Homework 3

Rikard Olsson and Sharan Sabi

April 2018

1

1.1

a) 1 and 3
b) Directed acyclic graph

1.2

Claim 1.1. *Yes, the predictions will be the same.*

Proof. Assume that $n = |w_1| = |w_2| = |w_3|$. Let's rewrite the given average as $y_{12} = x(\frac{w_1+w_2}{2})^T$ and set $y_3 = w_3^T x = f(w_1, w_2)^T x = ((w_1 + w_2)/2)^T x$. Now, we simply sum over both to see their equality

$$y_{12} = \sum_i^n x_i(\frac{w_{1i} + w_{2i}}{2})$$

$$y_3 = \sum_i^n x_i f(w1, w2)_i = \sum_i^n x_i(w_{1i} + w_{2i})/2$$

$$y_{12} = y_3$$

Or we can show it with vector equations

$$y_{12} = \mathbf{x}(\begin{bmatrix} w_{11} \\ w_{12} \\ ... \\ w_{1n} \end{bmatrix} + \begin{bmatrix} w_{21} \\ w_{22} \\ ... \\ w_{2n} \end{bmatrix})/2 = \mathbf{x}\begin{bmatrix} w_{11} + w_{21} \\ w_{12} + w_{22} \\ ... \\ w_{1n} + w_{2n} \end{bmatrix}/2 = \mathbf{x}\begin{bmatrix} \frac{w_{11}+w_{21}}{2} \\ \frac{w_{12}+w_{22}}{2} \\ ... \\ \frac{w_{1n}+w_{2n}}{2} \end{bmatrix}$$

$$y_3 = \mathbf{x}\begin{bmatrix} \frac{w_{11}+w_{21}}{2} \\ \frac{w_{12}+w_{22}}{2} \\ ... \\ \frac{w_{1n}+w_{2n}}{2} \end{bmatrix}$$

$$y_{12} = y_3$$

■

1.3

$$\frac{\partial E}{\partial \mathbf{w}_i} = \frac{\partial y}{\partial \mathbf{w}_i}\frac{\partial E}{\partial y}$$
$$= \mathbf{x}\frac{2}{2}(y-t)$$
$$= \mathbf{x}(\mathbf{w}^T\mathbf{x}-t)$$

Now, we apply this into a formula for the new $\mathbf{w}_i$

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} - \lambda \mathbf{x}_i(\mathbf{w}^T\mathbf{x}-t)$$

or we can write this as change of w_i as in $\delta\mathbf{w}_i = -\lambda\mathbf{x}_i(\mathbf{w}^T\mathbf{x}-t)$

1.4

a)

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

b) Here we need to sum over the nodes in k which are effected by the change of z_j, that's why there's a sum:

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

c)

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

d)

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j}\frac{\partial \sum_i w_{ij}y_i}{\partial w_{ij}} = \frac{\partial E}{\partial z_j}y_i$$

2

First, we need to clarify some sub- and super scripts. Let a super script I, J, K, IJ, JK denote a layer in the network. E.g. $w^{(JK)}$ denotes the matrix of weights between layer I and J. Let a sub script i, j, k, ij, jk denote the index of a matrix/vector. E.g. $w_{ij}^{(IJ)}$ denotes the weight between node $i \in I$ and node $j \in J$.

2.1

This is the derivative for the error classification function with respect to a single weight $w_{jk}^{(JK)}$ between the hidden and output layer

$$\frac{\partial E^{class}}{\partial w_{jk}^{(JK)}} = \frac{\partial E^{class}}{\partial z_k^{(K)}} \frac{\partial z_k^{(K)}}{w_{jk}^{(JK)}} =$$

$$= (y_k^{(K)} - t_k) \frac{\partial \sum_j w_{jk}^{(JK)} * y_j^J}{\partial w_{jk}^{(JK)}} =$$

$$= (y_k^{(K)} - t_k) y_j^{(J)}$$

This is the derivative for the error classification function with respect to a single weight between the input and hidden layer $w_{ij}^{(IJ)}$

$$\frac{\partial E^{class}}{\partial w_{ij}^{(IJ)}} = \frac{\partial y_j^{(J)}}{\partial w_{ij}^{(IJ)}} \frac{\partial z^{(K)}}{\partial y_j^{(J)}} \frac{\partial E^{class}}{\partial z_k^{(K)}} =$$

