IS-ZC444: ARTIFICIAL INTELLIGENCE

Lecture-14: Special Topics

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems,
BITS Pilani, Pilani, Jhunjhunu-333031, Rajasthan, INDIA

Nov 21, 2020 FLIPPED (WILP @ BITS-Pilani Jul-Nov 2020)

Markov Modal

- Andrev Markav: A canonical probabilistic model for temporal or sequential data. $X_0 \overset{A}{\rightarrow} X_1 \overset{A}{\rightarrow} ... \overset{A}{\rightarrow} X_n$
- Future is independent of past given the present. Assumption is that the present state encode all the history
- Order specifies how many evidences are important. Order three Markov Modal takes last three data
- iid[1] don't work.
- Temporal data, weather prediction, speech recognition, automatic music generation and handwriting recognition are some of the few applications

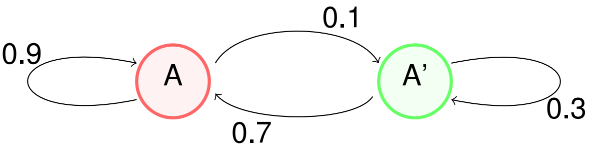
Example:

Suppose a company selling a product A (has market share of 20%), launches a advertise campaign that is expected to retain 90% old customers and attract 70% new. What maximum market share the product A can get?

[1] independent and identically distributed

Markov Modal

Transition diagram



Initial State

$$S_0 = \begin{bmatrix} 0.2 & 0.8 \end{bmatrix}$$

Transition matrix

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.7 & 0.3 \end{bmatrix}$$

- $S_0 = \begin{bmatrix} 0.2 & 0.8 \end{bmatrix}$
- $S_1 = S_0 \times A = \begin{bmatrix} 0.74 & 0.26 \end{bmatrix}$
- $S_2 = S_1 \times A = \begin{bmatrix} 0.848 & 0.152 \end{bmatrix}$
- $S_3 = S_2 \times A = \begin{bmatrix} 0.8696 & 0.1304 \end{bmatrix}$

Is it going to saturate?

Stationary matrix

$$\begin{bmatrix} a & b \end{bmatrix} \times A = \begin{bmatrix} a & b \end{bmatrix}$$

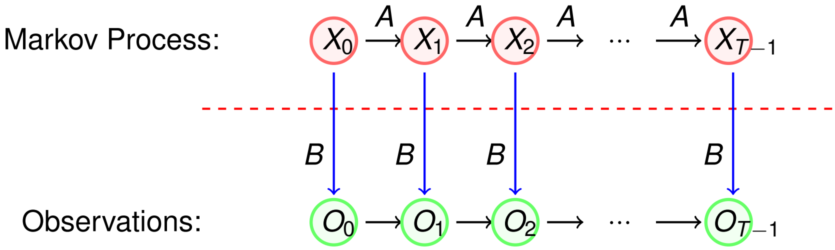
$$\begin{bmatrix} a & b \end{bmatrix} \times \begin{bmatrix} 0.9 & 0.1 \\ 0.7 & 0.3 \end{bmatrix} = \begin{bmatrix} a & b \end{bmatrix}$$

what are a and b? 0.875 and 0.125

- Does it always happen? No, only if matrix is **regular**
- When some power of the matrix has all positive values
- Which of these are regular?

$$\begin{bmatrix} 0.3 & 0.7 \\ 0.1 & 0.9 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0.2 & 0.8 \\ 1 & 0 \end{bmatrix}$$

Hidden Markov Modal (HMM)



Assume we observe news coverage (S/M/L) of some article, to know whether a day was Hot or Cold?

$$B = \begin{array}{cc} & \begin{array}{ccc} S & M & L \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} \end{array} \qquad A = \begin{array}{cc} & \begin{array}{cc} H & C \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \end{array}$$

Hidden Markov Modal (HMM)

$$B = \begin{array}{c} \\ H \\ C \end{array} \begin{array}{c} \begin{array}{ccc} S & M & L \end{array} \\ \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} \end{array} \qquad A = \begin{array}{c} \\ H \\ C \end{array} \begin{array}{c} \begin{array}{cc} H & C \end{array} \\ \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \end{array}$$

- Assume initial configuration for H and C be $\pi = \begin{bmatrix} 0.6 & 0.4 \end{bmatrix}$
- And let observations be S, M, S, L
- Then what is $P(HHCC)$?
  $0.6 \times 0.1 \times (0.7 \times 0.4) \times (0.3 \times 0.7) \times (0.6 \times 0.1) = 0.000212$

Hidden Markov Modal (HMM)

