Module 12

Machine Learning

Version 1 CSE IIT, Kharagpur

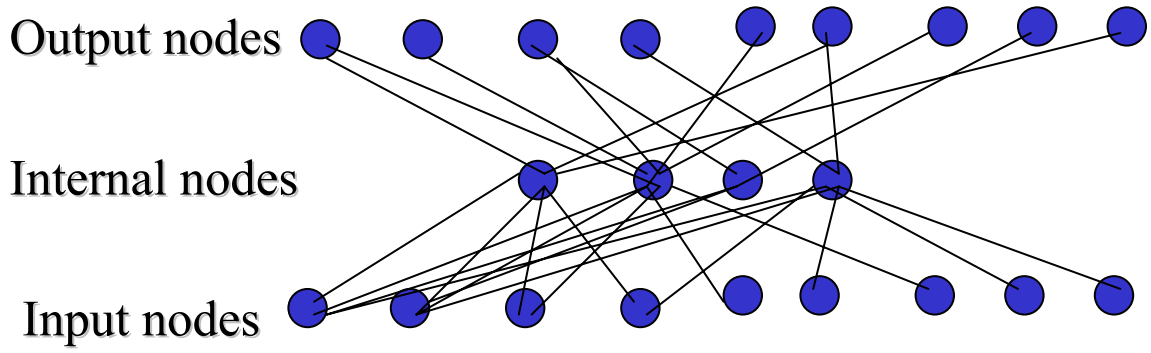
Lesson 39

Neural Networks - III

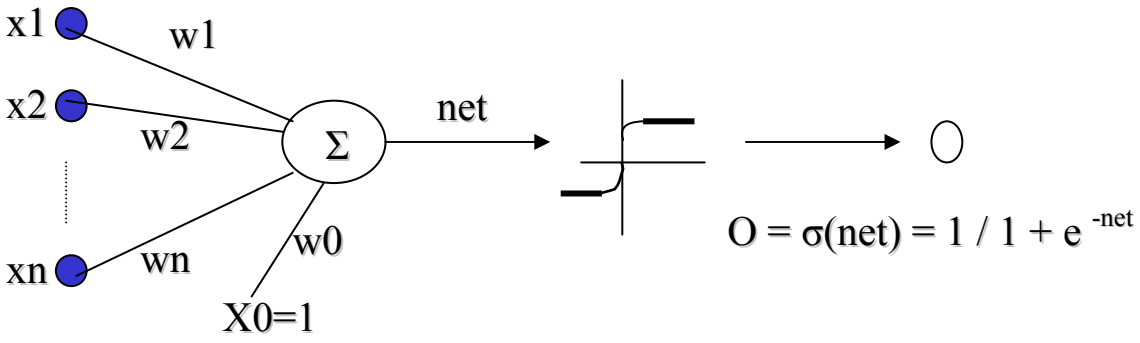
Version 1 CSE IIT, Kharagpur

12.4.4 Multi-Layer Perceptrons

In contrast to perceptrons, multilayer networks can learn not only multiple decision boundaries, but the boundaries may be nonlinear. The typical architecture of a multi-layer perceptron (MLP) is shown below.



To make nonlinear partitions on the space we need to define each unit as a nonlinear function (unlike the perceptron). One solution is to use the sigmoid unit. Another reason for using sigmoids are that they are continuous unlike linear thresholds and are thus differentiable at all points.



$$O(x1,x2,\ldots,xn) = \sigma (WX)$$

where: $\sigma (WX) = 1 / 1 + e^{-WX}$

Function σ is called the sigmoid or logistic function. It has the following property:

$$d\,\sigma(y) / dy = \sigma(y)\ (1 - \sigma(y))$$

12.4.4.1 Back-Propagation Algorithm

Multi-layered perceptrons can be trained using the back-propagation algorithm described next.

Goal: To learn the weights for all links in an interconnected multilayer network.

We begin by defining our measure of error:

$$E(W) = \tfrac{1}{2} \sum_d \sum_k (t_{kd} - o_{kd})^2$$

k varies along the output nodes and d over the training examples.

