Back-propagation as reinforcement in prediction tasks

André Grüning

Cognitive Neuroscience Sector
S.I.S.S.A.
via Beirut 4
34014 Trieste
Italy
gruening@sissa.it

Abstract. The back-propagation (BP) training scheme is widely used for training network models in cognitive science besides its well known technical and biological short-comings. In this paper we contribute to making the BP training scheme more acceptable from a biological point of view in cognitively motivated prediction tasks overcoming one of its major drawbacks.

Traditionally, recurrent neural networks in symbolic time series prediction (e. g. language) are trained with gradient decent based learning algorithms, notably with back-propagation (BP) through time. A major drawback for the biological plausibility of BP is that it is a supervised scheme in which a teacher has to provide a fully specified target answer. Yet, agents in natural environments often receive a summary feed-back about the degree of success or failure only, a view adopted in reinforcement learning schemes.

In this work we show that for simple recurrent networks in prediction tasks for which there is a probability interpretation of the network's output vector, Elman BP can be reimplemented as a reinforcement learning scheme for which the expected weight updates agree with the ones from traditional Elman BP, using ideas from the AGREL learning scheme (van Ooyen and Roelfsema 2003) for feed-forward networks.

Reinforcement learning where the teacher gives only feed-back about success or failure of an answer is thought to be biologically more plausible than supervised learning since a fully specified correct answer might not always be available to the learner or even the teacher ([1], especially for biological plausibility [2]).

In this article we extent the ideas of the AGREL scheme [3] about how to implement back-propagation (BP) in feed-forward (FF) networks for classification tasks to encompass Elman BP for simple recurrent networks (SRNs) in prediction tasks [4]. The results have relevance especially for the cognitive science community for which SRN models have become an important tool [5], since they improve the standing of SRNs with respect to biological and cognitive plausibility.

1 SRNs and Elman BP

A SRN (also called Elman network) in its simplest form resembles a 3-layer FF network, but in addition the hidden layer is self-recurrent [4]. It is a special case of a general recurrent neural network (RNN) and could thus be trained with full back-propagation through time (BPTT) and BPTT(n) [6]. However, instead of regarding the hidden layer as self-recurrent, one introduces a so-called *context layer* into which the activities of the hidden neurons are stored in each time step and which acts as an additional input to the hidden layer in the next time step and thus effects its recurrency. Regarding the forward-propagation of activity through the SRN these two views are equivalent.

For the back-propagation of error, the SRN is now viewed as a FF network with an additional set of inputs from the context layer. Hence standard BP in conjunction with copying the hidden layer into the context layer can be used for training [4]. This scheme is called *Elman BP* and has found wide application especially in linguistically motivated prediction task (for an overview see [7]).

Since as described above, with Elman BP training SRNs can be reduced to training layered FF networks and since the step back to a SRN with context units is obvious, it is sufficient to formulate our reinforcement implementation of Elman BP for layered FF networks.

2 Recapitulation: Standard back-propagation

We state some important formulae of BP for layered FF networks first since the reinforcement implementation will be based on them. In order to keep notation simple, we will only deal with networks that are strictly layered, i. e. there are connections only between subsequent layers. Generalisations where neurons receive input from other downstream layers are of course possible. Furthermore we refrain from explicitly introducing a bias term, since its effect can easily be achieved by a unit with constant activation 1 in each layer.

Let us assume we have a network with $p+1$ layers in total where the counting starts from 0 for the input layer. Let $y_i^r, r > 0$ denote the output of neuron i in layer r, while $y^0 = (y_i^0)_i$ denotes the input vector. Let us further assume that the set of different input patterns y^0 is classified into classes c and that the target vector t^c only depends on the class of a particular input vector y^0 (or rather the class of a sequence of input vectors in the case of a SRN).

Then, for each input y^0 the layers are updated consecutively from 1 to p, $f : x \mapsto \frac{1}{1+e^{-x}}$ is the activation function of the neuron that maps the net input a_i^r of neuron i in layer r to its output y_i^r. Finally the output is read off from y^p and the error E_c against the target vector t^c is computed as follows, $0 < r \leq p$:

$$y_i^r = f(a_i^r), \quad a_i^r = \sum_j w_{ij}^{r,r-1} y_j^{r-1}, \quad E_c(y^p) = \frac{1}{2} \sum_i (t_i^c - y_i^p)^2. \tag{1}$$

