UMBC CSEE

Foundations of Data Mining

CS-691

Homework Assignment - 4

Sandipan Dey

2009

1. Construct a feed-forward 3-layer (one input layer, one hidden layer, and one output node) perceptron network that correctly learns the XOR function. Clearly show the link weights and the thresholds. Construct this network through analytical observations, not by running a neural network simulation environment.

Answer:

Let's denote the input binary variables by $x_1, x_2 \in \{0,1\}$. We have to learn the binary function $XOR: \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$, so that the output binary variable $y = x_1 \oplus x_2$.
Since we know that XOR is linearly inseparable function, we can't design the perceptron using just 2-layer perceptron, for if we could, $\exists w_1, w_2$ (weights) and a non-negative threshold $t \geq 0$ such that

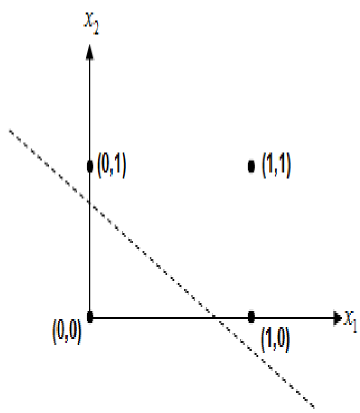
$$y = \begin{cases} 0, w_1 x_1 + w_2 x_2 < t \\ 1, w_1 x_1 + w_2 x_2 \geq t \end{cases}$$

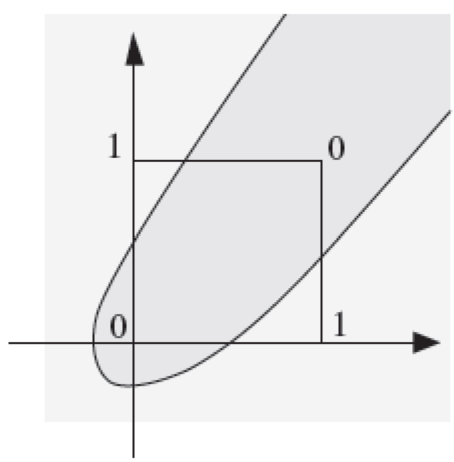
but since $y = x_1 \oplus x_2$, with XOR function defined by the following:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

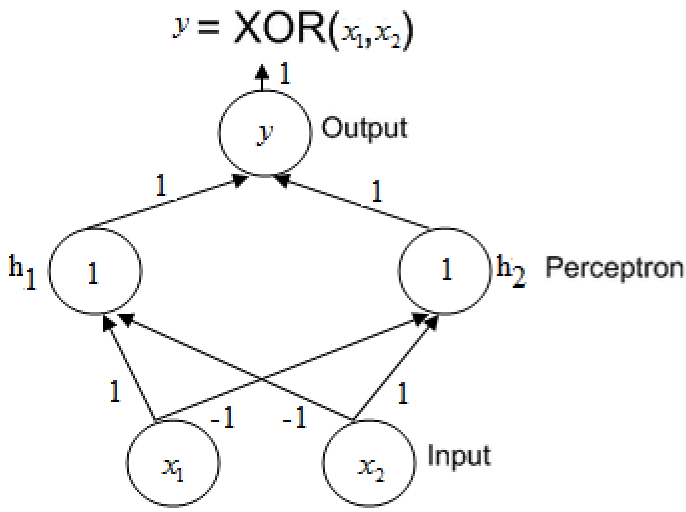
with all the following inequalities must be satisfied:
$w_1 \geq t,\ \ w_2 \geq t,\ \ 2t \leq w_1 + w_2 < t,\ \ t \geq 0 \Rightarrow t > 2t \wedge t \geq 0 \Rightarrow t < 0 \wedge t \geq 0$,
a contradiction.

Hence, we need a 3-layer perceptron network, with a hidden layer placed in between the input and output layer.





Now the XOR function can be realized as a linearly separable function of couple of linearly separable functions (in 2 steps): $y = x_1 \oplus x_2 = x_1 \overline{x_2} + \overline{x_1} x_2 = AND(OR(x_1, NOT(x_2)), OR(NOT(x_1), x_2))$, all of AND, OR, NOT functions being linearly separable.

