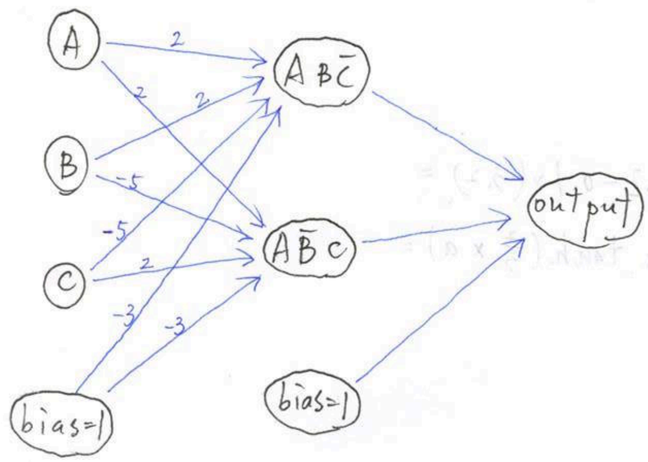
24-787 AIML HW3

1\. (a)

$$(A\cdot B)\oplus(A\cdot C)=(A\cdot B+AC)\cdot\overline{(AB\cdot AC)}=(AB+AC)\cdot(\overline{AB}+\overline{AC})$$

$$=(AB+AC)\cdot(\bar{A}+\bar{B}+\bar{A}+\bar{C})=AB\bar{C}+A\bar{B}C.$$

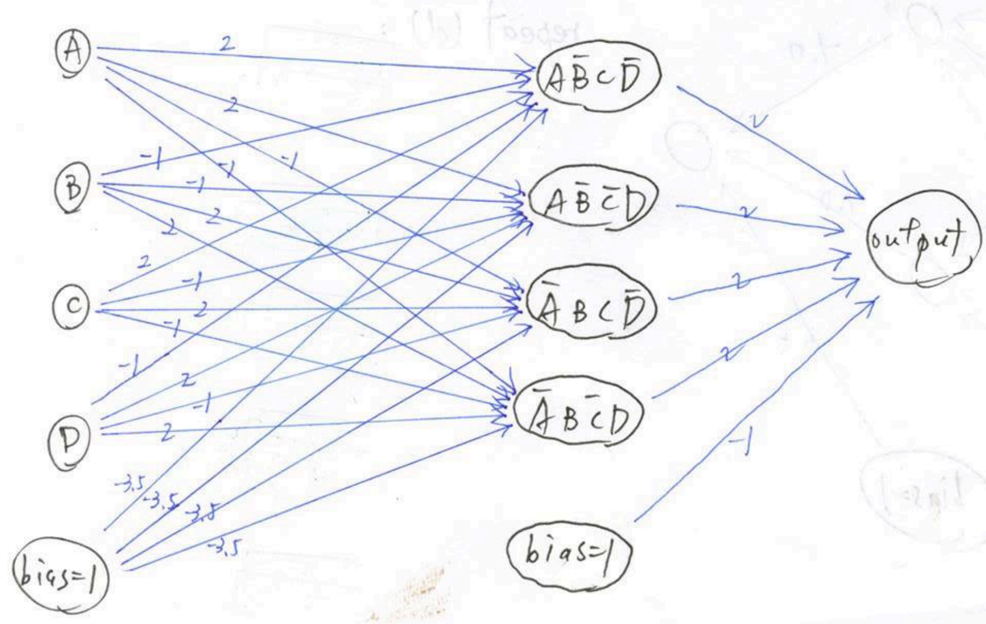


(b)

