CIAR Second Summer School Tutorial Lecture 2a

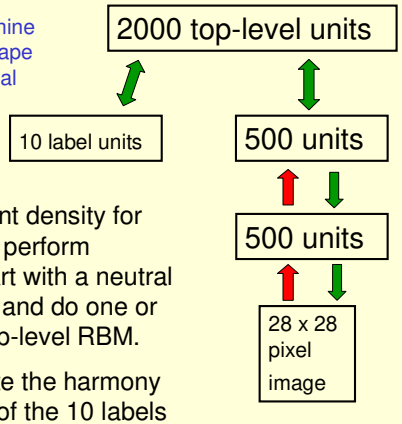
Learning a Deep Belief Net

Geoffrey Hinton

A neural network model of digit recognition

The top two layers form a restricted Boltzmann machine whose free energy landscape models the low dimensional manifolds of the digits.

The valleys have names:



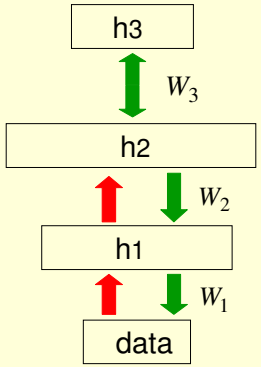
The model learns a joint density for labels and images. To perform recognition we can start with a neutral state of the label units and do one or two iterations of the top-level RBM.

Or we can just compute the harmony of the RBM with each of the 10 labels

The generative model

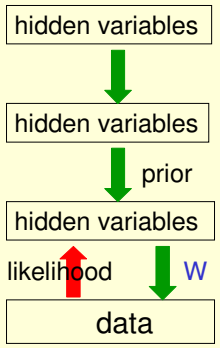
- To generate data:
  1. Get an equilibrium sample from the top-level RBM by performing alternating Gibbs sampling forever.
  2. Perform a top-down ancestral pass to get states for all the other layers.

  So the lower level bottom-up connections are not part of the generative model



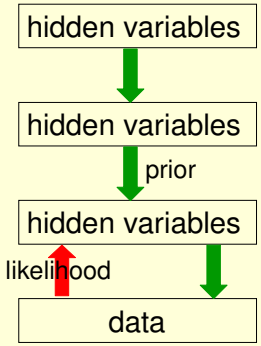
Why its hard to learn belief nets one layer at a time

- To learn W, we need the posterior distribution in the first hidden layer.
- Problem 1: The posterior is typically intractable 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 of the higher variables to get the prior for first hidden layer. Yuk!



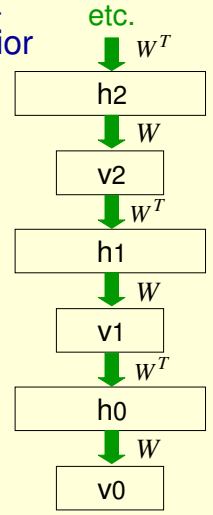
Using complementary priors to eliminate explaining away

- A "complementary" prior is defined as one that exactly cancels the correlations created by explaining away. So the posterior factors.
  - Under what conditions do complementary priors exist?
  - Complementary priors do not exist in general:
    - Parameter counting shows that complementary priors cannot exist if the relationship between the hidden variables and the data is defined by a separate conditional probability table for each hidden configuration.



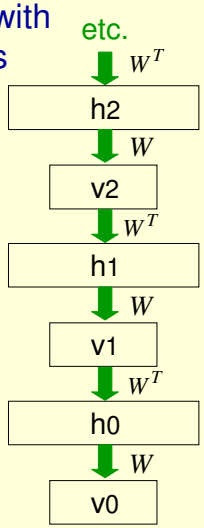
An example of a complementary prior

- The distribution generated by this infinite DAG with replicated weights is the equilibrium distribution for a compatible pair of conditional distributions: p(v|h) and p(h|v).
  - An ancestral pass of the DAG is exactly equivalent to letting a Restricted Boltzmann Machine settle to equilibrium.
  - So this infinite DAG defines the same distribution as an RBM.

