Writing Assignment

Jingyun Jia

(a) 4.1: The output of the perceptron is $o = sgn(w\_0 + w\_1 * x\_1 + w\_2 * x\_2)$, and as the coordinates of two points on the line are (-1, 0) and (0, 2), the equation of the line is $2 + 2 * x\_1 - x\_2 = 0$. To check the signs, we can use origin *(0, 0) (negative).* However, the output of perceptron of the origin is positive. Thus we should negate the previous values as: $w\_0 = -2$, $w\_1 = -2$, $w\_2 = 1$.

(b) 4.2

i: $A \wedge \neg B$

$w\_0 = -1$, $w\_1 = 1$, $w\_2 = -1$

A	B	output (before threshold)	output (after threshold)
-1	-1	-1	-1
-1	1	-3	-1
1	-1	1	1
1	1	-1	-1

ii: First convert A XOR B into DNF as $(A \wedge \neg B) \vee (\neg A \wedge B)$, we can define the perceptrons P1 and P2 for $(A \wedge \neg B)$ and $(\neg A \wedge B)$, then compose the output of P1 and P2 into a perceptron P3 that implements o(P1) v o(P2), thus we can define weights of P1 as w_1_0 = -1, w_1_1 = 1, w_1_2 = -1 (from question i). Similarly, we can define weights of P2 as w_2_0 = -1, w_2_1 = -1, w_2_2 = 1. And weights of P3 as w_3_0 = 1, w_3_1 = 1, w_3_2 = 1.

A	B	P1 (before threshold)	P1 (after threshold)	P2 (after threshold)	P2 (after threshold)	P3 (after threshold)	P3 (after threshold)
-1	-1	-1	**-1**	-1	**-1**	-1	**-1**
-1	1	-3	**-1**	1	**1**	1	**1**
1	-1	1	**1**	-3	**-1**	1	**1**
1	1	-1	**-1**	-1	**-1**	-1	**-1**

(c) 4.9: Such network (8 x 1 x 8) doesn't exit. There is values for the hidden unit weights that can create the hidden unit encoding like 0.1, 0.2 … 0.8. But they couldn't be correctly decoded. For example, if the weights from hidden layer to output i is w_i, to decode the hidden value of input 00000001 (0.1), we should have w_1 > 0 (sigmoid(net1) > 0.5). However, when the input is not 00000001, we would want w_1 < 0 (sigmoid(net1) < 0.5). Thus there is a contradiction.

(d)

i. output of three hidden units

[1. 0. 0. 0. 0. 0. 0. 0.] -> [0.02330608 0.94964275 0.03244465] -> [1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0.] -> [0.96715981 0.81395512 0.00859622] -> [0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0.] -> [0.25794738 0.01838361 0.00799996] -> [0. 0. 1. 0. 0. 0. 0. 0.]

[0. 0. 0. 1. 0. 0. 0. 0.] -> [0.98689529 0.98849353 0.94828497] -> [0. 0. 0. 1. 0. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 0.] -> [0.98823483 0.01025096 0.34575952] -> [0. 0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 0. 0. 1. 0. 0.] -> [0.01464087 0.96785581 0.95182017] -> [0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0.] -> [0.66511471 0.04564384 0.99481247] -> [0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 0. 1.] -> [0.00663294 0.02238707 0.61180058] -> [0. 0. 0. 0. 0. 0. 0. 1.]
output of four hidden units
[1. 0. 0. 0. 0. 0. 0. 0.] -> [0.98217337 0.9138675 0.17248933 0.98140507] -> [1. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0.] -> [0.3403114 0.02267727 0.0262758 0.76010772] -> [0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0.] -> [0.01182193 0.16999956 0.97386967 0.96892287] -> [0. 0. 1. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0. 0.] -> [0.51131049 0.59313516 0.01389186 0.02305874] -> [0. 0. 0. 1. 0. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 0.] -> [0.97464819 0.98226441 0.95672401 0.06855901] -> [0. 0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 0. 0. 1. 0. 0.] -> [0.05864502 0.48302108 0.96593779 0.01105608] -> [0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0.] -> [0.93979326 0.01436514 0.97224952 0.48460382] -> [0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 0. 1.] -> [0.0160151 0.97361508 0.15826047 0.84971748] -> [0. 0. 0. 0. 0. 0. 0. 1.]

If we round these hidden values, then output of three hidden units looks like
[0, 1, 0]
[1, 1, 0]
[0, 0, 0]
[1, 1, 1]
[1, 0, 0]
[0, 1, 1]
[1, 0, 1]
[0, 0, 1]
There are three hidden units, thus can represent 8 different input (2^3). And the rounded hidden values of four hidden units network looks like:
[1, 1, 0, 1]
[0, 0, 0, 1]
[0, 0, 1, 1]
[1, 1, 0, 0]
[1, 1, 1, 0]
[0, 0, 1, 0]
[1, 0, 1, 0]
[0, 1, 0, 1]
4 hidden values can represent 16 (2^4) different digits, thus it also works for the identity dataset.
The hidden values represent contracted features. And since two networks constructed different sets of features, the magnitudes of hidden values are different.

ii. The performances (accuracy) on iris-test data with and without validation set looks like:

Noise Rate (%)	w/ validation	w/o validation
0	0.96	0.96
2	0.98	0.96
4	0.94	0.94
6	0.96	0.96
8	1	0.96
10	0.92	0.92
12	0.96	0.96
14	0.98	0.96
16	0.88	0.76
18	0.94	0.96
20	0.96	0.96

I maintained two sets of weights during training process. The prediction results generated by the final weights (after 5000 iterations) and the weights having best performance on validation set (split 0.15 of training set), the goal is to reduce overfitting. Although I find that the accuracy varies when using different validation sets, it can be seen that in general, using the set of weights which has the highest accuracy on validation set results in higher accuracy on test set. And the model tend to be more robust (stable performance) when using validation set.