Belief Nets

Summary of video lectures by G.Hinton

Presented by: Ardi Tampuu

Institute of Computer Science, University of Tartu

December 2014

Problems with backpropagation

By the end of 90ies the AI world had given up on backpropagation, even though in theory it allowed to teach complicated networks to do fancy stuff. Neural nets were blamed to :

- not be able to make use of the hidden layers
- did not work well on recurrent nets, nor auto-encoders
- actually the fault of insufficient training data and bad initialization
- SVM worked a lot better for the things AI-people were trying to do back then

Different types of learning

Hinton points out that AI and Statistics deal with very different set of tasks (and I agree).

In statistics we aim to explain the relationship between our variables in a fairly simple model (that humans can understand). This means we usually have low dimensionality and we need to minimize the amount of noise.

In AI we do not care if we understand how the system achieves the goal, as long as the goal is achieved. (we can let back-prop to figure out what features to look for). As we do not aim to understand the complex interaction, the data can be of high-dimensions.

What is SVM ?

Support Vector Machine is a classification tool that searches for a hyperplane that best separates the two classes in the feature space. One can also project the data into a higher-dimensional space and search for a hyperplane in there.

- very few parameters to fit
- over-fitting can be easily avoided

Hinton claims the idea is the same as perceptrons, but presented in a fancier mathematical package.

The verdict on back-propagation

In conclusion Hinton says that back-prop is a powerful tool, but it is very needy :

- needs labelled data (almost all data is not labelled)
- learning through multiple layers takes a forever (and a half)
- can't see the big picture and gets stuck in local optima

This led to developing unsupervised methods that could discover the structure in the data.

Unsupervised learning

We aim to understand the regularities in the data.

Generative models - we want to train a model that would generate data similar to our training data (maximize the likelihood of our data being generated by the model)

Different types of generative models - Boltzmann machine (only one hidden layer), causal models

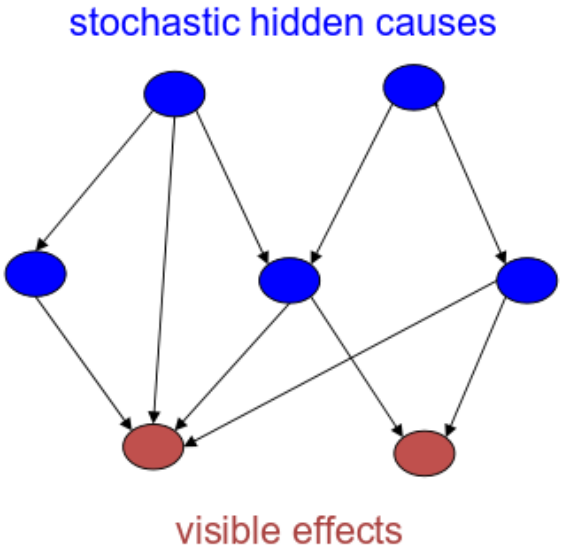
Types of causal models

Graphical models :

we define a graph structure that represents the dependencies, then we can (probabilistically) compute the expected values of some variables, knowing others

Belief nets : directed acyclic graph with stochastic nodes. We want two things :

- to be able to infer the states of hidden nodes knowing visible nodes.
- adjust weights so that the model generates data similar to



Belief nets

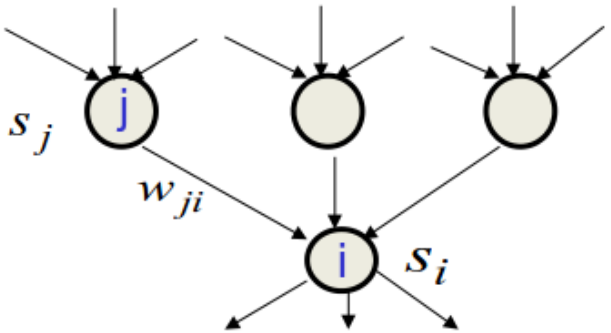
Unlike early graphical models, in Belief Nets the connectivity is not man-made and is learned from the data. Connectivity is not sparse and inference is difficult.

- We could make a belief net with Boltzmann machines (symmetric connections, easy to learn if RBM).
- We could also connect binary stochastic neurons with directed connections, getting a Sigmoid Belief Net

Learning Sigmoid Belief Nets

The learning rule for sigmoid belief nets

- Learning is easy if we can get an unbiased sample from the posterior distribution over hidden states given the observed data.
- For each unit, maximize the log prob. that its binary state in the sample from the posterior would be generated by the sampled binary states of its parents.



