Dimensionality Reduction and Principle Components Analysis
Outline

• What is dimensionality reduction?
• Principle Components Analysis (PCA)
  – Example (Bishop, ch 12)
  – PCA vs linear regression
  – PCA as a mixture model variant
  – Implementing PCA
• Other matrix factorization methods
  – Applications (collaborative filtering)
  – MF with SGD
  – MF vs clustering vs LDA vs ....
A DIMENSIONALITY REDUCTION METHOD YOU ALREADY KNOW....
Outline

• What’s new in ANNs in the last 5-10 years?
  – **Deeper networks**, more data, and faster training
    • Scalability and use of GPUs ✔
    • Symbolic differentiation ✔
    • Some subtle changes to cost function, architectures, optimization methods ✔

• What types of ANNs are most successful and why?
  – Convolutional networks (CNNs) ✔
  – Long term/short term memory networks (LSTM) ✔
  – Word2vec and embeddings ✔
  – **Autoencoders**

• What are the hot research topics for deep learning?
Neural network auto-encoding

• Assume we would like to learn the following (trivial?) output function:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000001</td>
<td>00000001</td>
</tr>
<tr>
<td>00000010</td>
<td>00000010</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
</tbody>
</table>

• One approach is to use this network

[Image of neural network diagram]
Neural network auto-encoding

Maybe learning something like this:

<table>
<thead>
<tr>
<th>Input</th>
<th>Hidden Values</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000 → .89 .04 .08 → 10000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01000000 → .01 .11 .88 → 01000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00100000 → .01 .97 .27 → 00100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00010000 → .99 .97 .71 → 00010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001000 → .03 .05 .02 → 00001000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000100 → .22 .99 .99 → 00000100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000010 → .80 .01 .98 → 00000010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000001 → .60 .94 .01 → 00000001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000001</td>
<td>00000001</td>
</tr>
<tr>
<td>00000010</td>
<td>00000010</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
</tbody>
</table>
Neural network autoencoding

• The hidden layer is a *compressed version* of the data
• If training error is zero then the training data is perfectly reconstructed
  – It probably won’t be unless you overfit
• Other similar examples will be imperfectly reconstructed
• High reconstruction error usually means the example is *different* from the training data
  – *Reconstruction error* on a vector $x$ is related to $P(x)$ on the probability that the auto-encoder was trained with.
• Autoencoders are related to generative models of the data
(Deep) Stacked Autoencoders

Train

Apply

new dataset H of hidden unit activations for X

http://ufldl.stanford.edu/wiki/

Input Features I Output

Input (Features I) Features II Output
(Deep) Stacked Autoencoders

http://ufldl.stanford.edu/wiki/
Modeling documents using top 2000 words.

- 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.
First compress all documents to 2 numbers. Then use different colors for different document categories.
Applications of dimensionality reduction

- Visualization
  - examining data in 2 or 3 dimensions
- Semi-supervised learning
  - supervised learning in the reduced space
- Anomaly detection and data completion
  - coming soon
- Convenience in data processing
  - Go from words (10^6 dimensions) to word vectors (hundreds of dimensions)
  - Use dense GPU-friendly matrix operations instead of sparse ones
PCA
PCA is basically this picture...

- with linear units (not sigmoid units)
- trained to minimize squared error
- with a constraint that the “hidden units” are orthogonal
Motivating Example of PCA

• The MNist digits problem was simplified because the digits were
  – Centered
  – In a canonical position
  – Scaled to the same size

• What if they weren’t?
A Motivating Example

- Take a *single* 64*64* digit and create a dataset by repeatedly
  - Move it to a 100*100* image
  - Shift by *x*, *y* and rotate by *θ*
- Dataset has 10,000 features but really only needs 3
A Motivating Example

"prototype" = a vector of the same dimension as the instances

- PCA: reduces each instance to a linear combination of a few "prototypes" (blue+, green-). These are the first 5:

A specific choice of prototypes are the principle components

Original  $M = 1$  $M = 10$  $M = 50$  $M = 250$
A Motivating Example

“prototype” = a vector of the same dimension as the instances

- PCA: reduces each instance to a linear combination of a few “prototypes” (blue+, green-). These are the first 5:
PCA as matrices

2 prototypes

10,000 pixels

1000 * 10,000,00

V[i,j] = pixel j in image i

1.4*PC1 + 0.5*PC2 =
PCA as matrices: the optimization problem

Given a zero-mean dataset

\[ X = \begin{bmatrix}
  x_1^1 & x_1^2 & \cdots & x_1^m \\
  \vdots & \ddots & \ddots & \vdots \\
  x_n^1 & \cdots & x_n^m 
\end{bmatrix} = \begin{bmatrix}
  \vdots \\
  -x^t \\
  \vdots 
\end{bmatrix} \]