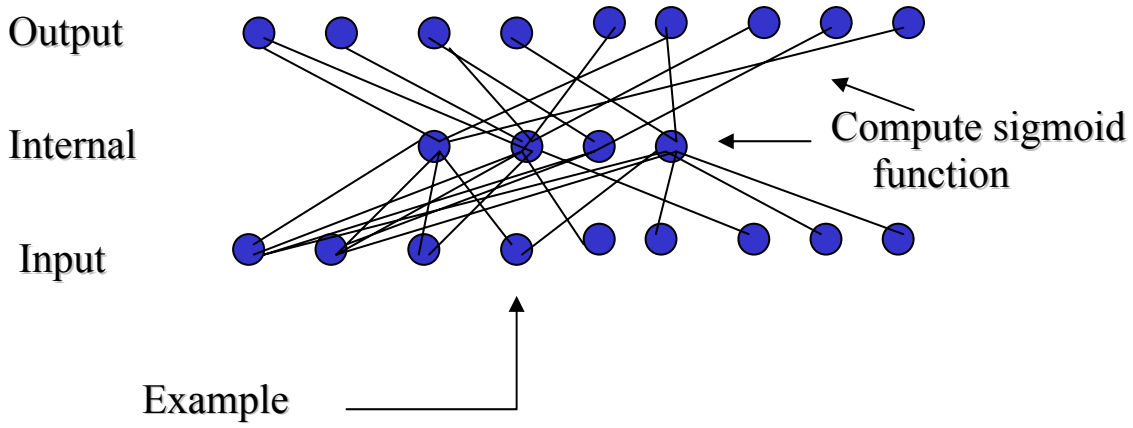
The idea is to use again a gradient descent over the space of weights to find a global minimum (no guarantee).

Algorithm:

1. Create a network with nin input nodes, nhidden internal nodes, and nout output nodes.
2. Initialize all weights to small random numbers.
3. Until error is small do:
   For each example X do
   - Propagate example X forward through the network
   - Propagate errors backward through the network

Forward Propagation

Given example X, compute the output of every node until we reach the output nodes:



Backward Propagation

A. For each output node k compute the error:
$\delta k = Ok\ (1-Ok)(tk - Ok)$

B. For each hidden unit h, calculate the error:
$\delta h = Oh\ (1-Oh)\ \Sigma k\ Wkh\ \delta k$

C. Update each network weight:

C. $Wji = Wji + \Delta Wji$
where $\Delta Wji = \eta\ \delta j\ Xji$ (Wji and Xji are the input and weight of node i to node j)

A momentum term, depending on the weight value at last iteration, may also be added to the update rule as follows. At iteration n we have the following:

$\Delta Wji\ (n) = \eta\ \delta j\ Xji + \alpha \Delta Wji\ (n)$

Where α ($0 <= \alpha <= 1$) is a constant called the momentum.

1. It increases the speed along a local minimum.
2. It increases the speed along flat regions.

Remarks on Back-propagation

1. It implements a gradient descent search over the weight space.
2. It may become trapped in local minima.
3. In practice, it is very effective.
4. How to avoid local minima?
    a) Add momentum.
    b) Use stochastic gradient descent.
    c) Use different networks with different initial values for the weights.

Multi-layered perceptrons have high representational power. They can represent the following:

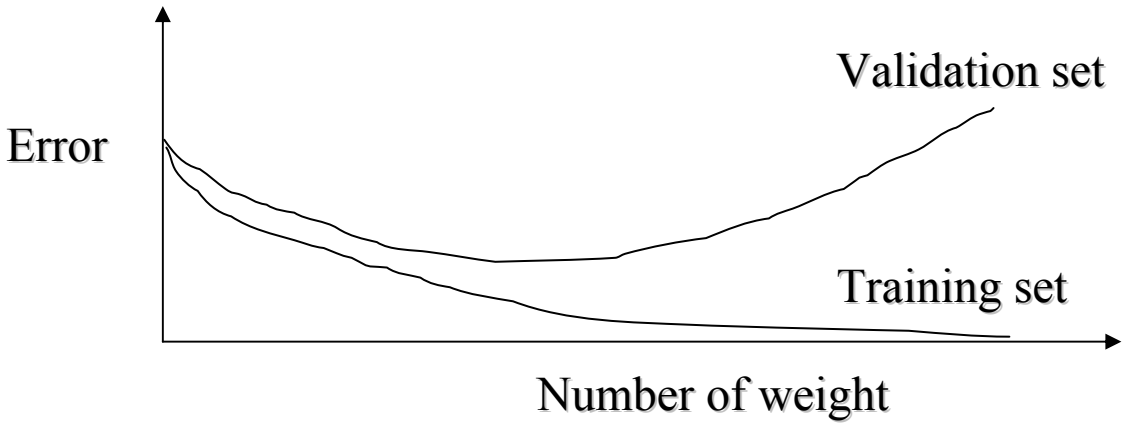
1. Boolean functions. Every boolean function can be represented with a network having two layers of units.

2. Continuous functions. All bounded continuous functions can also be approximated with a network having two layers of units.

3. Arbitrary functions. Any arbitrary function can be approximated with a network with three layers of units.

Generalization and overfitting:

One obvious stopping point for backpropagation is to continue iterating until the error is below some threshold; this can lead to overfitting.



