NEURAL NETWORKS 101

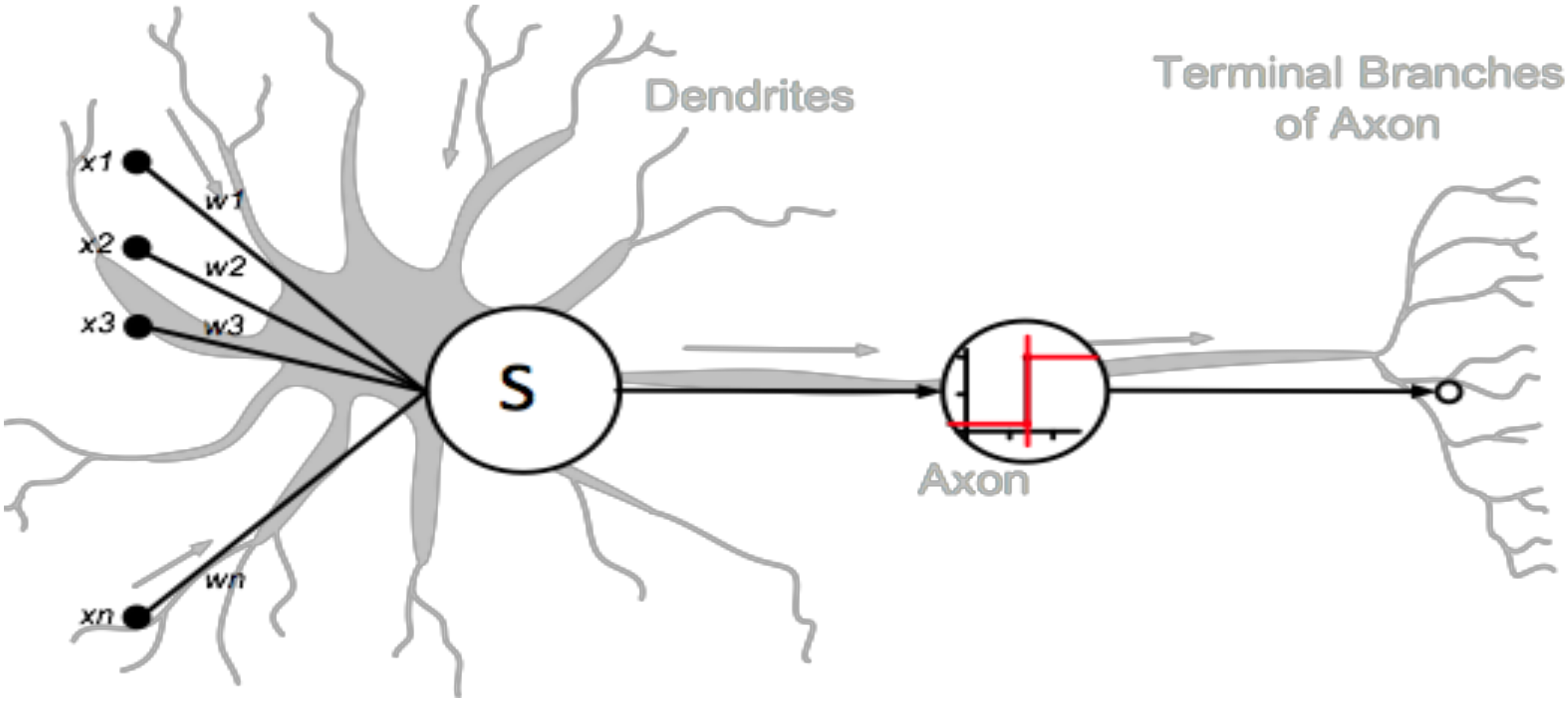
Varadharajan Mukundan

ThoughtWorks

Artificial Neural Networks (ANN)

