ReadmeJuliaNN

Milos Vukadinovic

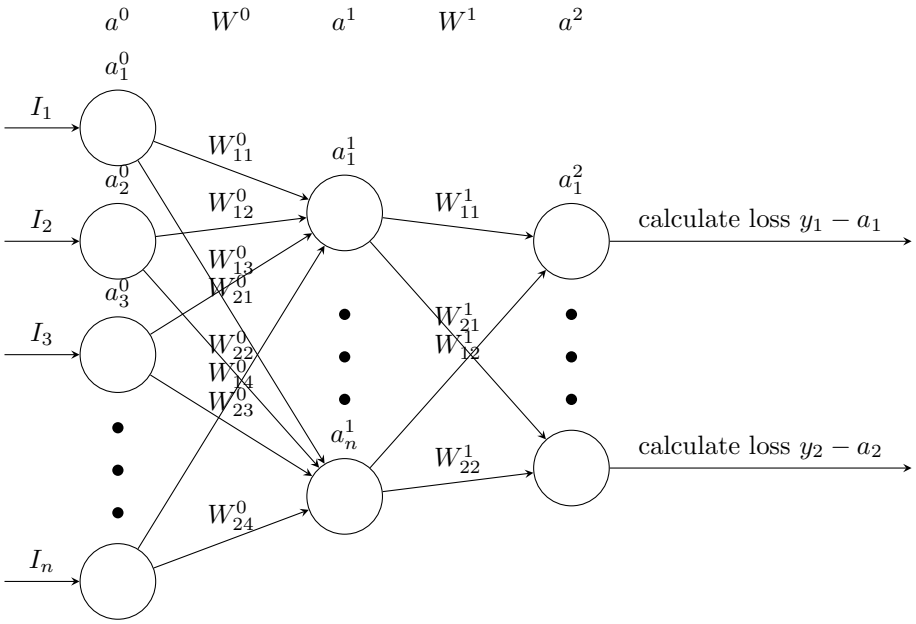

Figure 1: Neural network structure and notation

1 Notation

Activation of each neuron:

a_i^l where l is the index of the layer the neuron is in and i is the index of the neuron.

Weight connecting two neurons w_{jk}^l where l is the layer this weight is sending signals to (the next layer), j is the index of a neuron in l (to) layer and k is the index of a neuron in $l-1$ (from) layer. See why we use this notation 3

Denote notation to make things easier

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1}$$

Also note that

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1})$$

where σ is the sigmoid fcn *Cost function*
A cost or loss function calculates bad did our network do, higher the cost function more changes we'll need to make. It accepts the activations from the last layer a^n and the correct labels y. In our example we will use MSE - Mean Squared Error. (We will denote last layer as n.

$$C(a^n, y) = \frac{1}{2n} \sum_i (y_i - a_i^n)^2$$

2 Assumptions

- $C = \frac{1}{n} \sum_x C_x$
- Cost can be written as a fcn of the NN's output

3 Explaining W indexing

We think of W as a matrix operator so that we can do matrix multiplication as follows

$$w^l a^{l-1} = a^l$$

$$\begin{bmatrix} \mathrm{w}_{11} & \mathrm{w}_{12} & \mathrm{w}_{13} \\ \mathrm{w}_{21} & \mathrm{w}_{22} & \mathrm{w}23 \end{bmatrix} \begin{bmatrix} \mathrm{a}_1^{l-1} \\ \mathrm{a}_2^{l-1} \\ \mathrm{a}_3^{l-1} \end{bmatrix} = \begin{bmatrix} \mathrm{a}_1^l \\ \mathrm{a}_2^l \end{bmatrix}$$

Note the dimensions for the example $[2 \times 3][3 \times 1] = [2 \times 1]$ Thus you can see why

$$w_{jk} a_k = a_j$$

4 Backpropagation - Calculate Gradient

Now we will prove 3 formulas that will let us calculate the gradient for each w in the network and update it.
Let's start with calculating the partial derivative of the last (n) layer:

$$\frac{\partial C}{\partial a_j^n} = \frac{\partial(\frac{1}{2}\sum_i (y_i - a_i^n)^2}{\partial a_j^n} = (a_j - y_j)$$

a_j is present only in one sum so the derivative is straightforward
Written in a matrix form

$$\frac{\partial C}{\partial a^n} = (a^n - y)$$

Now, going backwards, we will calculate a partial with respect to z^n

$$\frac{\partial C}{\partial z^n} = \frac{\partial C}{\partial a^n}\frac{\partial a^n}{\partial z^n} = (a^n - y)\sigma'(z^n)$$

For the simplicity we will define

$$\delta^l = \frac{\partial C}{\partial z^l}$$

So in general we have the formula:

$$\delta^l = \frac{\partial C}{\partial a^l} \odot \sigma'(z^l) \tag{1}$$

We have the beginning, let's build a formula to propagate the network. We want to get δ^l in terms of δ^{l+1}

$$\frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial z^{l+1}}\frac{\partial z^{l+1}}{\partial z_j^l} = \delta^{l+1}\frac{\partial z^{l+1}}{\partial z_j^l} = \delta^{l+1}\frac{\partial z^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l} = \delta^{l+1}\frac{\partial z^{l+1}}{\partial a_j^l}\sigma'(z_j^l)$$

Now let's simplify $\frac{\partial z^{l+1}}{\partial a_j^l}$ and then we will put it back in the equation.

$$\frac{\partial z^{l+1}}{\partial a_j^l} = \frac{\partial(\sum_i \sum_k w_{kj}^{l+1} a_i^{l+1})}{\partial a_j^{l+1}} = \sum_k w_{kj}^{l+1}$$

Putting back we get

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

Or in the matrix form

$$\delta^l = w^{l+1}\delta^{l+1} \odot \sigma'(z^l) \tag{2}$$

This last formula is huge because it allows us to propagate back through the layers.
Finally, one left thing to do is calculate gradient for weights.

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$$

We don't need to find a formula for bias because I implemented a bias trick In the matrix form

$$\frac{\partial c}{\partial w^l} = a^{l-1}\delta^l \tag{3}$$

With formulas (1) (2) and (3) we are able to backpropagate!

(TODO:reference bias trick). (TODO: add proper citations) The whole implementation is based on the ideas from the book Neural Network by Michael Nielsen http://neuralnetworksanddeeplearning.com/