Homework 3: Feed-forward neural networks

TDA231 - Algorithms for Machine Learning & Inference

Theodor Åstrand, theast@student.chalmers.se, 931109-9114
Linnéa Otterlind, linott@student.chalmers.se, 921126-3620

Chalmers University of Technology

February, 2016

Problem 1.1

(a)

Figure (1) since there are no cycles or loops in this. In figure (2) we can see that there is a cycle and there is also a cycle in figure (4) which means that they are not feed-forward neural networks. In figure (3) information can travel between nodes in the inner layer which means it is not a feed-forward neural network.

(b)

The topological properties of the feed-forward neural network is that information always moves in one direction (forward), thus no cycles can exist and the information cannot travel between nodes in the same layer.

Problem 1.2

The output of the combined feed-forward neural networks is:

$y = \frac{1}{2}\mathbf{w}_1^T\mathbf{x} + \frac{1}{2}\mathbf{w}_2^T\mathbf{x}$

Since the networks have no hidden layers we can conclude that the outputs of the networks separately are:

$y_1 = \mathbf{w}_1^T\mathbf{x}$

$y_2 = \mathbf{w}_2^T\mathbf{x}$

Which means that the vectors $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{x}$ are all the same size.

Now we want to create a new neural network, which has an output equal to y. The new network has no hidden layers and has the weight $\mathbf{w}_3 = f(\mathbf{w}_1, \mathbf{w}_2)$.

$y_3 = \mathbf{w}_3^T\mathbf{x} = f(\mathbf{w}_1, \mathbf{w}_2)^T\mathbf{x} = \frac{1}{2}\mathbf{w}_1^T\mathbf{x} + \frac{1}{2}\mathbf{w}_2^T\mathbf{x}$

From this we can see that:

$\mathbf{w}_3^T = \frac{1}{2}\mathbf{w}_1^T + \frac{1}{2}\mathbf{w}_2^T \Leftrightarrow \mathbf{w}_3 = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_2)$

So we get:

$y_3 = \mathbf{w}_3^T\mathbf{x}$ where

$\mathbf{w}_3 = f(\mathbf{w}_1, \mathbf{w}_2) = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_2)$

Problem 1.3

$E = \frac{1}{2}(t-y)^2$

$y = \mathbf{w}^T\mathbf{x} = \Sigma_{j=1}^{n} w_j x_j$

The back-propagation algorithm uses the gradient descent method so we need to find the partial derivative of the error E with respect to the weight w_i.

$\frac{\partial E}{\partial w_i} = \frac{\partial(\frac{1}{2}(t-y)^2)}{\partial w_i} = \frac{\partial(\frac{1}{2}(t-y)^2)}{\partial y} \cdot \frac{\partial y}{\partial w_i} = -(t-y) \cdot \frac{\partial y}{\partial w_i} = -(t-y) \cdot \frac{\partial(\Sigma_{j=1}^{n} w_j x_j)}{\partial w_i} = -(t-y) \cdot x_i$

So for a learning rate λ we get:

$\Delta w_i = \lambda(t-y) \cdot x_i$

And the negative of this is what will be added to w_i to calculate w_i^+.

$w_i^+ = w_i + (-\lambda(t-y) \cdot x_i)$

Problem 1.4

If the activation function g is the logistic function we get: $g'(z) = g(z)(1 - g(z))$.

(a)

$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} = \frac{\partial E}{\partial y_k} \cdot g'(z_k)$

(b)

$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial z_j} = \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} = \frac{\partial E}{\partial y_k} \cdot g'(z_k) \cdot w_{jk} \cdot g'(z_j)$

(c)

$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{jk}} = \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{jk}} = \frac{\partial E}{\partial y_k} \cdot g'(z_k) \cdot y_j$

(d)

$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}} = \frac{\partial E}{\partial y_k} \cdot g'(z_k) \cdot w_{jk} \cdot g'(z_j) \cdot y_i$

Problem 2.1

$i =$ input layer, $h =$ hidden layer, $c =$ class layer

$g_j =$ activation function for layer j

$y_j =$ output of layer j

$w_{jk} =$ the matrix of all weights from layer j to layer k

$t_c =$ target values

$x_i = z_i = y_i$ (input values), $z_h = \sum_i w_{ih} y_i$, $z_c = \sum_h w_{hc} y_h$

