ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 383-388

Estimating Probabilities for Unbounded Categorization Problems

James Henderson

University of Geneva, Computer Science Department
24, rue Général Dufour, CH-1211 Genève 4, Switzerland

Abstract. We propose two output activation functions for estimating probability distributions over an unbounded number of categories with a recurrent neural network, and derive the statistical assumptions which they embody. Both these methods perform better than the standard approach to such problems, when applied to probabilistic parsing of natural language with Simple Synchrony Networks.

1 Introduction

Recurrent networks have the advantage over feed-forward networks that they can compute mappings from arbitrarily many input patterns to arbitrarily many output patterns. The work presented in this article focuses on the issues that arise with an unbounded number of output patters. In particular, we are concerned with the problem of estimating a probability distribution over an unbounded set of mutually exclusive categories. We propose two output activation functions which can be used for such tasks, and in each case derive the statistical assumptions which are necessary to prove that training networks which use these functions will result in estimates of the desired probability distributions. We then experimentally evaluate these activation functions in an application of a recurrent network architecture (Simple Synchrony Networks [4]) to probabilistic parsing of real natural language sentences. We find that both of these activation functions perform better than the standard approach of converting the task to a sequence of bounded categorization problems.

As an example of an unbounded categorization problem, consider the decision an incremental natural language parser must make when it reaches the last word of the sentence "John said Mary left yesterday." The parser must choose whether "yesterday" modifies "said" or "left". Each alternative modification can be thought of as a category, and they are mutually exclusive. In general there can be an unbounded number of verbs in such a sentence, so there can be an unbounded number of modifications which the parser must choose between. To make this choice, we want to be able to estimate a probability distribution over an unbounded number of mutually exclusive categories.

The reason a recurrent network can produce an unbounded number of out-

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 383-388

puts is that it can be run for an unbounded amount of time, and produce different outputs at each time.[1] We assume that one output y_i is produced at each time i, from one hidden layer activation pattern $\mathbf{z}_i$.[2] We will denote the total input as $\mathbf{x}$ and use Y_i to denote whether the output category associated with time i is the correct output category. Our objective is to have y_i to be an estimate of $P(Y_i|\mathbf{x})$. In the above example, the potential modifications are the different i, Y_i represents the correctness of modification i, and $\mathbf{x}$ is the whole sentence plus the question of how to parse "yesterday". We want y_1 to be an estimate of the probability that "yesterday" modifies "said", and y_2 to be the estimate for "left".

2 Using a Sequence of Bounded Categorizations

The standard method in statistical modeling and neural networks for handling unbounded categorization problems is to convert them into an unbounded sequence of bounded categorization problems. The first step in this process is to choose an ordering between the categories, which we will denote $c_1, \ldots, c_n$. The categorization probabilities which are then estimated are each conditional on the correct category not being any of those earlier in the sequence. These probabilities can each be estimated independently, because the fact that the alternatives are mutually exclusive has been taken care of in the conditioning.[3] The original probabilities can then be computed from these estimates by multiplying in accordance with the chain rule for conditional probabilities:[4]

$$\begin{aligned} P(Y_{c_2}|\mathbf{x}) &= P(Y_{c_2}|\neg Y_{c_1}, \mathbf{x}) \times (1 - P(Y_{c_1}|\mathbf{x})) \\ &\ldots \\ P(Y_{c_n}|\mathbf{x}) &= P(Y_{c_n}|\neg Y_{c_1}, \ldots, \neg Y_{c_{n-1}}, \mathbf{x}) \times \\ &\quad (1 - P(Y_{c_1}|\mathbf{x})) \times \ldots \times (1 - P(Y_{c_{n-1}}|\mathbf{x})) \end{aligned}$$

The main disadvantage of this method is that the sequential ordering chosen for the conditioning biases the estimate. When the estimates are smoothed, as with neural networks, then small probabilities will tend to be overestimated, thereby reducing the probability assigned to categories later in the sequence. This will bias the estimates towards categories earlier in the sequence, and, for most tasks, this bias will be detrimental no matter what ordering we choose. In the next section we will propose methods which do not require any ordering, and thus do not introduce this bias.

