# Supervision Assignments in AI
# Easter Term 2019
# Set 3: Artificial Neural Networks

Supervisor : Dionysis Manousakas
dm754@cam.ac.uk

May 25, 2019

1. (*AIMA18.22*) Suppose you had a neural network with linear activation functions. That is, for each unit the output is some constant $c$ times the weighted sum of the inputs.

   a. Assume that the network has one hidden layer. For a given assignment to the weights $w$, write down equations for the value of the units in the output layer as a function of $w$ and the input layer $x$, without any explicit mention of the output of the hidden layer. Show that there is a network with no hidden units that computes the same function.

   b. Repeat the calculation in part (a), but this time do it for a network with any number of hidden layers.

   c. Suppose a network with one hidden layer and linear activation functions has $n$ input and output nodes and $h$ hidden nodes. What effect does the transformation in part (a) to a network with no hidden layers have on the total number of weights? Discuss in particular the case $h \cdot n$.

For simplicity, we will assume that the activation function is the same linear function at each node: $g(x) = cx + d$. (The argument is the same (only messier) if we allow different $c_i$ and $d_i$ for each node.)

a. The outputs of the hidden layer are

$$H_j = g\left(\sum_k w_{kj} I_k\right) = c\sum_k w_{kj} I_k + d.$$

The final outputs are

$$O_i = g\left(\sum_j w_{ji} H_j\right) = c\left(\sum_j w_{ji}\left(c\sum_k w_{kj} I_k + d\right)\right) + d$$

Now we just have to see that this is linear in the inputs:

$$O_i = c^2 \sum_k I_k \sum_j w_{kj} w_{ji} + d\left(1 + c\sum_j w_{ji}\right)$$

Thus we can compute the same function as the two-layer network using just a one-layer perceptron that has weights $w_{ki} = \sum_j w_{kj} w_{ji}$ and an activation function $g(x) = c^2 x + d\left(1 + c\sum_j w_{ji}\right)$.

b. The above reduction can be used straightforwardly to reduce an $n$-layer network to an $(n-1)$-layer network. By induction, the $n$-layer network can be reduced to a single layer network. Thus, linear activation function restrict neural networks to represent only linearly functions.

c. The original network with $n$ input and output nodes and $h$ hidden nodes has $2hn$ weights, whereas the "reduced" network has $n^2$ weights. When $h \ll n$, the original network has far fewer weights and thus represents the input/output mapping more concisely. Such networks are known to learn much faster than the reduced network; so the idea of using linear activation functions is not without merit.

2. (*AIMA18.23*) Suppose that a training set contains only a single example, repeated 100 times. In 80 of the 100 cases, the single output value is 1; in the other 20, it is 0. What will a backpropagation 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 it to zero.)

Intuitively, the data suggest that a probabilistic prediction $P(Output = 1) = 0.8$ is appropriate. The network will adjust its weights to minimize the error function. The error is

$$E = (1/2)\sum_i (y_i - a_i)^2 = 50O_1^2 - 80O_1 + 50.$$

The derivative of the error with respect to the single output $a_1$ is

$$\frac{\partial E}{\partial a_1} = 100a_1 - 80.$$

Setting the derivative to zero, we find that indeed $a_1 = 0.8$.