Contextual Back-Propagation

Technical Report UT-CS-00-443

Bruce J. MacLennan*

Computer Science Department
University of Tennessee, Knoxville
maclennan@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 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 $\mathbf{Q}_{ij}$ associated with this connection. The dependence is given by:

$$W_{ij} \stackrel{\text{def}}{=} C(\mathbf{Q}_{ij}, c), \tag{1}$$

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

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_i^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, $\mathbf{x}^l$. The output $\mathbf{y}$ of the net is the activity of its last layer, $\mathbf{y} \stackrel{\text{def}}{=} \mathbf{x}^N$, and the input $\mathbf{x}$ to the net determines the activity of its "zeroth layer," $\mathbf{x}^0 \stackrel{\text{def}}{=} \mathbf{x}$.

The activity of a unit is the result of an activation function σ 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

$$s_i^l \stackrel{\text{def}}{=} \sum_{j=1}^{L_{l-1}} W_{ij}^l x_j^{l-1}. \tag{2}$$

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

$$x_i^l \stackrel{\text{def}}{=} \sigma(s_i^l), l = 1, \ldots, N,$$

which we may abbreviate $\mathbf{x}^l = \sigma(\mathbf{s}^l)$. The effective weights are given by Eq. 1.

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

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

$$\mathbf{p}^q \stackrel{c^q}{\mapsto} \mathbf{t}^q, q = 1, \ldots, T.$$

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

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

$$\mathcal{E}(\mathbf{Q}) \stackrel{\mathrm{def}}{=} \sum_{q=1}^{T} \|\mathbf{t}^q - \mathbf{y}^q\|^2 = \sum_{q=1}^{T} \|\mathbf{t}^q - \mathcal{N}(\mathbf{Q}, c^q)(\mathbf{p}^q)\|^2.$$

3 Contextual Back-Propagation

The basic equation of gradient descent is $\dot{\mathbf{Q}} = -\frac{1}{2}\eta\nabla\mathcal{E}(\mathbf{Q})$. 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:

$$\begin{aligned}
\nabla\mathcal{E} &= \nabla \sum_q \|\mathbf{t}^q - \mathbf{y}^q\|^2 \\
&= \sum_q \nabla \|\mathbf{t}^q - \mathbf{y}^q\|^2 \\
&= \sum_q 2(\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\mathrm{d}(\mathbf{t}^q - \mathbf{y}^q)}{\mathrm{d}\mathbf{Q}} \\
&= -2\sum_q (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\mathrm{d}\mathbf{y}^q}{\mathrm{d}\mathbf{Q}} \\
&= -2\sum_q (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}} \mathcal{N}(\mathbf{Q}, c^q)(\mathbf{p}^q).
\end{aligned}$$

Hence,

$$\dot{\mathbf{Q}} = \eta \sum_q (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}} \mathcal{N}(\mathbf{Q}, c^q)(\mathbf{p}^q).$$

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

$${}^q\mathbf{\Delta} \stackrel{\mathrm{def}}{=} (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\mathrm{d}\mathbf{y}^q}{\mathrm{d}\mathbf{Q}}.$$

This is a matrix of derivatives,

$${}^q\Delta^l_{ijk} = (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial Q^l_{ijk}},$$

where Q^l_{ijk} is the k-th (scalar) component of $\mathbf{Q}^l_{ij}$.

3.1 General Form

Since

$$(\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial Q^l_{ijk}} = (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial s^l_i} \frac{\partial s^l_i}{\partial Q^l_{ijk}},$$

it will be convenient to name the quantity:

$$\delta^l_i \stackrel{\text{def}}{=} (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial s^l_i}. \tag{3}$$

Thus,

$${}^q\Delta^l_{ijk} = \delta^l_i \frac{\partial s^l_i}{\partial Q^l_{ijk}}. \tag{4}$$

