Simple neural network to recognize handwritten digits

Ottokar Tilk

1 Description

The network has 3 layers (Figure 1). The input dimensionality is m and output dimensionality is o. Number of hidden units is n. For the MNIST dataset the m is 784 (each image is 28x28 pixels) and o is 10 (each image represents a digit in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$). For simplicity we are omitting biases.

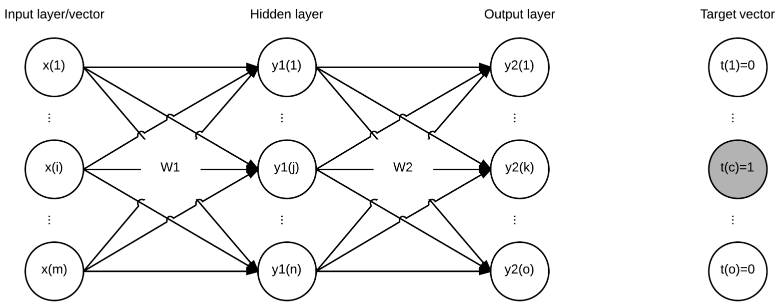

Figure 1: Architecture of the neural network.

2 Forward pass

2.1 Hidden layer

The input to the hidden unit is

$$z_1(j) = \sum_{i=1}^{m} x(i) W_1(i, j)$$

Hidden unit activation is the logistic sigmoid function

$$y_1(j) = \frac{1}{1 + e^{-z_1(j)}} \tag{1}$$

2.2 Output layer

The input to the output unit is

$$z_2(k) = \sum_{j=1}^{n} y_1(j) W_2(j,k)$$

Output unit activation is the softmax function

$$y_2(k) = \frac{e^{z_2(k)}}{\sum_{l=1}^{o} e^{z_2(l)}} \tag{2}$$

2.3 Error function

A suitable error function for our classification problem is the cross entropy function:

$$E = -\sum_{k=1}^{o} t(k) \ln y_2(k) = -\ln y_2(c)$$

where t is the one-hot encoded target vector. For example: if the correct answer c=6, then t would be [0 0 0 0 0 0 1 0 0 0].

3 Backward pass

3.1 Error derivative

The error derivative is non-zero only with respect to one output - the one that corresponds to the correct answer c. The derivative with respect to that output $y_2(c)$ is

$$\frac{\partial E}{\partial y_2(c)} = -\frac{1}{y_2(c)}$$

3.2 Output layer

The derivative of the softmax activation $y_2(c)$ with respect to its input $z_2(c)$ is

$$\frac{\partial y_2(c)}{\partial z_2(c)} = \frac{e^{z_2(c)} \sum_{l=1}^{o} e^{z_2(l)} - e^{z_2(c)} e^{z_2(c)}}{(\sum_{l=1}^{o} e^{z_2(l)})^2} = y_2(c)\big(1 - y_2(c)\big)$$

For all other units $k \neq c$ the derivative is

$$\frac{\partial y_2(c)}{\partial z_2(k)} = \frac{0 - e^{z_2(c)} e^{z_2(k)}}{(\sum_{l=1}^{o} e^{z_2(l)})^2} = y_2(c)\big(0 - y_2(k)\big)$$

These two cases can be written as one formula:

$$\frac{\partial y_2(c)}{\partial z_2(k)} = y_2(c)\big(\delta_{ck} - y_2(k)\big)$$

where δ_{ck} is Kronecker delta which equals 1 if $i = j$ and 0 otherwise. We can replace δ_{ck} with $t(k)$ and get

$$\frac{\partial y_2(c)}{\partial z_2(k)} = y_2(c)\big(t(k) - y_2(k)\big)$$

The derivative of the input to unit k with respect to its weight $W_2(j,k)$ is

$$\frac{\partial z_2(k)}{\partial W_2(j,k)} = y_1(j)$$

So the error derivative with respect to output layer weight $W_2(j,k)$ is

$$\frac{\partial E}{\partial W_2(j,k)} = \frac{\partial E}{\partial y_2(c)} \frac{\partial y_2(c)}{\partial z_2(k)} \frac{\partial z_2(k)}{\partial W_2(j,k)} =$$

$$= -\frac{1}{y_2(c)} y_2(c)\big(t(k) - y_2(k)\big) y_1(j) =$$

$$= \big(y_2(k) - t(k)\big) y_1(j)$$

The weight is updated with

$$\Delta W_2(j,k) = -\alpha \frac{\partial E}{\partial W_2(j,k)}$$

where α is the learning rate. The derivative of the input to the output unit $z_2(k)$ with respect to the output of the lower layer hidden unit $y_1(j)$ is

