A Bayesian Network for Temporal Segmentation

Roland Orre and Anders Lansner

SANS, Dept. of Numerical Analysis and Computing Science
Royal Institute of Technology, S–100 44 Stockholm, Sweden
E–mail: orre@sans.kth.se, FAX: +46–8–790 09 30

Abstract
A recurrent network which segments an unlabeled externally timed sequence of data is presented. The proposed method uses a Bayesian learning scheme earlier investigated, where the relaxation scheme is modified with a few extra parameters, a pairwise correlation threshold and a pairwise conditional probability threshold. The method studied is able to find start and end positions of words which are in an unlabeled continuous stream of characters. The robustness against noise during both learning and recall is studied.

1 INTRODUCTION

The segmentation problem is fundamental in pattern recognition. Given data with a sequential/temporal behaviour this shows up as the *temporal chunking problem* [1] which may be illustrated by the example:

$$thisisacontinuousstreamofdatathatispossibletoreadwithoutseparators$$

Here, we want unfamiliar lists of familiar items (characters) presented sequentially to be recognized as new items (words). In the first place just the characters are familiar. When we have seen different lists several times we will also recognize the words as familiar items. The method presented here detects segmentation points between words. Conceptually this means that we have grouped a sequence of elementary items into a new, composite item.

2 METHODS

2.1 Learning and Recall

The learning rule used for the ANN was earlier investigated [2]. The network can be characterized as a recurrent Hopfield type with graded output units. Weights, biases (1) and the transfer function (4) for the units are derived from Bayes' rule.

$$\beta_q = \log p(i) \quad (1)$$

$$w_{iq} = \log \frac{p(i|q)}{p(i)} = \log \frac{p(q\&i)}{p(i)p(q)} = \log \frac{p_{qi}}{p_i p_q} \quad (2)$$

$$s_q = \beta_q + \sum_{h \in A} w_{hq}\pi_h \quad (3)$$

The neurons sum their inputs as in (3) where s_q is the support of the receiving unit q, w_{hq} is the weight from unit h, (in the set of active units A) to unit q, β_q is the bias and π_h is the output from h. The relaxation process is performed as a statistical inference where explicit time is simulated by letting a *dynamic* support E_q being charged as a "leaky integrator" of the support s_q. An output value π_q is a hypothesis of $p(q|A)$ and is obtained by exponentiating the dynamic support E_q as is shown in (4). Here we have calculated the weight values as in (5).

$$\pi_q = \begin{cases} 1 & E_q \geq 0 \\ e^{E_q} & \beta_q \leq E_q < 0 \\ 0 & E_q < \beta_q \end{cases} \qquad (4) \qquad\qquad w_{iq} = \begin{cases} 0 & p_i < p_\nu \vee p_q < p_\nu \\ -log\ C_\theta & p_{iq} < p_\theta \\ -log\ C_\sigma & p(q|i) < p_\sigma \\ log \frac{p_{iq}}{p_i p_q} & otherwise \end{cases} \qquad (5)$$

The threshold p_ν in (5) acts as a noise limiter and has been set to a low constant value. The p_θ is a limit for correlations between units. A high p_θ causes uncommon patterns to be forgotten while p_σ is more independent of how frequent a certain pattern segment is. C_θ and C_σ are some large numbers that are characteristic for the network size and the number of patterns stored. They are not critical but should typically be set so $\forall i \forall q : (\frac{1}{C} < \frac{p_{iq}}{p_i p_q})$ to make the inhibition a monotonic function of the correlations.

2.2 Temporal Coding

The architecture used to map sequential information onto the recurrent network is illustrated in figure 1. The information is learned at every position. This is like a kind of translation invariant storage of the sequence.

2.3 Usage

One possibly way to use the detection of segmentation points (starts and ends) is illustrated by figure 2. The output from the segmentation algorithm can be used as a "print now" signal to a network that stores or recognizes each word at a fixed position.

2.4 Segmentation Principle

Assume that each temporal position, as in figure 1 is represented by a neural population, where each letter is coded with one unit. Further assume that the segmentation network has seen a noise free sequence like "`NEURALJUNIORNEURALEXPERTNEURAL`". The sequence consists of n words, in this case of equal length l, where the most frequent word occurs at least c times. These words are seen by the network in all positions. The following calculations depend on how the first and last word are treated. We assume here that a frequent word starts and stops a sequence. The probability for a unit to be active is, $p_\nu^u \geq (l(n-1)+1)^{-1}$ *i.e.* , a character has occurred at least once in a unique word and $p_\nu^c \geq c(l(n-1)+1)^{-1}$ or c times in a frequent word. The pairwise unit correlation p_{iq}, is $(l(n-1)+1)^{-1}$ within a unique word and $c(l(n-1)+1)^{-1}$ within a frequent word. To detect words that occur more frequently than their combinations we could use $p_\theta^c \geq c(l(n-1)+1)^{-1}$. If we also want to detect infrequent words combined with frequent words we could threshold the normalized correlation $p(i|q)$, as is expressed by p_σ. $p(i^c|q^c) \simeq \frac{p_\theta^c}{p_\nu^c} \simeq 1$ and $p(i^u|q^u) \simeq \frac{p_\theta^u}{p_\nu^u} \simeq 1$ while