State	Probability	Normalized Probability
HHHH	0.000412	0.042787
HHHC	0.000035	0.003635
HHCH	0.000706	0.073320
HHCC	0.000212	0.022017
HCHH	0.000050	0.005193
HCHC	0.000004	0.000415
HCCH	0.000302	0.031364
HCCC	0.000091	0.009451
CHHH	0.001089	0.114031
CHHC	0.000094	0.009762
CHCH	0.001882	0.195451
CHCC	0.000562	0.058573
CCHH	0.000470	0.048811
CCHC	0.000040	0.004154
CCCH	0.002822	0.293073
CCCC	0.000847	0.087963

Optimum state sequence

- In dynamic programming is CCCH
- HMM choses most probable symbol at each position. (by summation)

	0	1	2	3
P(H)	0.188182	0.519576	0.228788	0.804029
P(C)	0.811818	0.480424	0.771212	0.195971

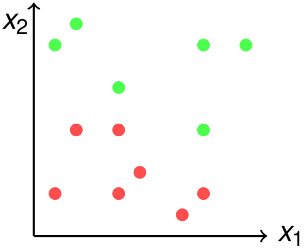
Optimum state sequence in HMM is ? CHCH

Linear Classification

Consider Following data

x_1	x_2	y
1	9	green
10	9	green
4	7	green
4	5	red
5	3	red
8	9	green
4	2	red
2	5	red
7	1	red
2	10	green
8	5	green
1	2	red
8	2	red

Data is in 2D, so let us visualize



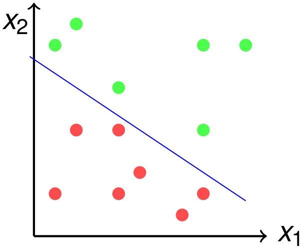
- Data looks linearly separable
- What is the decision boundary?

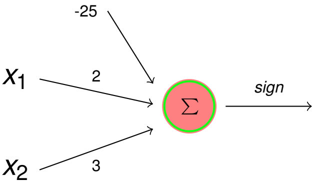
Many Possibilities, such as

if $(2x_1 + 3x_2 - 25 > 0)$ it is green
otherwise red

What about this arrangement?

With chosen *decision boundary* $2x_1 + 3x_2 - 25 = 0$





- This illustration is called as **perceptron**
- Provides a graphical way to represent the linear boundary
- Values 3, 2, -25 are its parameters or weights

Given a data

"How to find appropriate parameters?" is an important issue

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 1: Perceptron training rule

```
Begin with random weights w
repeat
    for each misclassified example do
        w_i = w_i + η(t − o)x_i
until all training examples are correctly classified;
return w
```

- **Why would this strategy converge?**
  1. Weight does not change when classification is correct
  2. If perceptron outputs -1 when target is +1: weight increases $\uparrow$
  3. If perceptron outputs +1 when target is -1: weight decreases $\downarrow$

Conversion with perceptron training rule is subject to linear separability of training example and appropriate η

Example

Consider the same data

x_1	x_2	y
1	9	green
10	9	green
4	7	green
4	5	red
5	3	red
8	9	green
4	2	red
2	5	red
7	1	red
2	10	green
8	5	green
1	2	red
8	2	red

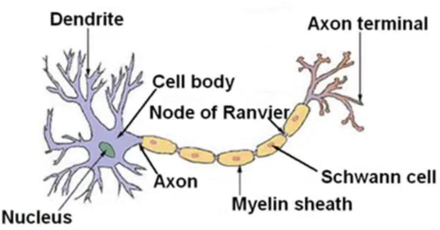
$$\eta = 0.01$$

w0=0.500, w1=0.500, w2=0.500	err=7
w0=0.360, w1=-0.120, w2=0.100	err=6
w0=0.300, w1=-0.180, w2=0.060	err=5
w0=0.240, w1=-0.140, w2=0.140	err=4
w0=0.180, w1=-0.200, w2=0.100	err=5
w0=0.120, w1=-0.160, w2=0.180	err=4
w0=0.080, w1=-0.060, w2=0.180	err=5
w0=0.020, w1=-0.120, w2=0.140	err=4
w0=-0.040, w1=-0.180, w2=0.100	err=5
w0=-0.100, w1=-0.140, w2=0.180	err=4
w0=-0.140, w1=-0.040, w2=0.180	err=5
w0=-0.200, w1=-0.100, w2=0.140	err=3
w0=-0.260, w1=-0.160, w2=0.100	err=4
w0=-0.320, w1=-0.120, w2=0.180	err=3
w0=-0.360, w1=-0.020, w2=0.180	err=3
w0=-0.420, w1=-0.080, w2=0.140	err=2
w0=-0.420, w1=-0.080, w2=0.240	err=2
Fourteen more iterations	
w0=-0.900, w1=-0.020, w2=0.180	err=1
w0=-0.900, w1=-0.020, w2=0.240	err=2
w0=-0.920, w1=0.020, w2=0.220	err=2
w0=-0.960, w1=-0.020, w2=0.220	err=3
w0=-0.980, w1=0.020, w2=0.200	err=2
w0=-1.000, w1=0.060, w2=0.180	err=2
w0=-1.040, w1=0.020, w2=0.180	err=0

