COMPSCI 689
Lecture 18: Autoencoders

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

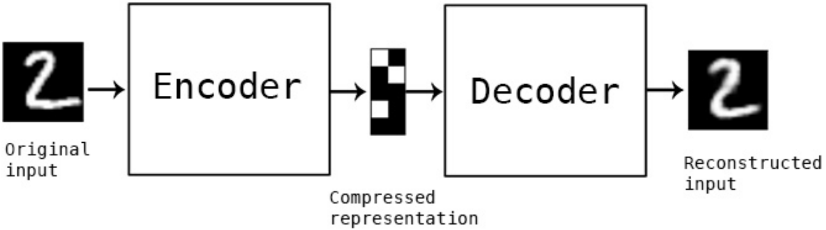
Latent Linear Models

- Latent linear models like factor analysis provide probability densities for linear manifolds in high dimensional spaces.
- Their main limitation is that they can only model linear manifolds.
- Generalized latent linear models can use different latent ad visible distributions, but they are still inherently linear.
- To build models for non-linear manifolds, we can apply basis expansions or look at kernelized latent linear models.
- However, these approaches require us to know the right expansion/kernel to apply.

Autoencoders

- An autoencoder is a deterministic model that consists of two components: an encoder function and a decoder function.
- The encoder function $f(\mathbf{x})$ maps a D-dimensional input vector $\mathbf{x} \in \mathcal{X}$ into a K-dimensional code vector $\mathbf{h} \in \mathcal{H}^K$.
- The decoder function maps the K-dimensional code vector $\mathbf{h} \in \mathcal{H}^K$ back to a D-dimensional reconstruction of the input vector $\mathbf{r} \in \mathcal{X}$.
- The goal is typically to learn to accurately copy $\mathbf{x}$ to $\mathbf{r}$ while constraining $\mathbf{h}$ in some way that forces it to encode salient features of the input while ignoring noise.

Example: Autoencoder for Digits



Linear Autoencoders

- A linear auto-encoder is an autencoder where the encoder and decoder are linear functions.

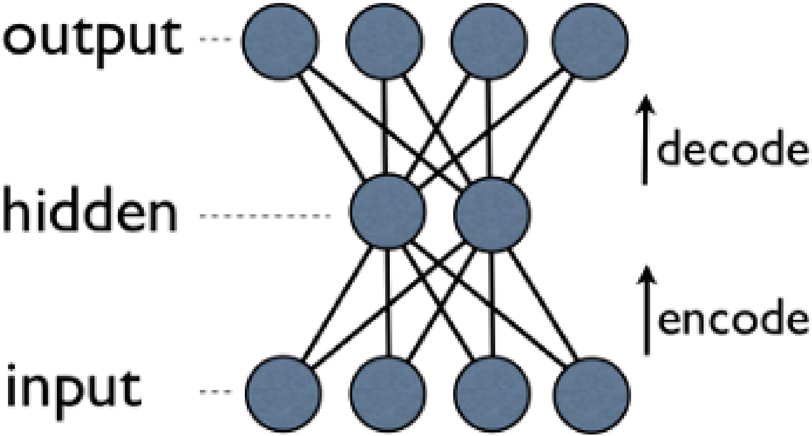
- The encoding and decoding functions are:

$$f(\mathbf{x}) = \mathbf{V}\mathbf{x}$$
$$g(\mathbf{h}) = \mathbf{W}\mathbf{h}$$

- Such a linear encoder-decoder model can be learned by minimizing the MSE between the inputs and the reconstructions, often called the *reconstruction error*.

$$\mathbf{V}^*, \mathbf{W}^* = \underset{\mathbf{V},\mathbf{W}}{\arg\min} \sum_{n=1}^{N} \|\mathbf{x}_n - g(f(\mathbf{x}_n))\|^2$$

Example: Basic Linear Autoencoder Model



Factor Analysis as an Autoencoder

- The factor analysis model can be seen as a particularly complex parameterization of a linear autoencoder trained using maximum likelihood estimation (the equations below assume $\mu = 0$)

