ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 377-382

Generalization by structural properties from sparse nested symbolic data

Mikael Bodén (mikael.boden@ide.hh.se)
Halmstad University, Sweden.

Abstract. A set of simulations demonstrate that recurrent networks can exhibit generalization by abstraction from extremely sparse but structurally homogenous symbolic data. By cascading two recurrent networks – feeding the second network with discretized hidden states of the first – it is also possible to generalize according to complex structure. By automatic discretization the cascaded architecture assists in scaling up sequential learning tasks and offers explanations to the apparent systematicity and generativity of language use.

1 Introduction

Inducing means in recurrent networks for processing structurally complex languages (e.g. context-free and context-sensitive) is possible but in many cases difficult [9, 1, 4]. Available demonstrations are limited in at least two senses.

1. Only a few terminal symbols are employed. It is unclear if large-scale languages can be used for induction.

2. The dynamics, which indicates that the network is actually going beyond regular language processing, is associated with individual terminals rather than word classes. It is unclear if the network will extrapolate, i.e. utilize the same principles for processing other related symbols possibly appearing at new levels of structural embedding.

Single-step prediction learning is a learning strategy that has proven particularly useful for organizing continuous internal state spaces according to contextual and functional similarities of untagged language data. Given a sequence of words presented to the network, the network essentially learns to output the probabilities of words occurring next but in a manner quite distinct from conventional n-gram modelling: by concurrently developing abstractions in state space.

The automatic availability of such grammatical and semantical abstractions, embodied as activation clusters, led us to investigate how such abstractions could improve learning the dynamics previously only detected for individual words. In partial support, results in machine learning show that performance when classifying continuous features can be improved by discretization prior to induction [2].

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 377-382

2 A case for structure and abstraction

A learner faces examples presented sequentially and in random order from the simple context-free language $a^n b^n$ where $1 \leq n \leq 3$: $aabbabaaabbbaabba...$. If we collect statistics for combinations of the six previous letters (a 6-gram), the prediction task can be carried out to perfection. There would be 25 examples in such a table so to estimate the probabilities used for generating the strings would not take long.

A tabular approach like the 6-gram would not be inclined to process strings generated by $a^n b^n$ where $1 \leq n \leq 4$. With a window of 6 symbols, $aaaabbb$ would be matched with $aaabbb \rightarrow a$. Even if n was allowed to take any value x while collecting statistics, and sufficient storage for storing $2x$ letter entries is assumed, the statistical learner would not generalize to process $a^{x+1}b^{x+1}$, or any n beyond x. Natural language contains many examples where embedding occurs. The inability to extrapolate beyond data led some to argue that a mechanism based on structure sensitivity (and recursion) is required [5].

Natural language also make use of a large number of components and, thus, a learner must abstract. We extend $a^n b^n$ by having two as, and two bs (a grammar which we refer to as a diversified center embedded language, DCEL). The statistically driven n-gram learner now needs to store $25 \cdot 2^6 = 1600$ entries in its table. More data is required to approximate the probabilities. If there are three variants of each a and b in DCEL 18225 entries are required. If 25 variants of each a and b is used, 6,103,515,625 entries are needed[1] and massive data sets are required. Hidden Markov Models (HMMs) can improve on tabular representation. By attaching transition probabilities to states (either observed or hidden) the amount of memory required to represent the statistics is significantly reduced (assuming the right abstractions are found). The catch is that HMMs are still finite and are consequently not able to go generalize beyond the level of embedding available in the training data.

Recurrent networks have been successful at learning small language fragments. As exemplified above there are two aspects to generalization of interest.

1. Abstraction. Through learning, networks group inputs in state space which are used similarly, and are thus able to generalize to other words within groups.

2. Dynamics for processing embedded structure. From a learning perspective there is a trade-off between available resources and the generality of mechanism. If state space is large and training set is small the network basically stores specific information about each sample, e.g. in a tabular fashion. Otherwise, the network is forced to seek other (more general) ways of representing the mechanism. A general mechanism may extrapolate beyond the limits of the training set. In Long Short Term Memory (LSTM) networks and some Simple Recurrent Networks (SRNs; [3]), simple linear counters are induced [4] for $a^n b^n$. In some SRNs and Sequential Cascaded Networks (SCNs; [8]) oscillating or spiralling dynamics is induced [1].

As pointed out by Hadley [5] and Marcus [6], generalization in humans sometimes goes beyond the regularities inferred with recurrent networks. As an example, if you

[1] "Don't care" markers can reduce this number.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 377-382

have heard “Smith fleedled Jones” you would infer that it is grammatical to say “Smith fleedled Belanger” (even though you have never seen ”Belanger” as the object in that context) [6]. In error based learning the likelihood that Belanger would be predicted (in that position) by the network is decreased every time another word occurs instead. If Belanger never occurs (in that particular training context), the probability eventually goes down to zero [6].

