Contextual Back-Propagation

Technical Report UT-CS-00-443

Bruce J. MacLennan*

Computer Science Department
University of Tennessee, Knoxville
macleman@cs.utk.edu

September 12, 2000

Abstract

A contextual neural network permits its connections to function differently in different behavioral contexts. This report presents an adaptation of the back-propagation algorithm to training contextual neural networks. It also addresses the special case of bilinear (sigma-pi) connections as well as the processing of continuous temporal patterns (signals).

1 Introduction

MacLennan (1998) provides a theoretical framework for contextual understanding for autonomous robots based on biological models. Contextual understanding allows expensive neural resources to be used for different purposes in different behavioral contexts; thus the function of these resources is context-dependent.

This flexibility means, however, that learning and adaptation must also be context-dependent. The basic idea is simple enough — hold the context constant while adjusting the other parameters — but it’s convenient to have explicit learning equations. MacLennan (1998) provides outer-product and convolutional learning rules for bilinear (“sigma-pi”) connections in one-layer networks for processing spatiotemporal patterns. The present report extends these algorithms to contextual back-propagation for multilayer networks processing spatial or spatiotemporal patterns; the algorithms

*This report may be used for any non-profit purpose provided that the source is credited.
are derived for general (differentiable) context dependencies and for the specific (common) case of bilinear dependencies.

I have tried to strike a balance in generality. On the one hand, this report goes beyond the simple second-order dependences discussed in MacLennan (1998). On the other, although the derivation is straight-forward and could be done in a very general framework, that generality seems superfluous at this point, and so it is limited to back-propagation.

2 Definitions

The context codes $c$ are drawn from some space, typically a vector space, but this restriction is not necessary.

The effective weight $W_{ij}$ of the connection to unit $i$ from unit $j$ is determined by the context $c$ and a vector of parameters $Q_{ij}$ associated with this connection. The dependence is given by:

$$ W_{ij} \overset{\text{def}}{=} C(Q_{ij}, c), $$

for some differentiable function $C$ on parameter vectors and contexts. In a simple particular case considered below (section 3.2), $C(Q_{ij}, c) = Q_{ij}^T c$.

We will be dealing with an $N$-layer feed-forward network in which the $l$-th layer has $L_l$ units (“neurons”). We use $x'_l$ to represent the activity of the $i$-th unit of the $l$-th layer, $i = 1, \ldots, L_l$. We often write the activities of a layer as a vector, $x'$. The output $y$ of the net is the activity of its last layer, $y \overset{\text{def}}{=} x'^N$, and the input $x$ to the net determines the activity of its “zeroth layer,” $x'^0 \overset{\text{def}}{=} x$.

The activity of a unit is the result of an activation function $\sigma$ applied to a linear combination of the activities of the units in the preceding layer. The coefficients of the linear combination are the effective weights. Thus the linear combination for unit $i$ of layer $l$ is given by

$$ s'_l \overset{\text{def}}{=} \sum_{j=1}^{L_{l-1}} W_{ij} x'_{j-1}. $$

That is, $s' = W^l x'^{l-1}$. The activities are then given by

$$ x'_l \overset{\text{def}}{=} \sigma(s'_l), l = 1, \ldots, N, $$

which we may abbreviate $x' = \sigma(s')$. The effective weights are given by Eq. 1.

We may then write the network as a function of the parameters and context, $y = N(Q, c)(x)$, and our goal is to choose the parameters $Q$ to minimize an error measure.

For training we have a set of $T$ triples $(p^q, c^q, t^q)$, where $p^q$ is an input pattern, $c^q$ is a context, and $t^q$ is a target pattern. The goal is to train the net so that pattern $p^q$ maps to target $t^q$ in context $c^q$, which we may abbreviate

$$ p^q \overset{C_q}{\rightarrow} t^q, q = 1, \ldots, T. $$
In effect, the context is additional input to the network, so we are attempting to map 
\((p^q, c^q) \mapsto t^q\). Contextual back-propagation, however, is not simply conventional 
back-propagation on the extended inputs \((p^q, c^q)\), since we must allow interactions 
between the components of \(p^q\) and \(c^q\).