Neural Network (NN)

NN is biologically motivated learning model that mimic human brain

- Started by *W. McCulloch* study on working of neurons in 1943
- MADALINE (1959), an adaptive filter that eliminates echoes on phone lines was the first neural network
- Popularity of Neural Network diminished in 90's but, due to advances in **processing power** and availability of **large data** it again became state-of-the-art



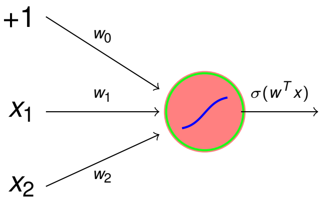
- Cell, Axon, Synopses, Molicules, and Dendrites
- Humans have 10^{11} neurons, each connected to 10^4 others, switches in 10^{-3} sec

An Example

Design a **perceptron** for

x_1	x_2	Classification
0	0	0
0	1	0
1	0	1
1	1	0

Let us assume following



We have following four equations

$$w_0 + w_1 \times (0) + w_2 \times (0) < 0 \quad (1)$$
$$w_0 + w_1 \times (0) + w_2 \times (1) < 0 \quad (2)$$
$$w_0 + w_1 \times (1) + w_2 \times (0) \geq 0 \quad (3)$$
$$w_0 + w_1 \times (1) + w_2 \times (1) < 0 \quad (4)$$

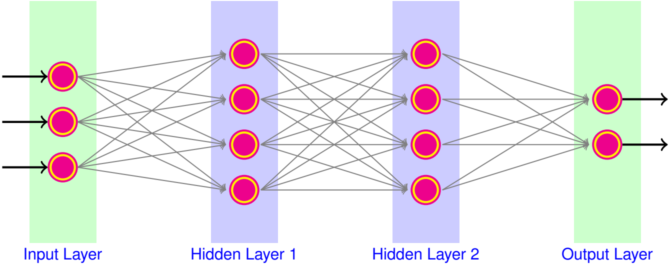
By (1) $w_0 < 0$ so let $w_0 = -1$
By (2) $w_0+w_2<0$ so let $w_2 = -1$
By (3) $w_0+w_1 \geq 0$ so let $w_1 = 1.5$
By (4) $w_0+w_1+w_2<0$ that is valid

$\boxed{\text{So } (w_0, w_1, w_2) = (-1, -1, 1.5)}$

Other possibilities are also there

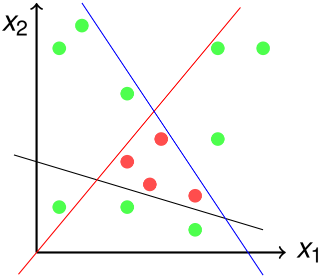
Neural Network

When neurons are interconnected in layers



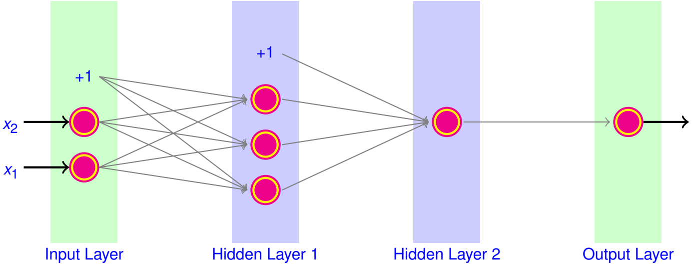
- Number of layers may differ
- Nodes in each intermediate layers may also differ
- Multiple output neurons are used for different class
- **Two levels deep** NN can represent any boolean function

More Example: Design NN for the following data



Whether it is **green**?

Red-line	Blue-line	Black-line	Color
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Neural Network Applications

NN is appropriate for problems with the following characteristics:

- Instances are provided by many attribute-value pairs (more data)
- The target function output may be discrete-valued, real-valued, or a vector of several real or discrete valued attributes
- The training examples may contain errors
- Long training times are acceptable
- Fast evaluation of the target function may be required
- The ability of humans to understand the learned target function is not important

Perceptron Training (delta rule)

When data is not linearly-separable, error fluctuates with parameter update so, it becomes difficult to decide when to stop

- Delta rule converges to a best-fit approximation of the target
- Uses gradient descent
- Consider unthresholded perceptron, $o(\vec{x}) = \vec{w}.\vec{x}$
- Training error is defined as

