CSC 2535: Computation in Neural Networks

Lecture 9 (extra material)

Geoffrey Hinton

Reminder: Why greedy learning works

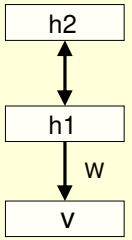
The weights, W, in the bottom level RBM define p(v|h1) and they also, indirectly, define p(h1).

So we can express the RBM model as

$$p(v) = \sum_h p(h1)\ p(v \mid h1)$$

If we leave p(v|h1) alone and build a better model of p(h1), we will improve p(v).

We need a better model of the posterior hidden vectors produced by applying W to the data.



Compositions of Experts

- In mixtures, we add probability distributions produced by different "experts".
- In products of experts we multiply the distributions together.
- In compositions of experts, each expert converts one distribution into another and we compose these transformations to extract representations of data.
  - We want to transform the data distribution into a distribution that is easier to model using an RBM.
  - This is what each RBM does because the distribution of p(h|data) is closer to the equilibrium distribution of the RBM than the data distribution (assuming we have equal numbers of hidden and visible units).

How to model real-valued data

- We need a way to model real valued data using binary stochastic hidden units. Consider the following energy function:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\,\varepsilon\,pixels} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j\,\varepsilon\,features} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j\, w_{ij}$$

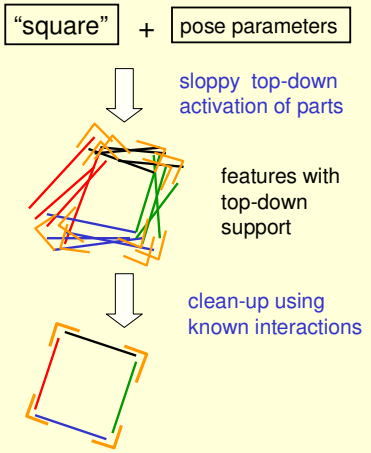
- The Gaussian around each visible unit stops it going to infinity.
- Alternating Gibbs sampling works as before for the hiddens. For the visibles we just compute their mean using the top down input and the bias, and then we add Gaussian noise with the right variance.

A 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.

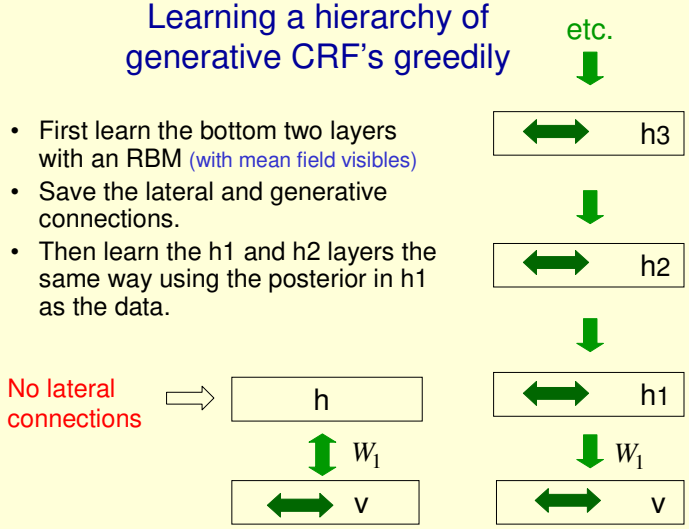
Generating the parts of an object

- One way to maintain the constraints between the parts is to generate each part very accurately
  - But this would require a lot of communication bandwidth.
- Sloppy top-down specification of the parts is less demanding
  - but it messes up relationships between features
  - so use redundant features and use lateral interactions to clean up the mess.
- Each transformed feature helps to locate the others
  - This allows a noisy channel



Learning a hierarchy of generative CRF's greedily

- First learn the bottom two layers with an RBM (with mean field visibles)
- Save the lateral and generative connections.
- Then learn the h1 and h2 layers the same way using the posterior in h1 as the data.



Some problems with backpropagation

- 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.

Results on permutation-invariant MNIST task

- Very carefully trained backprop net with one or two hidden layers (Platt; Hinton) 1.5%
- 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 1.1%
- Generative model of joint density followed by gentle backpropagation 1.0%

THE END