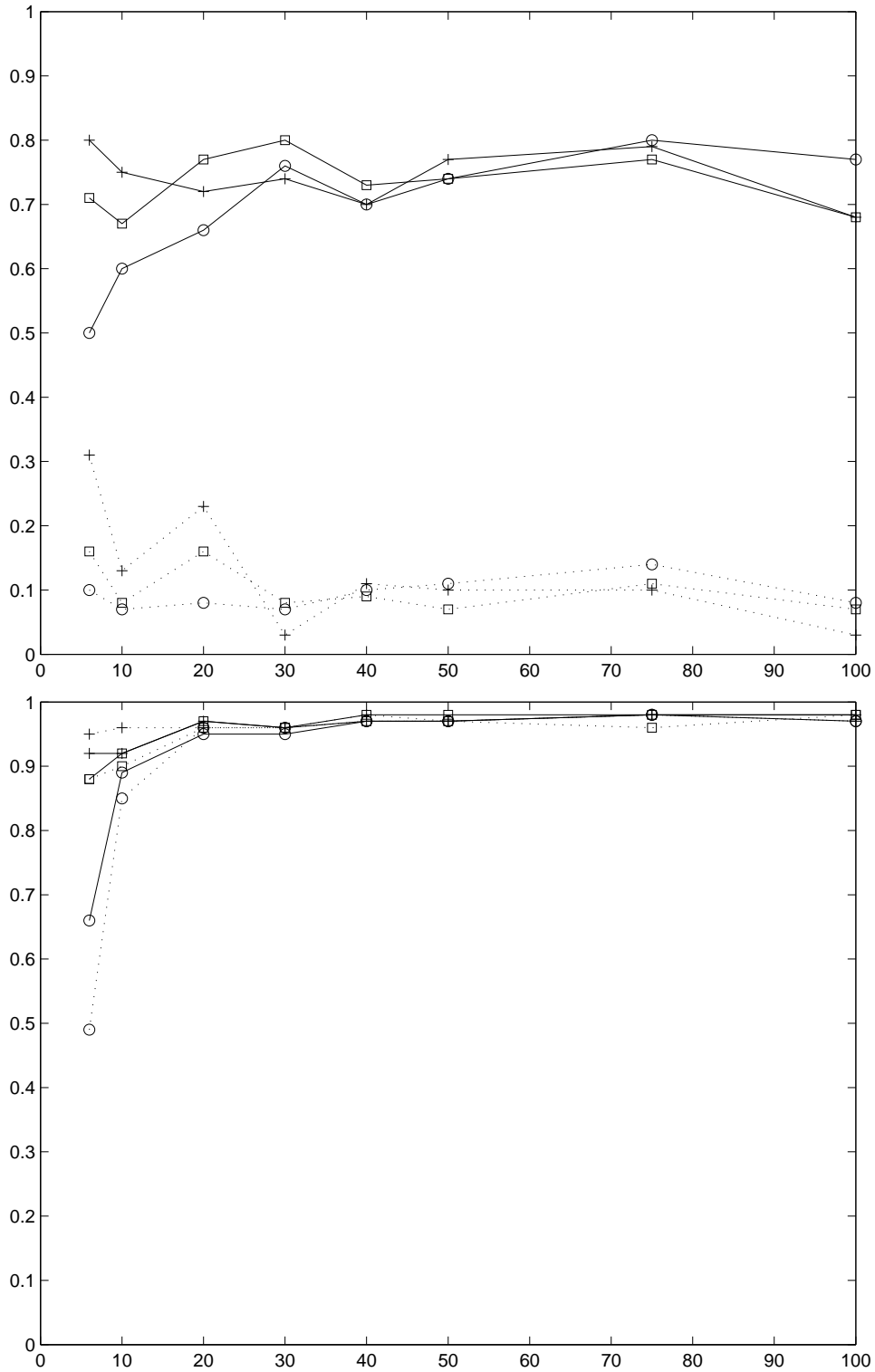


Figure 2: The average entropy (y-axis) for the two types of motifs (position dependent on top, shift invariant below) and for the two network types (solid line for feed forward networks, dotted line for recurrent networks) when the length of sequence is varied (x-axis). The motifs were set to half the sequence length. 1/3 of the motif symbols were wildcards. The number of symbols was varied between 2 (circles), 4 (squares) and 20 (pluses). All networks were equipped with 5 state units and $K = 10$.