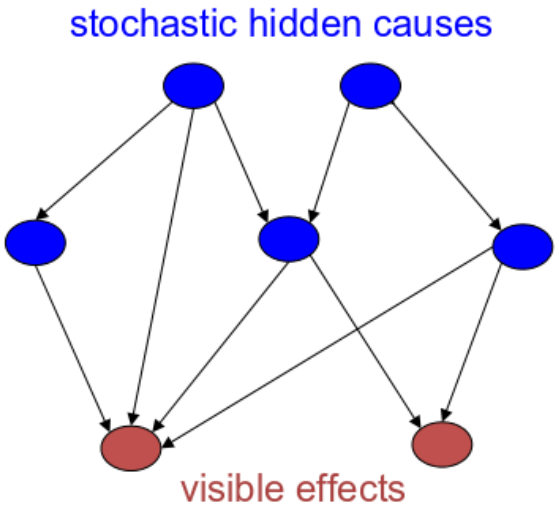
$$p_i \equiv p(s_i = 1) = \frac{1}{1 + \exp\left(-b_i - \sum_j s_j w_{ji}\right)}$$

$$\Delta w_{ji} = \varepsilon \, s_j (s_i - p_i)$$

Types of causal models

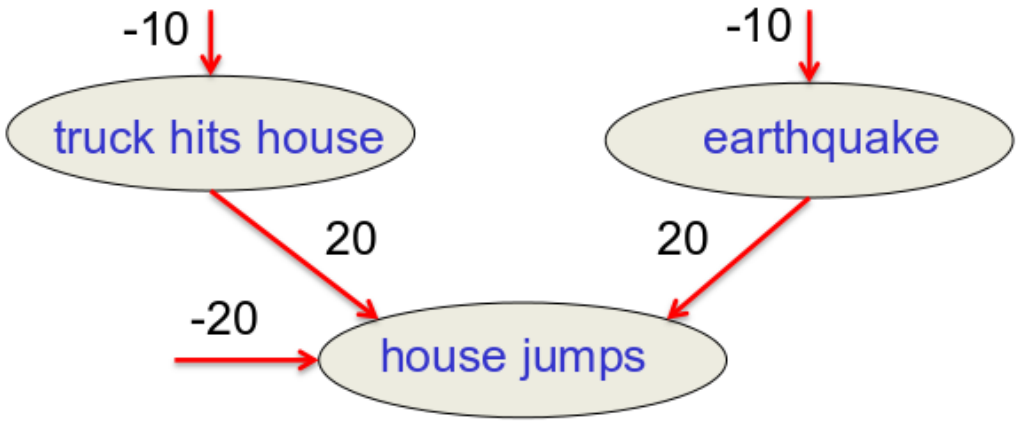
Learning Sigmoid Belief Nets

- It is easy to generate an unbiased example at the leaf nodes, so we can see what kinds of data the network believes in.
- It is hard to infer the posterior distribution over all possible configurations of hidden causes.
- It is hard to even get a sample from the posterior.
- So how can we learn sigmoid belief nets that have millions of parameters?



Explaining away

- Even if two hidden causes are independent in the prior, they can become dependent when we observe an effect that they can both influence.
  - If we learn that there was an earthquake it reduces the probability that the house jumped because of a truck.

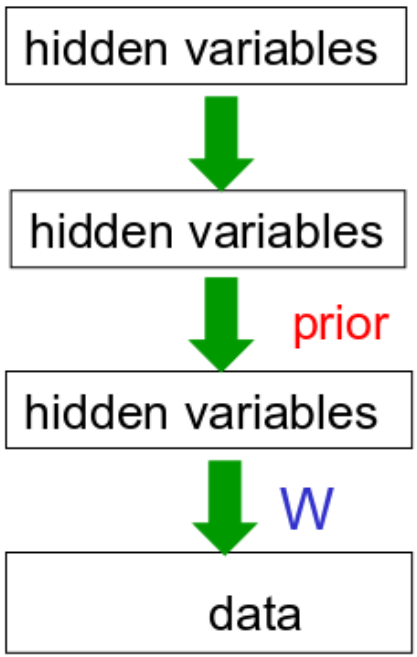


posterior over hiddens

p(1,1)=.0001
p(1,0)=.4999
p(0,1)=.4999
p(0,0)=.0001

What do we need to learn the weights ?

- To learn W, we need to sample from the posterior distribution in the first hidden layer.
- Problem 1: The posterior is not factorial because of "explaining away".
- Problem 2: The posterior depends on the prior as well as the likelihood.
  - So to learn W, we need to know the weights in higher layers, even if we are only approximating the posterior. All the weights interact.
- Problem 3: We need to integrate over all possible configurations in the higher layers to get the prior for first hidden layer. Its hopeless!



What do we need to learn the weights ?

