ELEG 5040 Midterm Solution

Xingyu ZENG

March 12, 2015

1 Problem 1

The gradient for the thresholding function f(s) will be zero except the zero point. And it's difficult for the error to backpropagate from top layers to bottom layers.

$$w_{new} := w_{old} - lr * \frac{\partial L}{\partial f}\frac{\partial f}{\partial w} = w_{old} - 0 = w_{old} \tag{1}$$

2 Problem 2

- initialization problem, the initial weights are set to the saturation positions where gradients are close to 0.
- the whole network doesn't find the global optimum due to the initialization problem or the learning method(such as backpropagation)
- limitation of iterations
- ...

3 Problem 3

The trivial solution is the identity matrix. The reasons is, the autoencoder wants to reconstruct the input data and the identity matrix can satisfy that requirement.

The ways of adding regularization to avoid such a trivial solution:

- the number of neuron nodes in the hidden layers should be less than the input dimension.
- using corrupted data or noise data to reconstruct correct data
- add L-x normalization for the weights between the layers.
- ...

4 Problem 4

False. Because the channel number of the convolution layer can be very large which means a lot of filters exist in the convolution layer, then the dimension of output can be larger than that of input data. Thus the number of neurons is not always reduced.

5 Problem 5

As $\sum_{k=1}^{c} z_{i,k} = 1$, the value $z_{i,k}$ can be regarded as the predicting probability for sample i to be class k.

The object function of cross entropy is

$$Loss = -\sum_{i=1}^{N}\sum_{j=1}^{c} 1(y_i == j) log z_{i,j} = -\sum_{i=1}^{N} log z_{i,y_i} \tag{2}$$

$$= -\sum_{i=1}^{N} log P_{i,y_i} = -log \Pi_{i=1}^{N} P_{i,y_i} \tag{3}$$

Here $1(x) = 1$ if x is true, else $1(x) = 0$.

Thus the cross entropy is equivalent to the negative log-likelihood on the training dataset.

6 Problem 6

$$\frac{\partial L}{\partial w_{j,i'}} = \frac{\partial L}{\partial net_j} * \frac{\partial net_j}{w_{j,i'}} = \frac{\partial L}{\partial net_j} * x_{i'} \tag{4}$$

$$\Rightarrow \frac{\partial L}{\partial w_{j,i'}} = 0 \text{ if } x_{i'} = 0 \tag{5}$$

Thus all the weights $w_{j,i'}$ cannot be updated.

The input-to-hidden weights will not be updated at the first iteration but will be updated in the following iterations.

$$\frac{\partial L}{\partial w_{k,j}} = \frac{\partial L}{\partial net_k} * \frac{\partial net_k}{\partial w_{k,j}} = \frac{\partial L}{\partial net_k} * y_j \tag{6}$$

$$\frac{\partial L}{\partial y_j} = \sum_{k=1}^{c} \frac{\partial L}{\partial net_k} \frac{\partial net_k}{\partial y_j} = \sum_{k=1}^{c} \frac{\partial L}{\partial net_k} w_{k,j} \tag{7}$$

$$\frac{\partial L}{\partial w_{j,i}} = \sum_{j=1}^{H} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{j,i}} = \sum_{j=1}^{H} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial net_j} x_i \tag{8}$$

If all the hidden-to-output weights are initialized as zeros, then $w_{k,j} = 0$, $\frac{\partial L}{\partial y_j} = 0$, $\frac{\partial L}{\partial w_{j,i}} = 0$. Thus the input-to-hidden weights will not be updated at the first iteration.

But in the following iterations $w_{k,j}$ will be updated to be non-zeros and the input-to-hidden weights will be updated.

7 Problem 7

$$\frac{\partial f}{\partial s} = \frac{4}{(e^s + e^{-s})^2} \tag{9}$$

$$\frac{\partial^2 f}{\partial^2 s} = -\frac{8(e^{2s} - e^{-2s})}{(e^{2s} + 2 + e^{-2s})^2} \tag{10}$$

Small weights mean that the input s for nonlinear function are small and close to 0. When s is close to 0, $\frac{\partial^2 f}{\partial^2 s}$ is close to 0 which means the nonlinear function can be approximated as one linear function. Thus the network approximates a linear classifier.

Large weights mean that the input s for nonlinear function are large and far away from 0. When s is far from 0, $\frac{\partial f}{\partial s}$ is close to 0 which means the gradients of weights should be very small. Thus it's hard to update the weights.

The weights should be set to make sure the input for nonlinear function is close to the extreme point of second derivative. At that point, the first-order derivative is not so small and the second-order derivative is a extreme value.

8 Problem 8

$$P(h|x) = \frac{P(h,x)}{P(x)} = \frac{P(h,x)}{\sum_h P(h,x)} \tag{11}$$

$$= \frac{e^{-E(x,h)}}{\sum_h e^{-E(x,h)}} = \frac{e^{b^T x + c^T h + h^T W x}}{\sum_h e^{b^T x + c^T h + h^T W x}} \tag{12}$$

$$= \frac{e^{c^T h + h^T W x}}{\sum_h e^{c^T h + h^T W x}} = \frac{e^{(c^T + x^T W^T) h}}{\sum_h e^{(c^T + x^T W^T) h}} \tag{13}$$

$$= \frac{e^{\sum_i (c^T + x^T W^T)_i h_i}}{\sum_h e^{\sum_i (c^T + x^T W^T)_i h_i}} = \frac{\Pi_i e^{(c^T + x^T W^T)_i h_i}}{\sum_h \Pi_i e^{(c^T + x^T W^T)_i h_i}} \tag{14}$$

$$= \frac{\Pi_i e^{(c^T + x^T W^T)_i h_i}}{\Pi_i \sum_{h_i} e^{(c^T + x^T W^T)_i h_i}} = \Pi_i \frac{e^{(c^T + x^T W^T)_i h_i}}{\sum_{h_i} e^{(c^T + x^T W^T)_i h_i}} \tag{15}$$

$$P(h_i|x) = \frac{\sum_{j \neq i} P(h_j, h_i, x)}{P(x)} = \frac{e^{(c^T + x^T W^T)_i h_i} \sum_{j \neq i} \Pi_j e^{(c^T + x^T W^T)_j h_j}}{\Pi_i \sum_{h_i} e^{(c^T + x^T W^T)_i h_i}} \tag{16}$$

$$= \frac{e^{(c^T + x^T W^T)_i h_i} \sum_{j \neq i} \Pi_j e^{(c^T + x^T W^T)_j h_j}}{\Pi_i \sum_{h_i} e^{(c^T + x^T W^T)_i h_i}} = \frac{e^{(c^T + x^T W^T)_i h_i}}{\sum_{h_i} e^{(c^T + x^T W^T)_i h_i}} \tag{17}$$

$$\Rightarrow P(h|x) = \Pi_i P(h_i|x) \tag{18}$$

As the possible value for h contains only 0 and 1, thus

$$P(h_i = 1|x) = \frac{e^{(c^T + x^T W^T)_i 1}}{e^{(c^T + x^T W^T)_i 0} + e^{(c^T + x^T W^T)_i 1}} = \frac{1}{1 + e^{-(c_i + (Wx)_i^T)}} = \sigma(c_i + W_i * x) \quad (19)$$

9 Problem 9

For the receptive field of a neuron output of the pooling layer in (a), it's 8×8; For the receptive field of a neuron of the second convolution layer in (b), it's 13×13.