Neural Network

Homework 2

Ning Ma, A50055399

1 Perceptron

1. For 2-D, the decision boundary is

$$w_1 x_1 + w_2 x_2 = \theta \tag{1}$$

Assume x is a point on the boundary. So, we have

$$w^T x = \theta \tag{2}$$

Let x^0 be the point from which we want to compute the distance to the line. So, the distance is

$$\frac{w^\top (x - x^0)}{||w||_2} = \frac{w^\top x - w^\top x^0}{||w||_2}$$

So, the distance from the origin to the boundary is

$$\frac{w^\top x - w^\top 0}{||w||_2} = \frac{w^\top x}{||w||_2} \tag{3}$$

2.

(a) The learning rule is

$$w_j(t+1) = w_j(t) + \alpha(Teacher - Output)x_j \tag{4}$$

(b) The perceptron learning goes as the following:
The final weights and threshold are w_1 = -1, w_2 = -1, $theta$ = -1.

(c) This solution is not unique. For example, $w_1 = -2$, $w_2 = -2$, and $\theta = -2$ is another solution. Namely, the solution will not change if you add same constant to all the parameters.

Table 1: perceptron learning for NAND

x_1	x_2	w_1	w_2	Net	$Output$	$Teacher$	$theta$
1	1	0	0	0	1	0	0
0	0	-1	-1	0	0	1	1
0	1	-1	-1	-1	0	1	0
1	1	-1	0	-1	1	0	-1
1	0	-2	-1	-2	0	1	0
-	-	-1	-1	-	-	-	-1

(d) i. See code file 'readProcessData.m' for details. After preprocessing, the data are unitless. If the data have different units, different data points may have very different numerical value but actually represent the same physical measurement. Secondly, if the original data is very large, this preprocessing may avoid overflowing and let the learning converge appropriately.

ii. According to Figure **??**, the class are linearly separable for most feature spaces.

iii. See code for more details. For the stopping criteria, I choose to stop when the iteration reaches 1000 or the error drops below 0.05, whichever occurs first.

iv. The test error rate is just 3.33%

v. my learning rate is 0.05. When I slightly increase the learning rate, the perceptron learning will converge a little bit fast and vice versa. However, if my learning rate is too large, the learning procedure will not converge.

2 Multilayer Perceptron

(a) The cross entropy loss function for softmax regression is

$$E = -\sum_{l=0}^{K-1} 1_{\{label=l\}} \log\, y_l \tag{5}$$

where $y_l = \frac{\exp(a_l)}{\sum_{m=0}^{K-1} \exp(a_m)}$

For the output layer, we have

$$\delta_k = -\frac{\partial E}{\partial a_k} = -\sum_l \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial a_k} \tag{6}$$

$$= -\sum_l -\frac{1_{\{label=l\}}}{y_l}(y_l\delta_{lk} - y_l y_k) \tag{7}$$

$$= \sum_l 1_{\{label=l\}}(\delta_{lk} - y_k) \tag{8}$$

$$= \sum_l \delta_{lk} 1_{\{label=l\}} - y_k \sum_l 1_{\{label=l\}} \tag{9}$$

$$= 1_{\{label=k\}} - y_k = t_k - y_k \tag{10}$$

For the hidden layer, $y_j = g(a_j)$, we have

$$\delta_j = -\frac{\partial E}{\partial a_j} = -\sum_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_j} \tag{11}$$

$$= \sum_k \delta_k \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial y_j}\frac{\partial y_j}{\partial a_j} \tag{12}$$

$$= \sum_k \delta_k \frac{\partial \sum_l w_{lk} y_l}{\partial y_j} y_j^{'} = \sum_k \delta_k w_{jk} y_j^{'} \tag{13}$$

$$= y_j^{'} \sum_k \delta_k w_{jk} \tag{14}$$

where δ_k has been computed from the output layer.
(b) For the output layer, we have

$$w_{jk} = w_{jk} - \alpha\frac{\partial E}{\partial w_{jk}} = w_{jk} - \alpha\frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{jk}} \tag{15}$$

$$= w_{jk} + \alpha\delta_k \frac{\partial \sum_l w_{lk} y_l}{\partial w_{jk}} \tag{16}$$

$$= w_{jk} + \alpha\delta_k y_j \tag{17}$$

For the hidden layer, we have

$$w_{ij} = w_{ij} - \alpha\frac{\partial E}{\partial w_{ij}} = w_{ij} - \alpha\frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} \tag{18}$$

$$= w_{ij} + \alpha\delta_j \frac{\partial \sum_l w_{lj} x_l}{\partial w_{ij}} \tag{19}$$

$$= w_{ij} + \alpha\delta_j x_i \tag{20}$$

where we have already computed the δ_k and δ_j in part (a).

(c) For the output layer, since $w_{jk} = w_{jk} + \alpha\delta_k y_j$, we have

$$W_{HO} = W_{HO} + \alpha y^{(j)} \otimes \delta^{(k)} \tag{21}$$

where $y^{(j)}$ is a column-vector output from hidden layer, $\delta^{(k)}$ is a column-vector δ from the output layer, and $\otimes$ is an outer product operator.

Similarly, for the hidden layer, since $w_{ij} = w_{ij} + \alpha\delta_j x_i$, we have

$$W_{IH} = W_{IH} + \alpha x^{(i)} \otimes \delta^{(j)} \tag{22}$$

where $x^{(i)}$ is a column-vector input, $\delta^{(j)}$ is a column-vector δ from the hidden layer, and $\otimes$ is an outer product operator.

Since $\delta_j = y_j^{'} \sum_k \delta_k w_{jk}$, we have

$$\delta^{(j)} = (y^{'})^{(j)} \odot (W_{HO} \bullet \delta^{(k)}) \tag{23}$$

where $\odot$ is an element-wise multiplication operator. Thus, we have

$$W_{IH} = W_{IH} + \alpha x^{(i)} \otimes \Big((y^{'})^{(j)} \odot (W_{HO} \bullet \delta^{(k)})\Big) \tag{24}$$

(d)

4.

(a) The result of the 10 2-way classification is in Table **??**

Table 2: 10 2-way classification

$0-all$	$1-all$	$2-all$	$3-all$	$4-all$	$5-all$	$6-all$	$7-all$	$8-all$	$9-all$
0.977	0.982	0.957	0.951	0.957	0.950	0.960	0.955	0.921	0.935

(b) The overall test accuracy is 0.846.

5.

(a)

(b) The test accuracy on the test data is 0.860.

(c) The test accuracy is a little bit hight than the one-vs-all logistic regression. It is because softmax regression can directly computes the probability of each class, and then vote for the one with highest probability. So, softmax regression will provide a little bit more accurate result.

3 Appendix

The following is the source code