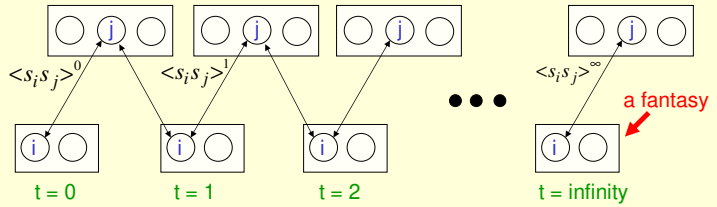


Inference in a DAG with replicated weights

- The variables in h0 are conditionally independent given v0.
  - Inference is trivial. We just multiply v0 by W^T
  - This is because the model above h0 implements a complementary prior.
- Inference in the DAG is exactly equivalent to letting a Restricted Boltzmann Machine settle to equilibrium starting at the data.



A picture of the Boltzmann machine learning algorithm for an RBM



Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\Delta w_{ij} = \varepsilon \left(<s_i s_j>^0 - <s_i s_j>^\infty \right)$$

- The learning rule for a logistic DAG is:

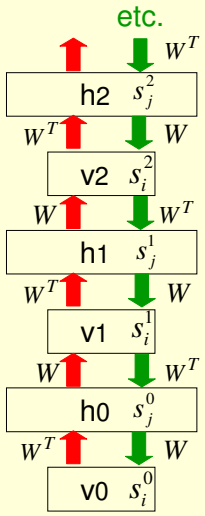
$$\Delta w_{ij} \propto s_j (s_i - \hat{s}_i)$$

- With replicated weights this becomes:

$$s_j^0 (s_i^0 - s_i^1) + s_i^1 (s_j^0 - s_j^1) + s_j^1 (s_i^1 - s_i^2) + \ldots$$

The derivatives for the recognition weights are zero.

$s_j^\infty s_i^\infty$

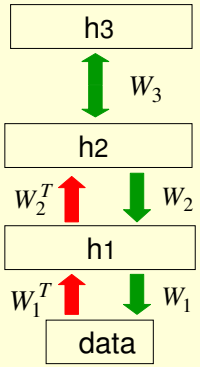


Multilayer contrastive divergence

- Start by learning one hidden layer.
- Then re-present the data as the activities of the hidden units.
  - The same learning algorithm can now be applied to the re-presented data.
- Can we prove that each step of this greedy learning improves a bound on the log probability of the data under the overall model?
  - What is the overall model?

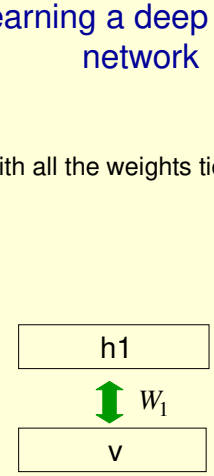
A simplified version with all hidden layers the same size

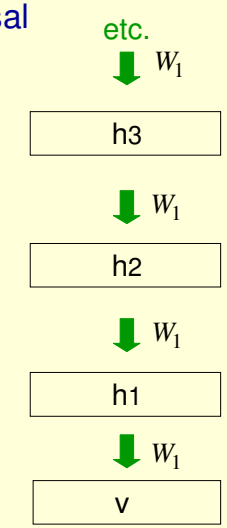
- The RBM at the top can be viewed as shorthand for an infinite directed net.
- When learning W_1 we can view the model in two quite different ways:
  - The model is an RBM composed of the data layer and h1.
  - The model is an infinite DAG with tied weights.
- After learning W_1 we untie it from the other weight matrices.
- We then learn W_2 which is still tied to all the matrices above it.