As soon as there are other relations in which Belanger plays similar roles as Jones (and Smith) it is quite likely that the network forms a “cluster” for these words. Some of the cluster members are closer than others – analogous to “typicality” in categorization.

3 Simulations

The idea this paper presents is simple: the “clusters” found at a hidden layer of a recurrent network can be segmented and discretized using a self-organizing map (or other unsupervised classifier) and be used as inputs and targets of another recurrent network, similarly trained to predict the next input (cf. [7] who used a cascaded setup to abstract from low level events in a robotic environment, and [10] who trained, on a variety of domains, an extra network to produce class representations (out of target data) which the main network is able to predict). Effectively, through continuous training, the second network approximates a coarser probability distribution over a different set of discrete variables (e.g. noun, verb, etc) at an abstraction level determined by the width of the segmentation network.

The abstracted variables can be mapped back to its members without taking into account the context by a simple association.[2] At the expense of specificity, a higher degree of generality is supported by the network.

The primary network is here an SRN consisting of two hidden layers of which only the second is recurrent. The recurrent layer feeds back to itself in a fully connected manner. The prediction task encourages the primary network to form context-independent representations at the first hidden layer which are then put into context at the second layer. The first hidden layer is used for segmentation since each input will, when weights are fixed, result in the same pattern independent of which context in which the input appears. The second network is also an SRN. Both SRNs are trained using backpropagation through time.

Specific network architectures were designed to address the question if the second recurrent network (through its own learning task and its own independent dynamics) supplies additional information to support a wider range of generalizations – including abstraction and dynamics for structural extrapolation. For benchmarking, conventional SRNs with various number of hidden layers and various number of hidden units were studied. Second, cascaded SRNs were evaluated. All networks were trained to predict the next word in strings presented sequentially with no specific end-of-string marker. In a center embedded language, the second half of each string is partly determined by the first half. Performance can thus be measured by the networks’ ability to

[2] Mapping abstractions back to specific members is not investigated here since we are dealing with a homogeneous data set. Simulation with a heterogeneous language data set which utilizes this feedback is currently underway.

	Mean errors (SD)		
Network	Train	Test	Test n=11 or 12
50/3R/50	0.50 (0.06)	0.55 (0.06)	0.84 (0.31)
50/2/2R/50	0.37 (0.01)	0.38 (0.01)	1.02 (0.17)
50/10R/50	0.86 (0.09)	0.92 (0.11)	1.67 (0.37)
50/10/10R/50	0.77 (0.06)	0.83 (0.06)	1.74 (0.51)
50/2R/50	0.40 (0.04)	0.45 (0.03)	0.95 (0.28)

Table 1: The RMS errors for 5 runs of each configuration (standard deviations shown within parentheses. The networks are described using the number of units in each layer (input to output). R indicates that the layer is recurrent.

predict the next symbol (or group of symbols) at any time after the first half plus one symbol of the string has been presented.

A set of 300 strings from 25-DCEL with $1 \leq n \leq 10$ (which encompasses a total of $9.1 \cdot 10^{27}$ possible strings) were used for training and another 300 strings (non-overlapping but generated by the same grammar) were used for testing.[3] In addition, test strings which go beyond level 10 were used for assessing the networks ability to generalize in terms of structural properties. Each symbol in each string is represented as a one-hot code, i.e. mutually orthogonal bit vectors.

Each network was trained for 100 rounds through the training set (300 strings) and updated after each completed string. Various learning rates were tested but significant differences were not observed. In a conventional SRN we used 0.05 for the results presented below. In the cascaded architecture we used 0.3. BPTT unfolded for 10 timesteps.

It turns out that when training the SRN on 25-DCEL the SRN abstracts the a-components and the b-components but is incapable of correctly predicting string endings even within the training set. The test errors were at the level of the training error (see Table 1). Hence, the network does not focus on specific strings or symbols. Instead, generalization is based on groups of symbols as discovered through training (verified by cluster analysis on hidden activation). This basis for generalization is not sufficient to address new levels of embedding. In Table 1 the errors on a test set which exclusively contains strings with $n \in 11, 12$ are shown. The errors are radically higher than the training- and test sets with shorter strings in each case. Hence, the network is not able to extrapolate to levels beyond 10.

The cascaded recurrent network was studied in the following way. A conventional 50/2/2R/50 SRN was trained on the 300 strings. At each presentation the first hidden layer (preceeding the recurrent layer) was fed into a self-organizing map (SOM) which had 2 outputs. The SOM produces a bit string $0, 1$ or $1, 0$ depending on the hidden activity. In all 10 networks the SOM produced (about half-way through training) one output for $a_1, a_2, ..., a_{25}$ and another for $b_1, b_2, ..., b_{25}$. Simultaneously, a secondary 2/2R/2 SRN was trained on the outputs of the SOM to predict the next word.[4] It

[3] The distribution of the training and test set was skewed so that shorter strings were more frequent.

