189 Neural Networks HW4

Max Johansen

November 3, 2016

1 BACKPROP DERIVATION

Let $l^{[0]}$ represent the input layer, $l^{[1]}$ represent the hidden layer and $l^{[2]}$ the output layer. Let the activation (activation function applied to weighted combination of inputs) at layer $l^{[k]}$ be represented by $\alpha^{[k]}$. Let $W_{ij}^{[k]}$ represent the weight from node $j \in l^{[k-1]}$ to $i \in l^{[k]}$. Let $g^{[k]}$ represent the activation function for $l^{[k]}$. Let $z_i^{[k]}$ represent the weighted combination of inputs to node $i \in l^{[k]}$.

$$z_i^{[k]} = \sum_{j=1} W_{ij}^{[k]} \alpha^{[k-1]} \tag{1.1}$$

$$\begin{aligned} \alpha^{[k]} &= g^{[k]}(W^{[k]}\alpha^{[k-1]}) \\ &= g^{[k]}(z^{[k]}) \end{aligned} \tag{1.2}$$

1.1 COST FUNCTION

We use cross entropy to represent cost J. Let $n_i(x)$ represent the ith component of the network output given features x.

$$J = -\sum_{i=1}^{n_{out}} y_i ln(n_i(x)) \tag{1.3}$$

We need to compute $\frac{\partial J}{\partial W_{ij}}$ using the chain rule.

$$\frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial z_i^{[2]}} \times \frac{\partial z_i^{[2]}}{\partial W_{ij}} \tag{1.4}$$

$$\frac{\partial z_i^{[2]}}{\partial W_{ij}} = \alpha_j^{[1]} \tag{1.5}$$

$$\begin{aligned}
\frac{\partial J}{\partial z_i^{[2]}} &= \frac{\partial - \sum_{j=1}^{n_{out}} y_j ln(n_j(x))}{\partial z_i^{[2]}} \\
&= \frac{\partial - \sum_{j=1}^{n_{out}} y_j ln(g^{[2]}(z_j^{[2]}))}{\partial z_i^{[2]}} \\
&= -\sum_{j=1}^{n_{out}} \frac{y_j}{g^{[2]}(z_j^{[2]})} \frac{\partial g^{[2]}(z_j^{[2]})}{\partial z_i^{[2]}} \\
&= -\sum_{j=i}^{n_{out}} \frac{y_j}{g^{[2]}(z_j^{[2]})} \frac{\partial g^{[2]}(z_j^{[2]})}{\partial z_i^{[2]}} - \sum_{j\neq i}^{n_{out}} \frac{y_j}{g^{[2]}(z_j^{[2]})} \frac{\partial g^{[2]}(z_j^{[2]})}{\partial z_i^{[2]}} \\
&= -\frac{y_i}{g^{[2]}(z_i^{[2]})} g^{[2]}(z_i^{[2]})(1 - g^{[2]}(z_i^{[2]})) - \sum_{j\neq i}^{n_{out}} \frac{y_j}{g^{[2]}(z_j^{[2]})} \frac{\partial g^{[2]}(z_j^{[2]})}{\partial z_i^{[2]}} \\
&= -y_i(1 - g^{[2]}(z_i^{[2]})) - \sum_{j\neq i}^{n_{out}} \frac{y_j}{g^{[2]}(z_j^{[2]})} \frac{\partial g^{[2]}(z_j^{[2]})}{\partial z_i^{[2]}} \\
&= -y_i(1 - g^{[2]}(z_i^{[2]})) + \sum_{j\neq i}^{n_{out}} \frac{y_j}{g^{[2]}(z_j^{[2]})} g^{[2]}(z_j^{[2]}) g^{[2]}(z_i^{[2]}) \\
&= -y_i(1 - g^{[2]}(z_i^{[2]})) + \sum_{j\neq i}^{n_{out}} y_j g^{[2]}(z_i^{[2]}) \\
&= -y_i + g^{[2]}(z_i^{[2]}) \sum_{j=1}^{n_{out}} y_j \\
&= -y_i + g^{[2]}(z_i^{[2]}) \\
&= g^{[2]}(z_i^{[2]}) - y_i
\end{aligned} \tag{1.6}$$

