The Maths behind Neural Networks

Alex Punnen

Contents

The Maths behind Neural Networks

Alex Punnen
© All Rights Reserved

Contents

Chapter 7

Back Propagation Pass 3 (Matrix Calculus)

Let's take the previous two layered simple neural network,with a Mean Square Error Loss function, and derive the Back Propagation formula with Matrix Calculus now.

Let's write the equation of the following neural network

```
x is the Input
y is the Output.
l is the number of layers of the Neural Network.
a is the activation function ,(we use sigmoid here)
```

$$x \rightarrow a^{l-1} \rightarrow a^{l} \rightarrow y$$

Where the activation a^l is

$$a^l = \sigma(w^l a^{l-1} + b^l).$$

and

$$a^l = \sigma(z^l) \quad where \quad z^l = w^l a^{l-1} + b^l$$

Our two layer neural network can be written as

$$\mathbf{a^0} \rightarrow \mathbf{a^1} \rightarrow \mathbf{a^2} \rightarrow \mathbf{y}$$

(a^2 does not denote the exponent but just that it is of layer 2)

Lets write down the derivative of Loss function wrto weight using chain rule

$$\frac{\mathbf{C}}{\mathbf{w^l}} = \frac{\mathbf{a^l}}{\mathbf{w^l}} . \frac{\mathbf{C}}{\mathbf{a^l}}$$

We will use the above equation as the basis for the rest of the chapter.

Gradient Vector/2D-Tensor of Loss function in last layer

$$C = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$

Assuming a neural net with 2 layers, we have the final Loss as

$$C = \frac{1}{2} \sum_j (y_j - a_j^2)^2$$

Where

$$a^2 = \sigma(w^2 . a^1)$$

We can then write

$$C = \frac{1}{2} \sum_j v^2 \quad \rightarrow (Eq\ A)$$

Where

$$v = y - a^2$$

Partial Derivative of Loss function with respect to Weight

For the last layer, lets use Chain Rule to split like below

$$\frac{\partial C}{\partial w^2} = \frac{\partial v^2}{\partial v} * \frac{\partial v}{\partial w^2} \quad \rightarrow (Eq\ B)$$

$$\frac{\partial v^2}{\partial v} = 2v \quad \rightarrow (Eq\ B.1)$$

$$\frac{\partial v}{\partial w^2} = \frac{\partial (y - a^2)}{\partial w^2} = 0 - \frac{\partial a^2}{\partial w^2} \quad \rightarrow (Eq\ B.2)$$

$$\frac{\partial C}{\partial w^2} = \frac{1}{2} * 2v(0 - \frac{\partial a^2}{\partial w^2}) \quad \rightarrow (Eq\ B)$$

Now we need to find

$$\frac{\partial a^2}{\partial w^2}$$

Let

$$a^2 = \sigma(sum(w^2 \otimes a^1)) = \sigma(z^2)$$

$$z^2 = sum(w^2 \otimes a^1)$$

$$z^2 = sum(k^2)\text{ , where } k^2 = w^2 \otimes a^1$$

We now need to derive an intermediate term which we will use later

$$\frac{\partial z^2}{\partial w^2} = \frac{\partial z^2}{\partial k^2} * \frac{\partial k^2}{\partial w^2}$$

$$= \frac{\partial sum(k^2)}{\partial k^2} * \frac{\partial (w^2 \otimes a^1)}{\partial w^2}$$

$$\frac{\partial z^2}{\partial w^2} = (1^{\rightarrow})^T * diag(a^1) = (a^1)^T \quad \rightarrow (Eq\ B.3)$$

Though these are written like scalar here; actually all these are partial differentiation of Vector by Vector, or Vector by Scalar. A set of vectors can be

represented as the matrix here.More details here https://explained.ai/matrix-calculus/#sec6.2

The Vector dot product $w.a$ when applied on matrices becomes the elementwise multiplication $w^2 \otimes a^1$ (also called Hadamard product)

Going back to $Eq\ (B.2)$

$$\frac{\partial a^2}{\partial w^2} = \frac{\partial a^2}{\partial z^2} * \frac{\partial z^2}{\partial w^2}$$

Using $Eq\ (B.3)$ for the term in left

$$= \frac{\partial a^2}{\partial z^2} * (a^1)^T$$

$$= \frac{\partial \sigma(z^2)}{\partial z^2} * (a^1)^T$$

$$\frac{\partial a^2}{\partial w^2} = \sigma^{'}(z^2) * (a^1)^T \quad \rightarrow (Eq\ B.4)$$

Now lets got back to partial derivative of Loss function wrto to weight

$$\frac{\partial C}{\partial w^2} = \frac{1}{2} * 2v(0 - \frac{\partial a^2}{\partial w^2}) \quad \rightarrow (Eq\ B)$$