The partials with respect to the parameters are computed:

$$\begin{aligned}
\frac{\partial s^l_i}{\partial Q^l_{ijk}} &= \frac{\partial}{\partial Q^l_{ijk}} \sum_{j'} W^l_{ij'} x^{l-1}_{j'} \\
&= \frac{\partial}{\partial Q^l_{ijk}} \sum_{j'} C(\mathbf{Q}^l_{ij'}, c) x^{l-1}_{j'} \\
&= \frac{\partial}{\partial Q^l_{ijk}} C(\mathbf{Q}^l_{ij}, c) x^{l-1}_j \\
&= \frac{\partial C(\mathbf{Q}^l_{ij}, c)}{\partial Q^l_{ijk}} x^{l-1}_j.
\end{aligned}$$

Hence we may write,

$$\frac{\partial s^l_i}{\partial \mathbf{Q}^l_{ij}} = \frac{\partial C(\mathbf{Q}^l_{ij}, c)}{\partial \mathbf{Q}^l_{ij}} x^{l-1}_j.$$

Therefore, the **parameter update rule for arbitrary weights** is:

$${}^q\mathbf{\Delta}^l_{ij} = \delta^l_i x^{l-1}_j \frac{\partial C(\mathbf{Q}^l_{ij}, c)}{\partial \mathbf{Q}^l_{ij}}, \tag{5}$$

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

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

$$\delta^N_i = (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial s^N_i} = (t^q_i - y^q_i) \frac{\mathrm{d} y^q_i}{\mathrm{d} s^N_i}$$

The derivative is simply,

$$\frac{\mathrm{d} y^q_i}{\mathrm{d} s^N_i} = \frac{\mathrm{d} x^N_i}{\mathrm{d} s^N_i} = \frac{\mathrm{d} \sigma(s^N_i)}{\mathrm{d} s^N_i} = \sigma'(s^N_i).$$

Thus the **delta values for the output layer** are:

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

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

The computation for the hidden layers $(0 \leq 1 < N)$ is very similar, but makes use of the delta values for the subsequent layers.

$$\begin{aligned}
\delta_i^l &= (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial s_i^l} \\
&= (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \sum_{m=1}^{L_{l+1}} \frac{\partial \mathbf{y}^q}{\partial s_m^{l+1}} \frac{\partial s_m^{l+1}}{\partial s_i^l} \\
&= \sum_m (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}} \frac{\partial \mathbf{y}^q}{\partial s_m^{l+1}} \frac{\partial s_m^{l+1}}{\partial s_i^l} \\
&= \sum_m \delta_m^{l+1} \frac{\partial s_m^{l+1}}{\partial s_i^l}.
\end{aligned}$$

The latter partials are computed as follows:

$$\begin{aligned}
\frac{\partial s_m^{l+1}}{\partial s_i^l} &= \frac{\partial}{\partial s_i^l} \sum_{i'} W_{mi'}^{l+1} x_{i'}^l \\
&= \sum_{i'} W_{mi'}^{l+1} \frac{\partial x_{i'}^l}{\partial s_i^l} \\
&= W_{mi}^{l+1} \frac{\mathrm{d}x_i^l}{\mathrm{d}s_i^l} \\
&= W_{mi}^{l+1} \sigma'(s_i^l).
\end{aligned}$$

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

$$\delta_i^l = \sigma'(s_i^l) \sum_m \delta_m^{l+1} W_{mi}^{l+1}, \tag{7}$$

which we may abbreviate $\boldsymbol{\delta}^l = \sigma'(\mathbf{s}^l)\hat{\times}[(W^{l+1})^{\mathrm{T}}\boldsymbol{\delta}^{l+1}]$. 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):

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

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

$${}^q\boldsymbol{\Delta}_{ij}^l = \delta_i^l x_j^{l-1} \frac{\partial C(\mathbf{Q}_{ij}^l, c)}{\partial \mathbf{Q}_{ij}^l}. \tag{10}$$

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,

$$C(\mathbf{Q}_{ij}, \mathbf{c}) \stackrel{\text{def}}{=} \mathbf{Q}_{ij}^{\mathrm{T}} \mathbf{c}.$$

In this case the partial derivative is simply,

$$\frac{\partial C(\mathbf{Q}_{ij}^{l}, \mathbf{c})}{\partial Q_{ijk}^{l}} = \frac{\partial}{\partial Q_{ijk}^{l}} \sum_{k'} Q_{ijk'}^{l} c_{k'} = c_k.$$

Hence, the **parameter update rule for bilinear weights** is,

$$^{q}\Delta_{ijk}^{l} = \delta_i^l x_j^{l-1} c_k, \tag{11}$$

which we may abbreviate $^{q}\boldsymbol{\Delta}^{l} = \boldsymbol{\delta}^{l} \wedge \mathbf{x}^{l-1} \wedge \mathbf{c}$, where "$\wedge$" represents outer product: $(\mathbf{u} \wedge \mathbf{v} \wedge \mathbf{w})_{ijk} \stackrel{\text{def}}{=} u_i v_j w_k$.

