Boltzmann Machine

A Brief Introduction

Ritajit Majumdar
Arunabha Saha

University of Calcutta

November 6, 2013

1. Hopfield Net
2. Stochastic Hopfield Nets with Hidden Units
3. Boltzmann Machine
4. Learning Algorithm for Boltzmann Machine
5. Applications of Boltzmann Machine
6. Restricted Boltzmann Machine
7. Reference

Hopfield Network

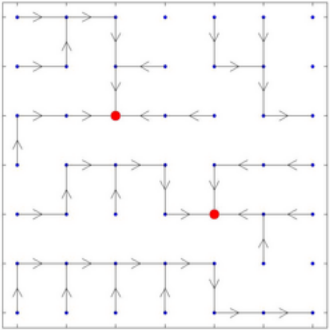


Figure: Two dimensional representation of motion in state space

Hopfield Net

A **Hopfield Net** is composed of binary threshold units with recurrent connections between them.

Recurrent networks of non-linear units are hard to analyze, since they can behave in many different ways -

1. Settle to a stable state.
2. Oscillate
3. Follow chaotic trajectory.

Hopfield Net

A **Hopfield Net** is composed of binary threshold units with recurrent connections between them.

Recurrent networks of non-linear units are hard to analyze, since they can behave in many different ways -

1. Settle to a stable state.
2. Oscillate
3. Follow chaotic trajectory.

John Hopfield introduced a **global energy function** for network with **symmetric** connections.

Hopfield Net

A **Hopfield Net** is composed of binary threshold units with recurrent connections between them.

Recurrent networks of non-linear units are hard to analyze, since they can behave in many different ways -

1. Settle to a stable state.
2. Oscillate
3. Follow chaotic trajectory.

John Hopfield introduced a **global energy function** for network with **symmetric** connections.

1. Each binary configuration of the whole network has an Energy.

Hopfield Net

A **Hopfield Net** is composed of binary threshold units with recurrent connections between them.

Recurrent networks of non-linear units are hard to analyze, since they can behave in many different ways -

1. Settle to a stable state.
2. Oscillate
3. Follow chaotic trajectory.

John Hopfield introduced a **global energy function** for network with **symmetric** connections.

1. Each binary configuration of the whole network has an Energy.
2. The binary threshold decision rule causes the network to settle to a minimum of this energy function.

Energy Function

The global energy is defined as -

$$E = -\sum_{i} s_i b_i - \sum_{i<j} s_i s_j w_{ij} \tag{1}$$

where b_i is the bias of the i^{th} unit, s is 0 or 1 [1] depending on whether the unit is turned off or on respectively. And w_{ij} is the weight of the connection between units i and j.

[1] For Bipolar Inputs the states will be -1 and 1 respectively.

Energy Function

The global energy is defined as -

$$E = -\sum_{i} s_i b_i - \sum_{i<j} s_i s_j w_{ij} \tag{1}$$

where b_i is the bias of the i^{th} unit, s is 0 or 1 [1] depending on whether the unit is turned off or on respectively. And w_{ij} is the weight of the connection between units i and j.

From this energy function, each unit computes locally how changing their state will affect the global energy.

[1] For Bipolar Inputs the states will be -1 and 1 respectively.

Energy Function

The global energy is defined as -

$$E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij} \tag{1}$$

where b_i is the bias of the i^{th} unit, s is 0 or 1 [1] depending on whether the unit is turned off or on respectively. And w_{ij} is the weight of the connection between units i and j.

From this energy function, each unit computes locally how changing their state will affect the global energy.

The **energy gap** is defined as -

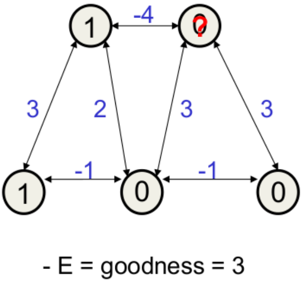
$$\triangle E_i = E(s_i = 0) - E(s_i = 1) = b_i + \sum_j s_j w_{ij} \tag{2}$$

[1] For Bipolar Inputs the states will be -1 and 1 respectively.

