CIAR Second Summer School Tutorial
Lecture 2b

Autoencoders
&
Modeling time series with Boltzmann machines

Geoffrey Hinton

Deep Autoencoders
(Hinton & Salakhutdinov, Science 2006)

- Autoencoders always looked like a really nice way to do non-linear dimensionality reduction:
  - They provide mappings both ways
  - The learning time and memory both scale linearly with the number of training cases.
  - The final model is compact and fast.
- But it turned out to be very very difficult to optimize deep autoencoders using backprop.
  - We now have a much better way to optimize them.

A toy experiment

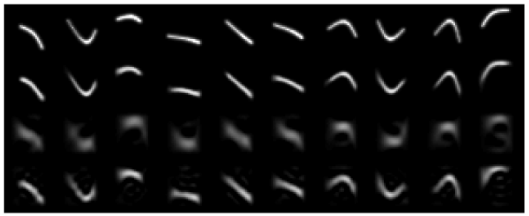
- Generate 100,000 images that have 784 pixels but only 6 degrees of freedom.
  - Choose 3 *x* coordinates and 3 *y* coordinates
  - Fit a spline
  - Render the spline using logistic ink so that it looks like a simple MNIST digit.
- Then use a deep autoencoder to try to recover the 6 dimensional manifold from the pixels.

The deep autoencoder

784 → 400 →200 →100 → 50 → 25 → 6 linear units → 25 ← 50 ← 100 ← 200 ← 400 ← 784

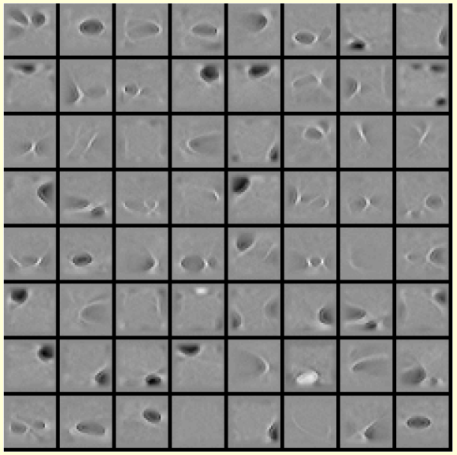
If you start with small random weights it will not learn. If you break symmetry randomly by using bigger weights, it will not find a good solution.

Reconstructions



	squared error
Data	0.0
Auto:6	1.5
PCA:6	10.3
PCA:30	3.9

Some receptive fields of the first hidden layer



An autoencoder for patches of real faces

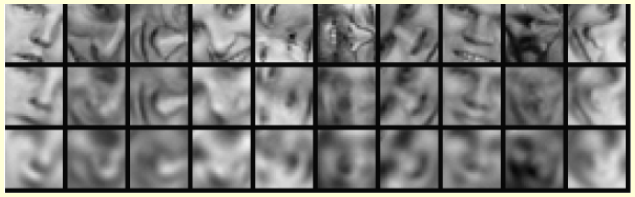
- 625→2000→1000→641→30 and back out again

linear logistic units linear

Train on 100,000 denormalized face patches from 300 images of 30 people. Use 100 epochs of CD at each layer followed by backprop through the unfolded autoencoder.

Test on face patches from 100 images of 10 new people.

Reconstructions of face patches from new people



Data

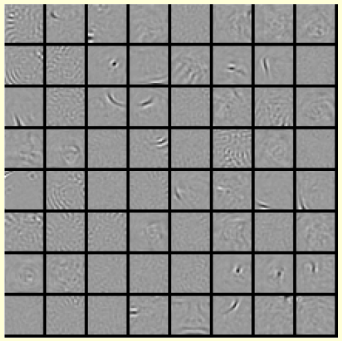
Auto:30 126

PCA:30 135

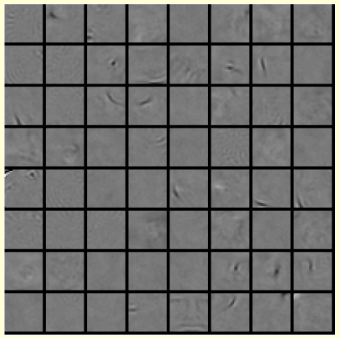
Fantasies from a full covariance Gaussian fitted to the posterior in the 30-D linear code layer

64 hidden units in the first hidden layer

Filters

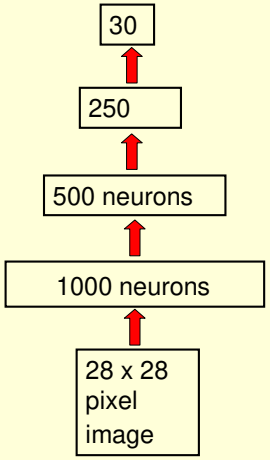


Basis functions

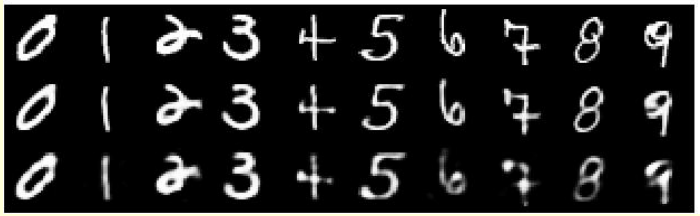


Another test of the learning algorithm

- Train an autoencoder with 4 hidden layers on the 60,000 MNIST digits
  - The training is entirely unsupervised..
- How well can it reconstruct?