For the update of the weights we need to know how much each single weight $w_{ij}^{r,r-1}$ contributes to the overall error E_c. For input y^0, E_c is an implicit function of all weights w. It helps in book-keeping of each weights' contribution $\frac{\partial E}{\partial w_{ij}^{r,r-1}}$ to the error by first calculating the contribution $\Delta_i^r := \frac{\partial E}{\partial a_i^r}$ of each neuron's net input to the error. The Δs can be recursively computed layer-wise starting from the output layer p and from them the weight updates δw with learning rate ϵ as:

$$\Delta_i^p := (y_i^p - t_i^c) f'(a_i^p), \tag{2}$$

$$\Delta_j^r := f'(a_j^r) \sum_i \Delta_i^{r+1} w_{ij}^{r+1,r}, \quad 0 < r < p. \tag{3}$$

$$\delta w_{ij}^{r,r-1} = -\epsilon \frac{\partial E}{\partial w_{ij}^{r,r-1}} = -\epsilon \frac{\partial E}{\partial a_i^r} \frac{\partial a_i^r}{\partial w_{ij}^{r,r-1}} = -\epsilon \Delta_i^r y_j^{r-1}, \tag{4}$$

3 Prediction Task for SRNs

Linguistically and cognitively inspired prediction learning means the following [4]: a sequence of unarily encoded symbols is input to the network one symbol at a time. The task is to predict the next symbol of the sequence. A context c for a prediction task would be given by a whole sequence of input symbols allowing for the same possible continuation(s). However it does not determine the next symbol with certainty but rather defines a distribution p_c accounting for linguistic variation. Thus the network has to learn the distribution of next symbols. Cognitively it is implausible to take the precalculated distribution as the target. Instead training is done against a target vector t^c where only one entry t_j^c drawn according to the appropriate distribution is one (all others zero), i. e. $p(t_j^c = 1) = p_c(j)$. For the error contributions Δ^p it follows, taking the expectation value over all possible targets t^c in context c:

$$< \Delta_i^p >_c = < (y_i^p - t_i^c) f'(a_i^p) >_c = (y_i^p - < t_i^c >_c) f'(a_i^p) = (y_i^p - p_c(i)) f'(a_i^p), \tag{5}$$

i. e. the expectation value of Δ_i^p in the output layer p coincides with the Δ derived from training against the distribution of target vectors. The same is true for all other Δs recursively computed from this due to the linearity of (3) in the Δs.

4 Reinforcement Learning

In prediction learning, the output's activation y_i^p corresponds to its symbol's estimated probability $p_{y^p}(i)$. Let us introduce a reinforcement scheme as follows: assume the network is only allowed to select one answer k as a response to its current input context c. It selects this answer according to the distribution $p_{y^p}(k) = y_k^p / |y^p|$: the neuron y_k^p corresponding to symbol k is called the winning neuron.

This answer is then compared to the target k_c drawn from the target distribution p_c and the network receives a reward $r = 1$ only when $k = k_c$, and $r = 0$ otherwise. Thus the objectively expected reward in input context c after selecting neuron k is $< r >_{c,k} = p_c(k)$. The network compares the received reward r to the subjectively expected reward, namely the activation y_k^p of the winning neuron. The relative difference

$$\delta := \frac{y_k^p - r}{p_{y^p}(k)} \tag{6}$$

is made globally available to all neurons in the network. From δ we then compute the error signals Δ^p for the output layer. Since attention is concentrated on the winning output k, only its Δ_k^p is different from zero, and we set $\Delta_i^p = 0$ for $i \neq k$ and

$$\Delta_k^p = \delta f'(a_k^p). \tag{7}$$

The other Δs and the weight updates δw can be recursively computed as before in (3) and (4). The expectation value of Δ_k^p in context c and with k as the winning unit calculates:

$$< \Delta_k^p >_{c,k} = f'(a_k^p) \frac{y_k^p - p_c(k)}{p_{y^p}(k)} \tag{8}$$

since t^c is drawn independently from k and $< r >_{c,k} = p_c(k)$. Compared to Elman BP an error for a certain output is calculated only when it is the winning unit, it is updated less frequently (by a factor $p_y(k)$). But its Δ is larger by $1/p_y(k)$ to compensate for this. Weighting the Δs with the probability $p_y(k)$ that k gets selected as the winning unit in context c, we get

$$< \Delta_k^p >_c = p_y(k) f'(a_k^p) \frac{y_k^p - p_c(k)}{p_y(k)} = f'(a_k^p)(y_k^p - p_c(k)) \tag{9}$$

and this agrees with (2) and (5), keeping in mind that $p_c(k)$ in our scheme would be the target entry t_i^c in the standard BP scheme. By linearity the expectation values of all other Δs and the δw in this scheme coincide as well with their counterparts in standard BP. When we update weights after each input presentation, the average of the Δs over several trials in context c will differ from the expectation value, but this is not a more severe moving target problem than encountered anyway in prediction learning in (5), and can also be dealt with by keeping ϵ low [6].

5 Discussion

We note that the order in which weights receive error signals form the outputs is altered: in standard BP weights receive error signals from all output neurons in each time step, while in this reinforcement scheme they receive a signal only from a single output, but with a greater amplitude. No major differences in the

course of learning are expected due to this changed order. In fact, it is known that training against the actual successors in the prediction task instead of against their probability distribution leads to faster and more reliable learning because higher error signals enable the network to leave local minima faster. A similar effect can be expected here.