$p(i^u|q^c) \simeq \frac{p^u_\theta}{p^c_\nu} \simeq \frac{1}{c}$. By setting the $p_\sigma \geq \frac{1}{c}$ in (5) we could make weights between units in frequent, and in infrequent, patterns inhibitory. When all words are about equally frequent we could set p_σ just above the greatest pairwise correlation between words. For n equally frequent words this implies that $\lim_{\frac{p_{qi}}{p_i p_q} \to 1} p_\sigma = \frac{1}{n^2}$.

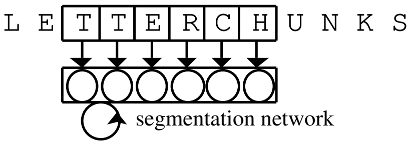

Figure 1: A continuous stream of data passes a tapped delay line that spreads the temporal information spatially over the network.

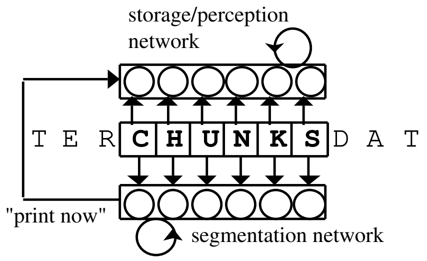

Figure 2: The segmentation network may signal "print now" to the actual storage or perception network when a segment start is found.

2.5 Segmentation Algorithm

A segmentation hypothesis S_τ (6), where τ is an index of the neural population that codes a character at a specific time-step, is propagated in parallel with the input sequence. S_T (8) is a sum of all S_τ. A decision about segmentation is done based on the current value of S_T and a decision threshold ξ (9). S_τ is based upon population activities A_τ close around each hypothesis point τ (6). A_τ is a normalized sum of the activity in population τ (7) where $n(\tau)$ is the number of units in this neural population.

$$S_\tau = \begin{cases} 0 & \tau = 0 \\ A_\tau & A_{\tau+1} < A_{\tau-1} - \xi \\ -A_\tau & A_{\tau+1} > A_{\tau-1} + \xi \\ 0 & otherwise \end{cases} \tag{6}$$

$$A_\tau = \frac{1}{n(\tau)} \sum_{i=1}^{n(\tau)} \pi_{i\tau} \tag{7}$$

$$S_T = \sum_{\tau=T-1}^{1} S_\tau \tag{8}$$

$$SEG = \begin{cases} BEGIN & S_T \geq \xi \\ END & S_T \leq -\xi \\ NO & otherwise \end{cases} \tag{9}$$

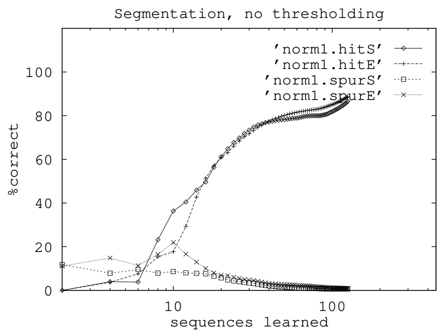

Figure 3: Random combinations of 17 words are learned and segmented. Upper curves show correct hits and lower show spurious hits.

3 EXAMPLE

Learn the sequences "NEURALJUNIOR", "NEURALEXPERT", "BAKERJUNIOR" and "BAKEREXPERT" on a network with six character populations as in figure 1. Each of the words "NEURAL", "JUNIOR", "EXPERT" and "BAKER" is twice as frequent as any of the combinations. Each sequence is learned at every position on the network. Biases and weights are calculated according to (1) and (5) and a relaxation is done for each position when the sequence is presented. In the following example a "_" means low output activity and a "+" stands for an ambiguous output hypothesis. To begin with we let the p_θ have a low value. We start by stimulating the network as in #1 below. For each time-step the network sees a 6 character long part of the sequence. After each stimulation in #1 we do a relaxation of the network and get outputs like in #2. We now do the same but before adjust p_θ to a higher value, like $p_\theta = 0.05$. We will then, after each relaxation, get the outputs in #3.

ex	t0	t1	t2	t3
#1	BAKERE	AKEREX	KEREXP	EREXPE
#2	BAKERJ	AKEREX	KEREXP	EREXPE
#3	BAKER_	AKER__	___EXP	__EXPE

The segmentation algorithm looks at the activity difference between adjacent populations. In example #3 it receives a strong indication for "R" in "ER_" to be an ending and "E" in "_EX" to be a start.