Overfitting can be avoided using the following strategies.

- Use a validation set and stop until the error is small in this set.
- Use 10 fold cross validation.
- Use weight decay; the weights are decreased slowly on each iteration.

Applications of Neural Networks

Neural networks have broad applicability to real world business problems. They have already been successfully applied in many industries.

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management
- target marketing

Because of their adaptive and non-linear nature they have also been used in a number of control system application domains like,

- process control in chemical plants
- unmanned vehicles
- robotics
- consumer electronics

Neural networks are also used a number of other applications which are too hard to model using classical techniques. These include, computer vision, path planning and user modeling.

Questions

1. Assume a learning problem where each example is represented by four attributes. Each attribute can take two values in the set {a,b}. Run the candidate elimination algorithm on the following examples and indicate the resulting version space. What is the size of the space?

((a, b, b, a), +)
((b, b, b, b), +)
((a, b, b, b), +)
((a, b, a, a), -)

2. Decision Trees

The following table contains training examples that help a robot janitor predict whether or not an office contains a recycling bin.

	STATUS	DEPT.	OFFICE SIZE	RECYCLING BIN?
1.	faculty	ee	large	no
2.	staff	ee	small	no
3.	faculty	cs	medium	yes
4.	student	ee	large	yes
5.	staff	cs	medium	no
6.	faculty	cs	large	yes
7.	student	ee	small	yes
8.	staff	cs	medium	no

Use entropy splitting function to construct a minimal decision tree that predicts whether or not an office contains a recycling bin. Show each step of the computation.

3. Translate the above decision tree into a collection of decision rules.

4. Neural networks: Consider the following function of two variables:

A	B	Desired Output
0	0	1
1	0	0
0	1	0
1	1	1

Prove that this function cannot be learned by a single perceptron that uses the step function as its activation function.

4. Construct by hand a perceptron that can calculate the logic function implies (=>). Assume that 1 = true and 0 = false for all inputs and outputs.

Solution

1. The general and specific boundaries of the version space are as follows:

S: {(?,b,b,?)}
G: {(?,?,b,?)}

The size of the version space is 2.

2. The solution steps are as follows:

1. **Selecting the attribute for the root node:**

```
                                YES                    NO
```

Status:

```
        Faculty:    (3/8)*[ -(2/3)*log_2(2/3) - (1/3)*log_2(1/3)]
        Staff:    + (3/8)*[ -(0/3)*log_2(0/3) - (3/3)*log_2(3/3)]
        Student:  + (2/8)*[ -(2/2)*log_2(2/2) - (0/2)*log_2(0/2)]
        TOTAL:    = 0.35 + 0 + 0 = 0.35
```

Size:

```
        Large:      (3/8)*[ -(2/3)*log_2(2/3) - (1/3)*log_2(1/3)]
        Medium:   + (3/8)*[ -(1/3)*log_2(1/3) - (2/3)*log_2(2/3)]
        Small:    + (2/8)*[ -(1/2)*log_2(1/2) - (1/2)*log_2(1/2)]
        TOTAL:    = 0.35 + 0.35 + 2/8 = 0.95
```

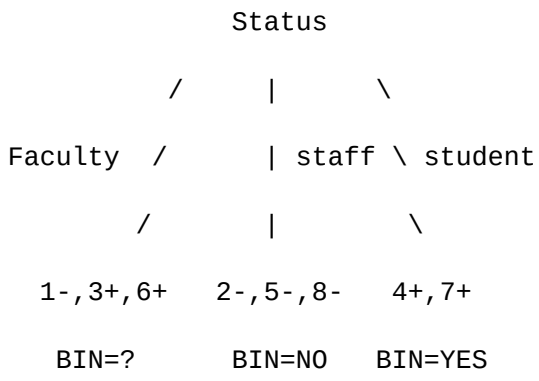
Dept:

ee: (4/8)*[-(2/4)*log_2(2/4) - (2/4)*log_2(2/4)]

cs: + (4/8)*[-(2/4)*log_2(2/4) - (2/4)*log_2(2/4)]

TOTAL: = 0.5 + 0.5 = 1

Since **status** is the attribute with the lowest entropy, it is selected as the root node:



Only the branch corresponding to Status=Faculty needs further processing.