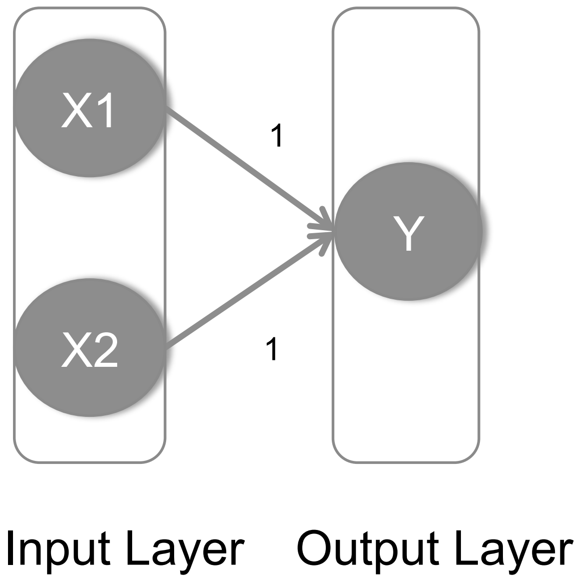
- Network of neurons (fundamental computational unit)
- Biologically inspired, massively parallel distributed computational model
- Generic, flexible models
  - Requires less / no feature engineering
  - Ability to generate feature representation (Auto Encoders)
- Applications
  - Computer vision
  - Function approximation (time series analysis)
  - Collaborative filtering / Recommender Systems (Netflix Prize)
  - Robotics
  - General BI Problems
  - NLP

Neuron



Slide Credit : Andrew L. Nelson

AND Gate (Perceptron)

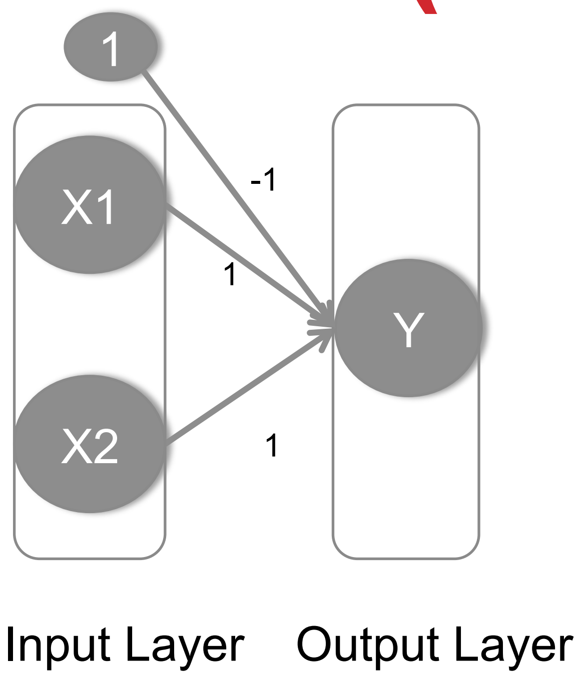


X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Net = \sum_i w_i x_i$$

$$y = f(Net) = \begin{cases} 1 & if\ Net > 2 \\ 0 & otherwise \end{cases}$$

AND Gate (Perceptron)

