COMP9444 Neural Networks and Deep Learning
Term 3, 2020

Solutions to Exercises 2: Backprop

This page was last updated: 09/22/2020 07:14:00

---

1. **Identical Inputs**

Consider a degenerate case where the training set consists of just a single input, repeated 100 times. In 80 of the 100 cases, the target output value is 1; in the other 20, it is 0. What will a back-propagation neural network predict for this example, assuming that it has been trained and reaches a global optimum? (Hint: to find the global optimum, differentiate the error function and set to zero.)

When sum-squared-error is minimized, we have

$$\begin{aligned} E &= 80*(z-1)^2/2 + 20*(z-0)^2/2 \\ dE/dz &= 80*(z-1) + 20*(z-0) \\ &= 100*z - 80 \\ &= 0 \text{ when } z = 0.8 \end{aligned}$$

When cross entropy is minimized, we have

$$\begin{aligned} E &= -80*\log(z) - 20*\log(1-z) \\ dE/dz &= -80/z + 20/(1-z) \\ &= (-80*(1-z) + 20*z)/(z*(1-z)) \\ &= (100*z - 80)/(z*(1-z)) \\ &= 0 \text{ when } z = 0.8\text{, as before.} \end{aligned}$$

2. **Linear Transfer Functions**

Suppose you had a neural network with linear transfer functions. That is, for each unit the activation is some constant c times the weighted sum of the inputs.

a. Assume that the network has one hidden layer. We can write the weights from the input to the hidden layer as a matrix $\mathbf{W}^{HI}$, the weights from the hidden to output layer as $\mathbf{W}^{OH}$, and the bias at the hidden and output layer as vectors $\mathbf{b}^{H}$ and $\mathbf{b}^{O}$. Using matrix notation, write down equations for the value $\mathbf{O}$ of the units in the output layer as a function of these weights and biases, and the input $\mathbf{I}$. Show that, for any given assignment of values to these weights and biases, there is a simpler network with no hidden layer that computes the same function.

Using vector and matrix multiplication, the hidden activations can be written as

$$\mathbf{H} = c * (\mathbf{b}^{H} + \mathbf{W}^{HI} * \mathbf{I})$$

The output activations can be written as

$$\begin{aligned}
\mathbf{O} &= c * [\mathbf{b}^{O} + \mathbf{W}^{OH} * \mathbf{H}] \\
&= c * [\mathbf{b}^{O} + \mathbf{W}^{OH} * c * (\mathbf{b}^{H} + \mathbf{W}^{HI} * \mathbf{I})] \\
&= c * [(\mathbf{b}^{O} + \mathbf{W}^{OH} * c * \mathbf{b}^{H}) + (\mathbf{W}^{OH} * c * \mathbf{W}^{HI}) * \mathbf{I}]
\end{aligned}$$

Because of the associativity of matrix multiplication, this can be written as

$$\mathbf{O} = c * (\mathbf{b}^{OI} + \mathbf{W}^{OI} * \mathbf{I})$$

where

$$\mathbf{b}^{OI} = \mathbf{b}^{O} + \mathbf{W}^{OH} * c * \mathbf{b}^{H}$$
$$\mathbf{W}^{OI} = \mathbf{W}^{OH} * c * \mathbf{W}^{HI}$$

Therefore, the same function can be computed with a simpler network, with no hidden layer, using the weights $\mathbf{W}^{OI}$ and bias $\mathbf{b}^{OI}$.

b. Repeat the calculation in part (a), this time for a network with any number of hidden layers. What can you say about the usefulness of linear transfer functions?

By removing the layers one at a time as above, a simpler network with no hidden layer can be constructed which computes exactly the same function as the original multi–layer network. In other words, with linear activation functions, you don't get any benefit from having more than one layer.