Deep Belief Networks: Underpinnings

Earl Spillar

2013-08-15

Outline

Purpose

- Our goal is to explain Deep Belief Networks based on Restricted Boltzmann Machines
- We may decide to play around with implementing these as well
- This will not be accomplished in one 20 min talk!
- There are quite a few materials available on the web- see references at end

Deep Belief networks

- First became aware through google tech talk by Geoff Hinton
- `http://www.youtube.com/watch?v=AyzOUbkUf3M` The Next Generation of Neural Networks
- Geoff Hinton and others have been cooking these since 1984 or so
- Talk was good, but left me confused
- Recently (last year) his group one several competitions- see his web site

History of Nerual Nets: Some Motivation

- Neural net craze started (for me) with Hopfield, early 80s, but-
- Hopfield nets, circa 1984, were shown to be insufficiently strong (Minsky)
- Backpropagation is slow at training things, and requires labeled examples to train
- How to make something that trains faster?

Deep Belief Networks Features

- Program by showing examples of "Patterns"
- Think of it as something in n-space, typically binary
- It will automatically categorize these into sets
- No Traning Sets (unless you want to)
- Can reverse and "dream" examples
- "holographic-" fill in missing pieces
- Works as well as vector support machines or better without traning
- Architecture similar to the mind

Basic architecture of Deep Belief Networks

- Built on (Restrictred) Boltzmann Machines- discussed this month
- Training (R)BMs has been speeded up with a trick so it's MANY times faster than it used to be
- One layer learns, basis for next, etc.
- Information flow can also be reversed and go back down the tree
- Multiple layers are essential for "abstracting" features, cascading allows complex things to go on

Deep Belief Networks/Deep Boltzmann Machines

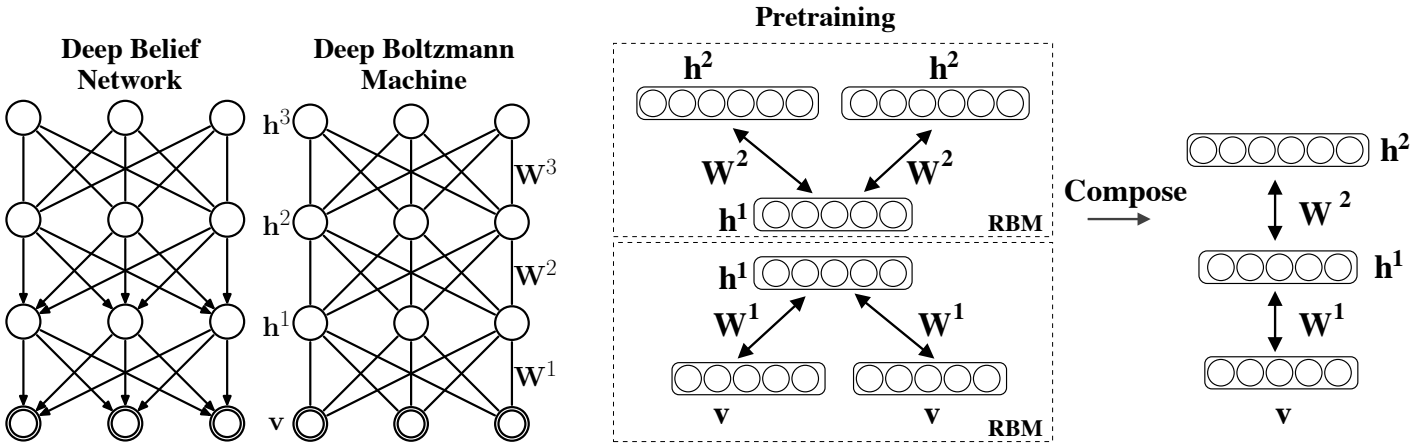
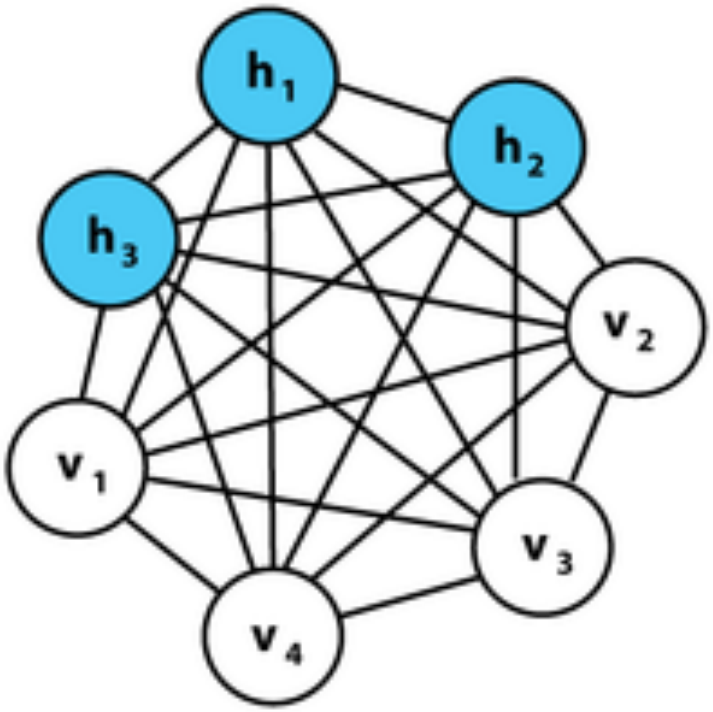