Find factors \( U \) and \( Z \) so that \( X \) is approximately their outer product:

\[
\begin{bmatrix}
  \vdots \\
  z^t \\
  \vdots 
\end{bmatrix}
\begin{bmatrix}
  \vdots \\
  -u_k \\
  \vdots 
\end{bmatrix} = \begin{bmatrix}
  \vdots \\
  -\hat{x}^t \\
  \vdots 
\end{bmatrix} = \hat{X} \quad \hat{x}^t = \sum_{i=1}^{k} z_k^t u_k
\]

Specifically minimizing the square of the reconstruction error

\[
J = \frac{1}{N} \sum_{t=1}^{N} \| x^t - \hat{x}^t \|^2
\]

under the constraint that the rows of \( U \) are orthogonal.
PCA as vectors: the optimization problem

Start with a zero-mean dataset, where $x_t$ is the $t$-th instance:

$$X = \begin{bmatrix}
x_1^1 & x_2^1 & \cdots & x_m^1 \\
\vdots & \ddots & \vdots \\
x_1^n & \cdots & \cdots & x_m^n
\end{bmatrix} = \begin{bmatrix}
- & \cdots & - \\
\vdots & \ddots & \vdots \\
- & \cdots & -
\end{bmatrix}$$

We want to find small number of orthogonal prototypes $u_1, \ldots, u_k$ and $k$ weights $z_{t1}, \ldots, z_{tk}$ for each instance $x_t$ so that if we approximate $x_t$ by

$$\hat{x}_t = \sum_{i=1}^{k} z_{tk} u_k$$

the approximation error will be small: we want to find $u$'s and $z$'s to minimize

$$J = \frac{1}{N} \sum_{t=1}^{N} \|x_t - \hat{x}_t\|^2$$
A cartoon of PCA

Green: the reconstruction of the original data

Magenta: the lower-dimensional model (linear combinations of one “prototype”)

In PCA we find a model that minimizes the “reconstruction error” (blue lines) ....
A 3D cartoon of PCA

http://www.nlpca.org/
More cartoons
PCA as matrices

2 prototypes

10,000 pixels

1000 * 10,000,00

1.4*PC1 + 0.5*PC2 =

V[i,j] = pixel j in image i
**PCA vs linear regression**

**R features (e.g., 4)**
- pl1
- pw1
- sl1
- sw1
- pl2
- pw2
- sl2
- sw2
- ...
- ...
- pln
- pwn

**n instances (e.g., 150)**

**W**

**m=1 regressors**
- w1
- w2
- w3
- w4

**H**

**predictions**
- y1
- ...
- yi
- yn

**Y**

\[ Y[i,1] = \text{instance i's prediction} \]
PCA vs linear regression

In contrast: in regression we’d minimize square error on one dimension ($x_2$) using a linear combination the other dimensions.
PCA vs mixtures of Gaussians

Mixture of Gaussians

For each point:
• Pick the index of the (latent) Gaussian $Z = k$
• Pick the point $x$ from that the $k$-th Gaussian, $x \sim N(\mu_k, \Sigma_k)$
PCA vs mixtures of Gaussians

Mixture of Gaussians
- Pick the index of the (latent) Gaussian $Z=k$
- Pick the point $x$ from that the $k$-th Gaussian, $x \sim N(\mu_k, \Sigma_k)$
PCA vs mixtures of Gaussians

**PCA**
- Pick a *continuous* value $z$, which will be used to combine the "prototypes" $u$ in the model.
- Pick the point $x$ from a spherical Gaussian centered on $zu$.
PCA vs mixtures of Gaussians

Comment: we can preprocess the data so that the mean is 0 to simplify the model.
PCA: IMPLEMENTATION
Finding principle components of a matrix

• There are two algorithms I like
  – EM (Roweis, NIPS 2007)
  – Eigenvectors of the correlation matrix (next)
PCA as vectors: the optimization problem

Start with a zero-mean dataset, where $x^t$ is a $t$-th instance:

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_m^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & \cdots & \cdots & x_m^n \end{bmatrix} = \begin{bmatrix} \vdots \\ -x^t \end{bmatrix}$$

We want to find small number of orthogonal prototypes $u_1, \ldots, u_k$ and $k$ weights $z_{k1}^t, \ldots, z_{kk}^t$ for each instance $x^t$ so that if we approximate $x^t$ by

$$\hat{x}^t = \sum_{i=1}^{k} z_{ki}^t u_k$$

the approximation error will be small: we want to find $u$’s and $z$’s to minimize

$$J = \frac{1}{N} \sum_{t=1}^{N} \|x^t - \hat{x}^t\|^2$$
PCA as matrices: the optimization problem

Given a zero-mean dataset

\[ X = \begin{bmatrix} x_1^1 & x_2^1 & \ldots & x_m^1 \\ \vdots & \ddots & \ddots & \vdots \\ x_1^n & \ldots & x_m^n \end{bmatrix} = \begin{bmatrix} \vdots \\ -x^t \end{bmatrix} \]