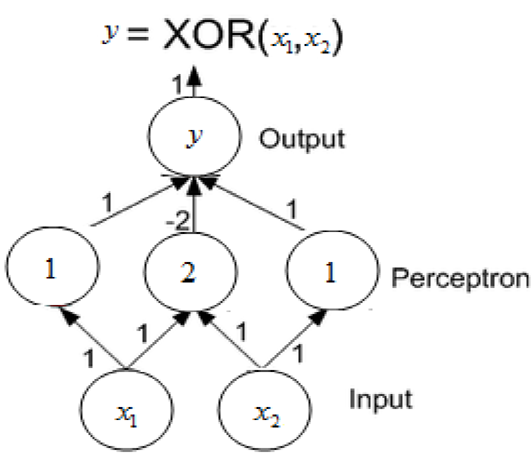


Threshold is 1 everywhere at both the nodes in the hidden layer and also at the output node.

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

<u>Alternative design</u>

In order design the following 3-layer perceptron, we see that the problem is due to the (1,1) input, we must make it less than the threshold somehow, to have output 0. We can see that it can be done by associating a negative weight to a node in the hidden layer, when both x_1, x_2 have value 1.



2. $\varphi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]$

$$\Rightarrow \varphi(u).\varphi(v) = [1, \sqrt{2}u_1, \sqrt{2}u_2, \sqrt{2}u_1 u_2, u_1^2, u_2^2].[1, \sqrt{2}v_1, \sqrt{2}v_2, \sqrt{2}v_1 v_2, v_1^2, v_2^2]^T$$

$$\Rightarrow K(u,v) = \varphi(u).\varphi(v) = 1 + 2u_1 v_1 + 2u_2 v_2 + 2u_1 v_1 u_2 v_2 + u_1^2 v_1^2 + u_2^2 v_2^2$$

$$\Rightarrow K(u,v) = \left(\begin{bmatrix} u_1 & u_2 \end{bmatrix} . \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + 1 \right)^2 = (u.v+1)^2$$

Now, the dual problem becomes

$$L_D(\lambda) = \sum_{i=1}^{N} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \phi(x_i).\phi(x_j)$$

$$\Rightarrow L_D(\lambda) = \sum_{i=1}^{N} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

We have the following training points:

x_1=(1, 1, -), x_2= (1, 0, +), x_3= (0, 1, +), x_4=(0, 0, -), y_1=y_3=+1, y_2=y_4=-1.

Assuming + label to be with value +1 and the – label with value -1, let's first construct the Kernel matrix:

$$K(x_1, x_1) = ([11].[11]^T + 1)^2 = (2+1)^2 = 9,$$

$$K(x_1, x_2) = ([11].[10]^T + 1)^2 = (1+1)^2 = 4$$

$$K(x_2, x_3) = ([10].[01]^T + 1)^2 = (0+1)^2 = 1$$

	Kernel	(1,1)	(1,0)	(0,1)	(0,0)
x_1	(1, 1)	9	4	4	1
X_2	(1, 0)	4	9	1	1
X_3	(0, 1)	4	1	9	1
X_4	(0, 0)	1	1	1	9

$$\sum_{i=1}^{4} \sum_{j=1}^{4} \lambda_i \lambda_j y_i y_j K(x_i, x_j) = 9\lambda_1^2 - 4\lambda_1\lambda_2 + 4\lambda_1\lambda_3 - \lambda_1\lambda_4$$

$$- 4\lambda_1\lambda_2 + 9\lambda_2^2 - \lambda_2\lambda_3 + \lambda_2\lambda_4 + 4\lambda_1\lambda_3 - \lambda_2\lambda_3 + 9\lambda_3^2 - \lambda_3\lambda_4 - \lambda_1\lambda_4 + \lambda_2\lambda_4 - \lambda_3\lambda_4 + 9\lambda_4^2$$

$$= 9(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2) - 8\lambda_1\lambda_2 + 8\lambda_1\lambda_3 - 2\lambda_1\lambda_4 - 2\lambda_2\lambda_3 + 2\lambda_2\lambda_4 - 2\lambda_3\lambda_4$$

$$L_D = \sum_{i=1}^{4} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

$$= (\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4) - (9/2)(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2) + 4\lambda_1\lambda_2 - 4\lambda_1\lambda_3 + \lambda_1\lambda_4 + \lambda_2\lambda_3 - \lambda_2\lambda_4 + \lambda_3\lambda_4$$