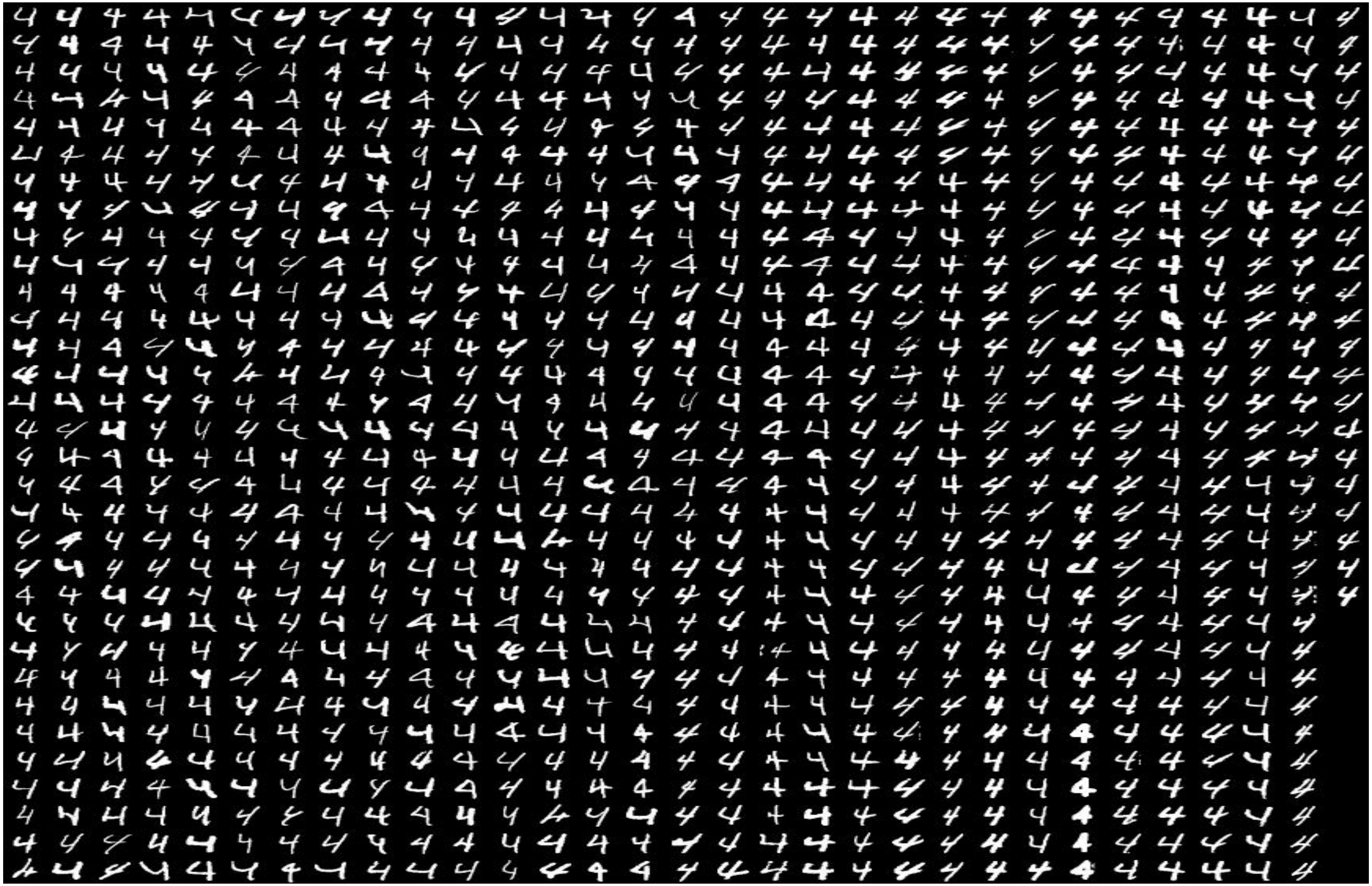


X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Net = \sum_i w_i x_i$$

$$y = f(Net) = \begin{cases} 1 & if\ Net > 0 \\ 0 & otherwise \end{cases}$$

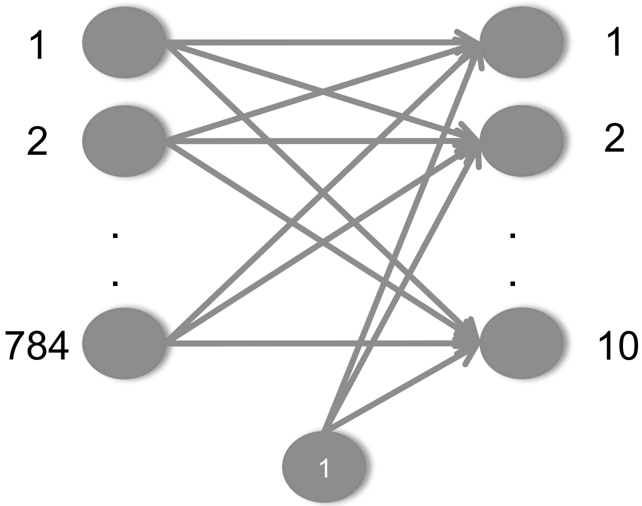
Digit Recognizer



Digit Recognizer

- Two implementations
  - Single layer Perceptron with step function
  - Single layer Perceptron with sigmoidal function
- Image scaled to 28 x 28 square matrix with values between 0 and 1
  - 0 represents black
  - 1 represents white
- Both the networks trained for 10 epochs

