COMS 4030A

Adaptive Computation and Machine Learning

EXERCISES:

(1) Suppose a neural network has five output nodes $n_1, \ldots, n_5$.

For each i, compute $softmax(n_i)$ if the z values at the output nodes are:

(a) $z_{n_1} = 3$, $z_{n_2} = 5$, $z_{n_3} = 0.5$, $z_{n_4} = -2$ and $z_{n_5} = 1.7$;

(b) $z_{n_1} = -2$, $z_{n_2} = -5$, $z_{n_3} = -0.5$, $z_{n_4} = 0.1$ and $z_{n_5} = -1.5$;

(c) $z_{n_1} = 0$, $z_{n_2} = 0.2$, $z_{n_3} = -0.1$, $z_{n_4} = 0.1$ and $z_{n_5} = -0.7$.

Solutions:

(a) 0.114, 0.844, 0.009, 0.001, 0.031

(b) 0.065, 0.003, 0.292, 0.532, 0.107

(c) 0.212, 0.258, 0.191, 0.234, 0.105

Note that rounding errors mean that the numbers may not add up to 1 exactly.

(2) Compute $L_{CE}(\boldsymbol{t}, \boldsymbol{y})$ for the following probability distributions:

(i) $\boldsymbol{y} = (0.5, 0.3, 0.2)$ and $\boldsymbol{t} = (1, 0, 0)$;

(ii) $\boldsymbol{y} = (0.5, 0.3, 0.2)$ and $\boldsymbol{t} = (0, 1, 0)$;

(iii) $\boldsymbol{y} = (0.5, 0.3, 0.2)$ and $\boldsymbol{t} = (0, 0, 1)$;

(iv) $\boldsymbol{y} = (0.1, 0.2, 0.7)$ and $\boldsymbol{t} = (0.3, 0.3, 0.4)$;

(v) $\boldsymbol{y} = (0.1, 0.2, 0.7)$ and $\boldsymbol{t} = (0.2, 0.2, 0.6)$.

Solutions:

(i) 0.693

(ii) 1.204

(iii) 1.609

(iv) 1.316

(v) 0.996

EXERCISES

(1) Rewrite the pseudocode for NEURAL NETWORK TRAINING ALGORITHM (with three layers) in such a way that the cross-entropy loss function is used.
At the hidden layer, you can use the σ activation function.

Solution:

The only change is the line:

for each output node n, where a_n is the output value, compute $\delta_n = a_n - t_n$

(2) Try the first exercise again, but use $relu$ at the hidden layer; then again with $tanh$.

Solution: For $relu$, the only changes are in the lines:

for each output node n, where a_n is the output value, compute $\delta_n = a_n - t_n$
for every node m in the hidden layer, where a_m is the activation value,
compute $\delta_m = \sum_n \delta_n \underline{w}_{mn}$ if $a_m > 0$ and $\delta_m = 0$ if $a_m = 0$,
where n ranges over all output nodes

For $tanh$, the only changes are in the lines:

for each output node n, where a_n is the output value, compute $\delta_n = a_n - t_n$
for every node m in the hidden layer, where a_m is the activation value,
compute $\delta_m = (\sum_n \delta_n \underline{w}_{mn})\,(1 - a_m^2)$,
where n ranges over all output nodes

(3) Consider a network with 2 input nodes, one hidden layer with 2 nodes, and 2 output nodes. The weights and the bias values are given by W_1, W_2, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$:

$$W_1 = \begin{bmatrix} -2 & -1 \\ 3 & 0 \end{bmatrix} \quad W_2 = \begin{bmatrix} 2 & 3 \\ -1 & -2 \end{bmatrix} \quad \boldsymbol{b}_1 = (0.5, 1.5) \quad \boldsymbol{b}_2 = (2, -1).$$

The activation function in the hidden layer is sigmoid (or $relu$, or $tanh$).
The output layer uses $softmax$ and the targets are one-hot encoded.
Using cross-entropy loss with input $\boldsymbol{x} = (-1, 1)$ and target $\boldsymbol{t} = (1, 0)$, do the following:
(a) First feed the input into the network to get the output and compute the loss.
(b) Perform one iteration of backpropagation training with $\eta = 0.1$.
(c) Feed the input into the network again and see if the loss has decreased.