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, $\mathbf{y}(t)$, $\mathbf{t}(t)$, as will the unit activities, $\mathbf{x}^l(t)$, and associated quantities such as $s_i^l(t)$. The parameters $\mathbf{Q}$ 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: $\mathbf{y}(t_1)$, $\mathbf{y}(t_2)$, …, $\mathbf{y}(t_n)$ 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:

$$\mathcal{E}(\mathbf{Q}) \stackrel{\text{def}}{=} \sum_q \int_{-\infty}^{0} \|\mathbf{t}^q(t) - \mathbf{y}^q(t)\|^2 \mathrm{d}t = \sum_q \|\mathbf{t}^q - \mathbf{y}^q\|^2.$$

The gradient is then easy to compute:

$$\begin{aligned}
\nabla \mathcal{E} &= \sum_q \int_{-\infty}^{0} \nabla \|\mathbf{t}^q(t) - \mathbf{y}^q(t)\|^2 \mathrm{d}t \\
&= -2 \sum_q \int_{-\infty}^{0} [\mathbf{t}^q(t) - \mathbf{y}^q(t)]^{\mathrm{T}} [\partial \mathbf{y}^q(t) / \partial \mathbf{Q}] \mathrm{d}t \\
&= -2 \sum_q \langle (\mathbf{t}^q - \mathbf{y}^q)^{\mathrm{T}}, \partial \mathbf{y}^q / \partial \mathbf{Q} \rangle.
\end{aligned}$$

Therefore, we can derive the change in a parameter ${}^{q}\Delta^{l}_{ijk}$:

$$\begin{aligned}
{}^{q}\Delta^{l}_{ijk} &\stackrel{\text{def}}{=} \left\langle (\mathbf{t}^{q} - \mathbf{y}^{q})^{\mathrm{T}}, \frac{\partial \mathbf{y}^{q}}{\partial Q^{l}_{ijk}} \right\rangle \\
&= \left\langle (\mathbf{t}^{q} - \mathbf{y}^{q})^{\mathrm{T}}, \frac{\partial \mathbf{y}^{q}}{\partial s^{l}_{i}} \frac{\partial s^{l}_{i}}{\partial Q^{l}_{ijk}} \right\rangle \\
&= \left\langle (\mathbf{t}^{q} - \mathbf{y}^{q})^{\mathrm{T}} \frac{\partial \mathbf{y}^{q}}{\partial s^{l}_{i}}, \frac{\partial s^{l}_{i}}{\partial Q^{l}_{ijk}} \right\rangle.
\end{aligned}$$

The delta values are therefore time-varying:

$$\delta^{l}_{i}(t) \stackrel{\text{def}}{=} [\mathbf{t}^{q}(t) - \mathbf{y}^{q}(t)]^{\mathrm{T}} \partial \mathbf{y}^{q}(t) / \partial s^{l}_{i}(t).$$

Thus,

$${}^{q}\Delta^{l}_{ijk} = \langle \delta^{l}_{i}, \partial s^{l}_{i} / \partial Q^{l}_{ijk} \rangle.$$

The connection W^{l}_{ij} to unit i from unit j will be modeled as a linear system, which can be characterized by its impulse response H^{l}_{ij},

$$W^{l}_{ij} x_{j}(t) = H^{l}_{ij}(t) \otimes x_{j}(t),$$

where "$\otimes$" represents (temporal) convolution. The impulse response is dependent on the parameters and context, $H^{l}_{ij} = C(\mathbf{Q}^{l}_{ij}, c)$. Thus, multiplication in the static case (Eq. 2) is replaced by convolution in the dynamic case:

$$s^{l}_{i}(t) \stackrel{\text{def}}{=} \sum_{j} H^{l}_{ij}(t) \otimes x^{l-1}_{j}(t).$$