- The encoder function is simply the mean of $P(\mathbf{z}|\mathbf{x})$:

$$f(\mathbf{x}) = \mathbb{E}_{P(\mathbf{z}|\mathbf{x})}[\mathbf{z}] = \mathbf{V}\mathbf{x}$$
$$\mathbf{V} = (I + \mathbf{W}^T \Psi \mathbf{W})^{-1} \mathbf{W}^T \Psi^{-1}$$

- The decoder function is the mean of $P(\mathbf{x}|\mathbf{z})$:

$$g(\mathbf{z}) = \mathbb{E}_{P(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mathbf{W}\mathbf{z}$$

PPCA as an Autoencoder

- The Probabilistic Principal Components Analysis (PPCA) model is a special case of Factor Analysis where $\Psi = \sigma^2 I$ and $\mathbf{W}$ is constrained to be an orthonormal matrix.
- The encoder function is the mean of $P(\mathbf{z}|\mathbf{x})$, but the isotropic assumption assumption results in simplifications:

$$f(\mathbf{x}) = \mathbb{E}_{P(\mathbf{z}|\mathbf{x})}[\mathbf{z}] = \mathbf{V}\mathbf{x}$$
$$\mathbf{V} = (\sigma^2 I + \mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T$$

- The decoder function is simply the mean function of $P(\mathbf{x}|\mathbf{z})$:

$$g(\mathbf{z}) = \mathbb{E}_{P(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mathbf{W}\mathbf{z}$$

Classical Principal Components Analysis as an Autoencoder

- In the limit as σ^2 goes to zero, we obtain the classical PCA model. The encoder further simplifies due to orthogonality of $\mathbf{W}$:

$$f(\mathbf{x}) = \mathbb{E}_{P(\mathbf{z}|\mathbf{x})}[\mathbf{z}] = \mathbf{V}\mathbf{x}$$
$$\mathbf{V} = (\sigma^2 I + \mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T \to \mathbf{W}^T$$

- The decoder function is again he mean of $P(\mathbf{x}|\mathbf{z})$:

$$g(\mathbf{z}) = \mathbb{E}_{P(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mathbf{W}\mathbf{z}$$

- This result shows that classical PCA is actually a directly parameterized linear autoencoder. General linear autoencoders do not require orthogonality of $\mathbf{W}$ and $\mathbf{V}$, but they identify exactly the same linear subspace!

Classical Principal Components Analysis

- One of the interesting properties of classical PCA is that you do not need iterative numerical optimization to find the optimal $\mathbf{W}$.
- Let $\mathbf{X}$ be an $N \times D$ data matrix that has been mean centered.
- Define the empirical scatter matrix to be $\mathbf{S} = \mathbf{X}^T\mathbf{X}$.
- Then the optimal $\mathbf{W}$ is given by the leading K eigenvectors of $\mathbf{S}$.
- These eigenvectors are orthogonal by definition and are the rank K maximum variance sub-space of $\mathbf{X}$, also called the *principal sub-space*.
- Interestingly, this estimation approach also identifies the optimal $\mathbf{W}$ for the PPCA model.
- Finally, the rank K singular value decomposition of $\mathbf{X}$ is given by $\mathbf{USV}^T$ where $\mathbf{U}$ is an $N \times K$ orthonormal matrix, $\mathbf{U}$ is a $D \times K$ orthonormal matrix and $\mathbf{S}$ is a diagonal matrix.

Singular Value Decomposition

- The rank K singular value decomposition (SVD) of $\mathbf{X}$ is given by $\mathbf{USV}^T$ where $\mathbf{U}$ is an $N \times K$ orthonormal matrix, $\mathbf{U}$ is a $D \times K$ orthonormal matrix and $\mathbf{S}$ is a positive diagonal matrix.
- The rank K SVD of $\mathbf{X}$ minimizes the objective function $\|\mathbf{X} - \mathbf{USV}^T\|_2^2$ subject to the stated conditions on $\mathbf{U}, \mathbf{S}, \mathbf{V}$.
- The matrix $\mathbf{V}^T$ identified using the SVD is exactly equal to the $\mathbf{W}$ matrix identified using PCA (or PPCA).
- This means that the SVD provides yet another way of identifying the optimal parameters for a linear autoencoder.