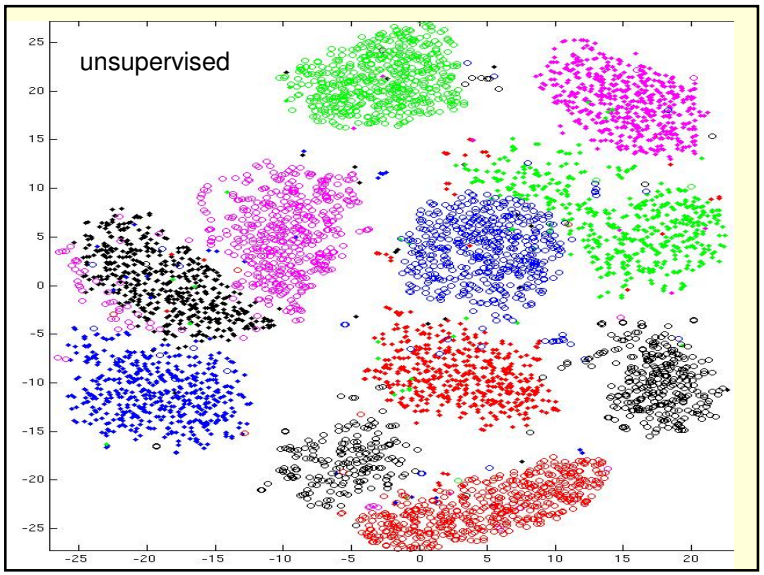
Reconstructions from 30-dimensional codes



- Top row is the data
- Second row is the 30-D autoencoder
- Third row is 30-D logistic PCA which works much better than standard PCA

Do the 30-D codes found by the autoencoder preserve the class structure of the data?

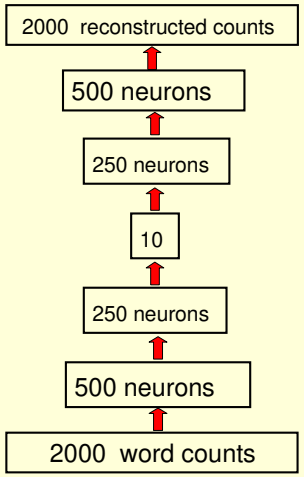
- Take the activity patterns in the top layer and display them in 2-D using a new form of non-linear multidimensional scaling.
- Will the learning find the natural classes?



A final example: Document retrieval

- We can use an autoencoder to find low-dimensional codes for documents that allow fast and accurate retrieval of similar documents from a large set.
- We start by converting each document into a "bag of words". This a 2000 dimensional vector that contains the counts for each of the 2000 commonest words.

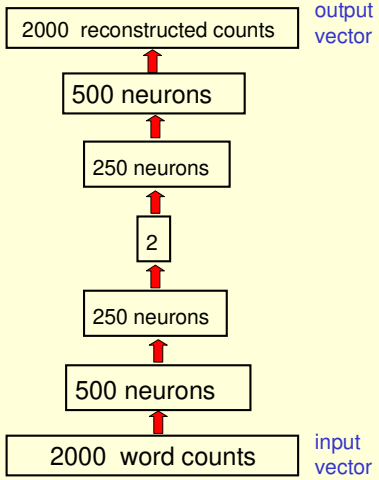
How to compress the count vector



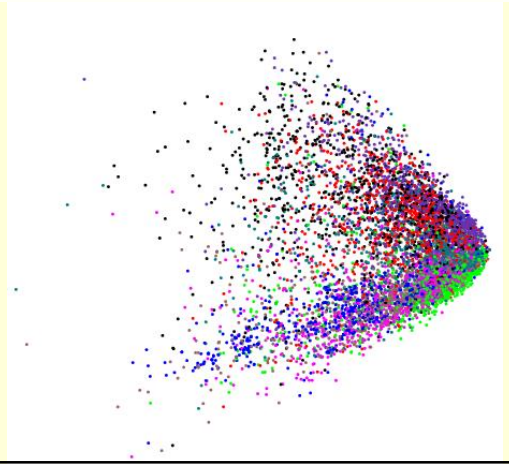
- We train the neural network to reproduce its input vector as its output
- This forces it to compress as much information as possible into the 10 numbers in the central bottleneck.
- These 10 numbers are then a good way to compare documents.
  - See Ruslan Salakhutdinov's talk

Using autoencoders to visualize documents

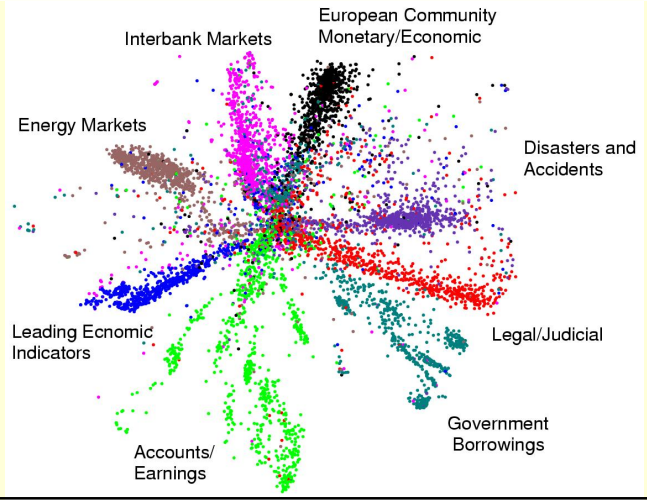
- Instead of using codes to retrieve documents, we can use 2-D codes to visualize sets of documents.
  - This works much better than 2-D PCA



First compress all documents to 2 numbers using a type of PCA
Then use different colors for different document categories



First compress all documents to 2 numbers with an autoencoder
Then use different colors for different document categories

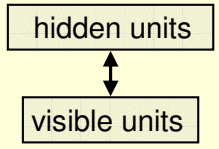


A really fast way to find similar documents

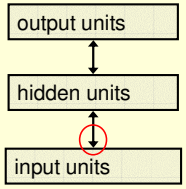
- Suppose we could convert each document into a binary feature vector in such a way that similar documents have similar feature vectors.
  - This creates a "semantic" address space that allows us to use the memory bus for retrieval.
- Given a query document we first use the autoencoder to compute its binary address.
  - Then we fetch all the documents from addresses that are within a small radius in hamming space.
  - This takes constant time. No comparisons are required for getting the shortlist of semantically similar documents.

Conditional Boltzmann Machines (1985)

- Standard BM: The hidden units are not clamped in either phase.
- The visible units are clamped in the positive phase and unclamped in the negative phase. The BM learns p(visible).

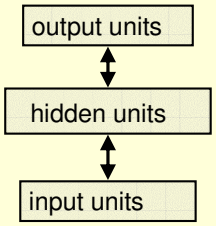


- Conditional BM: The visible units are divided into "input" units that are clamped in both phases and "output" units that are only clamped in the positive phase.
  - Because the input units are always clamped, the BM does not try to model their distribution. It learns p(output | input).

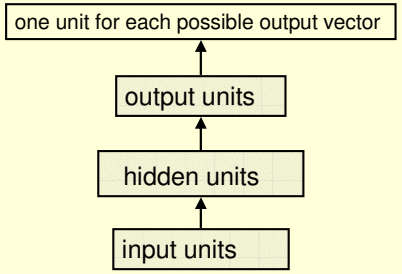


What can conditional Boltzmann machines do that backpropagation cannot do?

- If we put connections between the output units, the BM can learn that the output patterns have structure and it can use this structure to avoid giving silly answers.

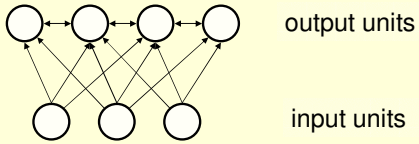


- To do this with backprop we need to consider all possible answers and this could be exponential.



Conditional BM's without hidden units

- These are still interesting if the output vectors have interesting structure.
  - The inference in the negative phase is non-trivial because there are connections between unclamped units.



Higher order Boltzmann machines

- The usual energy function is quadratic in the states:

$$E = bias\ terms\ -\sum_{i<j} s_i s_j\ w_{ij}$$

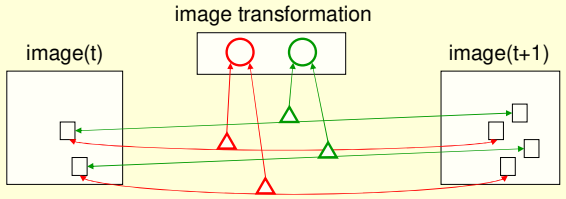
- But we could use higher order interactions:

$$E = bias\ terms\ -\sum_{i<j<k} s_i s_j s_k\ w_{ijk}$$

- Unit k acts as a switch. When unit k is on, it switches in the pairwise interaction between unit i and unit j.
  - Units i and j can also be viewed as switches that control the pairwise interactions between j and k or between i and k.

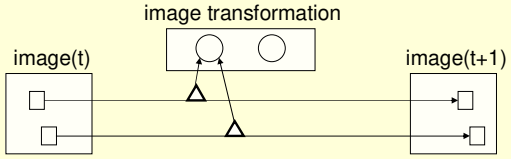
Using higher order Boltzmann machines to model transformations between images.

- A global transformation specifies which pixel goes to which other pixel.
- Conversely, each pair of similar intensity pixels, one in each image, votes for a particular global transformation.



Higher order conditional Boltzmann machines

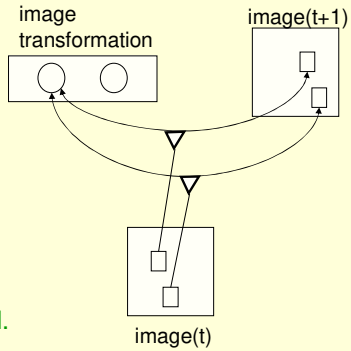
- Instead of modeling the density of image pairs, we could model the conditional density $p(\text{image(t+1)} \mid \text{image(t)})$



- See the talk by Roland Memisevic

Another picture of a conditional, higher-order Boltzmann machine

- We can view it as a Boltzmann machine in which the inputs create interactions between the other variables.
  - This type of model is sometimes called a conditional random field.



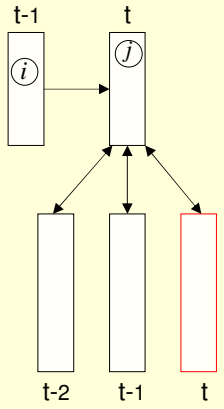
Time series models

- Inference is difficult in directed models of time series if we use distributed representations in the hidden units.
  - So people tend to avoid distributed representations (e.g. HMM's)
- If we really need distributed representations (which we nearly always do), we can make inference much simpler by using three tricks:
  - Use an RBM for the interactions between hidden and visible variables.
  - Include temporal information in each time-slice by concatenating several frames into one visible vector.
  - Treat the hidden variables in the previous time slice as additional fixed inputs.

The conditional RBM model

- Given the data and the previous hidden state, the hidden units at time t are conditionally independent.
  - So online inference is very easy.
- Learning can be done by using contrastive divergence.
  - Reconstruct the data at time t from the inferred states of the hidden units.
  - The temporal connections between hiddens can be learned as if they were additional biases

$$\Delta w_{ij} = s_i \left(< s_j >_{data} - < s_j >_{recon} \right)$$

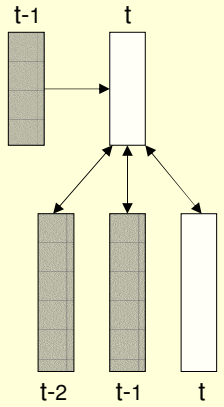


A three stage training procedure (Taylor, Hinton and Roweis)

- First learn a static model of pairs or triples of time frames ignoring the directed temporal connections between hidden units.
- Then use the inferred hidden states to train a "fully observed" sigmoid belief net that captures the temporal structure of the hidden states.
- Finally, use the conditional RBM model to fine tune all of the weights.

Generating from a learned model

- Keep the previous hidden and visible states fixed
  - They provide a time-dependent bias for the hidden units.
- Perform alternating Gibbs sampling for a few iterations between the hidden units and the current visible units.
  - This picks new hidden and visible states that are compatible with each other and with the recent history.



Comparison with hidden Markov models

- Our inference procedure is incorrect because it ignores the future.
- Our learning procedure is slightly wrong because the inference is wrong and also because we use contrastive divergence.
- But the model is exponentially more powerful than an HMM because it uses distributed representations.
  - Given N hidden units, it can use N bits of information to constrain the future.
  - An HMM can only use log N bits of history.
  - This is a huge difference if the data has any kind of componential structure. It means we need far fewer parameters than an HMM, so training is actually easier, even though we do not have an exact maximum likelihood algorithm.

An application to modeling motion capture data

- Human motion can be captured by placing reflective markers on the joints and then using lots of infrared cameras to track the 3-D positions of the markers.
- Given a skeletal model, the 3-D positions of the markers can be converted into the joint angles plus 6 parameters that describe the 3-D position and the roll, pitch and yaw of the pelvis.
  - We only represent changes in yaw because physics doesn't care about its value and we want to avoid circular variables.

Modeling multiple types of motion

- We can easily learn to model walking and running in a single model.
- Because we can do online inference (slightly incorrectly), we can fill in missing markers in real time.
- If we supply some static skeletal and identity parameters, we should be able to use the same generative model for lots of different people.

Show the movies