Thus our goal is to find \(Q\) so that \(t^q\) is as nearly equal to \(N(Q, c^q)(p^q)\) as possible. 
Therefore we define a least-squares error function:

\[
\mathcal{E}(Q) \overset{\text{def}}{=} \sum_{q=1}^{T} \|t^q - y^q\|^2 = \sum_{q=1}^{T} \|t^q - N(Q, c^q)(p^q)\|^2.
\]

### 3 Contextual Back-Propagation

The basic equation of gradient descent is \(\dot{Q} = -\frac{1}{\eta} \nabla \mathcal{E}(Q)\). Therefore we begin by 
computing the gradient of the error function, so far as we are able while remaining 
independent of the specifics of the \(C\) function:

\[
\nabla \mathcal{E} = \nabla \sum_{q} \|t^q - y^q\|^2
\]

\[
= \sum_{q} \nabla \|t^q - y^q\|^2
\]

\[
= \sum_{q} 2(t^q - y^q)^T \frac{d(t^q - y^q)}{dQ}
\]

\[
= -2 \sum_{q} (t^q - y^q)^T \frac{dy^q}{dQ}
\]

\[
= -2 \sum_{q} (t^q - y^q)^T \frac{d}{dQ} N(Q, c^q)(p^q).
\]

Hence,

\[
\dot{Q} = \eta \sum_{q} (t^q - y^q)^T \frac{d}{dQ} N(Q, c^q)(p^q).
\]

For online learning, omit the summation. Define the change resulting from the \(q\)-th 
pattern:

\[
^q \Delta \overset{\text{def}}{=} (t^q - y^q)^T \frac{dy^q}{dQ}.
\]

This is a matrix of derivatives,

\[
^q \Delta^j_{jk} = (t^q - y^q)^T \frac{\partial y^q}{\partial Q^j_{i,k}}
\]

where \(Q^j_{i,k}\) is the \(k\)-th (scalar) component of \(Q^j_{i,j}\).
3.1 General Form

Since

$$(t^q - y^q)^T \frac{\partial y^q}{\partial Q_{ijk}^l} = (t^q - y^q)^T \frac{\partial y^q}{\partial s_i^l} \frac{\partial s_i^l}{\partial Q_{ijk}^l},$$

it will be convenient to name the quantity:

$$\delta_i^l \overset{\text{def}}{=} (t^q - y^q)^T \frac{\partial y^q}{\partial s_i^l}.$$  \hfill (3)

Thus,

$$q \Delta_{ijk}^l = \delta_i^l \frac{\partial s_i^l}{\partial Q_{ijk}^l}.$$ \hfill (4)

The partials with respect to the parameters are computed:

$$\frac{\partial s_i^l}{\partial Q_{ijk}^l} = \frac{\partial}{\partial Q_{ijk}^l} \sum_j W_{ij} x_j^{l-1}$$

$$= \frac{\partial}{\partial Q_{ijk}^l} \sum_j C(Q_{ij}^l, c) x_j^{l-1}$$

$$= \frac{\partial}{\partial Q_{ijk}^l} C(Q_{ij}^l, c) x_j^{l-1}$$

$$= \frac{\partial C(Q_{ij}^l, c)}{\partial Q_{ijk}^l} x_j^{l-1}. $$

Hence we may write,

$$\frac{\partial s_i^l}{\partial Q_{ij}^l} = \frac{\partial C(Q_{ij}^l, c)}{\partial Q_{ij}^l} x_j^{l-1}. $$

Therefore, the parameter update rule for arbitrary weights is:

$$q \Delta_{ij}^l = \delta_i^l x_j^{l-1} \frac{\partial C(Q_{ij}^l, c)}{\partial Q_{ij}^l},$$ \hfill (5)

which we may abbreviate $q \Delta' = \left[ \delta' (x^{l-1}) \right] \hat{\times} \left[ \partial C(Q^l, c) / \partial Q^l \right]$, where $\hat{\times}$ represents component-wise multiplication $[(u \times v)_n \overset{\text{def}}{=} u_n v_n]$.