The derivative of $s^{l}_{i}(t)$ with respect to the parameters is then given by:

$$\begin{aligned}
\frac{\partial s^{l}_{i}(t)}{\partial Q^{l}_{ijk}} &= \frac{\partial}{\partial Q^{l}_{ijk}} H^{l}_{ij}(t) \otimes x^{l-1}_{j}(t) \\
&= \frac{\partial}{\partial Q^{l}_{ijk}} \int_{-\infty}^{+\infty} H^{l}_{ij}(u) x^{l-1}_{j}(t-u) \mathrm{d}u \\
&= \int_{-\infty}^{+\infty} \frac{\partial H^{l}_{ij}(u)}{\partial Q^{l}_{ijk}} x^{l-1}_{j}(t-u) \mathrm{d}u \\
&= \frac{\partial H^{l}_{ij}(t)}{\partial Q^{l}_{ijk}} \otimes x^{l-1}_{j}(t).
\end{aligned}$$

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

$${}^{q}\Delta^{l}_{ijk} = \left\langle \delta^{l}_{i}, \frac{\partial H^{l}_{ij}}{\partial Q^{l}_{ijk}} \otimes x^{l-1}_{j} \right\rangle. \qquad (12)$$

Notice that the computation involves a temporal convolution (i.e., processing by linear system with impulse response $\partial H_{ij}^l(t)/\partial Q_{ijk}^l$).

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 $H_{ij}^l(t)$ to be a linear superposition of component functions $h_{ijk}^l(t)$, which could be the impulse responses of individual branches of a dendritic tree. Let $v_{ijk}^l(t)$ be the output of one of these component filters:

$$v_{ijk}^l(t) \stackrel{\text{def}}{=} h_{ijk}^l(t) \otimes x_j^{l-1}(t).$$

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

$$H_{ij}^l(t) = \sum_k C_{ijk}^l h_{ijk}^l(t),$$

where

$$C_{ijk}^l \stackrel{\text{def}}{=} \mathbf{c}^{\mathrm{T}} \mathbf{Q}_{ijk}^l = \sum_m c_m Q_{ijkm}^l.$$

Therefore,

$$\frac{\partial H_{ij}^l(t)}{\partial Q_{ijkm}^l} = \frac{\partial}{\partial Q_{ijkm}^l} \sum_{k,m} c_m Q_{ijkm}^l h_{ijk}^l(t) = c_m h_{ijk}^l(t).$$

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

$$\frac{\partial s_i^l}{\partial Q_{ijkm}^l} = c_m h_{ijk}^l(t) \otimes x_j^{l-1}(t) = c_m v_{ijk}^l(t).$$

The **parameter update rule for a superposition of filters** is then

$$^q\Delta_{ijkm}^l = \langle \delta_i^l, v_{ijk}^l \rangle c_m. \tag{13}$$

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

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_{mi}^{l+1} are linear operators):

$$\begin{aligned} \frac{\partial s_m^{l+1}(t)}{\partial s_i^l(t)} &= \frac{\partial}{\partial s_i^l(t)} \sum_{i'} W_{mi'}^{l+1} x_{i'}^l(t) \\ &= W_{mi}^{l+1} \partial x_i^l(t)/\partial s_i^l(t) \\ &= W_{mi}^{l+1} \sigma'[s_i^l(t)]. \end{aligned}$$

Thus we get the following **delta values for spatiotemporal signals**:

$$\delta_i^N(t) = \sigma'[s_i^N(t)][t_i^q(t) - y_i^q(t)], \tag{14}$$

$$\delta_i^l(t) = \sum_{m=1}^{L_{l+1}} \delta_m^{l+1}(t) W_{mi}^{l+1} \sigma'[s_i^l(t)] \quad (\text{for } 0 \le l < N). \tag{15}$$

5 Reference

1. MacLennan, B. J. (1998). Mixing memory and desire: Want and will in neural modeling. In: Karl H. Pribram (Ed.), *Brain and values: Is a biological science of values possible*, Mahwah: Lawrence Erlbaum, 1998, pp. 31–42. (corrected printing).