Settling to an Energy Minima

The net is initially in a random state i.e., the units are on or off randomly. The binary threshold decision rule updates units **one at a time** in a random order.

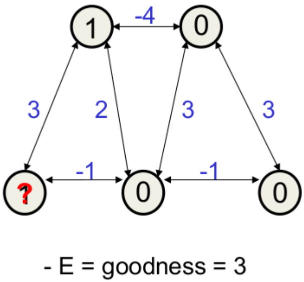
- Update each unit to whichever of its two states minimizes the global energy.
- Use binary threshold units, i.e., the states can be either 0 or 1.



Settling to an Energy Minima

The net is initially in a random state i.e., the units are on or off randomly. The binary threshold decision rule updates units **one at a time** in a random order.

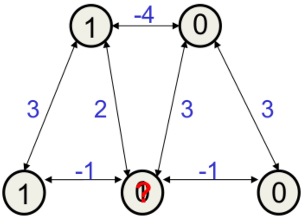
- Update each unit to whichever of its two states minimizes the global energy.
- Use binary threshold units, i.e., the states can be either 0 or 1.



Settling to an Energy Minima

The net is initially in a random state i.e., the units are on or off randomly. The binary threshold decision rule updates units **one at a time** in a random order.

- Update each unit to whichever of its two states minimizes the global energy.
- Use binary threshold units, i.e., the states can be either 0 or 1.



-E = goodness = 4

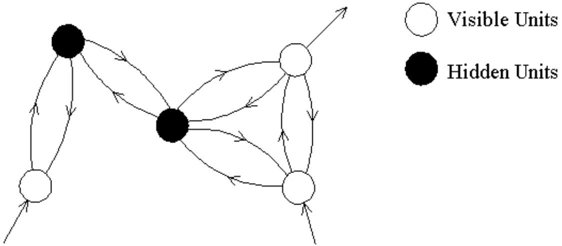
Using this type of Computation as memory

- Hopfield proposed that memories could be energy minima of a neural net.
  - The binary threshold decision rule can be used to "clean up" incomplete or corrupted memory.

Using this type of Computation as memory

- Hopfield proposed that memories could be energy minima of a neural net.
  - The binary threshold decision rule can be used to “clean up” incomplete or corrupted memory.
- Using energy minima to represent memories gives a content-addressable memory.
  - An item can be accessed just by knowing parts of it.
  - It is robust against hardware damage.

Stochastic Hopfield Nets with Hidden Units



$$p_i = p(\Delta E_i) = \frac{1}{1 + e^{-\Delta E_i / T}}$$

Hopfield Net with Hidden Units

Why add hidden units?

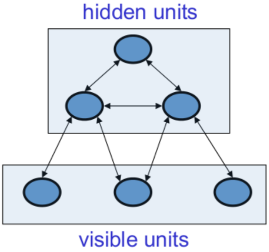
In Hopfield Net, there is no hidden layer of units. By adding hidden layers, the attention can be shifted from just storing memories to various types of interpretations of the inputs.

Hopfield Net with Hidden Units

Why add hidden units?

In Hopfield Net, there is no hidden layer of units. By adding hidden layers, the attention can be shifted from just storing memories to various types of interpretations of the inputs.

- Use the net to construct interpretations of sensory input.

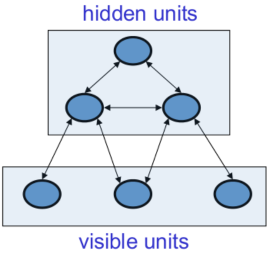


Hopfield Net with Hidden Units

Why add hidden units?

In Hopfield Net, there is no hidden layer of units. By adding hidden layers, the attention can be shifted from just storing memories to various types of interpretations of the inputs.

- Use the net to construct interpretations of sensory input.
- The input is represented by the visible units.

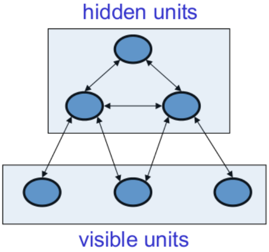


Hopfield Net with Hidden Units

Why add hidden units?

In Hopfield Net, there is no hidden layer of units. By adding hidden layers, the attention can be shifted from just storing memories to various types of interpretations of the inputs.