Taking partial derivative w.r.t . λ_i for optimizing L_D, we get the following 4 equations:

$$9\lambda_1 - 4\lambda_2 + 4\lambda_3 - \lambda_4 = 1$$

$$- 4\lambda_1 + 9\lambda_2 - \lambda_3 + \lambda_4 = 1$$

$$4\lambda_1 - \lambda_2 + 9\lambda_3 - \lambda_4 = 1$$

$$- \lambda_1 + \lambda_2 - \lambda_3 + 9\lambda_4 = 1$$

With solution: $\lambda_1 = 0.1765, \lambda_2 = 0.1838, \lambda_3 = 0.0662, \lambda_4 = 0.1176$

Also, $\varphi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]$

$$\Rightarrow \varphi(x_1) = [1, \sqrt{2}, \sqrt{2}, \sqrt{2}, 1, 1]$$

$$\Rightarrow \varphi(x_2) = [1, \sqrt{2}, 0, 0, 1, 0]$$

$$\Rightarrow \varphi(x_3) = [1, 0, \sqrt{2}, 0, 0, 1]$$

$$\Rightarrow \varphi(x_4) = [1, 0, 0, 0, 0, 0]$$

$$\Rightarrow w_o = \sum_i \lambda_i y_i \phi(x_i) = [-0.5690 \quad -0.3415 \quad -0.5078 - 0.6014 \;\; 0.0063 \;\; 0.6342]$$

$$\Rightarrow \max m \arg in = (1/2) \| w_o \|^2 = 0.5 \times 1.2091 = 0.6046 unit$$

3. (a) Applying Bayes theorem,

P(A=1|+) = $\dfrac{P(+ \mid A=1)P(A=1)}{P(+).} = \dfrac{(3/5).(1/2)}{(1/2)} = \dfrac{3}{5}$

P(B=1|+) = $\dfrac{P(+ \mid B=1)P(B=1)}{P(+).} = \dfrac{(2/4).(4/10)}{(1/2)} = \dfrac{2}{5}$

P(C=1|+) = $\dfrac{P(+ \mid C=1)P(C=1)}{P(+).} = \dfrac{(4/5).(1/2)}{(1/2)} = \dfrac{4}{5}$

P(A=1|-) = $\dfrac{P(- \mid A=1)P(A=1)}{P(-).} = \dfrac{(2/5).(1/2)}{(1/2)} = \dfrac{2}{5}$

P(B=1|-) = $\dfrac{P(- \mid B=1)P(B=1)}{P(-).} = \dfrac{(2/4).(4/10)}{(1/2)} = \dfrac{2}{5}$

P(C=1|-) = $\dfrac{P(- \mid C=1)P(C=1)}{P(-).} = \dfrac{(1/5).(1/2)}{(1/2)} = \dfrac{1}{5}$

(b) P(+|A=1, B=1, C=1)

$$= \frac{P(A=1, B=1, C=1 \mid +)P(+)}{P(A=1, B=1, C=1)} = \frac{P(A=1 \mid +)P(B=1 \mid +)P(C=1 \mid +)P(+)}{P(A=1, B=1, C=1)}$$

$$= \frac{(3/5).(2/5).(4/5).(1/2)}{1/10} = \frac{3.1.4.10}{5.5.5} = \frac{24}{25}$$

P(-|A=1, B=1, C=1)

$$= \frac{P(A=1, B=1, C=1 \mid -)P(-)}{P(A=1, B=1, C=1)} = \frac{P(A=1 \mid -)P(B=1 \mid -)P(C=1 \mid -)P(-)}{P(A=1, B=1, C=1)}$$

$$= \frac{(2/5).(2/5).(1/5).(1/2)}{1/10} = \frac{1.2.1.10}{5.5.5} = \frac{4}{25}$$

Hence, prediction using Naive Bayesian is class label +.