Constraints on Linear Autoencoders

- In order for an MSE-minimizing linear autoencoder to learn something useful about the structure of the input data, it is necessary to constrain the code vector in some way.
- If $K = D$ (complete representation) or $K > D$ (overcomplete representation), then we can achieve zero reconstruction error by setting the first D rows of V to the identity matrix and the first K columns of W to the identity matrix (all other entries are zero).
- One way to constrain a linear autoencoder is to require $K < D$ (an undercomplete representation).
- This dimensionality reduction forces the model to extract useful information from the inputs and to discard noise in order to minimize the reconstruction error.

Sparse Overcomplete Autoencoders

- A different way to constrain a linear autoencoder is to allow $K > D$ while constraining the number of non-zero elements of the code vector $\mathbf{h}$.

- *Sparse coding* is a dimensionality reduction model that has a linear decoder and an optimization-based encoder:

$$
\begin{aligned}
f(\mathbf{x}) &= \arg\min_{\mathbf{h}} \quad \|\mathbf{x} - \mathbf{W}\mathbf{h}\|_2^2 + \lambda \|\mathbf{h}\|_1 \\
g(\mathbf{h}) &= \mathbf{W}\mathbf{h}
\end{aligned}
$$

- The parameters $\mathbf{W}$ are learned on a data set using:

$$
\arg\min_{W, \mathbf{h}_1, \ldots, \mathbf{h}_N} \sum_{n=1}^{N} \left(\|\mathbf{x}_n - \mathbf{W}\mathbf{h}_n\|_2^2 + \lambda \|\mathbf{h}_n\|_1 \right)
$$

Sparse Overcomplete Linear Autoencoders

- Sparse coding can achieve sparse code vectors through direct ℓ_1 penalization during encoding, but they need to perform optimization to infer the code can be prohibitive.
- An alternative is to learn a linear autoencoder while penalizing the ℓ_1 norm of the codes it produces to constrain the complexity of the codes the model tends to generate.
- The resulting optimization problem looks like:

$$\mathbf{V}^*, \mathbf{W}^* = \arg\min_{\mathbf{V},\mathbf{W}} \sum_{n=1}^{N} \left(\|\mathbf{x}_n - g(f(\mathbf{x}_n))\|_2^2 + \lambda \|f(\mathbf{x}_n)\|_1 \right)$$

Linear Denoising Autoencoders

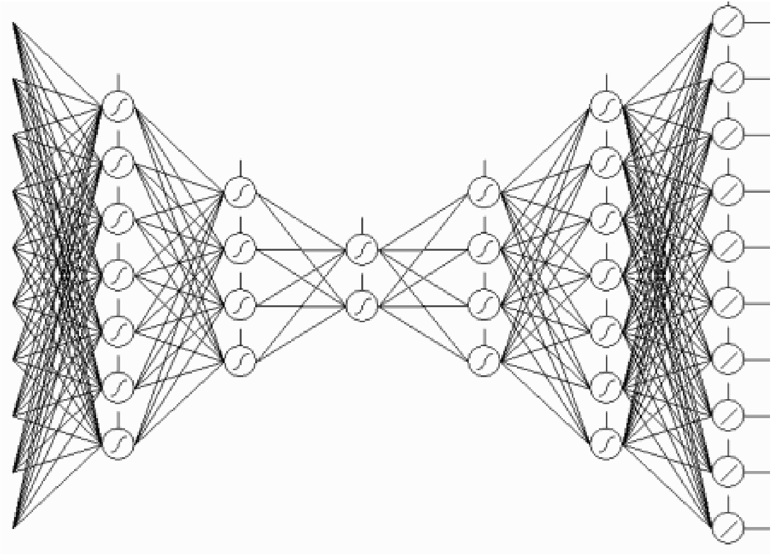
- Yet another way to stop an overcomplete autoencoder from simply learning the identity function is to make the problem that the autoencoder has to solve more difficult.
- A *denoising autoencoder* purposely corrupts the input $\mathbf{x}$ using a sample drawn from a stochastic noise process $q(\mathbf{x}'|\mathbf{x})$.
- It then provides $\mathbf{x}'$ as the input to the autoender while requiring the reconstruction that the model produced match the original $\mathbf{x}$.
- The model can be trained as shown below. The characteristics of the noise process are hyper-parameters of model.

$$\mathbf{V}^*, \mathbf{W}^* = \underset{\mathbf{V},\mathbf{W}}{\arg\min} \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{x}'|\mathbf{x}_n)} \left[||\mathbf{x}_n - g(f(\mathbf{x}'))||_2^2 \right]$$

Non-Linear Autoencoders

- All of the above models and training criteria can only accurately represent data defined on linear manifolds.
- To make an autoencoder non-linear, it suffices to make the encoder and decoder networks non-linear.
- However, the capacity of non-linear autoencoders is no longer limited by the length of the code vector.
- In fact, it is possible to construct a deep network where the code is 1-dimensional, but the network achieves zero reconstruction loss.

