Techniques for Dimensionality Reduction

PCA and Other Matrix Factorization Methods
Outline

• Principle Components Analysis (PCA)
  – Example (Bishop, ch 12)
  – PCA as a mixture model variant
    • With a continuous latent variable
  – Breaking down PCA
    • Optimization problem
    • Solution
    • Intuitions

• General matrix factorization
  – Application to collaborative filtering
  – Algorithms
  – Wrap-up
A Motivating Example

• 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 $\theta$
• Dataset has 10,000 features but really only needs 3
A Motivating Example

- 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

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

Original $\quad M = 1 \quad M = 10 \quad M = 50 \quad M = 250$
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

1000 images

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

10,000 pixels

1000 * 10,000,000

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

\[ 1.4 \times \text{PC1} + 0.5 \times \text{PC2} = \text{3} \]
A Cartoon of PCA

Red: the dataset
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/
Some more cartoons
PCA vs Linear Regression

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

$r$ features (e.g., 4)

\[
\begin{align*}
\text{pl1} & \quad \text{pw1} & \quad \text{sl1} & \quad \text{sw1} \\
\text{pl2} & \quad \text{pw2} & \quad \text{sl2} & \quad \text{sw2} \\
\ldots & \quad \ldots & \quad \ldots & \quad \ldots \\
\text{pln} & \quad \text{pwn}
\end{align*}
\]

\[W\]

$m=1$ regressors

\[
\begin{align*}
\text{w1} \\
\text{w2} \\
\text{w3} \\
\text{w4}
\end{align*}
\]

\[H\]

predictions

\[
\begin{align*}
y1 \\
\ldots \\
yi \\
\ldots \\
yn
\end{align*}
\]

\[Y\]

\[Y[i,1] = \text{instance } i\text{'s prediction}\]
In contrast: in regression we’d minimize square error on one dimension ($x_2$) using a linear combination of the other dimensions.
PCA vs mixture 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, $x \sim N(\mu_k, \Sigma_k)$

Plate notation
PCA vs mixture 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 mixture 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 mixture of Gaussians

Comment: we can preprocess the data so that the mean is 0 to simplify the model
Finding the Principle Components

• There are different algorithms that can be used
  – EM (Roweis, NIPS 2007)
  – Can also be turned into an eigenvector computation (next)
Outline

• PCA
  – Example (Bishop, ch 12)
  – PCA as a mixture model variant
    • With a continuous latent variable
  – Breaking down PCA
    • Optimization problem
    • Solution
    • Intuition
The PCA Problem (vectors)

Start with a zero-mean dataset, where \( x^t \) is the \( t \)-th instance:

\[
X = \begin{bmatrix}
  x^1_1 & x^1_2 & \cdots & x^1_m \\
  \vdots & \ddots & \vdots & \vdots \\
  x^n_1 & \cdots & x^n_m \\
\end{bmatrix} = \begin{bmatrix}
  \vdots \\
  -x^t \\
  \vdots 
\end{bmatrix}
\]

We want to find small number of orthogonal prototypes \( u_1, \ldots, u_k \) and \( k \) weights \( z^t_1, \ldots, z^t_k \) for each instance \( x^t \) so that if we approximate \( x^t \) by

\[
\hat{x}^t = \sum_{i=1}^{k} z^t_i u_k
\]

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

\[
J = \frac{1}{N} \sum_{t=1}^{N} \| x^t - \hat{x}^t \|^2
\]
The PCA Problem (matrices)

Given a zero-mean dataset

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

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

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

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.
A PCA Algorithm

Start with a zero-mean dataset, where

- $\mathbf{x}^t$ is the $t$-th instance
- $\mathbf{f}_i$ 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, $\mathbf{U}$.
- Now find $\mathbf{Z}$ given $\mathbf{X}$ and $\mathbf{U}$. 

\[
X = \begin{bmatrix}
    x_1^1 & x_2^1 & \cdots & x_m^1 \\
    \vdots & \ddots & \vdots \\
    x_1^n & \cdots & x_m^n
\end{bmatrix}
= \begin{bmatrix}
    \vdots \\
    -x^t t \\
    \vdots
\end{bmatrix}
\]
\[
X = \begin{bmatrix}
    x_1^1 & x_2^1 & \cdots & x_m^1 \\
    \vdots & \ddots & \vdots \\
    x_1^n & \cdots & x_m^n
\end{bmatrix}
= \begin{bmatrix}
    \vdots \\
    \vdots \\
    \vdots
\end{bmatrix}
\]
PCA Algorithm: Intuitions

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 \textit{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
PCA: Eigenfaces

