Feedforward Neural Network Study Notes

Sicong Zhao

Part1: The equations of backpropagation

Equation 1: The error vector in the output layer

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

Equation 2: The relationship of error between two consecutive layer

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

Equation 3: The derivative of cost function of biases

$$\frac{\mathrm{d}C}{\mathrm{d}b^l} = \delta^l$$

Equation 4: The derivative of cost function of weights

$$\frac{\mathrm{d}C}{\mathrm{d}w^l_{kj}} = \delta^l_k a^{l-1}_j$$

Part2: Proof of Backpropagate Formulas

(1) Equation 1

$$
\begin{aligned}
\delta_j^L &= \frac{\mathrm{d}C}{\mathrm{d}z_j^L} \\
&= \sum_k \frac{\mathrm{d}C}{\mathrm{d}a_k^L} \times \frac{\mathrm{d}a_k^L}{\mathrm{d}z_j^L} \\
&\text{Only when k=j, the second term could be non-zero} \\
&= \frac{\mathrm{d}C}{\mathrm{d}a_j^L} \times \frac{\mathrm{d}a_j^L}{\mathrm{d}z_j^L} \\
&= \frac{\mathrm{d}C}{\mathrm{d}a_j^L}\sigma'(z_j^L)
\end{aligned}
$$

$$
\text{The vectorized expression is:} \qquad \delta^L = \nabla_a C \odot \sigma'(z^L)
$$

(2) Equation 2

$$
\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)
$$

First, pull out a single element of the matrix calculation.

$$
\delta_j^l = \sum_k (w_{kj}^{l+1} \delta_k^{l+1}) \sigma'(z_j^l)
$$

Second, expand the left side.

$$
\begin{aligned}
\delta_j^l &= \frac{\mathrm{d}C}{\mathrm{d}z^{l+1}} \times \frac{\mathrm{d}z^{l+1}}{\mathrm{d}z_j^l} \\
&= \sum_k \frac{\mathrm{d}C}{\mathrm{d}z_k^{l+1}} \times \frac{\mathrm{d}z_k^{l+1}}{\mathrm{d}z_j^l} \\
&= \sum_k \delta_k^{l+1} \times \frac{\mathrm{dz_k^{l+1}}}{\mathrm{d}a_j^l} \times \frac{\mathrm{d}a_j^l}{\mathrm{d}z_j^l} \\
&= \sum_k \delta_k^{l+1} \times w_{kj}^{l+1} \times \sigma'(z_j^l)
\end{aligned}
$$

(3) Equation 3

$$
\begin{aligned}
\frac{\mathrm{d}C}{\mathrm{d}b^l} &= \frac{\mathrm{d}C}{\mathrm{d}z^l} \times \frac{\mathrm{d}z^l}{\mathrm{d}b^l} \\
&= \frac{\mathrm{d}C}{\mathrm{d}z^l} \\
&= \delta^l
\end{aligned}
$$

(4) Equation 4

$$\frac{\mathrm{d}C}{\mathrm{d}w_{kj}^l} = \sum_i \frac{\mathrm{d}C}{\mathrm{d}z_i^l} \times \frac{\mathrm{d}z_i^l}{\mathrm{d}w_{kj}^l}$$
$$\text{Only when i=k, the second term could be non-zero}$$
$$= \frac{\mathrm{d}C}{\mathrm{d}z_k^l} \times \frac{\mathrm{d}z_k^l}{\mathrm{d}w_{kj}^l}$$
$$= \delta_k^l \times a_j^{l-1}$$

Part3: Implementation in Python

In following code, `L` stands for last layer, `l` stands for second to last layer

(1) Equation 1

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

```
delta_L = cost_derivative(output, y) * sigmoid_prime(z_L)
```

(2) Equation 2

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

```
delta_l = np.dot(weights_L.transpose(), delta_L) *
sigmoid_prime(z_l)
```

(3) Euqtion 3

$$\frac{\mathrm{d}C}{\mathrm{d}b^l} = \delta^l$$

```
nabla_bias = delta_L # Just use the result of previous 2 equation
```

(4) Equation 4

$$\frac{\mathrm{d}C}{\mathrm{d}w_{kj}^l} = \delta_k^l a_j^{l-1}$$

```
nabla_weight = np.dot(delta_L, sigmoid(z_l).transpose())
```

```python
# Helper functions

def cost_derivative(out_put, y):
  return out_put - y

def sigmoid(z):
  return 1.0/(1.0 + np.exp(-z))

def sigmoid_prime(z):
  return sigmoid(z) * (1-sigmoid(z))
```