[1] We are using the word "time" simply to refer to the different states which a recurrent network passes through. Everything discussed here is equally applicable to the graph-based generalization of time, as discussed in [2].

[2] All this work could be generalized to any finite number of output categories per time, but we do not do so here in the interest of space.

[3] The unbounded amount of information in the conditional of the probability can be handled with any of a variety of methods, including making explicit independence assumptions and running a recurrent network on the sequence $c_1, \ldots, c_n$.

[4] Note that $P(Y_{c_1}|\mathbf{x})$ is estimated directly and $P(Y_{c_n}|\neg Y_{c_1}, \ldots, \neg Y_{c_{n-1}}, \mathbf{x})$ is actually equal to 1 because c_n is the only possible alternative remaining.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 383-388

3 Estimating the Probabilities Directly

The standard methods for estimating probabilities with neural networks apply only to feed-forward networks. However, for any (discrete time) recurrent network applied to a particular input, we can convert it into an equivalent feed-forward network with one copy of the recurrent network for each time step in the input [5]. The difficulty arises because the standard proofs for showing that a trained neural network will estimate a probability distribution assume that all the outputs are computed from the same hidden layer activation pattern [1]. But for recurrent networks the different outputs y_i are produced at different times from different hidden activation patterns $\mathbf{z}_i$. The fundamental issue in deriving new activation functions for estimating probability distributions over an unbounded set of categories is how to handle an unbounded number of hidden activation patterns.

3.1 An Unbounded Logistic Sigmoid Function

When we train a network to compute an output, the hidden layer learns a representation which is optimized for computing that particular output given the input. Thus it is natural to assume that a given output y_i can be accurately computed from its hidden activation pattern $\mathbf{z}_i$ without requiring additional information about the input $\mathbf{x}$. This independence assumption, plus the fact that $\mathbf{z}_i$ is a function of $\mathbf{x}$, results in the following approximation:

$$P(Y_i|\mathbf{x}) \;=\; P(Y_i|\mathbf{z}_i,\mathbf{x}) \;\approx\; P(Y_i|\mathbf{z}_i) \;=\; \frac{p(\mathbf{z}_i|Y_i)P(Y_i)}{p(\mathbf{z}_i)} \tag{1}$$

In this model we are not assuming any ordering or other differentiation between the different category indexes i, so the prior probability distribution over these categories $P(Y_i)$ must be uniform. Therefore $P(Y_i) = 1/n$, where n is the number of categories. We can write $p(\mathbf{z}_i)$ as a sum over output cases: $p(\mathbf{z}_i|Y_i)(1/n) + \Sigma_{j\neq i}p(\mathbf{z}_i|Y_j)(1/n)$. The first element of the sum is the numerator in the previous formula. The second element represents the probability of computing the hidden activation vector $\mathbf{z}_i$ in situations where Y_i is false. We assume that these later probabilities are only dependent on the fact that Y_i is false, and not dependent on the specific index j of the correct category. Thus:

$$p(\mathbf{z}_i) \;\approx\; p(\mathbf{z}_i|Y_i)(1/n) + p(\mathbf{z}_i|\neg Y_i)(1-1/n) \tag{2}$$

These assumptions allow us to estimate $P(Y_i|\mathbf{x})$ with a single output unit y_i, in a similar way to the two-category case for finite mappings:

$$P(Y_i|\mathbf{x}) \;\approx\; \frac{p(\mathbf{z}_i|Y_i)(1/n)}{p(\mathbf{z}_i|Y_i)(1/n) + p(\mathbf{z}_i|\neg Y_i)(1-1/n)} \tag{3}$$

$$\approx\; \frac{1}{1+(n-1)\exp(-a_i)} \quad \text{where} \quad a_i = \ln\left(\frac{p(\mathbf{z}_i|Y_i)}{p(\mathbf{z}_i|\neg Y_i)}\right) \tag{4}$$

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 383-388

Equation 4 is the logistic sigmoid activation function, except that the weighting of $(n-1)$ has the effect of shifting the function to the right by $\ln(n-1)$.