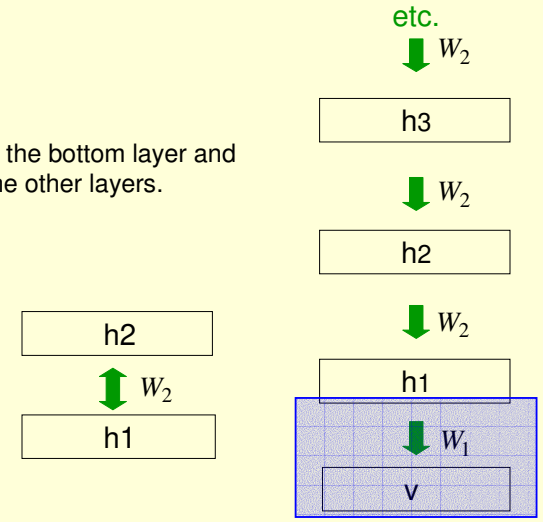
Learning a deep causal network

- First learn with all the weights tied

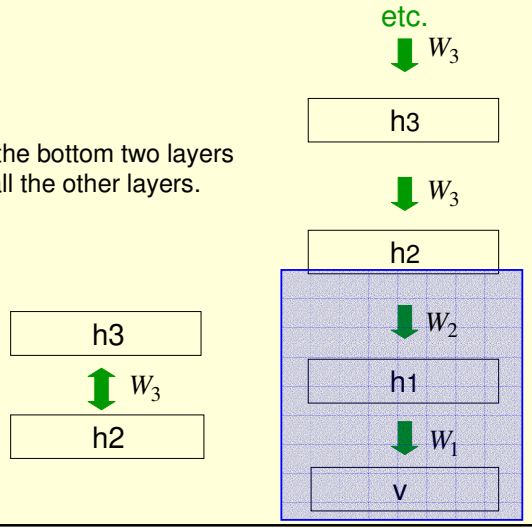




- Then freeze the bottom layer and relearn all the other layers.



- Then freeze the bottom two layers and relearn all the other layers.



Why the hidden configurations should be treated as data when learning the next layer of weights

- After learning the first layer of weights:

$$\log p(x) \geq -\langle energy(x) \rangle + entropy(h1 \mid x)$$
$$\geq \sum_{\alpha} p(h1 = \alpha \mid x) \left[\log p(h1 = \alpha) + \log p(x \mid h1 = \alpha)\right] + entropy$$

- If we freeze the generative weights that define the likelihood term and the recognition weights that define the distribution over hidden configurations, we get:

$$\log p(x) \geq \sum_{\alpha} p(h1 = \alpha \mid x) \left[\log p(h1 = \alpha)\right] + constant$$

- Maximizing the RHS is equivalent to maximizing the log prob of "data" α that occurs with probability $p(h1 = \alpha \mid x)$

Why greedy learning works

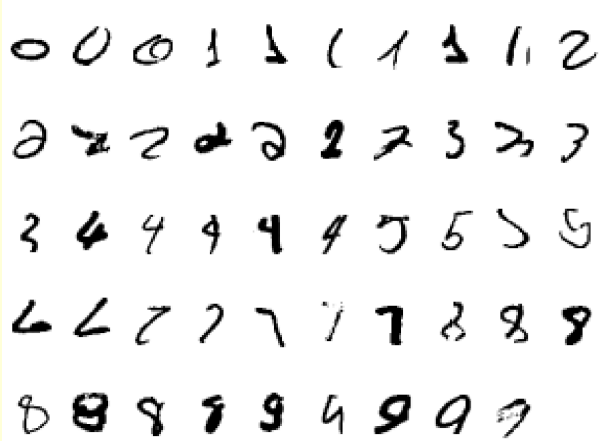
- Each time we learn a new layer, the inference at the layer below becomes incorrect, but the variational bound on the log prob of the data improves provided we start the learning from the tied weights that implement the complementary prior.
- Now that we have a guarantee we can loosen the restrictions and still feel confident.
  - Allow layers to vary in size.
  - Do not start the learning at each layer from the weights in the layer below.