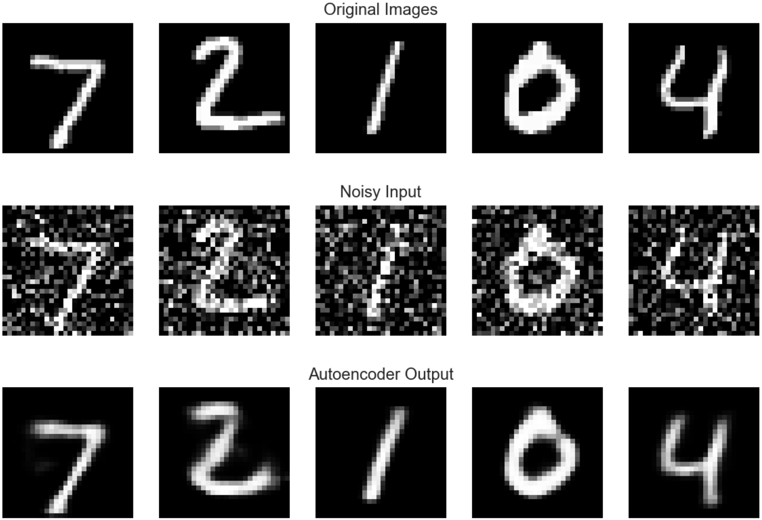
Example: Deep Autoencoders



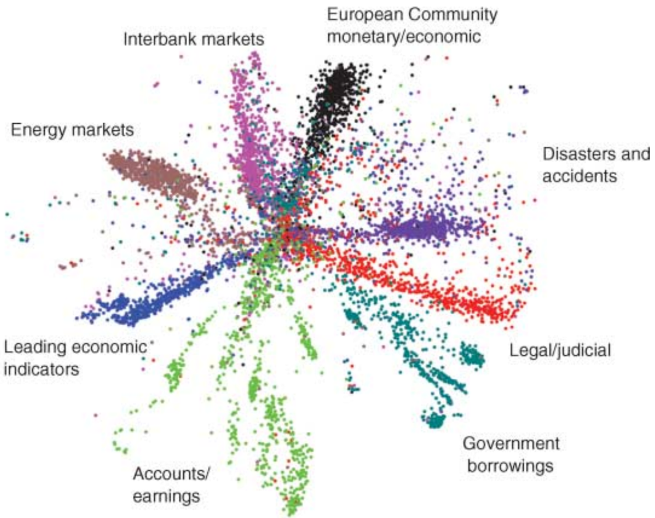
Constraining Non-Linear Autoencoders

- Like with overcomplete linear autoencoders, nonlinear autoencoders need to have constraints to force them to extract useful information from the data.
- Possible constraints include constraining the depth of the encoder and decoder networks as well as the length of the code vectors, using sparse codes, and using the denoising principle.
- As with supervised learning, it is known that some data distributions can be much more efficiently represented with deep, non-linear autoencoders instead of shallow non-linear or linear autoencoders.
- The key is determining the network architecture that gives the best performance for a given down-stream task.

Example: Deep Denoising Autoencoder for Images



Example: Deep Autoencoder Embedding for Documents



Autoencoder Applications

- Autoencoders have many applications including feature extraction, image and signal denoising, and semantic hashing for information retrieval.
- Unsupervised feature extraction (e.g., representation learning) is one of their most common applications and involves learning an encoder-decoder pair on a large unlabeled data set.
- The basic autoencoder architecture can also be modified for semi-supervised learning by including and encoder, a decoder, and a classifier that uses the code vector as its input.

Example: Semi-Supervised Autoencoder/Classifier