(c) P(A = 1) = 1/2.

P(B = 1)= 4/10=2/5.

P(A = 1, B=1) =2/10=1/5.

Since P(A=1, B=1)=1/5=(1/2).(2/5)=P(A=1)P(B=1), A and B are statistically independent.

4. The popular approaches to handle missing values are the following:

a) *Discard instances with missing values*

b) Imputation (by mean substitution, multiple regression, MLE, nearest neighbor etc)

Here we are going to apply both the techniques discard /mean substitution separately on the data. We use 10 fold cross-validation technique. To measure performance of the model, we use the following metrics:

- Accuracy = (TP+TN)/(TP+TN+FP+FN)
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)

Where TP = True Positive, FP = False Positive

TN = True Negative, FN = False Negative

Using **<u>Naïve Baysian Classifier</u>**, we obtain the following results using weka:

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===

Correctly Classified Instances	515	78.0303 %
Incorrectly Classified Instances	145	21.9697 %
Kappa statistic	0.5439	
Mean absolute error	0.2212	
Root mean squared error	0.434	
Relative absolute error	44.6569 %	
Root relative squared error	87.2083 %	
Total Number of Instances	660	(after discarding tuples with missing values)

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.601	0.072	0.873	0.601	0.712	0.901	+
	0.928	0.399	0.738	0.928	0.823	0.902	-
Weighted Avg.	0.78	0.251	0.799	0.78	0.772	0.902	

=== Confusion Matrix ===

```
  a   b   <-- classified as
 179 119 |   a = +
  26 336 |   b = -
```

with TP = 179, TN = 336, FP = 26, FN = 119.

Hence, Accuracy = $\frac{TP+TN}{TP+FP+TN+FN} = \frac{179+336}{179+26+336+119} = 0.78$

Precision = $\frac{TP}{TP+FP} = \frac{179}{179+26} = 0.83$

Recall = $\frac{TP}{TP+FN} = \frac{179}{179+119} = 0.60$

=== Stratified cross-validation ===

Correctly Classified Instances	537	77.8261 %
Incorrectly Classified Instances	153	22.1739 %
Kappa statistic	0.5372	
Mean absolute error	0.2227	
Root mean squared error	0.4355	
Relative absolute error	45.0874 %	
Root relative squared error	87.634 %	
Total Number of Instances	690	(without discarding tuples with missing values)

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.599	0.078	0.86	0.599	0.706	0.896	+
	0.922	0.401	0.742	0.922	0.822	0.896	-
Weighted Avg.	0.778	0.257	0.794	0.778	0.77	0.896	

=== Confusion Matrix ===

```
  a   b   <-- classified as
 184 123 |   a = +
  30 353 |   b = -
```

with TP = 184, TN = 353, FP = 30, FN = 123.

Hence, Accuracy = $\frac{TP+TN}{TP+FP+TN+FN} = \frac{183+353}{183+30+353+123} = 0.78$

Precision = $\frac{TP}{TP+FP} = \frac{184}{184+30} = 0.86$

Recall = $\frac{TP}{TP+FN} = \frac{184}{184+123} = 0.61$

As we can see, the performance is improved in this case (if we don't discard tuples with missing values).