- Use the net to construct interpretations of sensory input.
- The input is represented by the visible units.
- The interpretation is represented by the states of the hidden units.

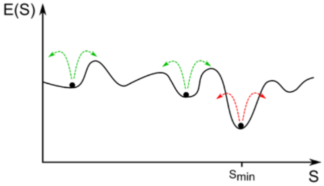


Noisy Networks

- The Binary Threshold Decision rule always goes downhill, i.e., reduces energy.

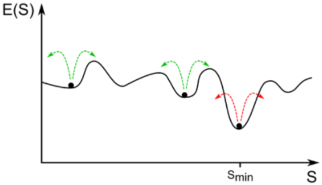
Noisy Networks

- The Binary Threshold Decision rule always goes downhill, i.e., reduces energy.
- Hence it is impossible to escape from a local minima.



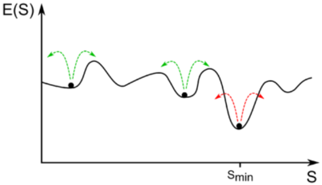
Noisy Networks

- The Binary Threshold Decision rule always goes downhill, i.e., reduces energy.
- Hence it is impossible to escape from a local minima.
- **Solution** - Use random noise to escape from poor, shallow minima.



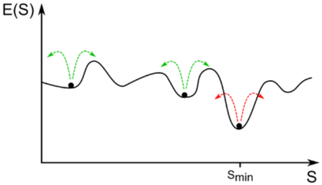
Noisy Networks

- The Binary Threshold Decision rule always goes downhill, i.e., reduces energy.
- Hence it is impossible to escape from a local minima.
- **Solution** - Use random noise to escape from poor, shallow minima.
  - Start with a lot of noise to escape the energy barriers of poor local minima.



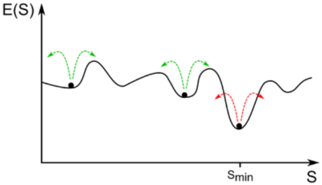
Noisy Networks

- The Binary Threshold Decision rule always goes downhill, i.e., reduces energy.
- Hence it is impossible to escape from a local minima.
- **Solution** - Use random noise to escape from poor, shallow minima.
    - Start with a lot of noise to escape the energy barriers of poor local minima.
    - Slowly reduce the noise so that the system ends up in a deep minima.



Noisy Networks

- The Binary Threshold Decision rule always goes downhill, i.e., reduces energy.
- Hence it is impossible to escape from a local minima.
- **Solution** - Use random noise to escape from poor, shallow minima.
  - Start with a lot of noise to escape the energy barriers of poor local minima.
  - Slowly reduce the noise so that the system ends up in a deep minima.
  - This process is called "**Simulated Annealing**".



Stochastic Binary Units

How to add noise?

Stochastic Binary Units

How to add noise?

1. Replace the binary threshold units by binary stochastic units that make biased random decisions.

Stochastic Binary Units

How to add noise?

1. Replace the binary threshold units by binary stochastic units that make biased random decisions.
2. The "**temperature**" controls the amount of noise.

Stochastic Binary Units

How to add noise?

1. Replace the binary threshold units by binary stochastic units that make biased random decisions.
2. The "**temperature**" controls the amount of noise.
3. Unit i then turns on with the probability given by the logistic function -

$$prob(s_i = 1) = \frac{1}{1 + e^{-\frac{\triangle E_i}{T}}} \tag{3}$$

Stochastic Binary Units

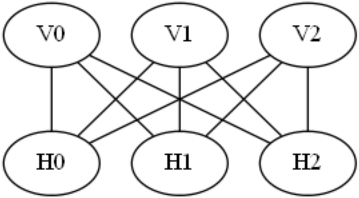
How to add noise?