Back-fitting

- After we have learned all the layers greedily, the weights in the lower layers will no longer be optimal. We can improve them in two ways:
  - Untie the recognition weights from the generative weights and learn recognition weights that take into account the non-complementary prior implemented by the weights in higher layers.
  - Improve the generative weights to take into account the non-complementary priors implemented by the weights in higher layers.
- What algorithm should we use for improving on the weights that are learned greedily?

Show the movie

Examples of correctly recognized MNIST test digits (the 49 closest calls)



How well does it discriminate on MNIST test set with no extra information about geometric distortions?

- Up-down net with RBM pre-training + CD10 1.25%
- SVM (Decoste & Scholkopf) 1.4%
- Backprop with 1000 hiddens (Platt) 1.5%
- Backprop with 500 -->300 hiddens 1.5%

- Separate hierarchy of RBM's per class 1.7%
- Learned motor program extraction ~1.8%
- K-Nearest Neighbor ~ 3.3%

- Its better than backprop and much more neurally plausible because the neurons only need to send one kind of signal, and the teacher can be another sensory input.

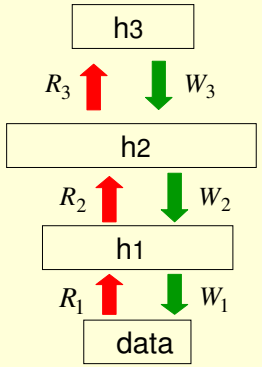
All 125 errors

Samples generated by running the top-level RBM with one label clamped. There are 1000 iterations of alternating Gibbs sampling between samples.

Samples generated by running top-level RBM with one label clamped. Initialized by an up-pass from a random binary image. 20 iterations between samples.

The wake-sleep algorithm

- Wake phase: Use the 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 the generative weights to generate samples from the model.
  - Train the recognition weights to reconstruct activities in each layer from the layer below.



The flaws in the wake-sleep algorithm

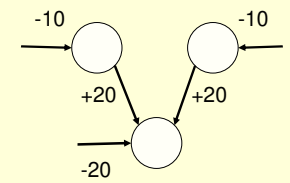
- The recognition weights are trained to invert the generative model in parts of the space where there is no data.
  - This is wasteful.
- The recognition weights follow the gradient of the wrong divergence. They minimize KL(P||Q) but the variational bound requires minimization of KL(Q||P).
  - This leads to incorrect mode-averaging
- The posterior over the top hidden layer is very far from independent because the independent prior cannot eliminate explaining away effects.

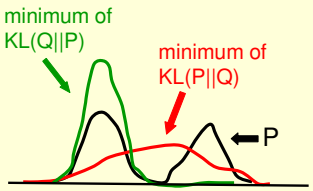
A contrastive divergence version of wake-sleep

- Replace the top layer of the DAG by an RBM
  - This eliminates bad variational approximations caused by top-level units that are independent in the prior.
  - It is nice to have an associative memory at the top.
- Replace the ancestral pass in the sleep phase by a top-down pass starting with the state of the RBM produced by the wake phase.
  - This makes sure the recognition weights are trained in the vicinity of the data.
  - It also reduces mode averaging. If the recognition weights prefer one mode, they will stick with that mode even if the generative weights like some other mode just as much.

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 very improbable hidden configurations.
- Its much better to just pick one mode and pay one bit.