Turk and Pentland, 1991
PCA: Eigenfaces
Turk and Pentland, 1991

Average face

Six eigenfaces (PC’s)
PCA: Eigenfaces

Turk and Pentland, 1991
PCA: Eigenfaces
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

\[
\begin{bmatrix}
\vdots & \mathbf{z}^t & \vdots \\
\end{bmatrix}
\begin{bmatrix}
\begin{array}{c}
\vdots \\
- - \mathbf{u}_k - - \\
\vdots \\
\end{array}
\end{bmatrix}
= \begin{bmatrix}
\vdots \\
- \hat{\mathbf{x}}^t - \\
\end{bmatrix} = \hat{\mathbf{X}}
\]
Matrix completion for image denoising
Outline

• Principle Components Analysis (PCA)
• Other types of/applications of matrix factorization
  – Collaborative filtering/recommendation
  – Matrix factorization for CF using gradient descent
What is collaborative filtering?
What is collaborative filtering?
What is collaborative filtering?
What is collaborative filtering?

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the $1M Grand Prize to team “BellKor’s Pragmatic Chaos”. Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>% Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:18:28</td>
</tr>
<tr>
<td>2</td>
<td>The Ensemble</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:36:22</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8582</td>
<td>9.90</td>
<td>2009-07-10 21:24:40</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8588</td>
<td>9.84</td>
<td>2009-07-10 01:12:31</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries !</td>
<td>0.8591</td>
<td>9.81</td>
<td>2009-07-10 00:32:20</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8594</td>
<td>9.77</td>
<td>2009-06-24 12:06:56</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8601</td>
<td>9.70</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.8612</td>
<td>9.59</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Feeds2</td>
<td>0.8622</td>
<td>9.48</td>
<td>2009-07-12 13:11:51</td>
</tr>
<tr>
<td>10</td>
<td>BigChaos</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-04-07 12:33:59</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-07-24 00:34:07</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</td>
<td>0.8624</td>
<td>9.46</td>
<td>2009-07-26 17:19:11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>% Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>xiangliang</td>
<td>0.8642</td>
<td>9.27</td>
<td>2009-07-15 14:53:22</td>
</tr>
<tr>
<td>14</td>
<td>Gravity</td>
<td>0.8643</td>
<td>9.26</td>
<td>2009-04-22 18:31:32</td>
</tr>
<tr>
<td>15</td>
<td>Ces</td>
<td>0.8651</td>
<td>9.18</td>
<td>2009-06-21 19:24:53</td>
</tr>
<tr>
<td>16</td>
<td>Invisible Ideas</td>
<td>0.8653</td>
<td>9.15</td>
<td>2009-07-15 15:53:04</td>
</tr>
<tr>
<td>17</td>
<td>Just a guy in a garage</td>
<td>0.8662</td>
<td>9.06</td>
<td>2009-05-24 10:02:54</td>
</tr>
<tr>
<td>18</td>
<td>J Dennis Su</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-03-07 17:16:17</td>
</tr>
<tr>
<td>19</td>
<td>Craig Carmichael</td>
<td>0.8666</td>
<td>9.02</td>
<td>2009-07-25 16:00:54</td>
</tr>
<tr>
<td>20</td>
<td>acmehill</td>
<td>0.8668</td>
<td>9.00</td>
<td>2009-03-21 16:20:50</td>
</tr>
</tbody>
</table>

**Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos**

**Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos**

**Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell**

**Cinematic Score - RMSE = 0.9525**
What is collaborative filtering?
Other examples of social filtering....
Other examples of social filtering....
Everyday Examples of Collaborative Filtering...

- Bestseller lists
- Top 40 music lists
- The “recent returns” shelf at the library
- Unmarked but well-used paths thru the woods
- The printer room at work
- “Read any good books lately?”
- ....

**Common insight:** personal tastes are correlated:
  - If Alice and Bob both like X and Alice likes Y then Bob is more likely to like Y
  - especially (perhaps) if Bob knows Alice
Outline

• Principle Components Analysis (PCA)
• Other types of/applications of matrix factorization
  – Collaborative filtering/recommendation
  – Algorithms:
    • K-NN type methods
    • Classification-base methods
    • ...
  – Matrix factorization
Recovering latent factors in a matrix

$V[i,j] = \text{user } i\text{'s rating of movie } j$

$n \text{ users}$

$m \text{ movies}$

$v_{11} \ldots$

$\ldots \ldots$

$v_{ij}$

$\ldots$

$v_{nm}$
Recovering latent factors in a matrix

Minimize squared error reconstruction error and force the “prototype” users to be orthogonal $\Rightarrow$ PCA

$V[i,j] = \text{user i’s rating of movie j}$
Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent

Rainer Gemulla

talk pilfered from →

Peter J. Haas   Yannis Sismanis   Erik Nijkamp
Collaborative Filtering

- **Problem**
  - Set of users
  - Set of items (movies, books, jokes, products, stories, ...)
  - Feedback (ratings, purchase, click-through, tags, ...)