1. Replace the binary threshold units by binary stochastic units that make biased random decisions.
2. The "**temperature**" controls the amount of noise.
3. Unit i then turns on with the probability given by the logistic function -

$$prob(s_i = 1) = \frac{1}{1 + e^{-\frac{\triangle E_i}{T}}} \tag{3}$$

$T = 0$	Deterministic (Hopfield Net)
$T \rightarrow \infty$	Complete Chaos
$T = 1$	Approaches Boltzmann Distribution

Boltzmann Machine



Statistical Mechanics

Consider a physical system with large number of possible states and many degrees of freedom.
Let p_i be the occurrence probability of state i

$$p_i \geq 0 \text{ for all } i \qquad \text{and} \qquad \sum_i p_i = 1.$$

At **thermal equilibrium** state i occurs with probability

$$p_i = \frac{1}{Z} exp\left(-\frac{E_i}{k_B T}\right)$$

where E_i is the energy of the system
Boltzmann constant, $k_B = 1.38x10^{-23}$ J/K
$exp\left(-\frac{E_i}{k_B T}\right)$ is the Boltzmann Factor
Partition function,(Zustadsumme)

$$Z = \sum_i exp\left(-\frac{E_i}{k_B T}\right)$$

Statistical Mechanics

Gibbs Distribution

1. States of low energy have a higher occurrence probability than states of high energy.
2. As T is reduced, the probability is concentrated on a smaller subset of low-energy states.

In the context of neural nets, the parameter T (which controls thermal fluctuations) represents the effect of **synaptic noise**. Hence $k_B = 1$ is set and p_i and Z getting the form

$$p_i = \frac{1}{Z} exp\left(-\frac{E_i}{T}\right)$$

$$Z = \sum_i exp\left(-\frac{E_i}{T}\right)$$

What is Boltzmann Machine?

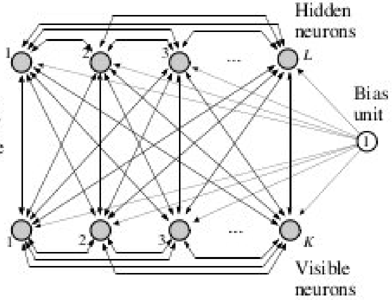
Boltzmann Machine

A Hopfield Net consisting of Binary Stochastic Neuron with hidden units is called Boltzmann Machine.

A **Boltzmann Machine** is a network of symmetrically connected, neuron like units that make stochastic decisions about whether to be on or off.

Structure of Boltzmann Machine

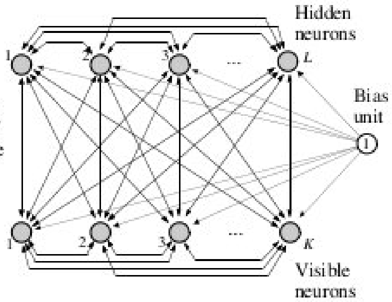
Figure 11.4:
Architectural graph of Boltzmann machine; K is the number of visible neurons and L is the number of hidden neurons.



- The stochastic neurons of Boltzmann machine are in two groups: **vissible** and **hidden**.

Structure of Boltzmann Machine

Figure 11.4: Architectural graph of Boltzmann machine; K is the number of visible neurons and L is the number of hidden neurons.



- The stochastic neurons of Boltzmann machine are in two groups: **vissible** and **hidden**.
- visible neurons provide an interface between the net and its environment.

Structure of Boltzmann Machine

- during the training phase, the visible neurons are **clamped**; the hidden neurons always operate freely, they are used to explain underlying constraints in the environmental input vectors.

Structure of Boltzmann Machine

- during the training phase, the visible neurons are **clamped**; the hidden neurons always operate freely, they are used to explain underlying constraints in the environmental input vectors.
- the hidden units do this (explain underlying constraints) by capturing higher-order correlations between the clamping vectors.

Structure of Boltzmann Machine

- during the training phase, the visible neurons are **clamped**; the hidden neurons always operate freely, they are used to explain underlying constraints in the environmental input vectors.
- the hidden units do this (explain underlying constraints) by capturing higher-order correlations between the clamping vectors.
- Boltzmann machine learning may be viewed as an unsupervised learning procedure for modelling a distribution that is specified by the clamping patterns.

Structure of Boltzmann Machine

- during the training phase, the visible neurons are **clamped**; the hidden neurons always operate freely, they are used to explain underlying constraints in the environmental input vectors.
- the hidden units do this (explain underlying constraints) by capturing higher-order correlations between the clamping vectors.
- Boltzmann machine learning may be viewed as an unsupervised learning procedure for modelling a distribution that is specified by the clamping patterns.
- the network can perform pattern completion: when a vector bearing part of the information is clamped onto a subset of the visible neurons, the network performs completion of the pattern on the remaining visible neurons (if it has learnt properly).