Digit Recognizer - I



$$\Delta w = -\frac{dE}{dw} = (t_n - y_n)x_n$$

$$w = w + \alpha \Delta w$$

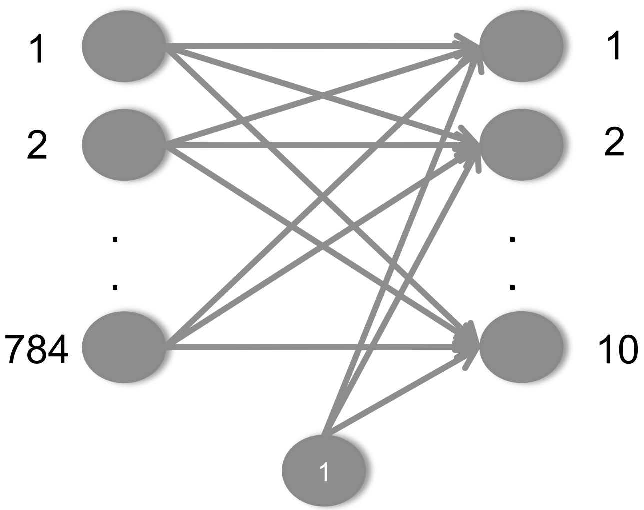
- Single Layer step Perceptron
- W = 785 X 10 Matrix
- Objective : Find optimal weight to minimize error
- X – Input vector, y – Output according to perceptron, t – original output

$$Net = \sum_i w_i x_i$$

$$y = f(Net) = \begin{cases} 1 & if\ Net > 0 \\ 0 & otherwise \end{cases}$$

$$E = \frac{1}{2}\sum_n (t_n - y_n)^2$$

Digit Recognizer - II



- Single Layer step Perceptron
- W = 785 X 10 Matrix
- Objective : Find optimal weight to minimize error
- X – Input vector, y – Output according to perceptron, t – original output

$$Net = \sum_i w_i x_i$$

$$y = f(Net) = \frac{1}{1+e^{-Net}}$$

$$E = \frac{1}{2}\sum_n (t_n - y_n)^2$$

$$\frac{dy}{dNet} = -\left(\frac{1}{\left(1+e^{-Net}\right)^{-2}} \cdot -e^{-Net}\right) = y(1-y)$$

$$\Delta w = -\frac{dE}{dw} = \frac{dE}{dy} \cdot \frac{dy}{dNet} \cdot \frac{\partial Net}{\partial w} = (t_n - y_n)(y_n)(1-y_n)x_n$$