Figure 2: **Left**: A three-layer Deep Belief Network and a three-layer Deep Boltzmann Machine. **Right**: Pretraining consists of learning a stack of modified RBM's, that are then composed to create a deep Boltzmann machine.

- Salakhutdinov et al.

Neural Nets: Boltzmann Machines

- A bunch of "neurons" connected by weights
- want particular configurations to have "greater probability", corresponding to reality



- from Scholarpedia

Probablility of Configuration

- Consider an array of $i = 1 \ldots n$ neurons
- let the state of the neurons in a particular configuration be s_i, either 0 or 1
- The weights between the cells can be expressed in a matrix w_{ij}
- Indivifual cells have "threshhold" or "bias" terms θ_i
- I think the people in this area started as physicists, so we have an analogy with statistical mechanics
- Define the energy as $E_{config} = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i$
- $w_{ii} = 0$ and $w_{ij} = w_{ji}$.
- basically the most likely states are the ones with the lowest energy

Cells With No Information

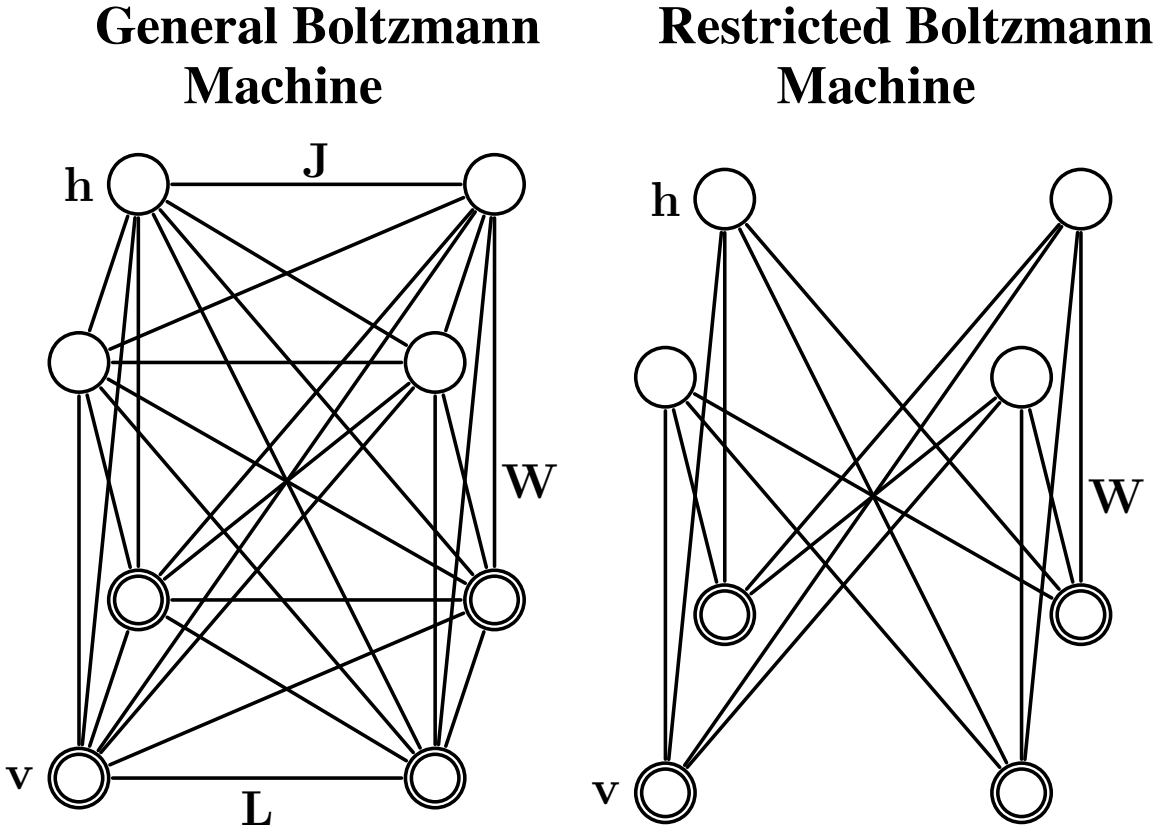
- Imagine now you have most of the cells filled in from an observation, but you don't have information about a few
- Using the best w and α fill in the known cells
- Solve for the "unknown" cells by looking for the values that would give the network the lowest energy
- "holographic"- missing information can be filled in
- Note that one network can store information on several patters
- All we need is a way to program the network based on examples