- **Predict additional items a user may like**
  - Assumption: Similar feedback $\implies$ Similar taste

- **Example**

\[
\begin{pmatrix}
Alice & The Matrix & Up \\
Alice & ? & 4 & 2 \\
Bob    & 3 & 2 & ? \\
Charlie & 5 & ? & 3
\end{pmatrix}
\]

- **Netflix competition**: 500k users, 20k movies, 100M movie ratings, 3M question marks
Recovering latent factors in a matrix

\[ V[i,j] = \text{user } i\text{'s rating of movie } j \]
Semantic Factors (Koren et al., 2009)

The Color Purple
Sense and Sensibility
The Princess Diaries
Dave
The Lion King
Lea"
Ocean’s 11
Amadeus
Braveheart
Lethal Weapon
Independence Day
Dumb and Dumber
Gus
Latent Factor Models

- Discover latent factors \((r = 1)\)

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(2.24)</td>
<td>(1.92)</td>
<td>(1.18)</td>
</tr>
<tr>
<td>Alice</td>
<td>?</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(1.98)</td>
<td>(4.4)</td>
<td>(3.8)</td>
<td>(2.3)</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>(1.21)</td>
<td>(2.7)</td>
<td>(2.3)</td>
<td>(1.4)</td>
</tr>
<tr>
<td>Charlie</td>
<td>5</td>
<td>?</td>
<td>3</td>
</tr>
<tr>
<td>(2.30)</td>
<td>(5.2)</td>
<td>(4.4)</td>
<td>(2.7)</td>
</tr>
</tbody>
</table>

- Minimum loss

\[
\min_{W, H, u, m} \sum_{(i, j) \in Z} (V_{ij} - \mu - u_i - m_j - [WH]_{ij})^2
\]

\[+ \lambda (\|W\| + \|H\| + \|u\| + \|m\|)\]

- Bias, regularization

user-specific bias term

movie-specific bias term
Stochastic Gradient Descent

- Find minimum $\theta^*$ of function $L$
- Pick a starting point $\theta_0$
- Approximate gradient $\hat{L}'(\theta_0)$
- Jump “approximately” downhill
- Stochastic difference equation

$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n)$$

- Under certain conditions, asymptotically approximates (continuous) gradient descent
Recovering latent factors in a matrix

\[ V[i,j] = \text{user i's rating of movie j} \]
... is like Linear Regression ....

- **n instances (e.g., 150)**
- **$m=1$ regressors**
- **$r$ features (eg 4)**

$$Y_{i,1} = \text{instance } i\text{'s prediction}$$
for many outputs at once….

where we also have to find the dataset!

$Y[I, j] = \text{instance } i\text{'s prediction for regression task } j$
Matrix factorization as SGD

require that the loss can be written as

\[ L = \sum_{(i,j) \in Z} l(V_{ij}, W_{i*}, H_{*j}) \]

Algorithm 1 SGD for Matrix Factorization

Require: A training set \( Z \), initial values \( W_0 \) and \( H_0 \)

while not converged do  
    Select a training point \((i, j) \in Z\) uniformly at random.
    \[
    W_{i*}' \leftarrow W_{i*} - \epsilon_n N \frac{\partial}{\partial W_{i*}} l(V_{ij}, W_{i*}, H_{*j})
    \]
    \[
    H_{*j}' \leftarrow H_{*j} - \epsilon_n N \frac{\partial}{\partial H_{*j}} l(V_{ij}, W_{i*}, H_{*j})
    \]
    \[
    W_{i*} \leftarrow W_{i*}'
    \]
end while

step size
Matrix factorization as SGD - why does this work?

require that the loss can be written as

\[ L = \sum_{(i,j) \in Z} l(V_{ij}, W_{i*}, H_{*j}) \]

Algorithm 1 SGD for Matrix Factorization

Require: A training set \( Z \), initial values \( W_0 \) and \( H_0 \)

while not converged do 

   step

   Select a training point \((i, j) \in Z\) uniformly at random.