$$w = w + \alpha \Delta w$$

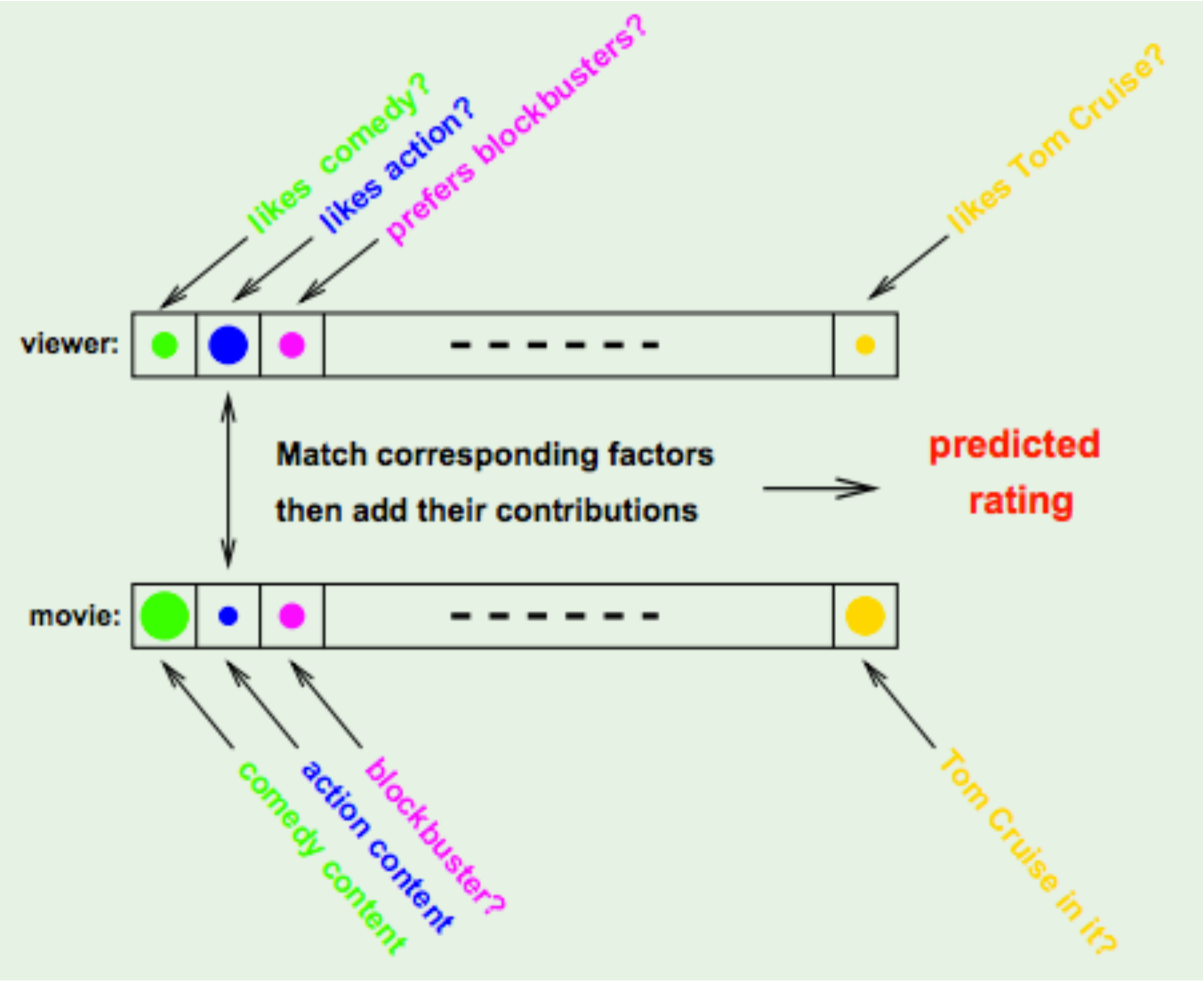
Digit Recognizer – I & II

- Initialize w = rand(795, 10)
- Repeat for n epochs
  - For each data point in training set
    - <x0,x1,x2…,x784> = <1,……> ; t = <0,0,0,0,1,0,0,0,0,0>
    - Apply activation function and get 'y'
    - Apply weight learning rule and update weights accordingly
- Error percentage for 1'st implementation after 10 epochs ~= 20%
- Error percentage for 2'nd implementation after 10 epochs ~= 8%

Collaborative Filtering

- Recommend a set of movies based on 'User Rating x Movie' Matrix.
- Uncovers latent / hidden feature vectors
- Reference Algorithms
  - LDA
  - Mixture Models / neighborhood Models
  - Matrix Factorizations

Collaborative Filtering



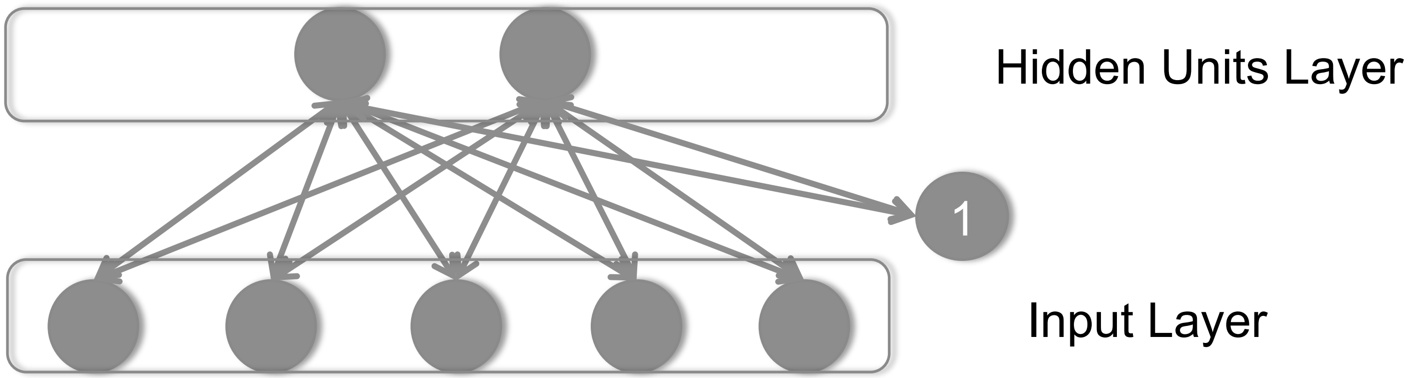
Slide Credit : Learning from Data course

