AI lmao

Martin Johnsrud

July 15, 2019

Theory

A neural network is made up of l layers, where the ith layer, $i \in \{1, ..., l\}$, has L_i neurons. Each neuron j has a bias $b_j^{(i)}$ and an activation $a_j^{(i)}$. The activation of the neurons is a function of the activation of the neurons in layer $i-1$. A weighted sum

$$z_j^{(i)} = \sum_{k=0}^{L_i} w_{jk}^{(i)} a_k^{(i-1)} + b_j$$

is passed through an activation function, in this case

$$f(x) = \frac{1}{1+\exp(-x)}.$$

The activation of a neurons then becomes

$$a_j^{(i)} = f\big(w_{jk}^{(i)} a_k^{(i-1)} + b_j\big),$$

using einstein summation. This means layer i is associated with a matrix $w^{(i)} \in \mathbb{R}^{L_i \times L_{i-1}}$. In the special case of

$i = 0$ is the activation of the neurons given by an input vector $x \in \mathbb{R}^{L_0}$, and there are no need for wights of biases.

With each input x is there a desired outputvector $y \in \mathbb{R}^{L_l}$, wich is compared to the activation of the last layer $a^{(l)}$, by the cost function

$$C = \big(y_j - a_j^{(l)}\big)^2.$$

The goal is to train the neural network, by using gradient decent to minimize C. C is a function of all the weights $w_{jk}^{(i)}$, the biases $b_j^{(i)}$ and the activations $a_j^{(i)}$ given an input x. To find all partial derivatives, back propagation is emploied.

Back propagation

The partial derivatives of the cost function with respect to the weights in the last layer is given by

$$\frac{\partial C}{\partial w_{jk}^{(l)}} = \frac{\partial C}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{jk}^{(i)}}.$$

We have

$$\frac{\partial C}{\partial a_j^{(l)}} = 2(a_j^{(l)} - y_j)$$

$$\frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} = \frac{\partial f}{\partial x} = \frac{-\exp(x)}{(\exp(x)+1)^2},$$

and

$$\frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)},$$

giving us this parital derivative. For subsequent layers, the derivative with respect to the activation is given by

$$\frac{\partial C}{\partial a_j^{(i-1)}} = \sum_{k=1}^{L_j} \frac{\partial C}{\partial a_k^{(i)}} \frac{\partial a_k^{(i)}}{\partial z_k^{(i)}} \frac{\partial z_k^{(i)}}{\partial a_k^{(i-1)}},$$

as the output depends on the activation of all the nodes in the former layer, the activation of that layer depnds on the activation of all the nodes in the layer before that, and som on. This can be calculated recursively. As before,

$$\frac{\partial a_j^{(i)}}{\partial z_j^{(i)}} = \frac{\partial f}{\partial x} = \frac{-\exp(x)}{(\exp(x)+1)^2}$$

and

$$\frac{\partial z_j^{(i)}}{\partial w_{jk}^{(i)}} = a_k^{(i-1)}.$$

To find the derivative with respect to the weight, we implicilty differentiate $a_j^{(i)}$,

$$\frac{\partial a_j^{(i)}}{\partial w_{jk}^{(i)}} = \frac{\partial}{w_{jk}^{(i)}} f\big(w_{jk}^{(i)} a_k^{(i-1)} + b_j\big) = \frac{\partial f}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial a_j^{(i-1)}} = f'(z) a_j^{(i-1)}.$$

This gives

$$\frac{\partial C}{\partial w_{jk}^{(i-1)}} = \frac{\partial C}{\partial a_j^{(i-1)}} \frac{\partial a_j^{(i-1)}}{\partial w_{jk}^{(i-1)}} = \frac{\partial C}{\partial a_j^{(i-1)}} f'(z) a_j^{(i-1)}.$$

To find the full gradient, we also need the partial derivatives with respect to the biases. These can now be easily found as,

$$\frac{\partial C}{\partial b_j^{(l)}} = \frac{\partial C}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} = \frac{\partial C}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}}$$

and

$$\frac{\partial C}{\partial b_j^{(i)}} = \sum_{k=1}^{L_j} \frac{\partial C}{\partial a_k^{(i)}} \frac{\partial a_k^{(i)}}{\partial z_k^{(i)}} \frac{\partial z_k^{(i)}}{\partial b_j^{(i)}} = \sum_{k=1}^{L_j} \frac{\partial C}{\partial a_k^{(i)}} \frac{\partial a_k^{(i)}}{\partial z_k^{(i)}}.$$

The backpropagating algorithm therefore consists of

1. Find $\partial C/\partial w_{jk}^{(l)}$ and $\partial C/\partial b_j^{(l)}$.
2. Loop backwards through the neural net to find Find $\partial C/\partial a_j^{(i)}$ and $\partial C/\partial b_j^{(i)}$ for $i \in \{1...l-1\}$.
3. Use $\partial C/\partial a_j^{(i)}$ to find $\partial C/\partial w_{jk}^{(i)}$-

This wil result in the gradient of C with respect to the weights and biases,

$$\nabla C = \begin{bmatrix} \partial C/\partial w_{11}^{(1)} \\ \vdots \\ \partial C/\partial w_{L_l L_l}^{(l)} \\ \\ \partial C/\partial b_j^{(1)} \\ \vdots \\ \partial C/\partial b_{L_l}^{(l)} \end{bmatrix}$$

Data structure

The class `Layer` contains N nodes (n) and biases (b), as well as an $m \times n$ matrix (w). A neural network, here represented by the class `neuralNet`, is a linked list of l layers. It takes a vector `L` of length l as input, where the ith element L_i, corresponds to the number of neurons in layer i of the network