Modelling Binary Data

The objective of Boltzmann Machine is -

Modelling Binary Data

Given a training set of binary vectors, fit the model that will assign a probability to every possible binary vector.

When unit i is given opportunity to update its state, it first computes its total input z_i,

$$z_i = b_i + \sum_j s_j w_{ij} \tag{4}$$

Unit i turns on with probability -

$$prob(s_i = 1) = \frac{1}{1 + e^{-z_i}} \tag{5}$$

If the units are updated sequentially in random order, the network will eventually reach a **Boltzmann Distribution**, also called its stationary distribution.

How Boltzmann Machine Generates Data

- It is **not** a causal generative model.

How Boltzmann Machine Generates Data

- It is **not** a causal generative model.
- Everything is defined in terms of energies of joint configurations of the visible and hidden units.

How Boltzmann Machine Generates Data

- It is **not** a causal generative model.
- Everything is defined in terms of energies of joint configurations of the visible and hidden units.
- The energies of the joint configurations are related to their probabilities by two ways:
  - either by defining the probability $p(\mathbf{v}, \mathbf{h}) \propto e^{-E(\mathbf{v},\mathbf{h})}$
  - define the probability to be the probability of finding the network in that joint configuration after we have updated all of the stochastic binary units many times.

How Boltzmann Machine Generates Data

- It is **not** a causal generative model.
- Everything is defined in terms of energies of joint configurations of the visible and hidden units.
- The energies of the joint configurations are related to their probabilities by two ways:
  - either by defining the probability $p(\mathbf{v}, \mathbf{h}) \propto e^{-E(\mathbf{v},\mathbf{h})}$
  - define the probability to be the probability of finding the network in that joint configuration after we have updated all of the stochastic binary units many times.

binary state of unit i in **v**

bias of unit k

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

Energy with configuration **v** on the visible units and **h** on the hidden units

indexes every non-identical pair of i and j once

weight between visible unit i and hidden unit k

How Boltzmann Machine Generates Data

In Boltzmann Machine, everything is defined in terms of the energies of joint configurations of the visible (v) and hidden (h) units.

The energies of joint configurations are related to their probabilities as -

$$p(v, h) \propto e^{-E(v,h)} \tag{6}$$

Probabilities in term of Energies

- The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energies of all other joint configurations.

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

partition function

Probabilities in term of Energies

- The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energies of all other joint configurations.

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

partition function

- The probability of a configuration of the visible units is the sum of the probabilities of all the joint configurations that contain it.

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

An Example

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

An example of how weights define a distribution

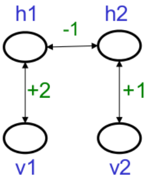


Figure: Credit: Geoffrey Hinton, Neural Networks for Machine Learning

Limitation and Solution

What if the network is remarkably large?

Limitation and Solution

What if the network is remarkably large?

- If there are large number of hidden units then we cannot calculate partition function, as it is exponentially many terms.
- We need to sample the data and to do this we can use Markov Chain Monte Carlo(MCMC).

Limitation and Solution

What if the network is remarkably large?

- If there are large number of hidden units then we cannot calculate partition function, as it is exponentially many terms.
- We need to sample the data and to do this we can use Markov Chain Monte Carlo(MCMC).
  - starting from a random global configuration pick units at random and allow them to stochastically update their states based on their energy gaps.
- Run MCMC until it reaches it stationary distribution(thermal eqb. at temp is 1)

Limitation and Solution

What if the network is remarkably large?

- If there are large number of hidden units then we cannot calculate partition function, as it is exponentially many terms.
- We need to sample the data and to do this we can use Markov Chain Monte Carlo(MCMC).
  - starting from a random global configuration pick units at random and allow them to stochastically update their states based on their energy gaps.
- Run MCMC until it reaches it stationary distribution(thermal eqb. at temp is 1)
  - the probability is related to its energy $p(v, h) \propto e^{-E(v,h)}$

Boltzmann Learning

"A surprising feature of this rule is that it uses only locally available information. The change of weight depends only on the behaviour of the two units it connects, even though the change optimizes a global measure."