$$\frac{\partial z_2(k)}{\partial y_1(j)} = W_2(j,k)$$

To get the error derivative with respect to lower layer outputs $y_1(j)$ we need to sum over all output units (see the variable dependence diagram in Figure 2)

$$\frac{\partial E}{\partial y_1(j)} = \sum_{k=1}^{o} \frac{\partial E}{\partial y_2(c)} \frac{\partial y_2(c)}{\partial z_2(k)} \frac{\partial z_2(k)}{\partial y_1(j)} =$$

$$= \sum_{k=1}^{o} -\frac{1}{y_2(c)} y_2(c)\big(t(k) - y_2(k)\big) W_2(j,k) =$$

$$= \sum_{k=1}^{o} \big(y_2(k) - t(k)\big) W_2(j,k) \tag{3}$$

Derivative in Equation 3 is *backpropagated* to lower layer.

3.3 Hidden layer

The derivative of hidden activation $y_1(j)$ with respect to its input $z_1(j)$ is

$$\frac{dy_1(j)}{dz_1(j)} = -\frac{1}{(1+e^{-z_1(j)})^2} e^{-z_1(j)}(-1) = \frac{e^{-z_1(j)} + 1 - 1}{(1+e^{-z_1(j)})^2} =$$

$$= \frac{1+e^{-z_1(j)}}{(1+e^{-z_1(j)})^2} - \frac{1}{(1+e^{-z_1(j)})^2} = y_1(j) - y_1(j)^2 = y_1(j)\big(1 - y_1(j)\big)$$

and the derivative of that input with respect to weight $W_1(i,j)$ is

$$\frac{\partial z_1(j)}{\partial W_1(i,j)} = x(i)$$

To get the error derivative with respect to the hidden layer weight $W_1(i,j)$ we use the backpropagated derivative from Equation 3

$$\frac{\partial E}{\partial W_1(i,j)} = \frac{\partial E}{\partial y_1(j)} \frac{\partial y_1(j)}{\partial z_1(j)} \frac{\partial z_1(j)}{\partial W_1(i,j)} =$$

$$= \sum_{k=1}^{o} \big(y_2(k) - t(k)\big) W_2(j,k) y_1(j) \big(1 - y_1(j)\big) x(i)$$

The weight is updated with

$$\Delta W_1(i,j) = -\alpha \frac{\partial E}{\partial W_1(i,j)}$$

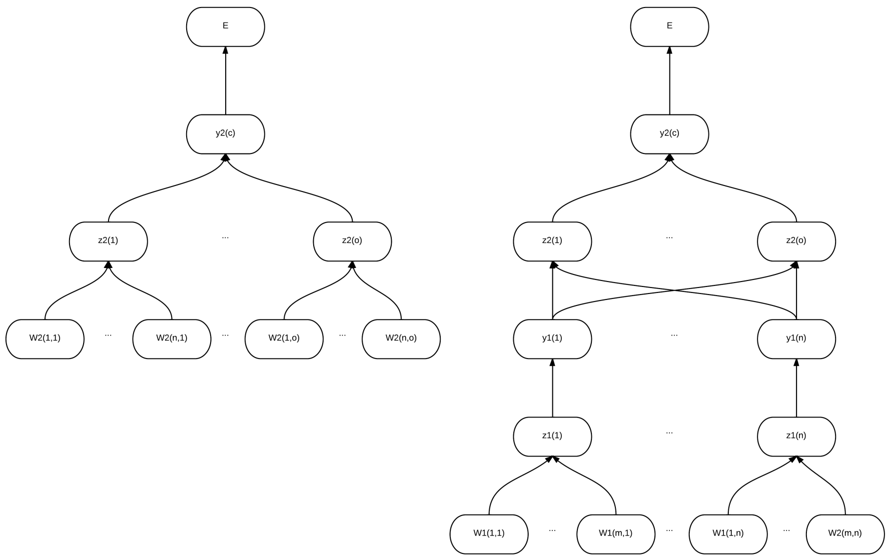

Figure 2: Variable dependence diagrams for W_2 (left) and W_1 (right)

4 Efficient implementation

Layer states and weight gradients can be computed using matrices and vectors and their operations. Layer inputs can be computed using

$$z_i = y_{i-1} W_i$$

where $y_0 = x$. Applying the activation function for each layer is the same as in Equations 1 and 2. For output layer gradients we get

$$\frac{\partial E}{\partial W_2} = y_1^T (y_2 - t)$$

and for hidden layer (* is element-wise multiplication)

$$\frac{\partial E}{\partial W_1} = x^T \big((y_2 - t) W_2^T * y_1 * (1 - y_1)\big)$$

where W_1 and W_2 are matrices and all others are row vectors.