The crucial facts why we can reimplement BP as a reinforcement scheme and thus replace a fully specified target vector with a single evaluative feedback are: (i) a probability interpretation of the output vector (and the target). This allows us to regard the network as directing its attention to a single output which is stochastically selected and subsequently to relate the evaluative feedback to this single output, (ii) the error contribution of each single neuron in the hidden layer(s) to the total error in BP is a linear superposition of the individual contributions of the neurons in the output layer. This enables us to concentrate on the contribution from a single output in each time step and still arrive at an expected weight update equal to the original scheme.

Obviously the ideas laid out in this paper are applicable to all kinds of SRNs, multilayered or not, and more general RNNs as long as their recurrence can be treated in the sense of introducing context units with immutable copy weights, and they ought also to be applicable to other networks and gradient based learning algorithms that fulfil the two above conditions (e. g. LSTM [8]).

As regards the biological plausibility, we list the following in favour of this BP-as-reinforcement scheme: (i) We have replaced a fully specified target with a single evaluative feed-back. (ii) The relative difference δ of actual and expected reward can be realised as a prediction error neuron [9, 2] whose activation is made globally available to all neurons in the network, e. g. by diffusion of a messenger such as dopamine [3]. (iii) Attentive concentration on the winning neuron is physiological plausible ([3] and references therein). (iv) Using the same sets of weights both for forward-propagation of activity and back-propagation of error is made plausible by introducing a second set of weights w' used only for the back-propagation of the Δs in (3) and updated with the same – mutatis mutandis – equation as in (4). This finds its functional equivalent in the ample evidence for recurrent connections in the brain. However we would consider this assumption as the scheme's weakest point that will need further elaboration. (v) Above and beyond [3]'s scheme, the only additional requirement for Elman BP as reinforcement has been that the activation of the hidden layer is retrievable in the next time step. This assumption is not implausible either in view of the ample recurrent connections in the brain which locally might well recycle activation from a neighbouring cell for some time from which this activation can be reconstructed. (vi) Finally we need to discuss how $p_{y^p}(k)$ can be derived from y_k^p in a plausible way, technically its is just an addition of the y^ps and dividing each output by this sum. Evidence for pools of neurons in visual cortex doing precisely a divisive normalisation is summarised in [10].[1]

Thus all quantities δ, y_j^r and Δ_i^{r+1} can be made locally available for the weight change at each synapse $w_{ij}^{r+1,r}$. While our reinforcement scheme naturally

[1] An anonymous referee pointed this reference out to me.

extends even to fully recurrent networks trained with BP through time (again it is mainly a question of the linearity of the Δs), there its application is of course less plausible since we would need to have access to previous activities of neurons for more than one time step.

6 Conclusion

Im sum, we have found a reinforcement learning scheme that behaves essentially like the standard BP scheme. It is biologically more plausible by using a success/failure signal instead of a precise target. Essential in transforming BP into a reinforcement scheme was (i) that the BP error signal for the complete target is a linear superposition of the error for each single output neuron, and (ii) the probabilistic nature of the task: select one possible output randomly and direct the network's attention towards it until it is rewarded. Furthermore we have briefly discussed the physiological or biological plausibility of other ingredients in the BP-as-reinforcement scheme. It seems that there is good evidence for all of them at least in some parts of the brain. Enhanced biological plausibly for BP thus gives SRN usage in cognitive science a stronger standing.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement learning: An Indroduction. Bradford Books, MIT Press, Cambridge (2002)
2. Wörgötter, F., Porr, B.: Temporal sequence learning, prediction, and control – a review of different models and their relation to biological mechanisms. Neural Computation **17** (2005) 245–319
3. van Ooyen, A., Roelfsema, P.R.: A biologically plausible implementation of error-backpropagation for classification tasks. In Kaynak, O., Alpaydin, E., Oja, E., Xu, L., eds.: Artificial Neural Networks and Neural Information Processing – Supplementary Proceedings ICANN/ICONIP, Istanbul (2003) 442–444
4. Elman, J.L.: Finding structure in time. Cognitive Science **14** (1990) 179–211
5. Ellis, R., Humphreys, G.: Connectionist Psychology. Psychology Press, Hove, East Sussex (1999)
6. Williams, R.J., Peng, J.: An efficient gradient-based algorithm for on-line training of recurrent network trajectories. Neural Computation **2** (1990) 490–501
7. Christiansen, M.H., Chater, N.: Toward a connectionist model of recursion in human linguistic performance. Cognitive Science **23** (1999) 157–205
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9** (1997) 1735–1780
9. Schultz, W.: Predictive reward signal of dopaminic neurons. J. Neurophysiol. **80** (1998) 1–27
10. Carandini, M., Heeger, D.J.: Summation and division by neurons in primate visual cortex. Sience **264** (1994) 1333–1336