COMP 790-125: Goals for today

We are going to talk about AutoEncoders.

- Linear Factor Models
- Encoding and Decoding
- Linear Autoencoders and PCA
- Nonlinear Autoencoders
- Sparse Autoencoders
- Denoising Autoencoders
- Contractive Autoencoders

PCA: Rotation matrix

In 2D

\[ R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \]

for example

\[ R \left( -\frac{\pi}{4} \right) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \]

of course this generalizes to higher dimensions.

Relevant observations to us are

- \( R^{-1} = R^T \) (orthogonal matrix)
- \(|R| = 1\) (no stretching)
PCA: Covariances and rotation matrices

Suppose we have two independent Gaussian distributions

\[
p(x) = \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{x^2}{2} \right\}
\]

\[
p(y) = \frac{1}{\sqrt{2\pi}9} \exp \left\{ -\frac{y^2}{18} \right\}
\]

\[
p(x, y) = \frac{1}{2\pi\sqrt{9}} \exp \left\{ -\frac{1}{2} \left( x^2 + \frac{y^2}{9} \right) \right\}
\]

or in matrix form

\[
p(x, y) = \frac{1}{2\pi\sqrt{|D|}} \exp \left\{ -(1/2) \begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} \right\}
\]

where

\[
D = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}
\]
Suppose I sample \[\begin{bmatrix} x \\ y \end{bmatrix}\] again independently from the same distribution. But then I use a rotation matrix to produce a different sample

\[
\begin{bmatrix} x^1 \\ y^1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

and we plot \((x^1, y^1)\)
Covariances and rotation

We performed a change of variables\(^1\) so what is

\[ p(x^1, y^1) \]

mapping back to original coordinate system is done by inverse

\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix} = A^{-1} \begin{bmatrix}
    x^1 \\
    y^1
\end{bmatrix}
\]

and we can write

\[
p(x^1, y^1) \propto \exp \left\{ -\frac{1}{2} \begin{bmatrix}
    x^1 \\
    y^1
\end{bmatrix} (ADA^T)^{-1} \begin{bmatrix}
    x^1 \\
    y^1
\end{bmatrix} \right\}
\]

and after computing normalization constant

\[
p(x^1, y^1) = \frac{1}{2\pi \sqrt{|ADA^T|}} \exp \left\{ -\frac{1}{2} \begin{bmatrix}
    x^1 \\
    y^1
\end{bmatrix}^T (ADA^T)^{-1} \begin{bmatrix}
    x^1 \\
    y^1
\end{bmatrix} \right\}
\]

\(^1\)if you remember Jacobians from your probability class, it is a multiplicative constant that will fall out of \(\int_{x^1, y^1} p(x^1, y^1) dx dy = 1\)
A Gaussian distribution

\[ p(x|\mu, \Sigma) \propto \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

can be thought of in terms of some underlying Gaussian with diagonal covariance and a rotation matrix \( A \)

\[ p(z|d) \propto \prod \exp \left\{ -\frac{z_i^2}{2d_i} \right\} \]

where

\[ x = Az + \mu \]
Suppose someone gives us a covariance matrix $\Sigma$.

Can we get rotation matrix $A$ and variances $d$?

Of course: we perform eigen-decomposition of the matrix (eig in matlab)

$$\Sigma = VDV^T$$

where $V$ contains eigenvectors of $\Sigma$ and $D$ is a diagonal matrix of eigenvalues of $\Sigma$.

And hence our $A$ is a matrix of eigenvectors of $\Sigma$ and $d$ are eigenvalues of $\Sigma$. 
Principal components

Now that we have a list of variances $d$ we can say which of the $z$ coordinates are the most variable.

Consequently we can translate observations about the most variable $z$ coordinates into observations about the most variable directions in the space of the original data $x$.

You order components based on their variability: the first component is the most variable.
In the probabilistic PCA model, we assume that

\[ z \sim \mathcal{N}(0, I_p) \]
\[ x \sim \mathcal{N}(Az, \sigma I_d) \]

where \( A \) is an orthogonal matrix.

In factor analysis, we make the same assumptions but \( A \) need not be orthogonal.
One well known application of PCA is Eigen-faces method. Each eigen-face is a principal component:

\[
\begin{align*}
\approx & 0.45^* z_1 \\
& -0.2^* z_2 \\
& +0.15^* z_3 \\
& -0.12^* z_4
\end{align*}
\]
Encoding and decoding

Encoding is the process of converting data to a code

\[ z = \text{enc}(x) \]

Decoding is the process of converting code into data.

\[ x = \text{dec}(z) \]

An objective for training encoders and decoders

\[ \min_{\text{enc,dec}} \| X - \text{dec}(\text{enc}(X)) \|_F^2 \]
In PCA:

- encoding
  \[ \text{enc}(z) = xA \]

- decoding
  \[ \text{dec}(x) = z^T A^T \]

An the objective is:

\[ \min_{A} \left\| X - XAA^T \right\|_F^2 \]
Auto-encoders consist of a coupled pair of encoders and decoders aimed at minimizing a loss in reconstruction of the data.

\[
\text{minimize } \text{loss}(X, \text{dec}(\text{enc}(X)))
\]

You can use different losses/reconstruction error

- mean square error
- mean absolute error
- cross entropy

You can use different decoders and encoders.