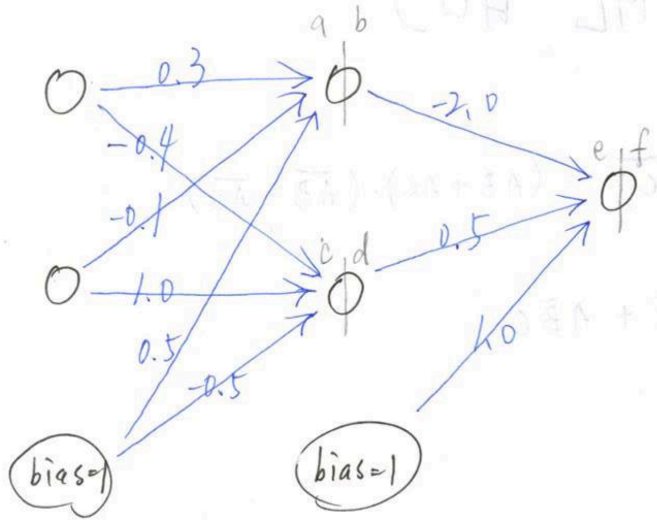
$$(A\oplus B)\cdot(C\oplus D)=(A+B)(\overline{AB})\cdot(C+D)(\overline{CD})=(A+B)(\bar{A}+\bar{B})\cdot(C+D)(\bar{C}+\bar{D})$$

$$=(A\bar{B}+\bar{A}B)(C\bar{D}+\bar{C}D)$$

$$=A\bar{B}C\bar{D}+A\bar{B}\bar{C}D+\bar{A}BC\bar{D}+\bar{A}B\bar{C}D$$



2. (a)



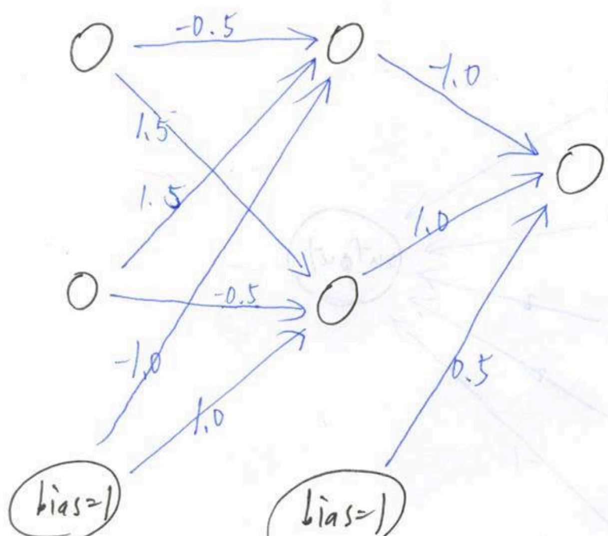
(b, c, d)

in the report.

(e)

repeat(a):

repeat(b, c, d) : in the report.

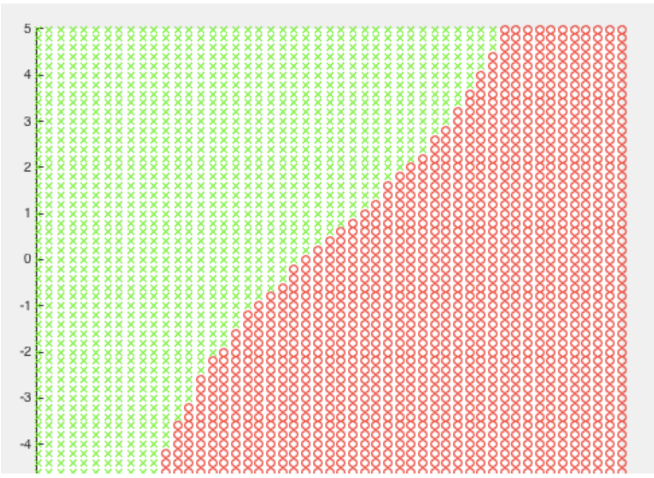


2.

(b) Matlab code:

```
indexi=1;
indexj=1;
output=zeros(51);
for i=-5:0.2:5
    for j=-5:0.2:5
        a=0.3*i-0.1*j+0.5;                   %equations
        b=1.716*tanh(2/3*a);                  %equations
        c=-0.4*i+1*j-0.5;                    %equations
        d=1.716*tanh(2/3*c);                 %equations
        e=-2*b+0.5*d+1;                       %equations
        f=1.716*tanh(2/3*e);                  %equations
        output(indexi,indexj)=f;
        indexj=indexj+1;
        if f>=0
            scatter(i,j,'x','g');
            hold on;
        else
            scatter(i,j,'o','r');
            hold on;
        end
    end
    indexi=indexi+1;
    indexj=1;
end
```