- Ackley, Hinton 1985

Goal of Learning

Learning Algorithm for Boltzmann Machine is an unsupervised learning algorithm. Unlike Backpropagation Algorithm, where the training set consists of input vector and desired output, in Boltzmann Machine only the input vector is provided.

- We want to maximize the product of the probabilities the Boltzmann Machine assigns to the binary vectors in training set.
- It is equivalent to maximizing the probability that we would obtain exactly the N training cases.

Assumptions

The goal of Boltzmann learning is to produce a NN that categorize input patterns according to Boltzmann distribution. Two assumptions are made:

Assumptions

The goal of Boltzmann learning is to produce a NN that categorize input patterns according to Boltzmann distribution. Two assumptions are made:

1. Each environmental vector persists long enough for the network to reach **thermal equilibrium**.

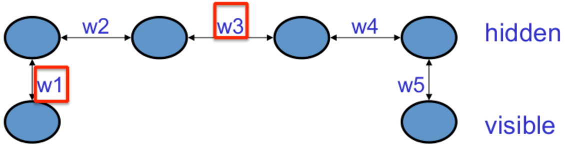
Assumptions

The goal of Boltzmann learning is to produce a NN that categorize input patterns according to Boltzmann distribution. Two assumptions are made:

1. Each environmental vector persists long enough for the network to reach **thermal equilibrium**.
2. There is no structure in the sequence in which environmental vectors are clamped to the visible units of the network.

Why Learning could be difficult

Consider a chain of hidden units with visibile units attached at two ends -

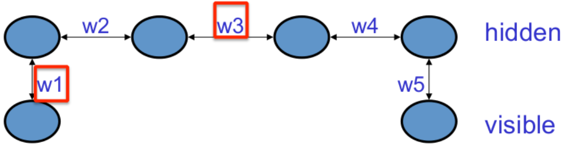


Training: We want the two visibile units to be in opposite states.

Solution: The product of all the weights must be negative. If all are positive, then turning on one unit will turn on the next unit and eventually the two visibile units will be in same state.

Why Learning could be difficult

Consider a chain of hidden units with visibile units attached at two ends -



Difficulty: To modify w1 and w5, we need to know w3 (and the weights of other hidden units too).

Because, if w3 is negative, then we need to modify w1 in a different way than what we would do if w3 is positive.

So to change one weight in a right direction, we need to know all the other weights.

Boltzmann Learning Algorithm

The learning procedure mainly divided into three phases:

Boltzmann Learning Algorithm

The learning procedure mainly divided into three phases:

1. Clamping phase

Boltzmann Learning Algorithm

The learning procedure mainly divided into three phases:

1. Clamping phase
2. Free-running phase

Boltzmann Learning Algorithm

The learning procedure mainly divided into three phases:

1. Clamping phase
2. Free-running phase
3. Learning phase

Algorithm

1 **Initialization**: Set weights to random numbers in [-1, 1]

Algorithm

1. **Initialization**: Set weights to random numbers in [-1, 1]
2. **Clamping Phase**: Present the net with the mapping it is supposed to learn by clamping input and output units to patterns. For each pattern perform simulated annealing on the hidden units in the sequence $T_0, T_1, ..., T_{final}$. At the final temperature, collect statistics to estimate the correlations

$$\rho_{ji}{}^{+} = \langle s_j s_i \rangle^{+} \qquad (j \neq i)$$

here $\langle s_j s_i \rangle^{+} = \sum_{\mathbf{s}_\alpha \in \Im} \sum_{\mathbf{s}_\beta} P(\mathbf{S}_\beta = \mathbf{s}_\beta | \mathbf{S}_\alpha = \mathbf{s}_\alpha) s_j s_i$

where $\mathbf{s}_\alpha$ and $\mathbf{s}_\beta$ represents the vector of visible and hidden neurons respectively

