Some notes on skip connections for TP1

Christopher Beckham

May 25, 2017

1 Introduction

Let us introduce a simple 3-hidden layer MLP, which I will denote as $\mathbf{x} \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 10$. That is to say, this is an MLP where the three hidden layers have 128, 64, and 32 units, respectively, before the classification (softmax) layer, which has 10 units (since we're dealing with the MNIST dataset). For concreteness of notation, suppose that $\mathbf{x}$ is a $(n \times p)$ matrix (where n is the number of examples/minibatch size and p is the number of features), and m_i denotes the number of units in the i'th hidden layer. In our example, $p = 28 \times 28 = 784$, $m_1 = 128$, $m_2 = 64$, and $m_3 = 32$.

We can write the math for our MLP as the following:

$$\begin{aligned} \mathbf{h}^{(1)} &= g(\mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}) \\ \mathbf{h}^{(2)} &= g(\mathbf{h}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}) \\ \mathbf{h}^{(3)} &= g(\mathbf{h}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)}) \end{aligned} \tag{1}$$

A good thing to do is to make sure that the math makes sense by checking the dimensionalities of all the parameters. For example, since the first hidden layer has m_1 units, its corresponding weight matrix $\mathbf{W}^{(1)}$ should be a $(p \times m_1)$ matrix. Then, the computation $\mathbf{x}\mathbf{W}^{(1)}$ is doing the dot product between a $(n \times p)$ matrix and a $(p \times m_1)$ matrix, whose result is then a $(n \times m_1)$ matrix!

Since the second hidden layer has m_2 units, its corresponding weight matrix $\mathbf{W}^{(2)}$ should be $(m_1 \times m_2)$, and so forth. Also note the biases: $\mathbf{b}^{(i)}$ will be a row vector of size m_i (or in other words, a matrix of dimension $(1 \times m_i)$. I have illustrated this network in Figure 1.

1.1 Skip connections

We can now tackle the task of adding skip connections to this MLP. There are many different ways to add skip connections, especially if you consider an MLP with more hidden layers than 3. You can decide to add skips every 2 layers (i.e., 'short skips'), every 4 layers (i.e., 'long skips'), etc. depending on how deep the network is. This is what makes your TP interesting in a sense: there are many different ways to do this, and you need to experiment to see what configuration gives you the best results.

In our case, we wish to make a skip connection between the two hidden layers $\mathbf{h}^{(1)}$ an $\mathbf{h}^{(3)}$. The skip connection can also technically be a layer, so we have to think about what its weight matrix and bias vector will be (though for this example we will omit a bias vector). For this example, let us denote that weight matrix to be $\mathbf{W}^{(s)}$, where the 's' is shorthand for 'skip'. Since the skip is a mapping from $\mathbf{h}^{(1)} \rightarrow \mathbf{h}^{(3)}$, its weight matrix will be $(m_1 \times m_3)$.

Then, the computation of the 3rd hidden layer, $\mathbf{h}^{(3)}$, will be some combination between the computation performed by the skip layer and the layer before it. In our case, we can choose to simply sum the result, like so:

$$\begin{aligned} \mathbf{h}^{(1)} &= g(\mathbf{x}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}) \\ \mathbf{h}^{(2)} &= g(\mathbf{h}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}) \\ \mathbf{h}^{(3)} &= g(\mathbf{h}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)} + \mathbf{h}^{(1)}\mathbf{W}^{(s)}) \end{aligned} \tag{2}$$

In this example, if we decide to not learn $\mathbf{W}^{(s)}$ – that is, we just leave it randomly initialised and don't update it during gradient descent – this becomes a random projection. If we decide to

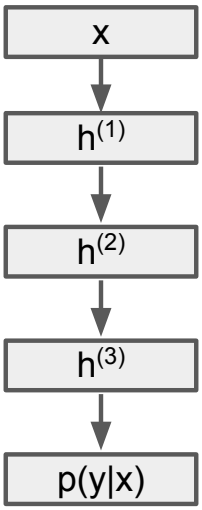

Figure 1: The MLP corresponding to Equation 1.

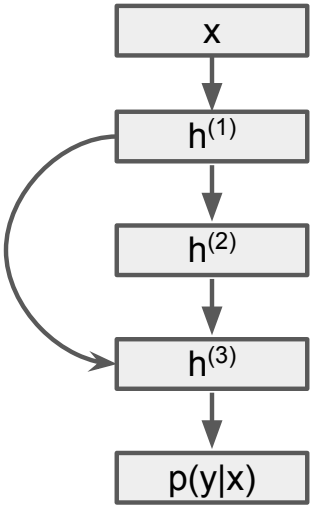

Figure 2: The MLP corresponding to Equation 2.

learn this matrix, then we increase model capacity, which will help us fit the training data better, but we have to be careful not to increase model capacity too much or else it will be more easy to overfit on the validation data! (Unless you introduce regularisation carefully, like dropout, L2, etc.)

There are other variants you can make to $\mathbf{h}^{(3)}$. For example, you could apply the nonlinearities separately:

$$\mathbf{h}^{(3)} = g(\mathbf{h}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)}) + g(\mathbf{h}^{(1)}\mathbf{W}^{(s)}) \tag{3}$$

You could even forgo the weight matrix for the skip connection and just concatenate the two representations $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(3)}$:

$$\mathbf{h}^{(3)} = concat(g(\mathbf{h}^{(2)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)}), \mathbf{h}^{(1)}) \tag{4}$$

By using $concat(\cdot, \cdot)$, the hidden layer $\mathbf{h}^3$ now has $128 + 32 = 160$ units![1]

1.2 Putting it all together

As I have mentioned in the TP, the main thing you need to do is try out some deep MLPs on MNIST, where you try out 1) sigmoids for all the activation functions, 2) relus for all the activation functions, and 3) sigmoids for all activations, but with an architecture that utilises skip connections. I will suggest an example, but since I have not written code myself for this TP, you should *take it with a grain of salt*, this is only for illustrative purposes!

You could for example compare 3 'deep' architectures: one with 3 hidden layers, one with 6, and one with 9, with these example configurations:

$$\begin{aligned}
&\mathbf{x} \to 256 \to 256 \to 256 \to 10 \\
&\mathbf{x} \to 256 \to 256 \to 256 \to 128 \to 128 \to 128 \to 10 \\
&\mathbf{x} \to 256 \to 256 \to 256 \to 128 \to 128 \to 128 \to 64 \to 64 \to 64 \to 10
\end{aligned} \tag{5}$$

For each of these architectures, figure out how you want to do the skip connections. Again, there are many ways you can do the skips. Once you figure this out, for each architecture, run experiments on MNIST where all the nonlinearities are sigmoid (skip connections disabled), all the nonlinearities are relu (skip connections disabled), and all the nonlinearities are sigmoid, with skips enabled. You should monitor many metrics, such as loss on the training set, loss on the validation set, and accuracy on the validation set. Depending on how deep your network is, you should expect to see the sigmoid + skip connection variant converging faster than the sigmoid without skip connections.

References

[1] Interesting side note: the U-Net paper, which is a commonly-used convolutional network architecture in medical imaging, uses exactly this trick in order to combine features at various hierarchies in the network.