Find factors U and Z so that X is approximately their outer product:

\[
\begin{bmatrix} \vdots & z^t & \ldots \end{bmatrix} \begin{bmatrix} \vdots & - & \ldots & -u_k & - & \ldots \end{bmatrix} = \begin{bmatrix} \vdots & - & \hat{x}^t & - \end{bmatrix} = \hat{X}
\]

Specifically minimizing the square of the reconstruction error

\[ J = \frac{1}{N} \sum_{t=1}^{N} \|x^t - \hat{x}^t\|^2 \]

under the constraint that the rows of U are orthogonal.
Implementing PCA

Start with a zero-mean dataset, where
- $x^t$ is the $t$-th instance
- $f_i$ is a column of feature values for the $i$-th feature.
- Compute the sample covariance matrix

$$C_X = X^T X$$

i.e.,

$$C_X(i, j) = \sum_t f_i^t f_j^t$$

- Find the largest $k$ eigenvectors of $C_X$. These are the prototypes, $U$.
- Now find $Z$ given $X$ and $U$. 
Implementing PCA: why does this work?

Start with a zero-mean dataset, where
- \( x_t \) is the \( t \)-th instance
- \( f_i \) is a column of feature values for the \( i \)-th feature.
- Compute the sample covariance matrix

\[
C_X = X^T X
\]

Some intuitions:
1. Suppose you wanted to predict feature \( i \) from feature \( j \). Your best guess would be
   \[
   f_i \text{ is predicted as } C_X(i, j) \cdot f_j
   \]
2. If you wanted to predict feature \( i \) from all other feature’s \( j \), a plausible guess is
   \[
   f_i \text{ is predicted as } \frac{1}{n} \sum_{j \neq i} C_X(i, j) \cdot f_j
   \]
3. Any eigenvector, \( e \), of \( C_X \) leads to an internally consistent* set of predictions
   \[
   \exists \lambda : \lambda e = C_X e \quad \Rightarrow \quad \forall i, \lambda e_i = \frac{1}{n} \sum_j C_X(i, j) e_j
   \]
   *
   
   up to a multiplier
Some more examples of PCA: Eigenfaces

Turk and Pentland, 1991
Some more examples of PCA: Eigenfaces

Average face

Six eigenfaces (PC’s)
Some more examples of PCA: Eigenfaces

Turk and Pentland, 1991
Some more examples of PCA:
Eigenfaces
Some more examples of PCA: Eigenfaces

How is this done?

Simplest approach:
- Add the image with missing values to the data matrix
- Minimize reconstruction error over the non-missing values

\[
\begin{bmatrix}
\vdots & \vdots \\
\end{bmatrix}
= 
\begin{bmatrix}
- & - & u_k & - & - \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
- & - & - & \hat{x^t} & - \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
- & - & - & - & ? \\
\end{bmatrix}
= \hat{X}
\]
Image denoising
Matrix completion for image denoising

Partially observed image

Reconstructed image
PCA FOR MODELING TEXT
(SVD = SINGULAR VALUE DECOMPOSITION)
A Scalability Problem with PCA

• Covariance matrix is large in high dimensions
  – With d features covariance matrix is d*d
• **SVD** is a closely-related method that can be implemented more efficiently in high dimensions
  – Don’t explicitly compute covariance matrix
  – Instead decompose $X$ to $X = USV^T$
  – $S$ is $k*k$ where $k<<<d$
  – $S$ is diagonal and $S[i,i] = \sqrt(\lambda_i)$ for $V[:,i]$
  – Columns of $V \sim =$ principle components
  – Rows of $US \sim =$ embedding for examples
SVD example
• The Neatest Little Guide to Stock Market Investing
• Investing For Dummies, 4th Edition
• The Little Book of Common Sense Investing: The Only Way to Guarantee Your Fair Share of Stock Market Returns
• The Little Book of Value Investing
• Value Investing: From Graham to Buffett and Beyond
• Rich Dad’s Guide to Investing: What the Rich Invest in, That the Poor and the Middle Class Do Not!
• Investing in Real Estate, 5th Edition
• Stock Investing For Dummies
• Rich Dad’s Advisors: The ABC’s of Real Estate Investing: The Secrets of Finding Hidden Profits Most Investors Miss

<table>
<thead>
<tr>
<th>Index Words</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dads</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dummies</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>estate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>guide</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>investing</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>market</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>real</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>rich</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>stock</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

TFIDF counts would be better

Recovering latent factors in a matrix

\[
\begin{array}{cccc}
    x_1 & y_1 \\
    x_2 & y_2 \\
    \vdots & \vdots \\
    x_n & y_n \\
\end{array}
\]

\[
\begin{array}{cccc}
    a_1 & a_2 & \ldots & a_m \\
    b_1 & b_2 & \ldots & b_m \\
\end{array}
\]

\[
V_{i,j} = \text{TFIDF score of term } j \text{ in doc } i
\]
Investing for real estate

Rich Dad’s Advisor’s: The ABCs of Real Estate Investment …
The little book of common sense investing:

Neatest Little Guide to Stock Market Investing