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

- Gradient would specify direction of steepest increase $\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$
- Weights can be learned as $w_i = w_i - \eta\frac{\partial E}{\partial w_i}$
- It can be seen that $\frac{\partial E}{\partial w_i} = \sum_{d\in D}(t_d - o_d)(-x_{id})$

Perceptron Training (delta rule)

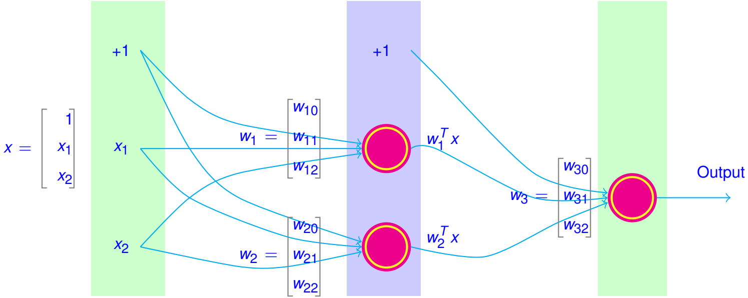
Algorithm 2: Gradient Descent (D,η)

Initialize w_i with random weights
repeat
  For each w_i, initialize $\triangle w_i = 0$
  for *each training example* $d \in D$ **do**
    Compute output o using model for d whose target is t
    For each w_i, update $\triangle w_i = \triangle w_i + \eta(t - o)x_i$
  For each w_i, set $w_i = w_i + \triangle w_i$
until *termination condition is met*;
return w

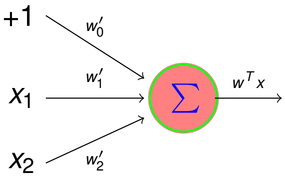
- A date item $d \in D$, is supposed to be multidimensional $d = (x_1, x_2, ..., x_n, t)$
- Algorithm converges toward the minimum error hypothesis.
- Linear programming can also be an approach

Linear Activation is Not Much Interesting

NN with perceptrons have limited capability, even with many layers



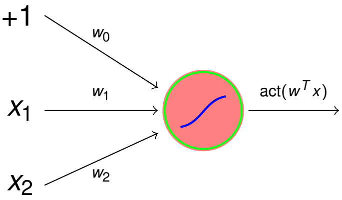
$$
\begin{aligned}
Output &= w_{30} \times 1 + w_{31} \times (w_1^T x) + w_{32} \times (w_2^T x) \\
&= w_{30} \times 1 + w_{31} \times [w_{10} \times 1 + w_{11} \times x_1 + w_{12} \times x_2] \\
&\quad + w_{32} \times [w_{20} \times 1 + w_{21} \times x_1 + w_{22} \times x_2] \\
&= (w_{30} + w_{31} w_{10} + w_{32} w_{20}) + (w_{31} w_{11} + w_{32} w_{21}) \times x_1 \\
&\quad + (w_{31} w_{12} + w_{32} w_{22}) \times x_2 \\
&= w'_0 + w'_1 \times x_1 + w'_2 \times x_2
\end{aligned}
$$

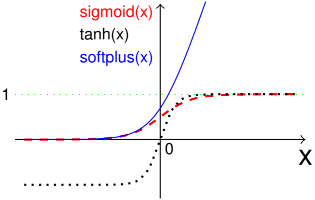


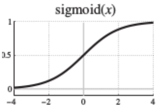
Expression of single perceptron

Neuron

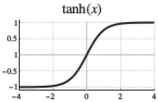
Neuron uses nonlinear activation functions (*sigmoid*, *tanh*, *ReLU*, *softplus etc*.) at the place of thresholding



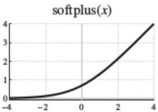




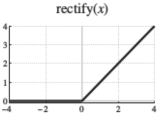
$$h(x) = \frac{1}{1 + \exp(-x)}$$



$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



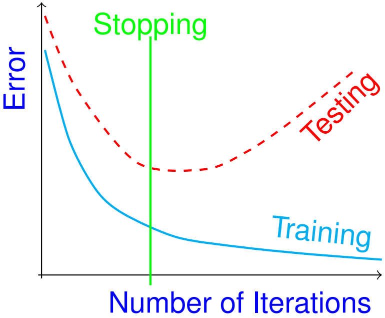
$$h(x) = \log(1 + \exp(x))$$



$$h(x) = \max(0, x)$$

Generalization, Overfitting, and Stopping Criterion

Continue training until the error on the training examples falls below some predetermined threshold could be a poor strategy



- Weight decay or use of validation set (k-fold ?) is suggested
- Input or output encoding can be used

Special methods and requirements

- Fuzzy membership
- Ant colony optimization
- Particle swarm optimization
- Fairness
- Interpretability

Thank You!

Thank you very much for your attention!

Queries ? Ref