$g_i(z_i) = z_i$, $g_h(z_h) =$ logistic function $= y_h$, $g_c(z_c) = \frac{e^{z_c}}{\sum_j e^{z_j}} = y_c$

$g_h'(z_h) = g_h(z_h)(1 - g_h(z_h))$

$\frac{\partial E_1}{\partial z_c} = y_c - t_c$

Gradients of the cost functions:

Classification cost:

$$\text{mean}\Big(\frac{\partial E_1}{\partial w_{ih}}\Big) = \text{mean}\Big(\frac{\partial E_1}{\partial z_c} \cdot \frac{\partial z_c}{\partial y_h} \cdot \frac{\partial y_h}{\partial z_h} \cdot \frac{\partial z_h}{\partial w_{ih}}\Big) = \text{mean}\Big((y_c - t_c) \cdot w_{hc} \cdot g_h'(z_h) \cdot y_i\Big) =$$

$$= \text{mean}\Big((y_c - t_c) \cdot w_{hc} \cdot (y_h(1 - y_h)) \cdot y_i\Big)$$

$$\text{mean}\Big(\frac{\partial E_1}{\partial w_{hc}}\Big) = \text{mean}\Big(\frac{\partial E_1}{\partial z_c} \cdot \frac{\partial z_c}{\partial w_{hc}}\Big) = \text{mean}\Big((y_c - t_c) \cdot x_i\Big)$$

Weight decay cost:

$$\frac{\partial E_2}{\partial w_{ih}} = \text{wd\_coeffecient} \cdot w_{ih}$$

$$\frac{\partial E_2}{\partial w_{hc}} = \text{wd\_coeffecient} \cdot w_{hc}$$

Total gradients of the combined cost functions:

$$E = E_1 + E_2$$

$$\frac{\partial E}{\partial w_{ih}} = \text{wd\_coeffecient} \cdot w_{ih} + \text{mean}\Big((y_c - t_c) \cdot w_{hc} \cdot (y_h(1 - y_h)) \cdot y_i\Big)$$

$$\frac{\partial E}{\partial w_{hc}} = \text{wd\_coeffecient} \cdot w_{hc} + \text{mean}\Big((y_c - t_c) \cdot x_i\Big)$$

Problem 2.2

t_c = data.targets, y_i = data.input, y_h = hid_output, y_c = class_probs,

w_{ih} = model.input_to_hid, w_{hc} = model.hid_to_class

```
[M, N] = size(data.inputs);

ce_class = (1/N)*(class_prob-data.targets)*hid_output';

ce_hid = (1/N)*((class_prob-data.targets)'*model.hid_to_class)'.*
         (hid_output-hid_output.^2)*data.inputs';

res.input_to_hid = (model.input_to_hid * wd_coefficient) + ce_hid;
res.hid_to_class = (model.hid_to_class * wd_coefficient) + ce_class;
```

The cost on the training data: 2.768381

Problem 2.3

Learning rate	Momentum = 0.0	Momentum = **0.9**
0.002	2.304283	2.300135
0.01	2.302117	2.284022
0.05	2.292967	2.008606
0.2	2.228969	**1.083429**
1.0	1.598844	2.018723
5.0	2.301322	2.302585
20.0	2.302585	2.302585

(a)

The best run was done with momentum.

(b)

The learning rate for the best run was 0.2.

Problem 2.4

(a)

Running net(0, 200, 1000, 0.35, 0.9, false, 100) yields that the cost on the validation data is: 0.430185.

(b)

Running net(0, 200, 1000, 0.35, 0.9, true, 100) yields that the cost on the validation data is: 0.334505.

(c)

Weight decay	Classification cost
0.0001	0.348294
0.001	**0.287910**
1	2.302585
5	2.302585
10	22.612774

The best generalization was achieved with weight decay 0.001.

(d)

Hidden layer size	Classification cost
10	0.421705
30	**0.317077**
100	0.368593
130	0.397597
200	0.430185

The cost on the validation data was lowest with 30 hidden layers.

(e)

Hidden layer size	Classification cost
18	0.306083
37	**0.265165**
83	0.311244
113	0.313749
236	0.343841

The cost on the validation data was lowest with 37 hidden layers.

(f)

Running net(0, 37, 1000, 0.35, 0.9, true, 100) yields the classification error rate on the test data: 0.084333.

By looking at our previous results we found that this could be beaten by running net(0.001, 38, 1000, 0.2, 0.9, true, 100) which yields the classification error rate on the test data: 0.073111.