Here's some more notation

$$\delta_i^{[k]} = \frac{\partial J}{\partial z_i^{[k]}} \tag{1.7}$$

This implies

$$\delta_i^{[2]} = g^{[2]}(z_i^{[2]}) - y_i \tag{1.8}$$

Vectorized

$$\delta^{[2]} = g^{[2]}(z^{[2]}) - y \tag{1.9}$$

We can use induction and chain rule here to help us out.

$$\begin{aligned}
\delta_i^{[k-1]} &= \frac{\partial J}{\partial z_i^{[k-1]}} \\
&= \sum_j \frac{\partial J}{\partial z_j^{[k]}} \frac{\partial z_j^{[k]}}{\partial \alpha_i^{[k-1]}} \frac{\partial \alpha_i^{[k-1]}}{\partial z_i^{[k-1]}} \\
&= g^{[k-1]'} z_i^{[k-1]} \sum_j \delta_j^{[k]} W_{ji}^{[k]}
\end{aligned} \tag{1.10}$$

Vectorized

$$\delta^{[k-1]} = g^{[k-1]'} z^{[k-1]} \odot W^{[k]T} \delta^{[k]} \tag{1.11}$$

Now, we combine $\frac{\partial J}{\partial z_i^{[k]}} \times \frac{\partial z_i^{[k]}}{\partial W_{ij}}$ to find $\frac{\partial J}{\partial W_{ij}^{[k]}}$

$$\frac{\partial J}{\partial W_{ij}^{[k]}} = \delta_i^{[k]} \alpha_j^{[k-1]} \tag{1.12}$$

Matrix notation

$$\frac{\partial J}{\partial W^{[k]}} = \delta^{[k]} \alpha^{[k-1]T} \tag{1.13}$$

Now we derive $\frac{\partial J}{\partial V}$ $(W^{[1]} = V)$ (x represents input to network).

$$\begin{aligned}
\frac{\partial J}{\partial W^{[1]}} &= \delta^{[1]} \alpha^{[0]T} \\
&= \delta^{[1]} x^T \\
&= g^{[1]'}(z^{[1]}) \odot W^{[2]T} \delta^{[2]} x^T \\
&= g^{[1]'}(z^{[1]}) \odot W^{[2]T} (g^{[2]}(z^{[2]}) - y) x^T
\end{aligned} \tag{1.14}$$

Now we explicitly state $\frac{\partial J}{\partial W^{[2]}}$.

$$\begin{aligned}
\frac{\partial J}{\partial W^{[2]}} &= \delta^{[2]} \alpha^{[1]T} \\
&= (g^{[2]}(z^{[2]}) - y) \alpha^{[1]T}
\end{aligned} \tag{1.15}$$

Now we define the SGD update rule for $W^{[k]}$. Let γ represent the learning rate.

$$W_{t+1}^{[k]} = W_t^{[k]} - \gamma \frac{\partial J}{\partial W^{[k]}} \tag{1.16}$$

Update rule for $W = W^{[2]}$

$$\begin{aligned} W_{t+1}^{[2]} &= W_t^{[2]} - \gamma \frac{\partial J}{\partial W^{[2]}} \\ &= W_t^{[2]} - \gamma (g^{[2]}(z^{[2]}) - y) \alpha^{[1]T} \end{aligned} \tag{1.17}$$

Update rule for $V = W^{[1]}$

$$\begin{aligned} W_{t+1}^{[1]} &= W_t^{[1]} - \gamma \frac{\partial J}{\partial W^{[1]}} \\ &= W_t^{[1]} - \gamma g^{[1]'}(z^{[1]}) \odot W^{[2]T} (g^{[2]}(z^{[2]}) - y) \end{aligned} \tag{1.18}$$

1.2 Implementation notes

I start with an initial learning rate of $1e{-}3$ and decay by $1/2$ every epoch. I stopped training after 5 epochs. Total train time was roughly 20 minutes. I initialized the weights to be zero mean gaussians normalized by $sqrt(n_{entries}$. I arrived at trainingAccuracy 0.9966 validationAccuracy 0.9397 which could absolutely be improved with regularization and the addition of a conv layer.