It remains to compute the delta values; we begin with the output layer $l = N$. Since the output units are independent, $\partial y_j^q / \partial s_i^N = 0$ for $j \neq i$, we have

$$\delta_i^N = (t^q - y^q)^T \frac{\partial y^q}{\partial s_i^N} = (t_i^q - y_i^q) \frac{\partial y_i^q}{\partial s_i^N}.$$ 

The derivative is simply,

$$\frac{dy_i^q}{ds_i^N} = \frac{dx_i^N}{ds_i^N} = \frac{d\sigma(s_i^N)}{ds_i^N} = \sigma'(s_i^N).$$
Thus the delta values for the output layer are:

$$\delta_i^N = (t_i^q - y_i^q)\sigma'(s_i^N),$$  \hspace{1cm} (6)$$

which we may abbreviate $$\delta^N = (t^q - y^q)\hat{\sigma}'(s^N).$$

The computation for the hidden layers (0 ≤ l < N) is very similar, but makes use of the delta values for the subsequent layers.

$$\delta_i^l = (t^q - y^q)^T \frac{\partial y^q}{\partial s_i^l}$$

$$\quad = (t^q - y^q)^T \sum_{m=1}^{L_{l+1}} \frac{\partial y^q}{\partial s_m^{l+1}} \frac{\partial s_m^{l+1}}{\partial s_i^l}$$

$$\quad = \sum_m (t^q - y^q)^T \frac{\partial y^q}{\partial s_m^{l+1}} \frac{\partial s_m^{l+1}}{\partial s_i^l}$$

$$\quad = \sum_m \delta_{m}^{l+1} \frac{\partial s_m^{l+1}}{\partial s_i^l}.$$ 

The latter partials are computed as follows:

$$\frac{\partial s_m^{l+1}}{\partial s_i^l} = \frac{\partial}{\partial s_i^l} \sum_{i'} W_{mi'}^{l+1} \delta_{i'}^{l+1}$$

$$\quad = \sum_{i'} W_{mi'}^{l+1} \frac{\partial \delta_{i'}^{l+1}}{\partial s_i^l}$$

$$\quad = W_{mi}^{l+1} \delta_i^{l+1}$$

$$\quad = W_{mi}^{l+1} \sigma'(s_i^l).$$

Hence the delta values for the hidden layers are computed by:

$$\delta_i^l = \sigma'(s_i^l) \sum_m \delta_{m}^{l+1} W_{mi}^{l+1},$$  \hspace{1cm} (7)$$

which we may abbreviate $$\delta^l = \sigma'(s_i^l)\hat{\times}[(W^{l+1})^T \delta^{l+1}].$$ Combining all of the preceding (Eqs. 6, 7, 5), we get the following equations for contextual back-propagation with arbitrary weights (showing here the updates for a single pattern q):

$$\delta_i^N = \sigma'(s_i^N)(t_i^q - y_i^q),$$  \hspace{1cm} (8)$$

$$\delta_i^l = \sigma'(s_i^l) \sum_{m=1}^{L_{l+1}} \delta_{m}^{l+1} W_{mi}^{l+1} \hspace{1cm} \text{(for } 0 \leq l < N),$$  \hspace{1cm} (9)$$

$$\Delta_{ij}^l = \frac{\partial C(Q_{ij}^l, c)}{\partial Q_{ij}^l}.$$

$$\Delta_{ij}^l = \delta_{i}^l \delta_{j}^{l-1} \frac{\partial C(Q_{ij}^l, c)}{\partial Q_{ij}^l}.$$

5
3.2 Bilinear Connections

Next we consider the special case in which the context dependent weights are simply bilinear interactions between unit activities and components of a context vector,

$$C(Q_{ij}, c) \triangleq Q^T_{ij} c.$$  

In this case the partial derivative is simply,

$$\frac{\partial C(Q_{ij}, c)}{\partial Q^T_{i,jk}} = \frac{\partial}{\partial Q^T_{i,jk}} \sum_{k'} Q^T_{i,jk'} c_{k'} = c_k.$$  