[4] Since the primary SRN produces the target output of the secondary SRN a delay is imposed on the

should be emphasized that the secondary SRN is trained to produce outputs on basis of inputs, both at a different granularity than in the original data set. This reduction in resolution turns out to have a marked effect on the ability to process deeper levels of embedding.

Of the 10 cascaded networks, 7 managed to process strings to level 10 and beyond. The best managed to correctly predict strings with $n \in 1, ..., 27$. On average the cascaded network managed to process strings up to $n = 15.3$. Qualitatively, the dynamics shows that the networks extrapolate. The two abstractions (the as and the bs) can be thought of as two autonomous dynamical systems (the input is fixed). For all of the networks that extrapolate, the first system oscillates towards an attractive point in a fractal fashion. When the second system starts (when the first b input is presented) the activation starts to diverge (by oscillation) from a repelling fixed point located in a separate part of the state space. The offset from the first fixed point determines the starting point for the second phase and indirectly determines how many oscillations that are required to reach the decision hyperplane which signals that the next input will be an a. The behavior of the systems can also be described by linearizing each dynamical system at the fixed points and calculating the corresponding eigenvalues and eigenvectors. It turns out that all 7 successful networks share the following properties.

- The largest absolute eigenvalues of the two systems are inversely proportional to one another. In practice one of them is usually around -0.7, the other around -1.4. The inverse proportionality ensures that the rate of contraction around the fixed point of the a system matches the rate of expansion around the fixed point of the b system ([9], p. 23).

- The fixed point of the a system lies on the axis given by the eigenvector corresponding to the smallest absolute eigenvalue of the b system (the direction in which the fixed point attracts) when projected through the second fixed point ([9], p. 23). This configuration basically entails that the first thing that happens in the b system is a long-distance state shift along its eigenvector to a part in state space close to the b fixed point. The positioning of the eigenvector ensures that the final state of the a system (which identifies the a-count) correctly sets the starting point for the expansion phase.

The output of the secondary SRN can be mapped back to the set of members. However, this mapping cannot be based on dependencies between individual members. Instead, if a property holds for a subset of members of the same group, that property holds for all of the members. Circularly, groups are only formed with those that actually share a substantial proportion of properties. In principle, the segmentation (implemented by the SOM) can happen at many levels. Consequently, generalization can be geared at different levels of abstraction.

4 Conclusion

In language processing there are at least two essential aspects to generalization: abstraction and structural extrapolation. Previous work has focused on one or the other.

training of the secondary SRN.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 377-382

We have shown that recurrent networks are well-suited to abstract symbols according to their similarities in a structurally homogeneous prediction task. However, the same networks are unable to reach the same level of performance on structural extrapolation as networks trained directly on abstract data. By feeding discretized state patterns to a second recurrent network we are able to associate the same dynamics previously observed only in small-scale experiments to a structurally similar but much larger data set.

Apart from scaling up to larger domains, the simulations indicate a potential resort for addressing the apparent systematicity of language in a completely unsupervised fashion. Hadley's "strong systematicity" requires that a token should be appropriately processed in novel syntactic positions, at any level of embedding [5]. Admittedly, the structure of the tested domain is homogeneous and has only a fraction of the complexity of natural language. However, the level of generalization that is achieved fulfills the requirements of strong systematicity.

References

[1] M. Bodén and J. Wiles. Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Sci*, 12(3):197–210, 2000.

[2] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. Russell, editors, *Machine Learning: Proc Twelfth International Conference*, pages 194–202, San Francisco, CA, 1995. Morgan Kaufmann.

[3] J. L. Elman. Finding structure in time. *Cognitive Sci*, 14:179–211, 1990.

[4] F. A. Gers and J. Schmidhüber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Trans Neural Networks*, 2001, 12(6).

[5] R. F. Hadley. Systematicity in connectionist language learning. *Mind and Language*, 9(3):247–272, 1994.

[6] G. F. Marcus. Language acquisition in the absence of explicit negative evidence: can simple recurrent networks obviate the need for domain-specific learning devices? *Cognition*, 73:293–296, 1999.

[7] S. Nolfi and J. Tani. Extracting regularities in space and time through a cascade of prediction networks: The case of a mobile robot navigating in a structured environment. *Connection Sci*, 11(2):125–148, 1999.

[8] J. B. Pollack. The induction of dynamical recognizers. *Machine Learning*, 7:227, 1991.

[9] P. Rodriguez, J. Wiles, and J. L. Elman. A recurrent neural network that learns to count. *Connection Sci*, 11(1):5–40, 1999.

[10] J. Schmidhuber and D. Prelinger. Discovering predictable classifications. *Neural Comp*, 5(4):625–635, 1993.