Solution:

(Please check - and note that rounding errors may give slightly different answers.)

(a) using σ in hidden layer, output is $(0.949, 0.051)$ and $L_{CE} = 0.05216$
using *relu* in hidden layer, output is $(05, 0.5)$ and $L_{CE} = 0.69315$
using *tanh* in hidden layer, output is $(0.952, 0.048)$ and $L_{CE} = 0.04919$

(b) using σ in hidden layer, the updated weights are:

$$W_1 = \begin{bmatrix} -2.032 & -1.001 \\ 3.032 & 0.001 \end{bmatrix} \quad W_2 = \begin{bmatrix} 2.016 & 2.984 \\ -0.999 & -2.001 \end{bmatrix}$$

$$\boldsymbol{b}_1 = (0.532, 1.501) \quad \boldsymbol{b}_2 = (2.005, -1.005).$$

using *relu* in hidden layer, the updated weights are:

$$W_1 = \begin{bmatrix} -1.95 & -1.05 \\ 2.95 & 0.05 \end{bmatrix} \quad W_2 = \begin{bmatrix} 2.0275 & 2.725 \\ -0.875 & -2.125 \end{bmatrix}$$

$$\boldsymbol{b}_1 = (0.45, 1.55) \quad \boldsymbol{b}_2 = (2.05, -1.05).$$

using *tanh* in hidden layer, the updated weights are:

$$W_1 = \begin{bmatrix} -2 & -1.0001 \\ 3 & 0.0001 \end{bmatrix} \quad W_2 = \begin{bmatrix} 2.005 & 2.995 \\ -0.995 & -2.005 \end{bmatrix}$$

$$\boldsymbol{b}_1 = (0.5, 1.5001) \quad \boldsymbol{b}_2 = (2.005, -1.005).$$

(c) using σ in hidden layer, the new output is $(0.951, 0.049)$ and $L_{CE} = 0.05024$
using *relu* in hidden layer, the new output is $(0.936, 0.064)$ and $L_{CE} = 0.06614$
using *tanh* in hidden layer, the new output is $(0.953, 0.047)$ and $L_{CE} = 0.04779$

EXERCISES

(1) Suppose you have the following dataset with only 5 data points:

$$\begin{bmatrix} 6 & -24 & 125{,}000 & A & -0.001 & T \\ 9 & -31 & 175{,}000 & C & 0.0023 & T \\ 3 & -7 & 95{,}000 & A & -0.004 & F \\ 11 & -17 & 300{,}000 & B & 0.0045 & F \\ 4 & -11 & 250{,}000 & B & 0.003 & T \end{bmatrix} \quad \begin{bmatrix} X \\ Y \\ X \\ Z \\ Y \end{bmatrix}$$

Do the preprocessing of the data for both the inputs and the targets (using one-hot encoding). For the input values, try both methods of normalisation described above.

Solution:

using max-min normalisation:

$$\begin{bmatrix} 0.375 & 0.292 & 0.146 & 1 & 0 & 0 & 0.353 & 1 & 0 \\ 0.75 & 0 & 0.390 & 0 & 0 & 1 & 0.741 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0.583 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0.125 & 0.833 & 0.756 & 0 & 1 & 0 & 0.826 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

using mean-standard deviation normalisation:

$$\begin{bmatrix} -0.200 & -0.692 & -0.834 & 1 & 0 & 0 & -0.640 & 1 & 0 \\ 0.798 & -1.500 & -0.183 & 0 & 0 & 1 & 0.437 & 1 & 0 \\ -1.197 & 1.268 & -1.231 & 1 & 0 & 0 & -1.619 & 0 & 1 \\ 1.463 & 0.115 & 1.453 & 0 & 1 & 0 & 1.155 & 0 & 1 \\ -0.865 & 0.807 & 0.799 & 0 & 1 & 0 & 0.666 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$