(c)



(d)Matlab code:

```
x1=2.2;
x2=-3.2;

a=0.3*x1-0.1*x2+0.5;       %equations
b=1.716*tanh(2/3*a);       %equations
c=-0.4*x1+1*x2-0.5;        %equations
d=1.716*tanh(2/3*c);       %equations
e=-2*b+0.5*d+1;            %equations
f=1.716*tanh(2/3*e);       %equations

y1=f  %display output y

x3=-3.2;
x4=2.2;

a=0.3*x3-0.1*x4+0.5;             %equations
b=1.716*tanh(2/3*a);             %equations
c=-0.4*x3+1*x4-0.5;              %equations
d=1.716*tanh(2/3*c);             %equations
e=-2*b+0.5*d+1;                  %equations
f=1.716*tanh(2/3*e);             %equations

y2=f   %display output y
```

answer:

the output of (x1,x2)=(2.2,-3.2)= -1.5897

the output of (x1,x2)=(-3.2,2.2)= 1.6735

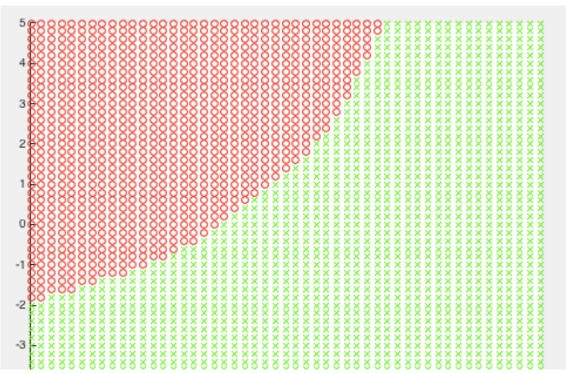
(e)

repeat(b)

Matlab code:

```
indexi=1;
indexj=1;
output=zeros(51);
for i=-5:0.2:5
    for j=-5:0.2:5
        a=-0.5*i+1.5*j-1;                  %equations
        b=1.716*tanh(2/3*a);               %equations
        c=1.5*i-0.5*j+1;                   %equations
        d=1.716*tanh(2/3*c);               %equations
        e=-1*b+1*d+0.5;                    %equations
        f=1.716*tanh(2/3*e);               %equations
        output(indexi,indexj)=f;
        indexj=indexj+1;
        if f>=0
            scatter(i,j,'x','g');
            hold on;
        else
            scatter(i,j,'o','r');
            hold on;
        end
    end
    indexi=indexi+1;
    indexj=1;
end
scatter(x,y,'x','g');
```

repeat(c)



repeat (d)

Matlab code:

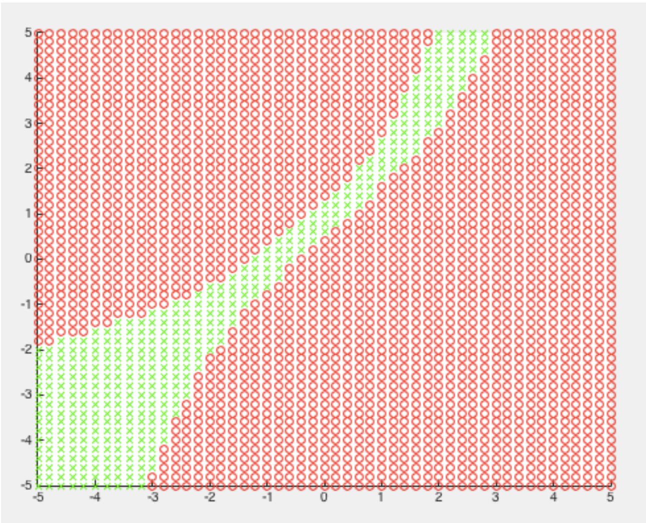
```
x1=2.2;

x2=-3.2;

x3=-3.2;

x4=2.2;


a=-0.5*x1+1.5*x2-1;            %equations

b=1.716*tanh(2/3*a);          %equations

c=1.5*x1-0.5*x2+1;            %equations

d=1.716*tanh(2/3*c);           %equations

e=-1*b+1*d+0.5;                %equations

f=1.716*tanh(2/3*e);           %equations


y1=f  %display output y


a=-0.5*x3+1.5*x4-1;            %equations

b=1.716*tanh(2/3*a);           %equations

c=1.5*x3-0.5*x4+1;            %equations

d=1.716*tanh(2/3*c);           %equations

e=-1*b+1*d+0.5;                %equations

f=1.716*tanh(2/3*e);           %equations


y2=f   %display output y
```