We could use Monte Carlo method to sample from the posterior, but it takes a lot of time.

On the other hand, we could try to approximate the distribution. This gives biased samples and we do not maximize the likelihood by doing it, we optimize something else.

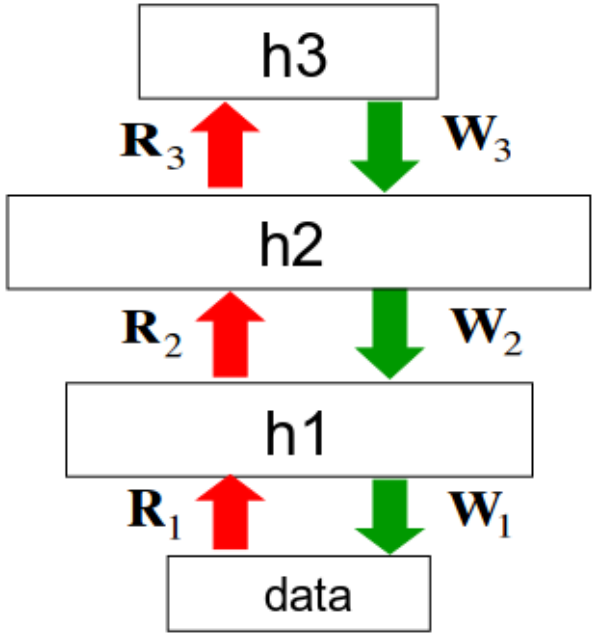
Learning with wrong distributions

It was clear that doing the right thing is computationally too expensive. Being desperate, people tried doing things wrong... and it worked !

Basically we omit the fact that the probabilities of hidden layers are dependent (due to explaining away). This means the probability of a whole vector (state of hidden layer) is the product of probabilities of each element.

Sleep-Wake algorithm ?

- Wake phase: Use recognition weights to perform a bottom-up pass.
  - Train the generative weights to reconstruct activities in each layer from the layer above.
- Sleep phase: Use generative weights to generate samples from the model.
  - Train the recognition weights to reconstruct activities in each layer from the layer below.



Sounds cool, are there any problems ?

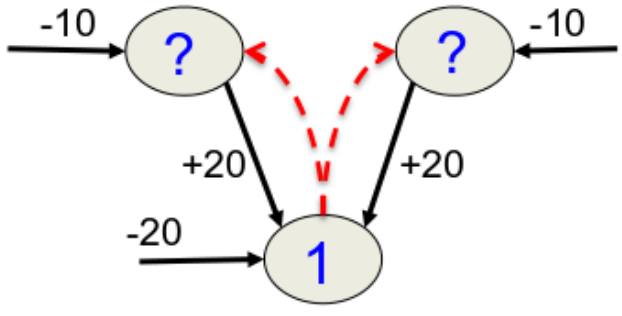
As we can imagine, doing something that is theoretically wrong has its downsides and special cases that it doesn't work :

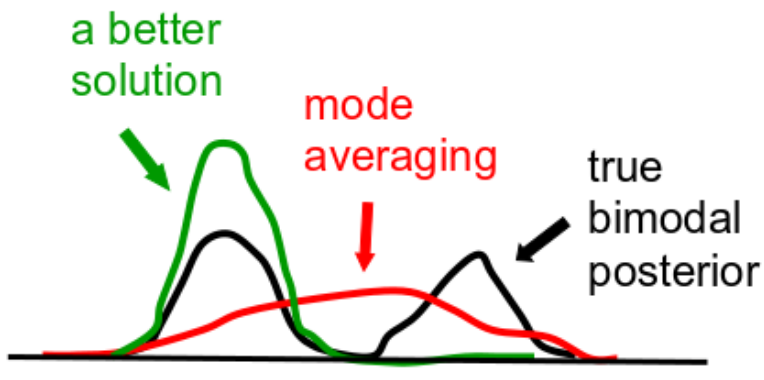
- The recognition weights are trained to invert the generative model in parts of the space where there is no data
- The posterior over the top hidden layer is very far from independent because of explaining away effects
- The recognition weights do not follow the gradient of the log probability of the data. Leads to *mode-averaging*

Sleep-Wake algorithm ?

Mode averaging

- If we generate from the model, half the instances of a 1 at the data layer will be caused by a (1,0) at the hidden layer and half will be caused by a (0,1).
  - So the recognition weights will learn to produce (0.5, 0.5)
  - This represents a distribution that puts half its mass on 1,1 or 0,0: very improbable hidden configurations.
- Its much better to just pick one mode.
  - This is the best recognition model you can get if you assume that the posterior over hidden states factorizes.





Thanks !

Thank you for your attention !