- Linear decoder and encoder – PCA
- Linear decoder, sparsity promoting encoder – sparse dictionary learning
- Neural net decoder and encoder
Interpolating codes$^2$

Figure 4: Top rows: Interpolation between a series of 9 random points in $Z$ show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

Consider encoder and decoder

\[ \text{enc}, \text{dec} \in \mathbb{R}^n \rightarrow \mathbb{R}^n \]

Hence the encoder maps a vector of length \( n \) into a vector of length \( n \).

Q: Give an example of mappings \text{enc} and \text{dec} that achieve perfect reconstruction?
Consider encoder and decoder

\[ \text{enc}, \text{dec} \in \mathbb{R}^n \rightarrow \mathbb{R}^n \]

Hence the encoder maps a vector of length \( n \) into a vector of length \( n \).

Q: Give an example of mappings \text{enc} and \text{dec} that achieve perfect reconstruction?
In order to avoid trivial solutions we can require

\[ \text{enc} : \mathbb{R}^n \rightarrow \mathbb{R}^k, \text{dec} : \mathbb{R}^k \rightarrow \mathbb{R}^n \]

where \( k < n \).

Note that in PCA \( k \) is the number of principal components.

If we choose to use a code that is longer than the data \( k > n \) – **overcomplete** code – we need to introduce regularization.
Training overcomplete auto-encoders

- Sparsity in the code

$$\text{minimize } \text{loss}(\text{data, dec(} \text{enc(X)}) + \lambda \| \text{enc(X)} \|_1$$

- Change loss by introducing noise in the input

$$\text{minimize } \text{loss}(\text{data, dec(} \text{enc(X + noise)})$$

- Encourage smooth changes in codes

$$\text{minimize } \text{loss}(\text{data, dec(} \text{enc(X)}) + \lambda \sum_t \| \nabla \text{enc(x)} \|_2^2$$
Sparse auto-encoders

\[
\text{minimize}_{\text{enc, dec}} \text{loss}(X, \underbrace{\text{dec}(\text{enc}(X))}_{\text{reconstructed data}}) + \lambda \|\text{enc}(X)\|_1
\]

Even if the code is overcomplete \(\text{enc}(X) : \mathbb{R}^n \rightarrow \mathbb{R}^k\) by requiring it to be sparse – most of entries of the code are zero – we reduce dimensionality of the code.

Note that Sparse PCA is not an example a sparse autoencoder

\[
\text{minimize}_{A,B} \left\| X - XB^T A \right\| + \lambda_1 \sum_{k=1}^{K} \|\beta_k\|_1 + \lambda_2 \sum_k \|\beta_k\|^2
\]

subject to \(AA^T = I\)

where \(B = [\beta_1, \beta_2, ... \beta_k]\)

Q: What is the difference between the Sparse PCA and a Sparse linear auto-encoder? Hard but important question.
Denoising Auto-encoders (DAE)

\[
\text{minimize } \text{loss}(\text{data}, \text{dec(enc(}X + \text{noise})) )
\]

Q: How well would identity mapping for encoder and decoder be?
A one dimensional manifold constructed by vertically translated image of 9 was encoded and code projected to 2D space.

At each point on the manifold we can compute a tangent – direction to move in order to stay on the line.

Tangent is visualized – it shows the difference between two translated version of a 9.
Contractive autoencoders

$$\text{minimize} \quad \text{loss}(\underbrace{X}_{\text{data}}, \underbrace{\text{dec}(\text{enc}(X))}_{\text{reconstructed data}}) + \lambda \sum_t \left\| \nabla \text{enc}(x^t) \right\|^2$$

Figure 14.10: Illustration of tangent vectors of the manifold estimated by local PCA and by a contractive autoencoder. The location on the manifold is defined by the input image of a dog drawn from the CIFAR-10 dataset. The tangent vectors are estimated by the leading singular vectors of the Jacobian matrix $\frac{\partial h}{\partial x}$ of the input-to-code mapping. Although both local PCA and the CAE can capture local tangents, the CAE is able to form more accurate estimates from limited training data because it exploits parameter sharing across different estimates that share a subset of active hidden units. The CAE tangent directions typically correspond to moving or changing parts of the object (such as the head or legs). Images reproduced with permission from Rifai et al. (2011c).
We will see more powerful and interesting architectures soon. However, if you are training an autoencoder

- work with denoising autoencoders, but don’t overwhelm the data with noise
- adding small amount of noise to binary variables will not necessarily mess-up the training
- you can stack DAEs – train an autoencoder on code – to obtain a deep network
- ReLU autoencoders do work
- you can train a deep ReLU autoencoder from scratch – no need for stacking
- interpretation is easy for one layer autoencoders, hard for deeper ones
COMP 790-125: Goals for today

We are going to talk about AutoEncoders.

- Linear Factor Models
- Encoding and Decoding
- Linear Autoencoders and PCA
- Nonlinear Autoencoders
- Sparse Autoencoders
- Denoising Autoencoders
- Contractive Autoencoders