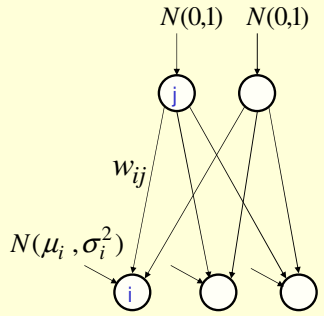
CSC2515:

Lecture 7 (post)
Independent Components Analysis, and Autoencoders

Geoffrey Hinton

Factor Analysis

- The generative model for factor analysis assumes that the data was produced in three stages:
  - Pick values independently for some hidden factors that have Gaussian priors
  - Linearly combine the factors using a factor loading matrix. Use more linear combinations than factors.
  - Add Gaussian noise that is different for each input.



A degeneracy in Factor Analysis

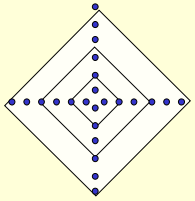
- We can always make an equivalent model by applying a rotation to the factors and then applying the inverse rotation to the factor loading matrix.
  - The data does not prefer any particular orientation of the factors.
- This is a problem if we want to discover the true causal factors.
  - Psychologists wanted to use scores on intelligence tests to find the independent factors of intelligence.

What structure does FA capture?

- Factor analysis only captures pairwise correlations between components of the data.
  - It only depends on the covariance matrix of the data.
  - It completely ignores higher-order statistics
    - Consider the dataset: 111, 100, 010, 001
    - This has no pairwise correlations but it does have strong third order structure.

Using a non-Gaussian prior

- If the prior distributions on the factors are not Gaussian, some orientations will be better than others
  - It is better to generate the data from factor values that have high probability under the prior.
  - one big value and one small value is more likely than two medium values that have the same sum of squares.



If the prior for each hidden activity is $p(s) \propto e^{-s}$ the iso-probability contours are straight lines at 45 degrees.

$$Laplace: \quad p(2,0) > p(\sqrt{2},\sqrt{2})$$

$$Gauss: \quad p(2,0) = p(\sqrt{2},\sqrt{2})$$

The square, noise-free case

- We eliminate the noise model for each data component, and we use the same number of factors as data components.
- Given the weight matrix, there is now a one-to-one mapping between data vectors and hidden activity vectors.
- To make the data probable we want two things:
  - The hidden activity vectors that correspond to data vectors should have high prior probabilities.
  - The mapping from hidden activities to data vectors should compress the hidden density to get high density in the data space. i.e. the matrix that maps hidden activities to data vectors should have a small determinant. Its inverse should have a big determinant

The ICA density model

Mixing matrix

- Assume the data is obtained by linearly mixing the sources → $\mathbf{x} = \mathbf{A}\mathbf{s}$ ← Source vector
- The filter matrix is the inverse of the mixing matrix. → $\mathbf{s} = \mathbf{W}^{\mathbf{T}}\mathbf{x}, \qquad \mathbf{W}^{\mathbf{T}} = \mathbf{A}^{-\mathbf{1}}$
- The sources have independent non-Gaussian priors. → $p(\mathbf{s}) = \prod_i p_i(s_i)$
- The density of the data is a product of source priors and the determinant of the filter matrix → $p(\mathbf{x}) = \prod_i p_i(\mathbf{w}_i^T \mathbf{x}) \,|\det \mathbf{W}|$

The information maximization view of ICA

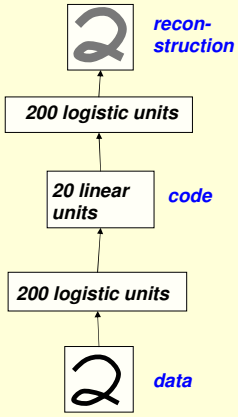
- Filter the data linearly and then applying a non-linear "squashing" function.
- The aim is to maximize the information that the outputs convey about the input.
  - Since the outputs are a deterministic function of the inputs, information is maximized by maximizing the entropy of the output distribution.
    - This involves maximizing the individual entropies of the outputs and minimizing the mutual information between outputs.

Overcomplete ICA

- What if we have more independent sources than data components? (independent \= orthogonal)
  - The data no longer specifies a unique vector of source activities. It specifies a distribution.
    - This also happens if we have sensor noise in square case.
  - The posterior over sources is non-Gaussian because the prior is non-Gaussian.
- So we need to approximate the posterior:
  - MCMC samples
  - MAP (plus Gaussian around MAP?)
  - Variational

Self-supervised backpropagation

- Autoencoders define the desired output to be the same as the input.
  - Trivial to achieve with direct connections
    - The identity is easy to compute!
- It is useful if we can squeeze the information through some kind of bottleneck:
  - If we use a linear network this is very similar to Principal Components Analysis



Self-supervised backprop and PCA

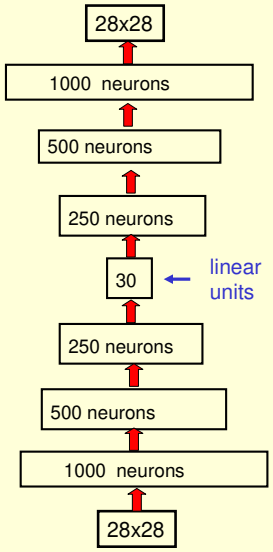
- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared reconstruction error.
- The m hidden units will span the same space as the first m principal components
  - Their weight vectors may not be orthogonal
  - They will tend to have equal variances

Self-supervised backprop in deep autoencoders

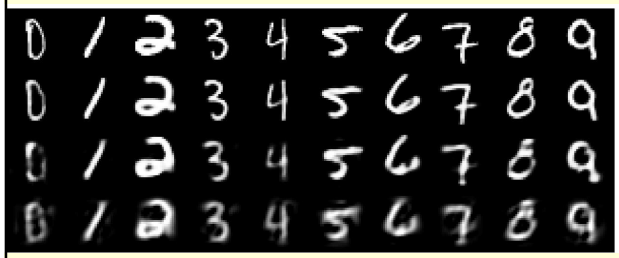
- We can put extra hidden layers between the input and the bottleneck and between the bottleneck and the output.
  - This gives a non-linear generalization of PCA
- It should be very good for non-linear dimensionality reduction.
  - It is very hard to train with backpropagation
  - So deep autoencoders have been a big disappointment.
    - But we recently found a very effective method of training them which will be described next week.

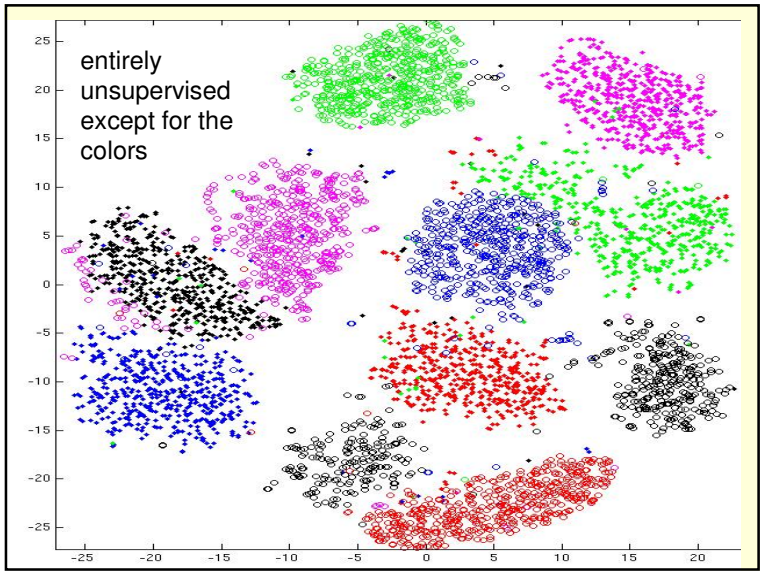
A Deep Autoencoder
(Ruslan Salakhutdinov)

- They always looked like a really nice way to do non-linear dimensionality reduction:
  - But it is very difficult to optimize deep autoencoders using backpropagation.
- We now have a much better way to optimize them.



A comparison of methods for compressing digit images to 30 real numbers.



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

- Take the 30-D activity patterns in the code layer and display them in 2-D using a new form of non-linear multi-dimensional scaling (UNI-SNE)
- Will the learning find the natural classes?