Backpropagation in Netz

Nick Ewing

0.1 Descripton

The following definitions and algorithm characterize the implementation of `netz.core`.

0.2 Definitions

The following variables and parameters are used in the Backpropagation algorithm:

λ: The regularization constant.

α: The learning rate constant.

γ: The learning momentum constant.

τ_{max}: The maximum number of epochs allowed.

ε_{min}: The desired error.

m: The number of training examples.

n: The number of input features for an example.

k: The number of outputs for an example.

$\{(x^i, y^i), \ldots, (x^m, y^m)\}$: The training set, composed of m input-output example pairs where $x^{(i)}$ is an n dimensional vector containing the i^{th} input and $y^{(i)}$ a k dimensional vector containing the i^{th} expected output.

X: An $m \times n$ matrix where the i^{th} row contains $(x^{(i)})^T$.

Y: An $m \times k$ matrix where the i^{th} row contains $(y^{(i)})^T$.

L: The number of layers in the network.

s_l: The number of neurons in layer l. $s_1 = n$ and $s_L = k$.

ϵ: The sum squared error of all training examples forward propagated through the network in the current epoch.

ε: The mean squared error of all training examples forward propagated through the network in the current epoch.

τ: The current epoch.

$\Theta_{ij}^{(l)}$: The synapse weight between neuron i of layer l and neuron j of layer $l+1$. $\Theta^{(l)}$ is thus a $s_{(j+1)} \times s_j + 1$ matrix.

$A_{ij}^{(l)}$: The change in synapse weight between neuron i of layer l and neuron j of layer $l+1$ of the last epoch. $A^{(l)}$ is thus a $s_{(j+1)} \times s_j + 1$ matrix.

$a^{(l)}$: A vector of length $s_l + 1$ when $l < L$ and s_l when $l = L$, containing the activation values for neurons in layer l where $a_0^{(l)}$ is the bias neuron when $l < L$.

$\delta^{(l)}$: A vector of length $s_l + 1$ when $l < L$ and s_l when $l = L$, containing the back propagated error values associated with neurons in layer l.

$\Delta_{ij}^{(l)}$: The sum change from all examples in synapse weight between neuron i of layer l and neuron j of layer $l+1$. $\Delta^{(l)}$ is thus a $s_{(j+1)} \times s_j + 1$ matrix.

$D_{ij}^{(l)}$: The regularized mean change from all examples in synapse weight between neuron i of layer l and neuron j of layer $l+1$. $D^{(l)}$ is thus a $s_{(j+1)} \times s_j + 1$ matrix.

$.*$: Element-wise matrix multiplication operator.
The sigmoid activation function:

$$g(z) = (1 - e^{-z})^{-1} \tag{1}$$

0.3 Algorithm

$\Theta^{(j)} \leftarrow$ RandomValuedMatrix$(s_{(j+1)},\, s_j + 1)$ **for** $j = 1 \dots L-1$
$A^{(j)} \leftarrow$ ZeroValuedMatrix$(s_{(j+1)},\, s_j + 1)$ **for** $j = 1 \dots L-1$

repeat
$\epsilon \leftarrow 0$

for $i \leftarrow 1 \dots m$ **do**
{Forward propagate input activations through network}
$a^{(1)} \leftarrow x^{(i)}$
$a^{(l)} \leftarrow g(\Theta^{(l-1)} a^{(l-1)})$ **for** $l = 2 \dots L$

{Find error of output layer from training outputs}
$\delta^{(L)} \leftarrow a^{(L)} - y^{(i)}$

{Add to sum squared error}
$\epsilon \leftarrow \epsilon + \sum_{j=1}^{k} (\delta_j^{(L)})^2$

{Backpropagate error through network}
$\delta^{(L)} \leftarrow (\Theta^{(l)})^T \delta^{(l+1)} .*\ a^{(l)} .* (1 - a^{(l)})$ **for** $l = L-1 \dots 2$

{Multiply errors by activations}
$\Delta^{(l)} \leftarrow \Delta^{(l)} + \delta^{(l+1)}\ (a^{(l)})^T$ **for** $l \leftarrow 1 \dots L-1$
end for

{Divide sum squared error by no. examples to get MSE}
$\varepsilon \leftarrow \epsilon / m$
return if $\varepsilon < \varepsilon_{min}$

{Divide errors by the number of training examples and add in regularization}
$D_{ij}^{(l)} \leftarrow \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ **for all** l, i where $j \neq 0$
$D_{ij}^{(l)} \leftarrow \frac{1}{m} \Delta_{ij}^{(l)}$ **for all** l, i where $j = 0$

{Adjust weights}
for $j \leftarrow 1 \dots L-1$ **do**
$A^{(j)} \leftarrow \alpha D^{(j)} + \gamma A^{(j)}$
$\Theta^{(j)} \leftarrow \Theta^{(j)} - A^{(j)}$
end for

$\tau \leftarrow \tau + 1$
until $\tau > \tau_{max}$