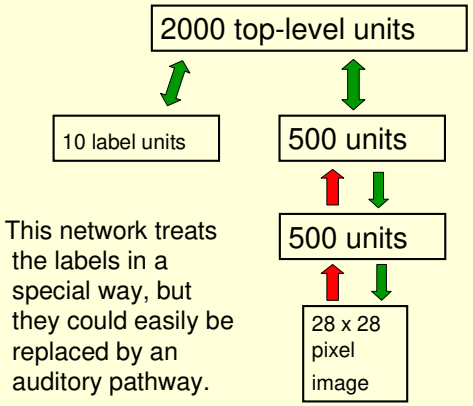




A different way to capture low-dimensional manifolds

- Instead of trying to explicitly extract the coordinates of a datapoint on the manifold, map the datapoint to an energy valley in a high-dimensional space.
- The learned energy function in the high-dimensional space restricts the available configurations to a low-dimensional manifold.
  - We do not need to know the manifold dimensionality in advance and it can vary along the manifold.
  - We do not need to know the number of manifolds.
  - Different manifolds can share common structure.
- But we cannot create the right energy valleys by direct interactions between pixels.
  - So learn a multilayer non-linear mapping between the data and a high-dimensional latent space in which we can construct the right valleys.

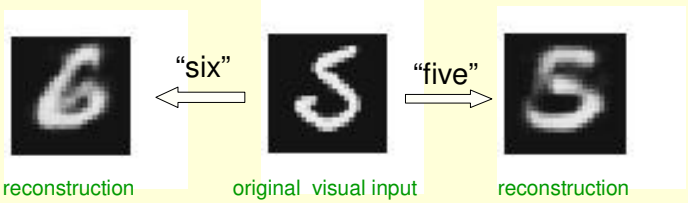
Learning with realistic labels



This network treats the labels in a special way, but they could easily be replaced by an auditory pathway.

Learning with auditory labels

- Alex Kaganov replaced the class labels by binarized cepstral spectrograms of many different male speakers saying digits.
- The auditory pathway then had multiple layers, just like the visual pathway. The auditory and visual inputs shared the top level layer.
- After learning, he showed it a visually ambiguous digit and then reconstructed the visual input from the representation that the top-level associative memory had settled on after 10 iterations.



Some problems with backpropagation (again!)

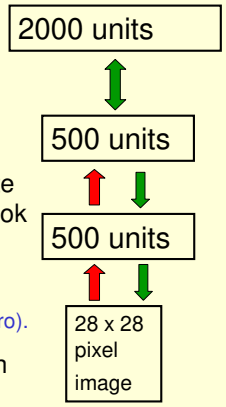
- The amount of information that each training case provides about the weights is at most the log of the number of possible output labels.
  - So to train a big net we need lots of labeled data.
- In nets with many layers of weights the backpropagated derivatives either grow or shrink multiplicatively at each layer.
  - Learning is tricky either way.
- Dumb gradient descent is not a good way to perform a global search for a good region of a very large very non-linear space.
  - So deep nets trained by backpropagation are rare in practice.

The obvious solution to all of these problems: Use greedy unsupervised learning to find a sensible set of weights one layer at a time. Then fine-tune with backpropagation

- Greedily learning one layer at a time scales well to really big networks, especially if we have locality in each layer.
- Most of the information in the final weights comes from modeling the distribution of input vectors.
  - The precious information in the labels is only used for the final fine-tuning.
- We do not start backpropagation until we already have sensible weights that already do well at the task.
  - So the learning is well-behaved and quite fast.

Modelling the distribution of digit images

The top two layers form a restricted Boltzmann machine whose free energy landscape should model the low dimensional manifolds of the digits.



The network learns a density model for unlabeled digit images. When we generate from the model we often get things that look like real digits of all classes.

More hidden layers make the generated fantasies look better (YW Teh, Simon Osindero).

But do the hidden features really help with digit discrimination? Add 10 softmaxed units to the top and do backprop.

Results on permutation-invariant MNIST task

- Very carefully trained backprop net with one or two hidden layers (Platt; Hinton) 1.6%
- SVM (Decoste & Schoelkopf) 1.4%

- Generative model of joint density of images and labels (with unsupervised fine-tuning) 1.25%
- Generative model of unlabelled digits followed by gentle backpropagtion 1.15%
- Generative model of joint density followed by gentle backpropagation 1.1%