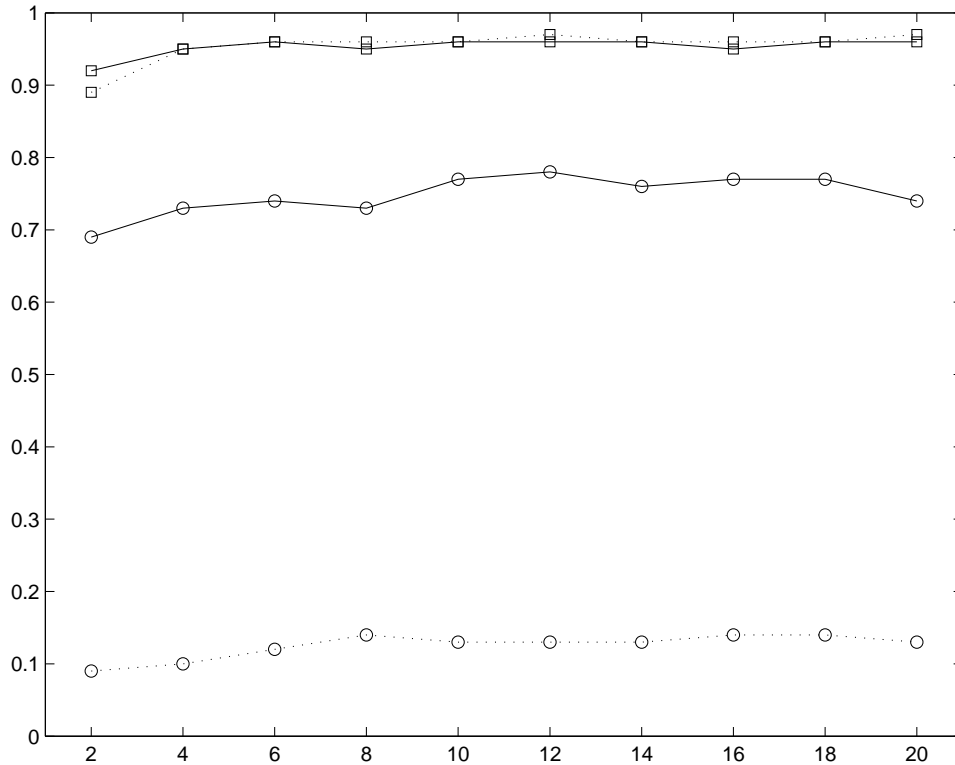


Figure 3: The average entropy (y-axis) for the two network types (solid line for feed forward networks, dashed line for recurrent networks) when the number of symbols is varied (x-axis). Entropy for position dependent motifs is marked with circles and shift invariant motifs are marked with squares. The sequence lengths were $L \in \{6, 10, 20, 30, 40, 50, 75, 100\}$, and the motif was adjusted to be half the sequence length. 1/3 of the motif consisted of wildcards. All networks were equipped with 5 state units and $K = 10$.

Motif-length	Wildcards			
	0	1/5	1/3	1/2
10 (1/5)	0.87 : 0.04	0.89 : 0.19	0.90 : 0.20	0.92 : 0.29
17 (1/3)	0.78 : 0.04	0.83 : 0.04	0.86 : 0.11	0.90 : 0.14
25 (1/25)	0.61 : 0.24	0.71 : 0.11	0.75 : 0.10	0.82 : 0.18

Table 1: The average entropy of the two network architectures (FN : RN) discriminating between sequences with position-dependent motifs, for varying sizes of motifs and proportions of wildcards. All sequences were 50 symbols long. Number of symbols was varied between 2, 4 and 20. All networks were equipped with 5 state units and $K = 10$.