Using $Eq\ (B.4)$ to substitute in the last term

$$= v(0 - \sigma^{'}(z^2) * (a^1)^T)$$

$$= v * -1 * \sigma^{'}(z^2) * (a^1)^T$$

$$= (y - a^2) * -1 * \sigma^{'}(z^2) * (a^1)^T$$

$$\frac{\partial C}{\partial w^2} = (a^2 - y) * \sigma^{'}(z^2) * (a^1)^T \quad \rightarrow \mathbb{E}q\ (3)$$

Gradient Vector of Loss function in Inner Layer

Now let's do the same for the inner layer. This is bit more tricky and we use the Chain rule to derive this

$$\frac{\partial C}{\partial w^1} = \frac{\partial a^1}{\partial w^1}.\frac{\partial C}{\partial a^1} \quad \rightarrow (4.0)$$

We can calculate the first part of this from $Eq\ (B.4)$ that we derived above

$$\frac{\partial a^2}{\partial w^2} = \sigma^{'}(z^2) * (a^1)^T \quad \rightarrow (Eq\ B.4)$$

$$\frac{\partial a^1}{\partial w^1} = \sigma'(z^1) * (a^0)^T \quad \rightarrow (4.1)$$

For the second part, we use Chain Rule to split like below, the first part of which we calculated in the earlier step.

$$\frac{\partial C}{\partial (a^1)} = \frac{\partial C}{\partial (a^2)}.\frac{\partial (a^2)}{\partial (a^1)}$$

$$\frac{\partial C}{\partial (a^2)} = \frac{\partial (\frac{1}{2}\|y - a^2\|^2)}{\partial (a^2)} = \frac{1}{2} * 2 * (a^2 - y) = (a^2 - y) = \delta^2$$

$$\frac{\partial C}{\partial (a^2)} = \delta^2 \rightarrow (4.2)$$

$$\text{Now to calculate} \quad \frac{\partial (a^2)}{\partial (a^1)} \quad where \quad a^2 = \sigma(w^2 a^1 + b^2)$$

$$\frac{\partial (a^2)}{\partial (a^1)} = \frac{\partial (\sigma(w^2 a^1 + b^2))}{\partial (a^1)} = w^2.\sigma'(w^2 a^1 + b^2) = w^2.\sigma'(z^2) \rightarrow (4.3)*$$

*https://math.stackexchange.com/a/4065766/284422

Putting (4.1) (4.2) and (4.3) together

Final Equations

$$\frac{\mathbf{C}}{\mathbf{w^1}} = \ '(\mathbf{z^1}) * (\mathbf{a^0})^{\mathbf{T}} * \ ^{\mathbf{2}} * \mathbf{w^2}. \ '(\mathbf{z^2}) \quad \rightarrow \mathbb{E}\mathbf{q}\ (\mathbf{5})$$

$$\delta^2 = (a^2 - y)$$

Adding also the partial derivate of loss funciton with respect to weight in the final layer

$$\frac{\mathbf{C}}{\mathbf{w^2}} = \ ^{\mathbf{2}} * \ '(\mathbf{z^2}) * (\mathbf{a^1})^{\mathbf{T}} \quad \rightarrow \mathbb{E}\mathbf{q}\ (\mathbf{3})$$

You can see that the inner layer derivative have terms from the outer layer. So if we store and use the result; this is like dynamic program; maybe the back=propagation algorithm is the most elegant dynamic programming till date.

$$\frac{\mathbf{C}}{\mathbf{w^1}} = \ ^{\mathbf{2}} * \ '(\mathbf{z^2}) * (\mathbf{a^0})^{\mathbf{T}} * \mathbf{w^2}. \ '(\mathbf{z^1}) \quad \rightarrow \mathbb{E}\mathbf{q}\ (\mathbf{5})$$

Using Gradient Descent to find the optimal weights to reduce the Loss function

With equations (3) and (5) we can calculate the gradient of the Loss function with respect to weights in any layel - in this example

$$\frac{\partial C}{\partial w^1}, \frac{\partial C}{\partial w^2}$$

We now need to adjust the previous weight, by gradient descent.

So using the above gradients we get the new weights iteratively like below. If you notice this is exactly what is happening in gradient descent as well; only chain rule is used to calculate the gradients here. Backpropagation is the algorithm that helps calculate the gradients for each layer.

$$\mathbf{W^{l-1}_{new}} = \mathbf{W^{l-1}_{old}} - \mathbf{learningRate} * \ \mathbf{C_0}/\ \mathbf{w^{l-1}}$$

Reference

- https://cedar.buffalo.edu/~srihari/CSE574/Chap5/Chap5.3-BackProp.pdf
- http://neuralnetworksanddeeplearning.com/chap2.html

Next: Back Propagation in Full - With Softmax & CrossEntropy Loss