<u>Using J48 classification algorithm (decision tree)</u>

```
J48 pruned tree
------------------

A9 = t
|   A10 = t: + (228.0/21.0)
|   A10 = f
|   |   A15 <= 444
|   |   |   A7 = v
|   |   |   |   A4 = u
|   |   |   |   |   A14 <= 112: + (16.57/1.57)
|   |   |   |   |   A14 > 112
|   |   |   |   |   |   A15 <= 70: - (30.0/10.0)
|   |   |   |   |   |   A15 > 70: + (2.0)
|   |   |   |   A4 = y
|   |   |   |   |   A13 = g: - (12.0/2.0)
|   |   |   |   |   A13 = s: + (3.0/1.0)
|   |   |   |   |   A13 = p: - (0.0)
|   |   |   |   A4 = l: - (0.0)
|   |   |   A7 = h: + (27.24/8.24)
|   |   |   A7 = bb
|   |   |   |   A3 <= 1.375: + (5.0/1.0)
|   |   |   |   A3 > 1.375: - (9.13/1.0)
|   |   |   A7 = ff: - (5.05/1.0)
|   |   |   A7 = j: - (1.01)
|   |   |   A7 = z: + (0.0)
|   |   |   A7 = o: + (0.0)
|   |   |   A7 = dd: + (1.01/0.01)
|   |   |   A7 = n: + (0.0)
|   |   A15 > 444: + (21.0/1.0)
A9 = f
|   A3 <= 0.165
|   |   A7 = v
|   |   |   A2 <= 35.58: - (18.72/3.44)
|   |   |   A2 > 35.58: + (3.6/0.16)
|   |   A7 = h: - (0.0)
|   |   A7 = bb: + (1.24/0.08)
|   |   A7 = ff: - (4.96/0.64)
|   |   A7 = j: + (1.24/0.08)
|   |   A7 = z: - (0.0)
|   |   A7 = o: - (0.0)
|   |   A7 = dd: - (0.0)
|   |   A7 = n: + (1.24/0.08)
|   A3 > 0.165: - (298.0/12.0)
```

Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===

Correctly Classified Instances 595 86.2319 %
Incorrectly Classified Instances 95 13.7681 %
Kappa statistic 0.7208
Mean absolute error 0.1907
Root mean squared error 0.3303
Relative absolute error 38.6088 %
Root relative squared error 66.4545 %
Total Number of Instances 690

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.837	0.117	0.851	0.837	0.844	0.886	+
	0.883	0.163	0.871	0.883	0.877	0.886	-
Weighted Avg.	0.862	0.143	0.862	0.862	0.862	0.886	

=== Confusion Matrix ===

```
  a   b   <-- classified as
257  50 |   a = +
 45 338 |   b = -
```

Hence, Accuracy = $\dfrac{TP+TN}{TP+FP+TN+FN} = \dfrac{257+338}{257+50+338+45} = 0.86$

Precision = $\dfrac{TP}{TP+FP} = \dfrac{257}{257+50} = 0.83$

Recall = $\dfrac{TP}{TP+FN} = \dfrac{257}{257+45} = 0.85$

As we can see, the performance is still improved in this case.

<u>Linear Regression Model</u>

Class =

```
 0.1246 * A5=g,gg +
 0.8492 * A6=d,i,k,j,aa,m,c,w,e,q,r,cc,x +
-0.6328 * A6=j,aa,m,c,w,e,q,r,cc,x +
 0.6058 * A6=aa,m,c,w,e,q,r,cc,x +
 0.1715 * A6=m,c,w,e,q,r,cc,x +
-0.3223 * A6=r,cc,x +
 0.5857 * A6=cc,x +
-0.6266 * A7=dd,j,v,bb,o,n,h,z +
 0.731  * A7=j,v,bb,o,n,h,z +
-0.7769 * A7=v,bb,o,n,h,z +
```

0.5557 * A7=n,h,z +
-0.4779 * A7=h,z +
1.1705 * A9=t +
0.2488 * A10=t +
0.0153 * A11 +
1.0218 * A13=p +
-0.0004 * A14 +
0 * A15 +
-1.2233

Time taken to build model: 0.27 seconds

=== Cross-validation ===

Correlation coefficient	0.7437
Mean absolute error	0.4588
Root mean squared error	0.6672
Relative absolute error	46.329 %
Root relative squared error	66.9622 %
Total Number of Instances	690

<u>Multilayer Perceptron</u>

Time taken to build model: 103.25 seconds
=== Cross-validation ===

Correlation coefficient	0.521
Mean absolute error	0.6858
Root mean squared error	1.0861
Relative absolute error	69.2607 %
Root relative squared error	108.9969 %
Total Number of Instances	690