4 Discussion

When a sequence space is large and sparse (long sequences consisting of a large set of symbols) it is essential for a machine learning algorithm to receive some guidance to “meaningful” patterns. The architectural bias allows the algorithm to search a constrained portion of the problem space.

Performing a comparative evaluation of the bias’ of recurrent and feedforward neural networks is intrinsically difficult. The common VC dimension technique will not work for recurrent networks with variable length input sequences [3]. In accordance with previous work [9, 3, 8], we have opted to perform the evaluation on the networks prior to training. After being presented with a sequence, the activation values of the hidden nodes are sampled and treated as individual dimensions of a state space for analysis. Proximity of two responses in this space indicates that the network is biased to consider the input sequences similar.

The state space of the recurrent network *architecture* reflects the presence of a bias. This bias makes sequential patterns, here expressed as motifs, more accessible to the recurrent than to the feed forward network architecture. Furthermore, the recurrent network seems to cope with domains which are extremely high-dimensional – the distinct organization of sequences at the state layer is almost unaffected with an increase of input dimensionality. We observe that short gaps in patterns (wildcard symbols) cause only small perturbations in the organization of states. However, when motifs are allowed to move freely in the sequence, considerably worse recognition can be expected. The recognition of shift-invariant patterns was not addressed specifically in previous work [9, 3, 8]. Our results indicate that if invariance is present it is wise to keep the number of symbols small and sequence length short (cf. work on learning complex formal languages [7], for a brief review and illustration of limitations).

So far undeservably little attention from bioinformatics has been paid to recurrent networks. Chiefly, Baldi and colleagues [1, 6] have explored a bi-directional variant of the recurrent network for protein structure prediction. A bi-directional recurrent network is – from the perspective of generating states – equivalent to two separate recurrent networks. Each network is dedicated to traversing the sequence in one direction (upstream or downstream). The results presented herein provides an explanation to the improvements noted by Baldi and colleagues (comparing with feed forward networks themselves).

It is important to acknowledge that the number of weights (free parameters) is much lower for the recurrent network compared to the feed forward network operating on the same sequence space. We can thus generally expect better generalization performance from a recurrent network after training – if both architectures accommodate training data equally well. On a cautionary note, it remains to be shown that training can be based on, and reinforce such patterns. It is well-known that learning algorithms for recurrent networks suffer from some deficiencies of learning long-term dependencies. However, with the emergence of algorithms such as Long Short Term Memory [4] the outlook is promising.

References

- [1] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15:937–946, 1999.

- [2] M. Christiansen and N. Chater. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23:157–205, 1999.
- [3] B. Hammer and P. Tino. Recurrent neural networks with small weights implement definite memory machines. *Neural Comp.*, 15(8):1897–1929, 2003.
- [4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [5] J. F. Kolen. Recurrent networks: State machines or iterated function systems? In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 203–210, Hillsdale, NJ, 1994. Erlbaum Associates.
- [6] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins*, 47:228–235, 2002.
- [7] J. Schmidhuber, F. Gers, and D. Eck. Learning nonregular languages: A comparison of simple recurrent networks and LSTM. *Neural Comp.*, 14(9):2039–2041, 2002.
- [8] P. Tino, M. Cernansky, and L. Benuskova. Markovian architectural bias of recurrent neural networks. *IEEE Transactions on Neural Networks*, 15(1):6–15, 2004.
- [9] P. Tino and B. Hammer. Architectural bias in recurrent neural networks: Fractal analysis. *Neural Comp.*, 15(8):1931–1957, 2003.