CS F425: Deep Learning

26

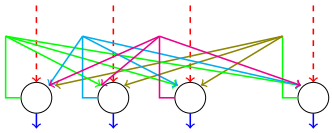
Boltzmann Machines
RBM, Belief Network

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

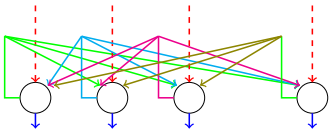
Mar 28, 2023 ON-CAMPUS Campus @ BITS-Pilani [Jan-May 2023]

http://ktiwari.in/dl

Energy Based Models: Hopfield Nets (1982)



Energy Based Models: Hopfield Nets (1982)

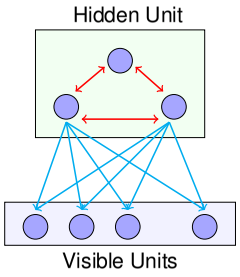


- Memory view of NN: a content-addressable (associative) one
- Feed-forward energy based model with recurrent connection

$$E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$$

- Nodes are binary threshold and weights are symmetric
- Low energy is good. Guarantee to converge on local minimum
- Energy gap $\triangle E_i = E(s_i = 0) - E(s_i = 1) = b_i + \sum_j s_j w_{ij}$

Hopfield Nets with hidden units



- States of hidden units can represent interpretations
- With lower energy one can get interpretation better. Weights represents constraints on interpretation.
- Hidden units may consider the data in higher dimensional space
- Temperature (noise) can help to escape local minima (**simulated annealing**)

Boltzmann Machine

- Energy based model defining joint probabilities of visible and hidden units

$$p(v, h) \propto e^{-E(v,h)}$$

- Probability of finding the network in that joint configuration after thermal equilibrium

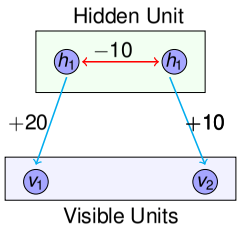
$$-E(v, h) = \sum_i v_i b_i + \sum_k h_k b_k + \sum_{i<j} v_i v_j w_{ij} + \sum_{i,k} v_i h_k w_{ik} + \sum_{k<l} h_k h_l w_{kl}$$

$$p(v, h) = \frac{e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

- We know

$$p(v) = \sum_h p(v, h)$$

Boltzmann Machine as a probabilistic model



v	h	$-E$	e^{-E}	p(v,h)	p(v)
0 0	0 0	0	1	.025	0.084
0 0	0 1	0	1	.025	
0 0	1 0	0	1	.025	
0 0	1 1	-1	0.37	.009	
0 1	0 0	0	1	.025	0.144
0 1	0 1	1	2.72	.069	
0 1	1 0	0	1	.025	
0 1	1 1	0	1	.025	
1 0	0 0	0	1	.025	0.305
1 0	0 1	0	1	.025	
1 0	1 0	2	7.39	.186	
1 0	1 1	1	2.72	.069	
1 1	0 0	0	1	.025	0.466
1 1	0 1	1	2.72	.069	
1 1	1 0	2	7.39	.186	
1 1	1 1	2	7.39	.186	
			39.70		

Challenges

- If number of units is large we cannot compute the normalization term
- So one can use Markov Chain Monte Carlo to get samples from the model starting from some random global initialization
- Keep picking units at random and allow then to stochastically update their states based on their energy gaps
- Run the Markov Chain until it reaches its stationary distribution (thermal equilibrium)
- It can be used as a probabilistic model of the data

Learning in Boltzmann Machine

- Unsupervised. Just give input vector
- Algorithm build a model of set of input
- Maximize the product of the probabilities for vectors in training set.
- That is maximizing sum of log probabilities of training vectors.

$$\frac{\partial \log p(v)}{\partial w_{ij}} = < s_i s_j >_v - < s_i s_j >_{model}$$

- Derivative of the log probability of a visible vector is the simple difference of correlations, we can make change in weight to be proportional to the

$$\triangle w_{ij} = < s_i s_j >_{data} - < s_i s_j >_{model}$$

Learning in Boltzmann Machine

$$-\frac{\partial E}{\partial w_{ij}} = s_i s_j$$

- Two stages
  1. Settle with data
  2. and Settle with no data

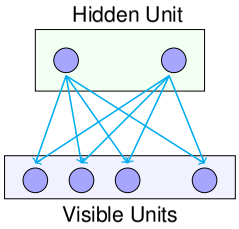
  Recall

$$p(v) = \frac{\sum_h e^{-E(v,h)}}{\sum_u \sum_g e^{-E(u,g)}}$$

- what two stages for
  1. +ve phase: Find hidden configuration that works well with v
  2. −ve phase: find best joint configuration raising the energy
- Need efficient way to collect +ve and −ve statistics

Restricted Boltzmann Machine (RBM)

- For making inference and learning easier, connectivity is restricted
- Single hidden layer; and NO connection in same layer
- Takes only one step to reach thermal equilibrium



$$p(h_j = 1) = \frac{1}{1 + e^{-(b_j + \sum_i v_i w_{ij})}}$$

- RBM can be used for classification
- It handle noisy-data, data imbalance, missing values, and unstructured- data

Belief Networks (Deep)

- Belif Net is a directed graph composed of stochastic variables
- It is a generative graphical model can be made by stacking RBM
- Backpropagation is slow in multiple hidden layers and gets stuck to local minima
- We don't initialize weights in a sensible way. Most of the data was used to be unlabeled
- It handle noisy-data, data imbalance, missing values, and unstructured- data
- Learning objective is $p(data)$ not $p(label|data)$

Constructive divergence using Gibbs Sampling

1. Set the status of visible units to the training dataset
2. $+ve$ Phase: Update hidden layers $Positive(E_{ij})$ is

$$p(h_j = 1|v) = \sigma(b_j + \sum_i w_{ij} v_i)$$

3. $-ve$ Phase: Update visible layers $Negative(E_{ij})$ is

$$p(v_i = 1|h) = \sigma(b_i + \sum_j w_{ij} h_j)$$

4. Update weights: $w_{ij} = w_{ij} + \alpha \times (Positive(E_{ij}) - Negative(E_{ij}))$

Repeat until convergence

Thank You!

Thank you very much for your attention!