Neural Network Notes

Manuel Serna-Aguilera

These are typeset notes cover the general architecture of a neural network model and applies it to the MNIST classification problem.

Notation

Weights

W is the set of weights connecting every activation in consecutive layers, where

$$W = \{W^{(1)}, W^{(2)}, \dots, W^{(L)}\}$$

from layer 1 up to layer L, $(1 \leq l \leq L)$. Layer $l = 0$ refers to the input data points as a column vector of shape $(n_0 \times 1)$. Subsequent layers have n_l activations (n_L is the number of activations in the output layer, i.e. the number of classes to predict on). Each element in W is a matrix. Each $W^{(l)}$ is a matrix with shape $(n_l \times n_{l-1})$, where

$$w_{jk}^{(l)} \in W^{(l)}$$

which can be read as "to activation j in layer l from activation k in layer $l-1$." The constraints of the indices are $1 \leq j < n_l$, and $1 \leq k \leq n_{l-1}$.

Biases

b is the set of biases for each activation, where

$$b = \{b^{(1)}, b^{(2)}, \dots, b^{(L)}\}$$

from layer 1 up to L, each $b^{(l)}$ is a column vector with shape $(n_l \times\ 1)$. Each entry $b_j^{(l)} \in b^{(l)}$, where $1 \leq j \leq n_l$.

Forward Propagation

Weighted Sum

The letter z will denote the "raw" weighted sum. Formally,

$$\begin{aligned} z &= \{z^{(1)}, z^{(2)}, \dots, z^{(L)}\} \\ z_j^{(l)} &\in z^{(l)} \\ 1 &\leq j \leq n_l \end{aligned}$$

where

$$\begin{aligned} z_j^{(l)} &= w_{j0}^{(l)} a_0^{(l-1)} + w_{j1}^{(l)} a_1^{(l-1)} + \ldots + w_{jm}^{(l)} a_m^{(l-1)} + b_j^{(l)} \\ m &= n_{l-1} \end{aligned}$$

Using Matrices and Vectors

Forward propagation using vectors and matrices from layer $l = 1$ up to L. Note that $a^{(0)}$ refers to the current model training input (which can also be referred to as x).

$$\begin{aligned} z^{(l)} &= W^{(l)} a^{(l-1)} + b^{(l)} \\ a^{(l)} &= f^{(l)}\big(z^{(l)}\big) \end{aligned}$$

Activation

The letter a refers to the output of the activation function $f^{(l)}$ that is applied to the weighted sums in layer l.

$$\begin{aligned} a &= \{a^{(1)}, a^{(2)}, \dots, a^{(L)}\} \\ a_j^{(l)} &\in a^{(l)} \\ 1 &\leq j \leq n_l \end{aligned}$$

where

$$a_j^{(l)} = f^{(l)}(z_j^{(l)})$$

Functions

Sigmoid (for intermediate layers)

$$\begin{aligned} \sigma(z^{(l)}) &= \frac{1}{1 + e^{z^{(l)}}} \\ \sigma'(z^{(l)}) &= \sigma(z^{(l)})\big(1 - \sigma(z^{(l)})\big) \end{aligned}$$

Softmax (for output layers)

$$\text{softmax}\big(z_i^{(l)}\big) = \frac{e^{z_i}}{\Sigma_j^{n_l} e^{z_j}}, 1 \leq i \leq n_l, 1 \leq j \leq n_l$$

Note that when using softmax as $f^{(L)}$ and the cost/loss/error as cross-entropy, the combined derivative will simply be $a^{(L)} - y$, where y is the true label for input x.

Backpropagation

Given some cost function (e.g. cross-entropy), we want to find how the cost/loss/error is influenced by the change of a particular weight or bias. In particular, we want to find the below gradients

$$\begin{aligned}\frac{\partial C}{\partial w_{jk}^{(l)}} &= \frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial C}{\partial a_j^{(l)}} \\ &= a_k^{(l-1)} \cdot (f^{(l)}(z_j^{(l)}))' \cdot \frac{\partial C}{\partial a_j^{(l)}}\end{aligned}$$

$$\begin{aligned}\frac{\partial C}{\partial b_j^{(l)}} &= \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial C}{\partial a_j^{(l)}} \\ &= 1 \cdot (f^{(l)}(z_j^{(l)}))' \cdot \frac{\partial C}{\partial a_j^{(l)}}\end{aligned}$$

where

$$\begin{aligned}\frac{\partial C}{\partial w_{jk}^{(l)}} &\in \nabla C(W) \\ \frac{\partial C}{\partial b_j^{(l)}} &\in \nabla C(b)\end{aligned}$$

(the $\nabla C(W)$ and $\nabla C(b)$ terms are the gradients for the parameters) and

$$\frac{\partial C}{\partial a_j^{(l)}} = \begin{cases} \sum_{i=1}^{n_{l+1}} \left(w_{ij}^{(l+1)} \cdot (f^{(l+1)}(z_i^{(l+1)}))' \cdot \frac{\partial C}{\partial a_i^{(l+1)}} \right) & \text{if } l \neq L \\ \text{derivative of cost function} & \text{if } l = L. \end{cases}$$

Finally, update parameters as below (the ticks represent the new, updated, parameters) to minimize cost/loss/error. The λ is the learning rate of the model.

$$\begin{aligned}W' &= W - \lambda \nabla C(W) \\ b' &= b - \lambda \nabla C(b)\end{aligned}$$

Using Vectors and Matrices

Computing each individual gradient is inefficient, and modern libraries and various programming languages already do the matrix multiplication for the user. Below are the four equations needed to compute the gradients for each layer.

$$\begin{aligned}
\delta^{(L)} &= \big(f^{(L)}(z^{(L)})\big)' \odot \frac{\partial C}{\partial a^{(L)}} && (1)\\
\delta^{(l)} &= \big(f^{(l)}(z^{(l)})\big)' \odot \big((W^{(l+1)})\delta^{(l+1)}\big) && (2)\\
\nabla C(W^{(l)}) &= \delta^{(l)}\big(a^{(l-1)}\big)^T && (3)\\
\nabla C(b^{(l)}) &= \delta^{(l)} && (4)
\end{aligned}$$

Note that L in the above equations specifically refers to the output layer. The term $a^{(0)}$ refers to the input x of the model. The symbol $\odot$ represents the element-wise multiplication between vectors or matrices. Finally, update the gradients.