answer:

the output of (x1,x2)=(2.2,-3.2)= 1.6979

the output of (x1,x2)=(-3.2,2.2)= -1.6464 extra credit:

the final plot has to be exactly the combination of two plot above. So I multiplied two outputs from two networks above and check their results. If the result larger than 0,then plot a “x”, meaning I got positive output, and plot a ”o” otherwise.



3. (a)

$$f(x) = (1+e^{-x})^{-1}$$

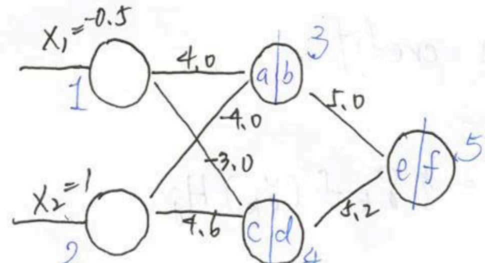
$$f'(x) = -(1+e^{-x})^{-2} \cdot (e^{-x}) \cdot (-1) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{1+e^{-x}} \cdot \frac{1}{1+e^{-x}} = f(x)\cdot(1-f(x))$$

(b) output =

$$\frac{1}{1+\exp\left(-\left(\frac{5}{1+\exp(-(4\times(-5)-4\times 1))} + \frac{5.2}{1+\exp(-(-3\times(-0.5)+4.6\times 1))}\right)\right)}$$

$$= 0.9945$$

(c)



$$a = 4.0\times(-0.5) + (-4.0)\times 1 = -6$$

$$b = (1+e^{6})^{-1} = 0.0025$$

$$c = (-3.0)\times(-0.5) + 4.6\times 1 = 6.1$$

$$d = (1+e^{-6.1})^{-1} = 0.9978$$

$$e = 5\times 0.0025 + 5.2\times 0.9978 = 5.201$$

$$f = 0.9945$$

$$\Delta W_{54} = \eta\, \delta_k f'(y_k) H_j$$

$$= 0.1 \times (0.9 - 0.9945) \times f(5.201)(1 - f(5.201)) \times d$$

$$= 0.1 \times (-0.0945) \times 0.0054 \times 0.9978 = -5.09 \times 10^{-5}$$ #

$$\Delta W_{31} = \eta \left[\sum_k \delta_k f'(y_k) W_{jk} \right] f'(h_j) x_i$$

$$= 0.1 \times \left((0.9 - 0.9945) \times 0.0054 \times 5.0 \right) \times f(-6)(1 - f(-6)) \times (-0.5)$$

$$= 0.1 \times (-0.0945) \times 0.0054 \times 5 \times 0.00246 \times (-0.5) = 3.14 \times 10^{-10}$$ #

(d) extra credit:

$$\Delta W_{53} = \eta\, \delta_k f'(y_k) H_j$$

$$= 0.1 \times (0.9 - 0.9945) \times 0.0054 \times b = -1.276 \times 10^{-7}$$

$$\Delta W_{41} = \eta \left(\sum_k \delta_k f'(y_k) W_{jk} \right) f'(h_j) x_i$$

$$= 0.1 \times (-0.0945) \times 0.0054 \times 5.2 \times 0.0022 \times (-0.5) = 2.92 \times 10^{-7}$$