Selecting the attribute to split the branch Status=Faculty:

YES NO

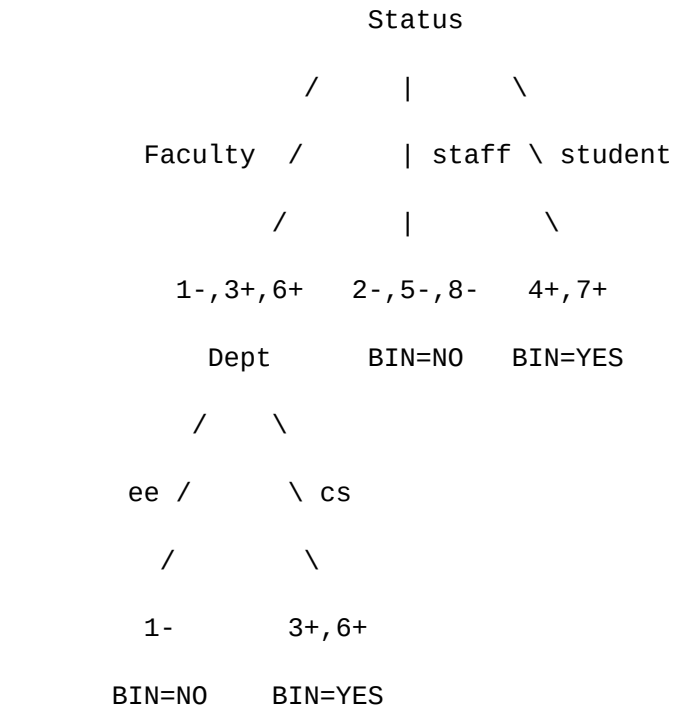
Dept:

ee: (1/3)*[-(0/1)*log_2(0/1) - (1/1)*log_2(1/1)]

cs: + (2/3)*[-(2/2)*log_2(2/2) - (0/2)*log_2(0/2)]

TOTAL: = 0 + 0 = 0

Since the minimum possible entropy is 0 and **dept** has that minimum possible entropy, it is selected as the best attribute to split the branch Status=Faculty. Note that we don't even have to calculate the entropy of the attribute **size** since it cannot possibly be better than 0.



Each branch ends in a homogeneous set of examples so the construction of the decision tree ends here.

3. The rules are:

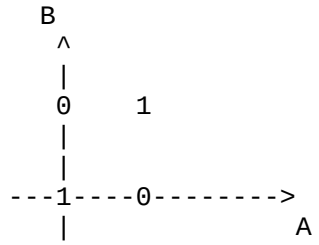
```
IF status=faculty AND dept=ee THEN recycling-bin=no

IF status=faculty AND dept=cs THEN recycling-bin=yes

IF status=staff THEN recycling-bin=no

IF status=student THEN recycling-bin=yes
```

4. This function cannot be computed using a single perceptron since the function is not linearly separable:



5. Let

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i} a_j\right), \quad a_0 = -1.$$

Choose g to be the threshold function (output 1 on input > 0, output 0 otherwise). There are two inputs to the implies function, $X_1 \Rightarrow X_2$. Therefore,

$$\sum_{j=0}^{n} W_{j,i} a_j = -W_0 + W_1 X_1 + W_2 X_2$$

Arbitrarily **choose** $W2 = 1$, and substitute into above for all values of X1 and X2.

X_1	X_2	$X_1 \Rightarrow X_2$	$-W_0 + W_1 X_1 + (1) X_2$		
0	0	1	$-W_0 > 0$	$\therefore$	$W_0 < 0$
0	1	1	$-W_0 + 1 > 0$	$\therefore$	$W_0 < 1$
1	0	0	$-W_0 + W_1 < 0$	$\therefore$	$W_0 > W_1$
1	1	1	$-W_0 + W_1 + 1 > 0$	$\therefore$	$W_0 - W_1 < 1$

Note that greater/less than signs are decided by the threshold fn. and the result of $X_1 \Rightarrow X_2$. For example, when $X_1=0$ and $X_2=0$, the resulting equation must be greater than 0 since g is threshold fn. and $X_1 \Rightarrow X_2$ is True. Similarly, when $X_1=1$ and $X_2=0$, the resulting equation must be less than 0 since g is the threshold fn. and $X_1 \Rightarrow X_2$ is False.

Since $W_0 < 0$, and $W_0 > W_1$, and $W_0 - W_1 < 1$, **choose** $W_1 = -1$. Substitute where possible.

X_1	X_2	$X_1 \Rightarrow X_2$	$-W_0 + (-1)X_1 + (1)X_2$
0	0	1	$W_0 < 0$
0	1	1	$W_0 < 1$
1	0	0	$W_0 > -1$
1	1	1	$W_0 - (-1) < 1 \quad \therefore \quad W_0 < 0$

Since $W_0 < 0$, and $W_0 > -1$, **choose** $W_0 = -0.5$.

Thus… $W_0 = -0.5$, $W_1 = -1$, $W_2 = 1$.

Finally,