Hence, the parameter update rule for bilinear weights is,

$$a\Delta^f_{ij} = \delta^f_i x^f_j c_k,$$

which we may abbreviate $$a\Delta^f = \delta^f \wedge x^f \wedge c,$$ where “$$\wedge$$” represents outer product: $$(u \wedge v \wedge w)_{ijk} \triangleq u_{i}v_{j}w_{k}.$$  

4 Spatiotemporal Patterns

Next, the preceding results will be extended to processing spatiotemporal patterns, in particular, continuously varying vector signals. Thus, the outputs and targets will be vector signals, $$y(t), t(t),$$ as will the unit activities, $$x'(t),$$ and associated quantities such as $$s'_i(t).$$ The parameters $$Q$$ will not be time-varying, except insofar as they are modified by learning (i.e., they vary on the slow time-scale of learning as opposed to the fast time-scale of the signals).

The simplest way to handle time-varying inputs is to make them discrete: $$y(t_1), y(t_2), \ldots, y(t_n)$$ etc.; then the time samples simply increase the dimension of all the vectors, and the preceding methods may be used. Instead, in this section we will take a signal-processing approach in which continuously-varying signals are processed in real time.

To begin, the error measure must integrate the difference between the output and target signals over time:

$$E(Q) \triangleq \sum_q \int_{-\infty}^{0} \|t^q(t) - y^q(t)\|^2 dt = \sum_q \|t^q - y^q\|^2.$$  

The gradient is then easy to compute:

$$\nabla E = \sum_q \int_{-\infty}^{0} \nabla\|t^q(t) - y^q(t)\|^2 dt$$

$$= -2 \sum_q \int_{-\infty}^{0} [t^q(t) - y^q(t)]^T \frac{\partial y^q(t)}{\partial Q} dt$$

$$= -2 \sum_q \langle (t^q - y^q)^T, \frac{\partial y^q}{\partial Q} \rangle.$$
Therefore, we can derive the change in a parameter $q\Delta_{i,j,k}^t$:

$$q\Delta_{i,j,k}^t \overset{\text{def}}{=} \left\langle (t^q - y^q)^T, \frac{\partial y^q}{\partial Q_{ij,k}^t} \right\rangle$$

$$= \left\langle (t^q - y^q)^T, \frac{\partial y^q}{\partial s^q_i} \frac{\partial s^q_i}{\partial Q_{ij,k}^t} \right\rangle$$

$$= \left\langle (t^q - y^q)^T \frac{\partial y^q}{\partial s^q_i} \frac{\partial s^q_i}{\partial Q_{ij,k}^t} \right\rangle.$$

The delta values are therefore time-varying:

$$\delta_i^t(t) \overset{\text{def}}{=} [t^q(t) - y^q(t)]^T \frac{\partial y^q(t)}{\partial s^q_i(t)}.$$

Thus, $q\Delta_{i,j,k}^t = \left\langle \delta_i^t, \frac{\partial s^q_i}{\partial Q_{ij,k}^t} \right\rangle$.

The connection $W_{ij}^t$ to unit $i$ from unit $j$ will be modeled as a linear system, which can be characterized by its impulse response $H_{ij}^t$,

$$W_{ij}^t x_j(t) = H_{ij}^t(t) \otimes x_j(t),$$

where “$\otimes$” represents (temporal) convolution. The impulse response is dependent on the parameters and context, $H_{ij}^t = C(Q_{ij,k}^t, c)$. Thus, multiplication in the static case (Eq. 2) is replaced by convolution in the dynamic case:

$$s_i^t(t) \overset{\text{def}}{=} \sum_j H_{ij}^t(t) \otimes x_j^{t-1}(t).$$

The derivative of $s_i^t(t)$ with respect to the parameters is then given by:

$$\frac{\partial s_i^t(t)}{\partial Q_{ij,k}^t} = \frac{\partial}{\partial Q_{ij,k}^t} H_{ij}^t(t) \otimes x_j^{t-1}(t)$$

$$= \frac{\partial}{\partial Q_{ij,k}^t} \int_{-\infty}^{+\infty} H_{ij}^t(u) x_j^{t-1}(t - u) du$$

$$= \int_{-\infty}^{+\infty} \frac{\partial H_{ij}^t(u)}{\partial Q_{ij,k}^t} x_j^{t-1}(t - u) du$$

$$= \frac{\partial H_{ij}^t(t)}{\partial Q_{ij,k}^t} \otimes x_j^{t-1}(t).$$

Therefore the spatiotemporal parameter update rule for arbitrary linear systems is given by

$$q\Delta_{i,j,k}^t = \left\langle \delta_i^t, \frac{\partial H_{ij}^t}{\partial Q_{ij,k}^t} \otimes x_j^{t-1} \right\rangle. \quad (12)$$
Notice that the computation involves a temporal convolution (i.e., processing by linear system with impulse response $\partial H^t_{ij}(t)/\partial Q^t_{ijkl}$).

To see how this might be accomplished, we consider a special case analogous to the bilinear weights considered in Sec. 3.2. Here we take the impulse response $H^t_{ij}(t)$ to be a linear superposition of component functions $h^t_{ijk}(t)$, which could be the impulse responses of individual branches of a dendritic tree. Let $v^t_{ijk}(t)$ be the output of one of these component filters:

$$v^t_{ijk}(t) \overset{\text{def}}{=} h^t_{ijk}(t) \otimes x^{t-1}_j(t).$$

The coefficients of the components of this superposition depend on the parameters and context vector. Thus,

$$H^t_{ij}(t) = \sum_k C^t_{ijk} h^t_{ijk}(t),$$

where

$$C^t_{ijk} \overset{\text{def}}{=} c^T Q^t_{ijkl} = \sum_m c_m Q^t_{ijklm}.$$  

Therefore,

$$\frac{\partial H^t_{ij}(t)}{\partial Q^t_{ijklm}} = \frac{\partial}{\partial Q^t_{ijklm}} \sum_k c_m Q^t_{ijklm} h^t_{ijk}(t) = c_m h^t_{ijk}(t).$$

Thus, the change in the input to the activation function is given by

$$\frac{\partial s^t_i}{\partial Q^t_{ijklm}} = c_m h^t_{ijk}(t) \otimes x^{t-1}_j(t) = c_m v^t_{ijk}(t).$$

The parameter update rule for a superposition of filters is then

$$q \Delta^t_{ijklm} = \langle \delta^t_i, v^t_{ijk} \rangle c_m. \quad (13)$$

Notice that this requires $v^t_{ijk}(t)$, the output from the component filters, to be saved, so that an inner product can be formed with $\delta^t_i(t)$.

The delta values are computed as before (Eqs. 8, 9), except that all the quantities are time-varying. Nevertheless, it may be helpful to write out the derivation for hidden layer deltas (keeping in mind that the $W^{l+1}_{mi}$ are linear operators):

$$\frac{\partial s^{l+1}_m(t)}{\partial s^t_i(t)} = \frac{\partial}{\partial s^t_i(t)} \sum_{\ell} W^{l+1}_{mi\ell} x^t_\ell(t)$$

$$= W^{l+1}_{mi} \partial x^t_\ell(t)/\partial s^t_i(t)$$

$$= W^{l+1}_{mi} \sigma'[s^t_i(t)].$$

Thus we get the following delta values for spatiotemporal signals:

$$\delta^N_i(t) = \sigma'[s^N_i(t)] [y^0_i(t) - y^0_i(t)], \quad (14)$$

$$\delta^l_i(t) = \sum_{m=1}^{L_{l+1}} \delta^{l+1}_m(t) W^{l+1}_{mi} \sigma'[s^l_i(t)] \quad \text{(for } 0 \leq l < N). \quad (15)$$
5 Reference