Algorithm

3 **Free-running Phase**:Repeat calculations in step 2, but this time only clamp the input units. Hence, at the final temperature, estimate the correlations

$$\rho_{ji}^{-} = \langle s_j s_i \rangle^{-} \quad (j \neq i)$$

here $\langle s_j s_i \rangle^{-} = \sum_{\mathbf{s}_\alpha \in \Im} \sum_{\mathbf{s}_\beta} P(\mathbf{S}_\beta = \mathbf{s}_\beta) s_j s_i$

Algorithm

3 **Free-running Phase**:Repeat calculations in step 2, but this time only clamp the input units. Hence, at the final temperature, estimate the correlations

$$\rho_{ji}{}^{-} = \langle s_j s_i \rangle^{-} \quad (j \neq i)$$

here $\langle s_j s_i \rangle^{-} = \sum_{\mathbf{s}_\alpha \in \Im} \sum_{\mathbf{s}_\beta} P(\mathbf{S}_\beta = \mathbf{s}_\beta) s_j s_i$

4 **Learning Phase**: updating the weights using the learning rule

$$\triangle w_{ji} = \eta(\rho_{ji}{}^{+} - \rho_{ji}^{-})$$

η is **learning parameter** depending upon T $(\eta = \frac{\varepsilon}{T})$

Algorithm

3. **Free-running Phase**:Repeat calculations in step 2, but this time only clamp the input units. Hence, at the final temperature, estimate the correlations

$$\rho_{ji}{}^{-} = \langle s_j s_i \rangle^{-} \qquad (j \neq i)$$

   here $\langle s_j s_i \rangle^{-} = \sum_{\mathbf{s}_\alpha \in \Im} \sum_{\mathbf{s}_\beta} P(\mathbf{S}_\beta = \mathbf{s}_\beta) s_j s_i$

4. **Learning Phase**: updating the weights using the learning rule

$$\triangle w_{ji} = \eta(\rho_{ji}{}^{+} - \rho_{ji}^{-})$$

   η is **learning parameter** depending upon T ($\eta = \frac{\varepsilon}{T}$)

5. **Iteration**: Iterate steps 2 to 4 until the learning procedure converges with no more changes with synaptic weight $w_{ji}\ \forall\ j, i$.

Learning Algorithm

Correlation

Everything that one weight needs to know of other weights and the data is contained in the difference of two correlations -

$$\frac{\partial logp(v)}{\partial w_{ij}} = \langle s_i s_j \rangle_v - \langle s_i s_j \rangle_{model} \tag{7}$$

where $\langle . \rangle$ is the expectation value.

- The first term in R.H.S. denotes the expectation value of product of states at equilibrium when the state vector (or data) v is clamped on the visibile units.
- The second term in R.H.S. denotes the expectation value of product of states at equilibrium without any clamping.

So we can make the change in weight -

$$\triangle w_{ij} \propto \langle s_i s_j \rangle_v - \langle s_i s_j \rangle_{model} \tag{8}$$

Why is it so?

We know the probability of a global configuration **at equilibrium** is -

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

So, the logarithm of probability is a linear function of the energy.

And energy, on its own term, is a linear function of weights and states.

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

Hence,

$$\frac{\partial E}{\partial w_{ij}} = -s_i s_j \tag{9}$$

Why is it so?

Differentiating equation 1, we get -

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

1. The process of settling to equilibrium state propagates information about the weights.
2. No need of back-propagation.
3. The following two stages are required -
   - The machine needs to settle to equilibrium with data.
   - The machine needs to settle to equilibrium without data.
4. However, in both cases the learning process is similar with different boundary conditions.

Why we need negative phase

- The combine use of positive and negative phase stabilizes the distribution of synaptic weights.

Why we need negative phase

- The combine use of positive and negative phase stabilizes the distribution of synaptic weights.
- The both phases are important equally due to the presence of **partition function** Z.

Why we need negative phase

- The combine use of positive and negative phase stabilizes the distribution of synaptic weights.
- The both phases are important equally due to the presence of **partition function** Z.
- The direction of **steepest descent** in energy space in **not** the same as the direction of **steepest ascent** in probability space.

Shortcomings

There are few problems with the Boltzmann algorithm

Shortcomings

There are few problems with the Boltzmann algorithm

- It is not prefixed that how many times we need to iterate.