Hidden cells

- When you program the network, you might insert a few cells NOT tied to inputs
- When you program the network, it turns out that these will be pulled to be "identifiers" for particular patterns
- In programming the network, each cell essentially becomes a flag raised for a particular class of input
- I don't have a good explanation for this (yet)
- So, if we have a way to train the network, it will recognize patterns and throw this information into these hidden cells

Restricted vs Unrestricted Boltzmann Machines

- Restricted Bolzmann machinees have fewer connections
- They are faster to train
- The hidden cells are in one layer, sensed cells in another layer



- from Scholarpedia

Programming the network- finding the lowest energy state

- Statical mechanics, and monte carlo method ala Metropolis, Ulam, von Neumann, Fermi
- Simulatd annealing process
- How do you find the lowest energy state for particular α and w?
- Randomly change state "locally"- that is change an n_i
- accept the change if the energy has decreased
- occasionally accept changes if energy worse, based on a decreasing "temperature"
- This is analogous to the cooling of a liquid into a solid
- If you go slow, you get low energy organized states: crystals
- These are the lowest energy states

Programming the network itself: w and θ

Here was the original programming algorithm:

- How to determine w and θ?
- Find the values of w and θ which minimize the energy for the training set
- Use a variant of simulated annealing:
- Feed in the traning set, that is points
- Vary all the w and θ to minimze energy

Incorporating hidden cells into the training

- We want to find a way to include the "hidden units" into the program, so that they will correspond to particular common states of the system.
- These enable the network to understand more complicated patterns then can be encapsulated with simple pair connections
- An example in an early paper was recognizing a shift register

Wake Sleep Learning: A Preview

- To train these more complex nets, a somewhat more complex annealing algorithm is run:
- It can be shown $\Delta w_{ij} - \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$
- The first term is the expectation w.r.t. the data
- The first expectation is done by using a sample input vector, drawing random h_j using a sigmoid distribution, and averaging
- The second expectation is the expectation w.r.t the programming
- The second expectation is done by starting things at random, and Gibbs sampling for a very long time
- Let's dive in here next time!

An algorithm to imlement today:

- Create nets with hidden nodes
- Create matrices of connections
- Create code that calculates energies
- Create code that perturbs
- Accept perturbation with probability $\frac{1}{1+e^{\Delta E/T}}$

Refrences (1/2)

- Google Tech Talk Video: `http://www.youtube.com/watch?v=AyzOUbkUf3M`
- You can find most of these at Google Scholar: scholar.google.com
- Hinton's group's website: `http://learning.cs.toronto.edu/~hinton/` many references there
- Scholarpedia: `http://www.scholarpedia.org/article/Boltzmann_machine`
- Early paper on training Boltzmann Machines: Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski. "A learning algorithm for Boltzmann machines." Cognitive science 9.1 (1985): 147-169. `http://www.learning.cs.toronto.edu/~hinton/absps/cogscibm.pdf`

References (2/2)

- Hinton, Geoffrey E., and Terrance J. Sejnowski. "Learning and relearning in Boltzmann machines." MIT Press, Cambridge, Mass 1 (1986): 282-317.
- A Practical Guide to Training Restricted Boltzmann Machines - Hinton, from his web site
- Deep review paper: Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127. `http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf`
- Wikipeida of course `http://en.wikipedia.org/wiki/Boltzmann_machine`
- Deep Boltzmann Machines Salakhutdinov, Ruslan, and Geoffrey E. Hinton. "Deep boltzmann machines." In International Conference on Artificial Intelligence and Statistics, pp. 448-455. 2009