Because the above independence assumptions are not exactly correct, it is sometimes desirable to add a normalization factor by taking advantage of the fact that $\Sigma_k p(Y_k|\mathbf{x}) = 1$. To normalize, equation 4 simply needs to be divided by $\Sigma_k 1/(1+(n-1)\exp(-a_k))$. This is optional, but it has been found to help in the later stages of training, as is done in the experiments in section 4.

For the two-category case for finite mappings, the logistic sigmoid activation function is used with the cross-entropy error function. This ensures that after training the outputs do indeed estimate the desired posterior probabilities, assuming that the class-conditioned probability of the hidden activation vector is in the exponential family of distributions [1]. Translating this to our case, we want to ensure that the output y_i estimates $P(Y_i|\mathbf{x})$, assuming that $p(\mathbf{z}_i|Y_i)$ is in the exponential family. $P(Y_i)$ and $P(Y_{\bar{i}})$ are independent of the input so they are constant factors. To complete the parallel with the finite case we simply need to assume that $p(\mathbf{z}_i|\neg Y_i)$ is also in the exponential family. Given these assumptions, the proof is the same as for the finite case.

The training algorithm for the unnormalized version of this model is the same as for the finite case. We can see this by noting that equation 4 is equivalent to the usual logistic sigmoid applied to $b_i = a_i - \ln(n-1)$, and $\partial b_i/\partial a_i = 1$. For the normalized version a factor of one minus the unnormalized output needs to be added to $\partial E/\partial a_i$.

3.2 An Unbounded Softmax Function

As an alternative to the unbounded sigmoid activation function, we consider what statistical assumptions would be necessary to derive a version of the normalized exponential, or "softmax", activation function. This is the proper activation function to use in the case of a bounded number of categories greater than 2, so it is natural to expect it would generalize to the unbounded case. For this derivation we do not assume that one hidden layer activation pattern is sufficient to estimate a given category's probability, but instead make the much weaker assumption that the total set of all hidden patterns is sufficient. This alternative provides us with a normalization factor equal to $p(\mathbf{z}_1, \ldots, \mathbf{z}_n)$:

$$P(Y_i|\mathbf{x}) \approx P(Y_i|\mathbf{z}_1, \ldots, \mathbf{z}_n) = \frac{p(\mathbf{z}_1, \ldots, \mathbf{z}_n|Y_i)P(Y_i)}{\sum_k p(\mathbf{z}_1, \ldots, \mathbf{z}_n|Y_k)P(Y_k)} \quad (5)$$

We then make the main independence assumption of this method; we assume that the probability distributions over hidden activation patterns are conditionally independent, given the correct category i:

$$p(\mathbf{z}_1, \ldots, \mathbf{z}_n|Y_i) \approx p(\mathbf{z}_i|Y_i) \times \prod_{j \neq i} p(\mathbf{z}_j|Y_i) \approx p(\mathbf{z}_i|Y_i) \times \prod_{j \neq i} p(\mathbf{z}_j|\neg Y_j) \quad (6)$$

In the second step we have assumed, as for the previous function, that the only information about Y_i which is relevant to $\mathbf{z}_j$ is the fact that Y_j is false.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 383-388

Using, as above, the uniform prior $P(Y_i) = 1/n$, and substituting formula 6 into formula 5 and simplifying, we get:

$$P(Y_i|\mathbf{x}) \approx \frac{p(\mathbf{z}_i|Y_i)}{p(\mathbf{z}_i|\neg Y_i)} \Big/ \sum_k \frac{p(\mathbf{z}_k|Y_k)}{p(\mathbf{z}_k|\neg Y_k)} \quad (7)$$

$$\approx \frac{\exp(a_i)}{\sum_k \exp(a_k)} \quad \text{where} \quad a_i = \ln\left(\frac{p(\mathbf{z}_i|Y_i)}{p(\mathbf{z}_i|\neg Y_i)}\right) \quad (8)$$

Equation 8 is precisely the softmax activation function, applied to an unbounded number of categories.