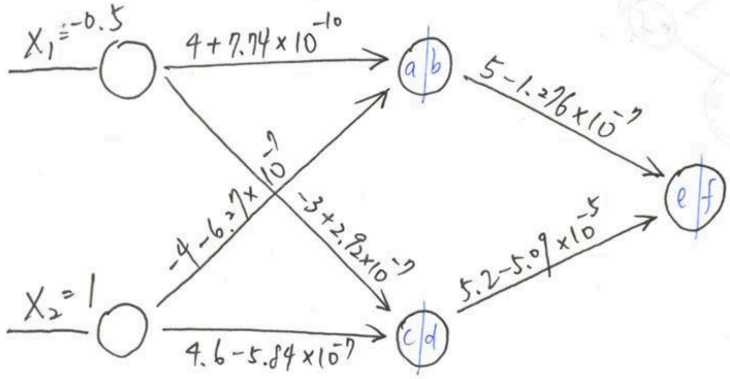
$$\Delta W_{32} = \eta \left(\sum_k \delta_k f'(y_k) W_{jk} \right) f'(h_j) x_i$$

$$= 0.1 \times (-0.0945) \times 0.0054 \times 5 \times 0.00246 \times 1 = -6.27 \times 10^{-7}$$

$$\Delta W_{42} = \eta \left(\sum_k \delta_k f'(y_k) W_{jk} \right) f'(h_j) x_i$$

$$= 0.1 \times (-0.0945) \times 0.0054 \times 5.2 \times 0.0022 \times 1 = -5.84 \times 10^{-7}$$

new network :



$a = -6 - 6.27 \times 10^{-7}$

$b = 0.0025$

$c = 6.099$

$d = 0.9978$

$e = 5.201$

$f \approx 0.9945 \ (< 0.9945)$

Conclusion :

Originally, the prediction error is very small, and the learning rate is small as well. So the ΔW will be extremely small, which doesn't lead to obvious error changing. (only slightly decreased)

4.

(a)

In my matlab code, I extracted the data from Netlab first. Then I split the data into two parts, one of which is training data and the other one is validation data. I put the training data into “netopt” function. Then I put training data, validation data and test data, which is also extracted from Netlab, into “mlpfwd” function to calculate all kinds of error.

What’s more, the cross-validation method I used is “holdout”, meaning that I hold part of the data as validation data, instead of training all.

(b)

parameters:

net: structure

a: training data

b: validation data

y: output

output: transformed output(0~9)

(c)

training error: difference between training labels and the training output from training data

validation error: difference between the outputs of networks using training data and validation data as inputs.

Test error: difference between the outputs of networks using training data and test data as inputs.

First of all, for training error, using less training sample I got good performance without error. However, when increasing number of training samples, I got some training error. I think the reason was, using only 50 epochs is not enough to update all the weights to get the result correct if using too many samples. I think it will be improved with higher epochs or higher learning rate.

Hidden units \ Training samples	100	1000	10000	60000
100	0%	0%	0.34%	4.05%
500	0%	0%	3.08%	4.62%
1000	0%	0%	4.89%	6.83%

Training error(%), epochs=50

For validation error and test error, the more training samples I used, the lower error rate I got. And I think in this question, the number of hidden units didn't affect the error rate too much.
What's more, there is high error rate when I used less training sample with high hidden unit. I think that was because of overfitting. And the overfitting problem came from the fact that I used too many hidden units to train relatively less samples. So when I increased the training samples, the error became lower.

Hidden units \ Training samples	100	1000	10000	60000
100	31.38%	12.12%	5.74%	
500	31.52%	13.04%	6.52%	
1000	31.4%	14.15%	7.99%	

validation error(%), epochs=50

Training samples / Hidden units	100	1000	10000	60000
100	31.8%	11.7%	5.77%	4.46%
500	32.29%	12.47%	6.27%	4.84%
1000	31.64%	13.68%	7.7%	6.97%

test error(%), epochs=50

It was obvious that using more training samples, epochs or hidden units takes more time to train, so we have to make a trade-off between time and accuracy. Finally, I chose the network with 30000 training samples, 200 epochs and 500 hidden units as my best network.

Error type / Epochs	Training error	Validation error	Test error
50	4.51%	5.44%	4.99%
100	0.07%	2.68%	2.42%
200	0%	2.44%	2.27%

Training sample=30000, hidden units=500

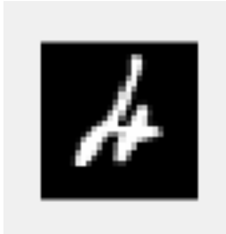
(d)

misclassified:

9th:

5→6

248th :



4→6

correctly classified: