CSC321 Lecture 20: Autoencoders

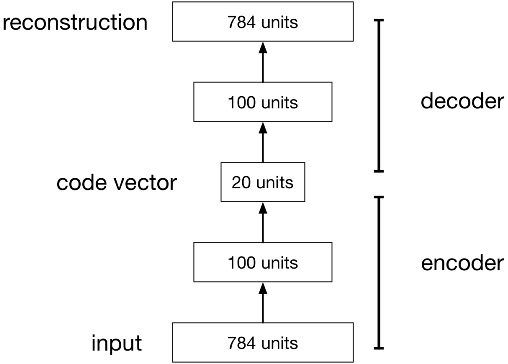
Roger Grosse

Overview

- Latent variable models so far:
  - mixture models
  - Boltzmann machines
- Both of these involve discrete latent variables. Now let's talk about continuous ones.
- One use of continuous latent variables is dimensionality reduction

Autoencoders

- An autoencoder is a feed-forward neural net whose job it is to take an input $\mathbf{x}$ and predict $\mathbf{x}$.
- To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input.



Autoencoders

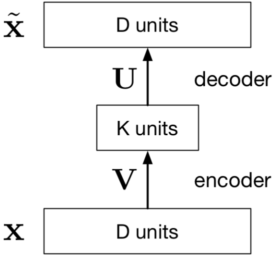
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Compression (i.e. reducing the file size)
  - Note: autoencoders don't do this for free — it requires other ideas as well.
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabled data can be much more plentiful than labeled data

Principal Component Analysis

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$



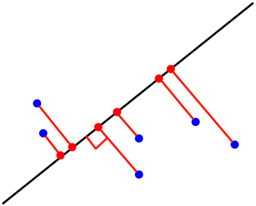
- This network computes $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{V}\mathbf{x}$, which is a linear function.
- If $K \geq D$, we can choose $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{U}\mathbf{V}$ is the identity. This isn't very interesting.
- But suppose $K < D$:
  - $\mathbf{V}$ maps $\mathbf{x}$ to a K-dimensional space, so it's doing dimensionality reduction.
  - The output must lie in a K-dimensional subspace, namely the column space of $\mathbf{U}$.

Principal Component Analysis

- We just saw that a linear autoencoder has to map D-dimensional inputs to a K-dimensional subspace $\mathcal{S}$.
- Knowing this, what is the best possible mapping it can choose?

Principal Component Analysis

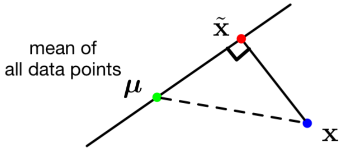
- We just saw that a linear autoencoder has to map D-dimensional inputs to a K-dimensional subspace $\mathcal{S}$.
- Knowing this, what is the best possible mapping it can choose?
  - By definition, the projection of $\mathbf{x}$ onto $\mathcal{S}$ is the point in $\mathcal{S}$ which minimizes the distance to $\mathbf{x}$.



- Fortunately, the linear autoencoder can represent projection onto $\mathcal{S}$: pick $\mathbf{U} = \mathbf{Q}$ and $\mathbf{V} = \mathbf{Q}^\top$, where $\mathbf{Q}$ is an orthonormal basis for $\mathcal{S}$.

Principal Component Analysis

- The autoencoder should learn to choose the subspace which minimizes the squared distance from the data to the projections.
- This is equivalent to the subspace which maximizes the variance of the projections.



By the Pythagorean Theorem,

$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\tilde{\mathbf{x}}^{(i)} - \boldsymbol{\mu}\|^2}_{\text{projected variance}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2}_{\text{reconstruction error}}$$

$$= \underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)} - \boldsymbol{\mu}\|^2}_{\text{constant}}$$

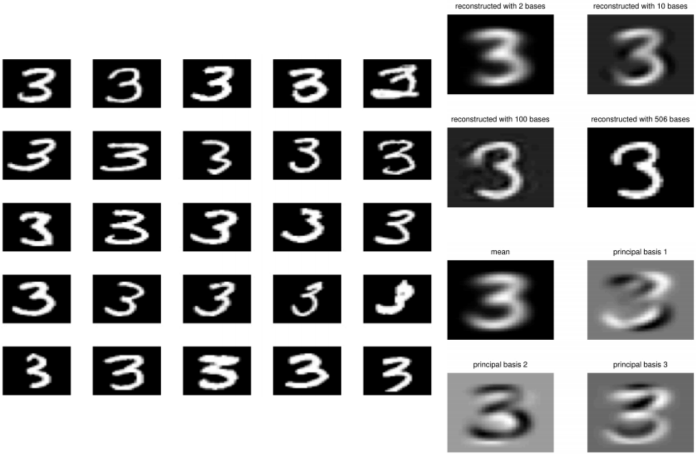
- You wouldn't actually sove this problem by training a neural net. There's a closed-form solution, which you learn about in CSC 411.
- The algorithm is called principal component analysis (PCA).

Principal Component Analysis

PCA for faces ("Eigenfaces")

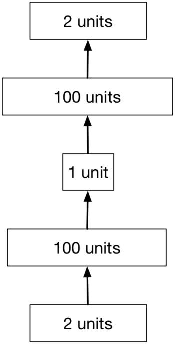
Principal Component Analysis

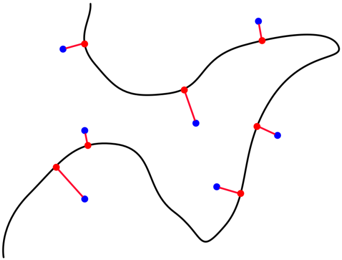
PCA for digits



Deep Autoencoders

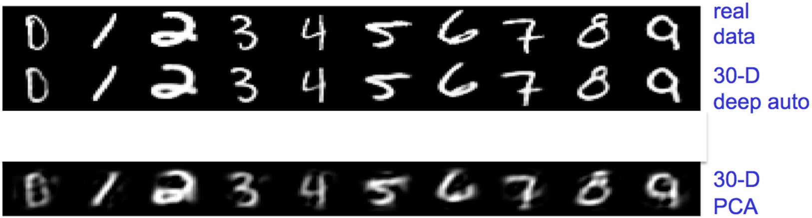
- Deep nonlinear autoencoders learn to project the data, not onto a subspace, but onto a nonlinear manifold
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.





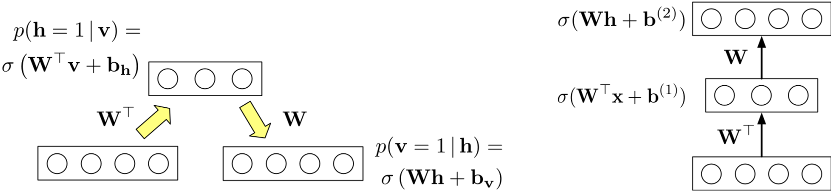
Deep Autoencoders

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)



Layerwise Training

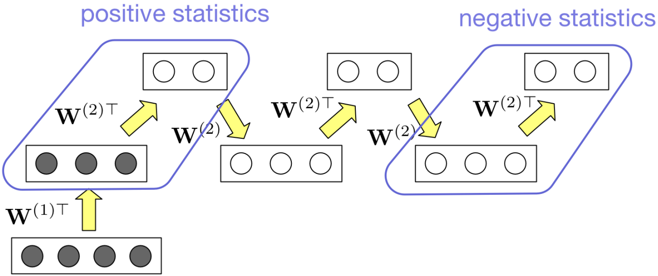
- There's a neat connection between autoencoders and RBMs.



- An RBM is like an autoencoder with tied weights, except that the units are sampled stochastically.

Layerwise Training

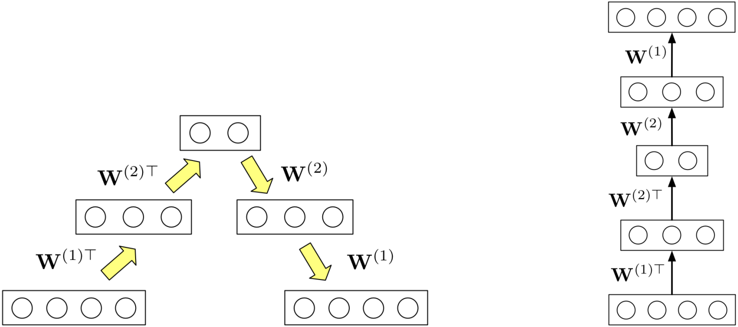
- Suppose we've already trained an RBM with weights $\mathbf{W}^{(1)}$.
- Let's compute its hidden features on the training set, and feed that in as data to another RBM:



- Note that now $\mathbf{W}^{(1)}$ is held fixed, but $\mathbf{W}^{(2)}$ is being trained using contrastive divergence.

Layerwise Training

- A stack of two RBMs can be thought of as an autoencoder with three hidden layers:



- This gives a good initialization for the deep autoencoder. You can then fine-tune the autoencoder weights using backprop.
- This strategy is known as layerwise pre-training.

- Autoencoders are not a probabilistic model.
- However, there is an autoencoder-like probabilistic model called a variational autoencoder (VAE). These are beyond the scope of the course, and require some more advanced math.
- Check out David Duvenaud's excellent course "Differentiable Inference and Generative Models": `https://www.cs.toronto.edu/~duvenaud/courses/csc2541/index.html`

Deep Autoencoders

(Professor Hinton's slides)