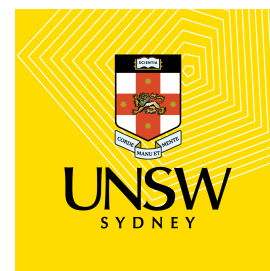


COMP9444

Neural Networks and Deep Learning

Term 2, 2024



Week 4 Tutorial: Softmax, Hidden Unit Dynamics

1. Softmax

Recall the formula of Softmax:

$$\text{Prob}(i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Consider a neural network being trained on a classification task with three classes 1, 2, 3. When the network is presented with a particular input, the output values are:

$$z_1 = 1.0, z_2 = 2.0, z_3 = 3.0$$

Suppose the correct class for this input is Class 2. Compute the following, to two decimal places:

- (a) $\text{Prob}(i)$, for $i = 1, 2, 3$
- (b) $d(\log \text{Prob}(2))/dz_j$, for $j = 1, 2, 3$

2. 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 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? If the loss function is changed from *Sum Squared Error* to *Cross Entropy*, does it give the same result? (Hint: to find the global optimum, differentiate the loss function and set to zero.)

3. Hidden Unit Dynamics

Consider a fully connected feedforward neural network with 6 inputs, 2 hidden units and 3 outputs, using a tanh activation at the hidden units and sigmoid at the outputs. Suppose this network is trained on the following data, and that the training is successful.

Draw a diagram showing:

- (a) for each input, a point in hidden unit space corresponding to that input, and

Item	Inputs	Outputs
	123456	123
1.	100000	000
2.	010000	001
3.	001000	010
4.	000100	100
5.	000010	101
6.	000001	110

- (b) for each output, a line dividing the hidden unit space into regions for which the value of that output is greater/less than one half.

4. 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 vector \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.
- (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?