Shortcomings

There are few problems with the Boltzmann algorithm

- It is not prefixed that how many times we need to iterate.
- Due to the presence of negative phase it takes a greater computation time

Shortcomings

There are few problems with the Boltzmann algorithm

- It is not prefixed that how many times we need to iterate.
- Due to the presence of negative phase it takes a greater computation time
- This algorithm computes averages of two phases and take their **difference**. When these two correlations similar, the presence of sampling noise makes the difference more noisy.

Shortcomings

There are few problems with the Boltzmann algorithm

- It is not prefixed that how many times we need to iterate.
- Due to the presence of negative phase it takes a greater computation time
- This algorithm computes averages of two phases and take their **difference**. When these two correlations similar, the presence of sampling noise makes the difference more noisy.
- It runs very slow. Take very much time to learn.

Shortcomings

There are few problems with the Boltzmann algorithm

- It is not prefixed that how many times we need to iterate.
- Due to the presence of negative phase it takes a greater computation time
- This algorithm computes averages of two phases and take their **difference**. When these two correlations similar, the presence of sampling noise makes the difference more noisy.
- It runs very slow. Take very much time to learn.
- Weight explosion: If weights get too big too early, then the network get struck in one goodness optimum.

Shortcomings

There are few problems with the Boltzmann algorithm

- It is not prefixed that how many times we need to iterate.
- Due to the presence of negative phase it takes a greater computation time
- This algorithm computes averages of two phases and take their **difference**. When these two correlations similar, the presence of sampling noise makes the difference more noisy.
- It runs very slow. Take very much time to learn.
- Weight explosion: If weights get too big too early, then the network get struck in one goodness optimum.

- This shortcomings can be eliminated by **sigmoid belief network**

Applications

- stock market trend prediction

Applications

- stock market trend prediction
- character recognition

Applications

- stock market trend prediction
- character recognition
- Face recognition

Applications

- stock market trend prediction
- character recognition
- Face recognition
- Internet Application

Applications

- stock market trend prediction
- character recognition
- Face recognition
- Internet Application
- Cancer Detection

Applications

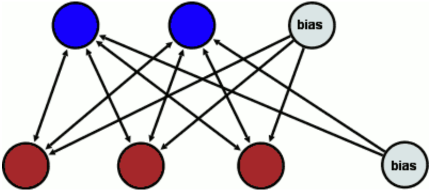
- stock market trend prediction
- character recognition
- Face recognition
- Internet Application
- Cancer Detection
- Loan Application

Applications

- stock market trend prediction
- character recognition
- Face recognition
- Internet Application
- Cancer Detection
- Loan Application
- Decision making

Restricted Boltzmann Machine

Restricted Boltzmann Machine is a stochastic neural network (random behaviour when activated). It consist of one layer of visible units (neurons) and one layer of hidden units. Units in each layer have no connections between them and are connected to all other units in other layer. Connections between neurons are bidirectional and symmetric . This means that information flows in both directions during the training and during the usage of the network and that weights are the same in both directions.



How RBM Works

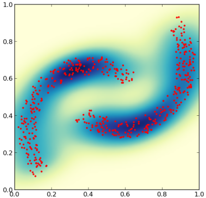
- First the network is trained by using some data set and setting the neurons on visible layer to match data points in this data set.

How RBM Works

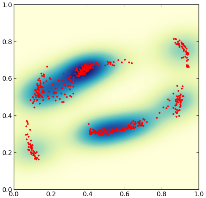
- First the network is trained by using some data set and setting the neurons on visible layer to match data points in this data set.
- After the network is trained we can use it on new unknown data to make classification of the data (this is known as unsupervised learning)

Continuous Restricted Boltzmann Machine

CRBM have very close implementation to original RBM with binomial neurons (0,1) as possible values of activation.



Training data[2]



Reconstructed data

[2] 500 training data were taken

Reference

Geoffrey Hinton
Neural Networks for Machine Learning
www.coursera.org

Geoffrey Hinton
http://www.scholarpedia.org/article/Boltzmann_machine

Simon Haykin, 2ed

Geoffrey Hinton, David Ackley
A learning Algorithm for Boltzmann Machine
Cognitive Science 9, 147-169 (1985)

THANK YOU