As with the sigmoid activation function, cross-entropy error is the appropriate error function to use for the softmax activation function when you wish to estimate a probability distribution over a bounded number of categories [1]. As for the unbounded sigmoid function, generalizing this result to the unbounded case requires the additional assumption that $p(\mathbf{z}_i|\neg Y_i)$ is in the exponential family. Given this assumption and the assumption that $p(\mathbf{z}_i|Y_i)$ is in the exponential family, the same proof as for the bounded case can be used to show that the network outputs y_i trained with cross-entropy error do estimate $P(Y_i|\mathbf{x})$. It is straightforward to show that the training algorithm is also the same as in the bounded case.

4 Experiments on Probabilistic Parsing

The recurrent network architecture to which we apply the above output activation functions is Simple Synchrony Networks (SSNs) [4]. SSNs are similar to Simple Recurrent Networks, but rather than computing one output pattern for each position in the input sequence, SSNs compute one output pattern for each pairing of a sequence position with any of its preceding sequence positions. When applied to natural language parsing, this provides enough outputs so that there is one output value for every possible parent of every word or phrase in the sentence (see [3]). These probabilities plus probabilities over phrase labels (noun phrase, verb phrase, etc.) are all the parser needs to find the most probable parse. In the experiments reported here, the same network architecture and the same probabilistic parser are applied, with varying output activation functions and their associated error functions.

Table 1 shows the results for the three methods.[5] The standard measure for comparing parser performance is shown in the last column. For comparison, the testing results for an un-smoothed statistical model (a probabilistic context

[5] For each method, we trained multiple networks with 60, 80, and 100 hidden units, using momentum and weight decay regularization. Training error was used to automatically reduce the learning rate, and cross validation was used to automatically reduce the regularization and to decide when to stop training. In each case the best network was chosen based on cross validation, and results were then computed on a held-out testing set. In each case one of the two networks with 100 hidden units was chosen, but similar cross validation results were achieved by all the networks run with 80 or 100 hidden units.

free grammar) were 29.2% recall, 53.7% precision, and 37.8% $F_{\beta=1}$, but state-of-the-art results using a much larger training corpus (1 million words versus the 26,480 words used here) are in the upper 80's.

	Training			Cross Validation			Testing		
	Rec	Prec	$F_{\beta=1}$	Rec	Prec	$F_{\beta=1}$	Rec	Prec	$F_{\beta=1}$
Sequence	60.2	62.3	61.3	57.9	60.0	58.9	58.7	59.9	59.3
Unb Sig	69.6	71.8	70.7	62.9	64.3	63.6	65.1	66.6	65.8
Unb Soft	72.1	72.5	72.3	64.5	63.9	64.2	64.8	64.2	64.5

Table 1: Results: percent phrase recall, precision, and a combination of both.

5 Conclusions

The experimental results show a clear advantage for both of the unbounded activation functions we have proposed in this article over the standard method of converting the unbounded distribution into a sequence of bounded distributions ("Sequence" in table 1). This shows that these alternative statistical estimation methods could improve the performance of many applications in unbounded domains, in particular statistical parsers. The unbounded sigmoid function ("Unb Sig" in table 1) shows an advantage over the unbounded softmax function ("Unb Soft" in table 1), but the results on the cross validation set show that this is not likely to be a reliable difference.

References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1995.

[2] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9:768–786, 1998.

[3] James Henderson and Peter Lane. A connectionist architecture for learning to parse. In *Proceedings of COLING-ACL*, pages 531–537, Montreal, Quebec, Canada, 1998.

[4] Peter Lane and James Henderson. Incremental syntactic parsing of natural language corpora with simple synchrony networks. *IEEE Transactions on Knowledge and Data Engineering*, 13(2), 2001.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol 1.* MIT Press, Cambridge, MA, 1986.