We need to compute the sum for a single node $j \in J$, that is, the effect its value has on every node it's connected to in K.

$$= \frac{\partial y_j^{(J)}}{\partial w_{ij}^{(IJ)}} \frac{\partial z^{(K)}}{\partial y_j^{(J)}} \sum_j w_{kj}^{(JK)} (y_k^{(K)} - t_k) =$$

$$= y_i^{(I)} g'(z_j^{(J)}) \sum_j w_{kj}^{(JK)} (y_k^{(K)} - t_k)$$

where $g(x)$ is the logistic function and its derivative $g'(x) = g(x) - g(x)^2$.

The weight decay derivative is just the part of θ corresponding to what weight matrix is being respected to.

$$\alpha \frac{\partial E^{weight}}{\partial w^{(JK)}} = \alpha w^{(JK)}$$

$$\alpha \frac{\partial E^{weight}}{\partial w^{(IJ)}} = \alpha w^{(IJ)}$$

Now, we can put it all together

$$\frac{\partial E^{class}}{\partial w_{jk}^{(JK)}} = (y_k^{(K)} - t_k) y_j^{(J)} + \alpha w_{jk}^{(JK)}$$

$$\frac{\partial E^{class}}{\partial w_{ij}^{(IJ)}} = y_i^{(I)} g'(z_j^{(J)}) \sum_j w_{kj}^{(JK)} (y_k^{(K)} - t_k) + \alpha w_{ij}^{(IJ)}$$

2.2

Following is the written code. Note that "Just with weight decay" or "With cost" -parts are (obviously) run separately. Also note that here is no summation used but instead matrix multiplications. So instead of using for-loops to get each separate δw_{ij} and δw_{jk} for every input image, we can do some transposes and multiply each layer given in previous question, since they contains computation of the data for every image. E.g. data.inputs is a 256 x 1000 matrix, with 256 pixels times 1000 images.

```
% Partial calculation: dlog = derivative of logistic function,
% diff_pt = (prediction - target)
dlog = hid_output-hid_output.^2;
diff_pt = (class_prob-data.targets);

% Get the size of images
[pixel_size,image_size] = size(data.inputs);

% Just with weight decay
res.input_to_hid = wd_coefficient*model.input_to_hid;
res.hid_to_class = wd_coefficient*model.hid_to_class;

% With cost
res.input_to_hid = ((diff_pt.' * model.hid_to_class).' .* dlog)*data.inputs.' /
+ wd_coefficient*model.input_to_hid;
res.hid_to_class = (diff_pt*hid_output.')/image_size
+ wd_coefficient*model.hid_to_class;
```

The training data cost when running $net(0.1, 7, 10, 0, 0, false, 4)$ was 2.76838.

2.3

a) The best run was with momentum
b) The best training cost was 1.08343 (rounded to 5 decimals) and was run with LR=0.2 and M=0.9.

2.4

a) Validation cost is now 0.43016 (rounded to 5 decimals)
b) Validation cost is now 0.33451 (rounded), which is better than last
c) The best was with WD=0.001 which gave a validation cost for classification 0.28791
d) The best was with having 30 hidden nodes. This gave a validation cost of 0.31708
e) The best was with having 37 hidden nodes. This gave a validation cost of 0.26517.

f) The classification error rate on the test data is 0.07178 when run $net(0.001, 37, 1000, 0.35, 0.9, true, 100)$, which seams pretty good comparing to the other results.