4 RESULTS

These experiments have been done by forming sequences of words picked from an English dictionary. Every next word in a sequence has been randomly selected with a certain probability. In figure 3 it is shown how the algorithm performs on a network tuned for pattern completion. The network learns an increasing amount of random sequences for a set of 17 words.

Segmentation performance, versus p_θ-level for noise free data, when just a few sequences are learned, is illustrated in figure 4. In figure 5 we see how the p_ν-level affects the performance. When varying the p_ν-level we could not reach the same performance as when varying the p_θ-level. When the p_θ was set to the optimal level the performance could not be improved, only decreased, by adjusting p_ν. This was also the case when some words were 15 times more frequent than the least common words. This would indicate that p_θ could be used for good segmentation performance but not p_ν.

One way to do segmentation using a recurrent network is to look at the distance between the stimulus pattern and the resulting pattern after relaxation. In figure 6 we compare the segmentation on noise free data with the *ideal* case where the words are learned at a fix position (f13) versus all positions (a13). In the former case the words were learned until no further improvement in segmentation could be achieved. The words were in this case of the same length as the network.

In figure 7 we see how the segmentation quantitatively depends on p_θ when we have noise in the learning data. A noise level of 0.4 means that the probability $p(\textit{wrong character}) = 0.4$ during stimulation for each population.

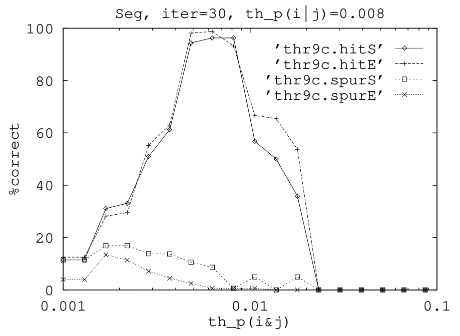

Figure 4: The segmentation performance dependency on the p_θ-level.

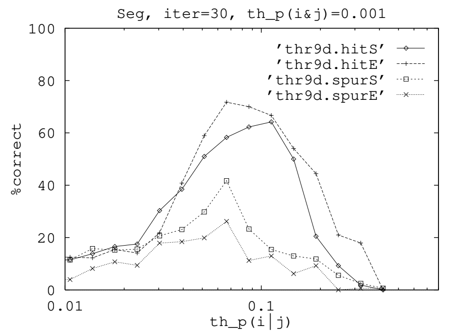

Figure 5: The segmentation performance dependency on the p_ν-level.

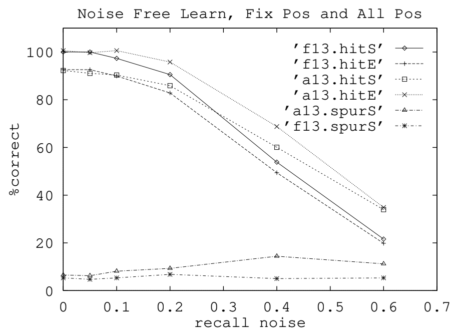

Figure 6: Segmentation when 13 words are learned at a fix position (f13) compared with when they are learned at all positions (a13).

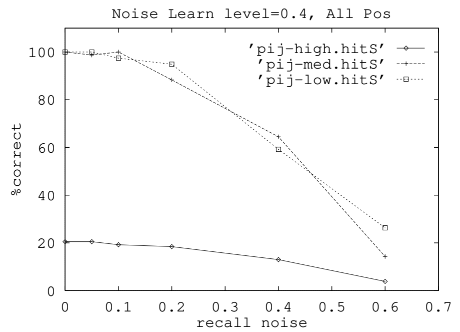

Figure 7: The performance of segmentation for three different p_θ-levels when there is noise (prob 0.4) in the training data.

References

[1] Grossberg, S. Unitization, automaticity, temporal order and word recognition. *Cognition and Brain Theory*, 7:263–283, 1984.

[2] Lansner, A. and Ekeberg, Ö. A one-layer feedback, artificial neural network with a bayesian learning rule. *Int. J. Neural Systems*, 1(1):77–87, 1989. Also extended abstract in Proceedings from the Nordic symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland.

[3] Schmiduber, J. Neural sequence chunkers. Tech. Rep. FKI-148-91, Institut für Informatik, Technische Universität München, 1991.

[4] Doutriaux, A. and Zipser, D. Unsupervised discovery of speech segments using recurrent networks. In Touretzky, D., Elman, J., Sejnowski, T., and Hinton, G., editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 52–61. Morgan Kaufmann, San Mateo, 1990.

[5] Lansner, A. A recurrent bayesian ann capable of extracting prototypes from unlabeled and noisy examples. In *Artificial Neural Networks, Proceedings ICANN-91*, pages 247–254. Royal Institute of Technology, Stockholm, Sweden, 1991.