Collaborative Filtering

- Scenario
  - 7 Movies, 10 Users preferences
  - Preference in binary format
    - 1 : Like
    - 0 : No Information / Not prefer
- Algorithm : **Restricted Boltzmann Machine (RBM)**
- Limited synthetic data might the performance of RBM
  - But good enough to illustrate the algorithm

Restricted Boltzmann Machine (RBM)

- Stochastic, probabilistic, binary, two layer Neural Network
- Hidden units represents Latent Features
    - First hidden unit might represent / cluster action movies
    - Second hidden unit might represent / cluster oscar movies
- Contrastive Divergence (CD) is used for training RBMs



Contrastive Divergence

- Two phase training for each data point
  - Phase 1 : Reality Phase
  - Phase 2 : Dreaming phase

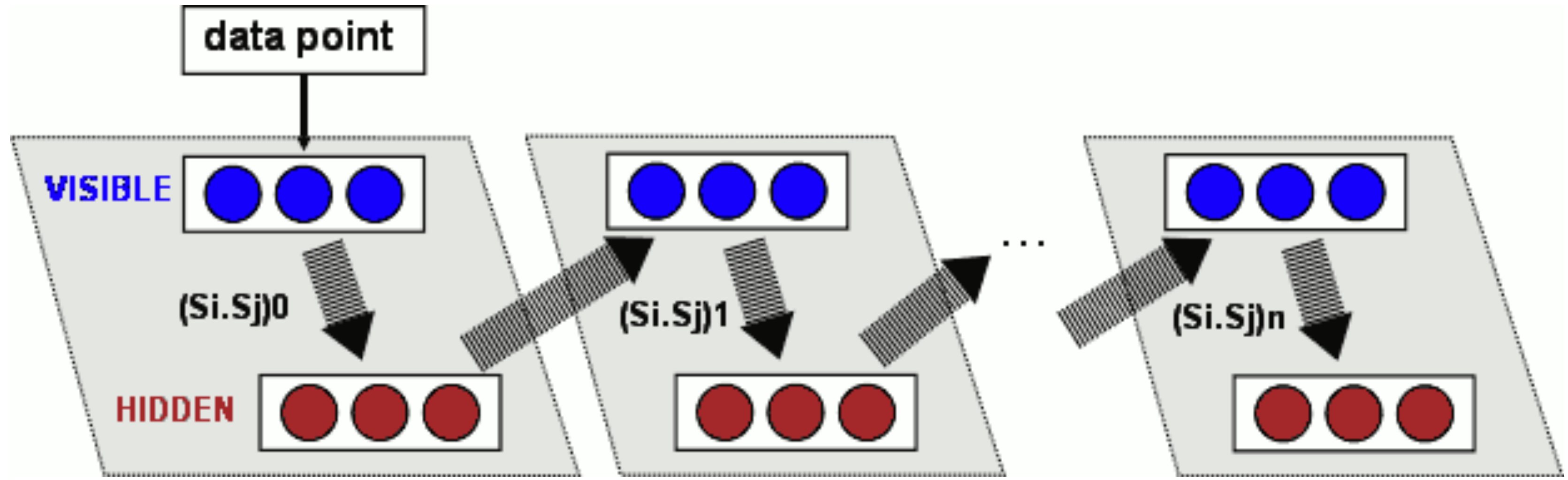


Image Credit : imonad.com

Reality Phase

$$Net = \sum_i w_i x_i$$

$$S = f(Net) = \frac{1}{1+e^{-Net}} > \phi$$

$$P(n_{1,} n_2) = S_{n_1} S_{n_2}$$

- Set up the input units to user preferences
- Calculate Net
- Apply sigmoidal threshold function to get the state of hidden units
- Calculate P(n1,n2) being a |hidden units| X |visible units+1| matrix

Dreaming Phase

$$Net = \sum_i w_i x_i$$

$$S = f(Net) = \frac{1}{1+e^{-Net}} > \phi$$

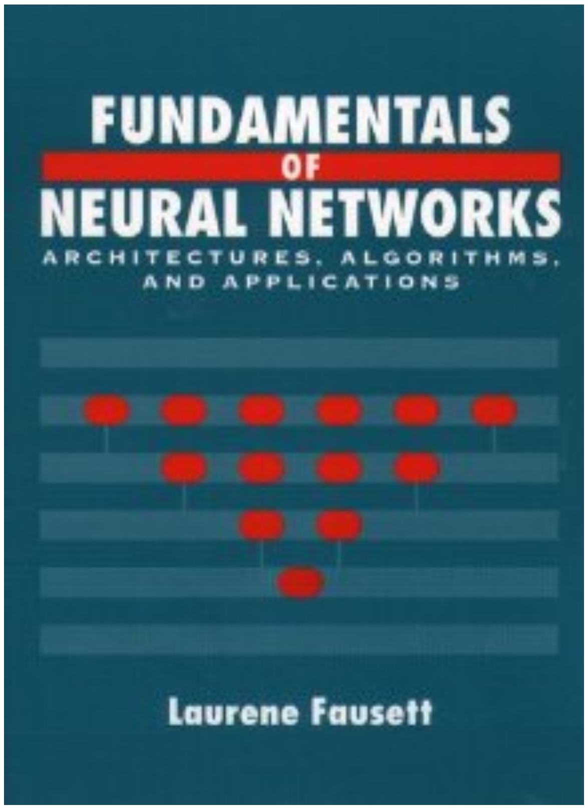
$$N(n_{1,}n_2) = S_{n_1} S_{n_2}$$

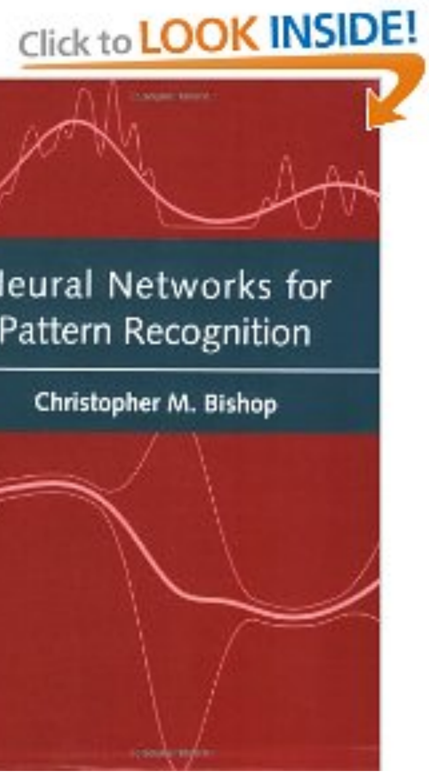
$$\Delta w = P(n_1, n_2) - N(n_1, n_2)$$

$$w = w + \alpha \Delta w$$

- Similar to reality phase, but consider the states from reality phase and determine the states of visible units
- Calculate N(n1,n2) similar to P(n1,n2)
- Update weights based on the weight update rule
- Repeat both the phase for each data point for n epochs

References


FUNDAMENTALS
OF
NEURAL NETWORKS
ARCHITECTURES, ALGORITHMS, AND APPLICATIONS
Laurene Fausett


Click to LOOK INSIDE!
Neural Networks for Pattern Recognition
Christopher M. Bishop

That's All Folks

- Code @ github.com/varadharajan/geek-night