   \[ W_{i*}' \leftarrow W_{i*} - \epsilon_n N \frac{\partial}{\partial W_{i*}} l(V_{ij}, W_{i*}, H_{*j}) \]

   \[ H_{*j}' \leftarrow H_{*j} - \epsilon_n N \frac{\partial}{\partial H_{*j}} l(V_{ij}, W_{i*}, H_{*j}) \]

   \[ W_{i*} \leftarrow W_{i*}' \]

end while

step size
Matrix factorization as SGD - why does this work? Here's the key claim:

require that the loss can be written as

\[ L = \sum_{(i,j) \in Z} l(V_{ij}, W_{i*}, H_{*j}) \]

\[
\frac{\partial}{\partial W_{i'k}} L_{ij}(W, H) = \begin{cases} 
0 & \text{if } i \neq i' \\
\frac{\partial}{\partial W_{ik}} l(V_{ij}, W_{i*}, H_{*j}) & \text{otherwise}
\end{cases}
\]

\[
\frac{\partial}{\partial H_{kj'}} L_{ij}(W, H) = \begin{cases} 
0 & \text{if } j \neq j' \\
\frac{\partial}{\partial H_{kj}} l(V_{ij}, W_{i*}, H_{*j}) & \text{otherwise}
\end{cases}
\]
Checking the claim

\[ \frac{\partial}{\partial W_{i*}} L(W, H) = \frac{\partial}{\partial W_{i*}} \sum_{(i',j) \in Z} L_{i'j}(W_{i'\ast}, H_{\ast j}) = \sum_{j \in Z_{i*}} \frac{\partial}{\partial W_{i*}} L_{ij}(W_{i*}, H_{\ast j}), \]

where \( Z_{i*} = \{ j : (i, j) \in Z \} \).

\[ \frac{\partial}{\partial H_{\ast j}} L(W, H) = \sum_{i \in Z_{\ast j}} \frac{\partial}{\partial W_{\ast j}} L_{ij}(W_{i*}, H_{\ast j}), \]

where \( Z_{\ast j} = \{ i : (i, j) \in Z \} \).

Think for SGD for logistic regression

• LR loss = compare \( y \) and \( \hat{y} = \text{dot}(w,x) \)
• similar but now update \( w \) (user weights) and \( x \) (movie weight)
What loss functions are possible?

\[ L_{NZSL} = \sum_{(i,j) \in Z} (V_{ij} - (WH)_{ij})^2 \]

\[ L_{L2} = L_{NZSL} + \lambda (\|W\|_F^2 + \|H\|_F^2) \]

\[ L_{NZL2} = L_{NZSL} + \lambda (\|N_1W\|_F^2 + \|HN_2\|_F^2) \]

N1, N2 - diagonal matrixes, sort of like IDF factors for the users/movies.
What loss functions are possible?

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Definition and Derivatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{NZSL}$</td>
<td>$L_{NZSL} = \sum_{(i,j) \in Z} (V_{ij} - [WH]_{ij})^2$</td>
</tr>
<tr>
<td></td>
<td>$\frac{\partial}{\partial W_{ik}} L_{ij} = -2(V_{ij} - [WH]<em>{ij})H</em>{kj}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{\partial}{\partial H_{kj}} L_{ij} = -2(V_{ij} - [WH]<em>{ij})W</em>{ik}$</td>
</tr>
</tbody>
</table>
What loss functions are possible?

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Definition and Derivatives</th>
</tr>
</thead>
</table>
| $L_{L2}$      | $L_{L2} = L_{NZSL} + \lambda \left( \|W\|_F^2 + \|H\|_F^2 \right)$  
|               | $= \sum_{(i,j) \in Z} \left[ (V_{ij} - [WH]_{ij})^2 + \lambda \left( \frac{\|W_{i*}\|_F^2}{N_{i*}} + \frac{\|H_{*j}\|_F^2}{N_{*j}} \right) \right]$  
|               | $\frac{\partial}{\partial W_{ik}} L_{ij} = -2(V_{ij} - [WH]_{ij})H_{kj} + 2\lambda \frac{W_{ik}}{N_{i*}}$  
|               | $\frac{\partial}{\partial H_{kj}} L_{ij} = -2(V_{ij} - [WH]_{ij})W_{ik} + 2\lambda \frac{H_{kj}}{N_{*j}}$ |
Stochastic Gradient Descent on Netflix Data

ALS = alternating least squares
Wrapup: Matrix Multiplications in Machine Learning
Recovering latent factors in a matrix

\[ r \]

\[ W \]

\[ m \text{ movies} \]

\[ H \]

\[ m \text{ movies} \]

\[ V[i,j] = \text{user } i\text{'s rating of movie } j \]
... vs PCA

Minimize squared error reconstruction error and force the “prototype” users to be orthogonal \( \Rightarrow \) PCA

\[ V[i,j] = \text{user i’s rating of movie j} \]
... vs autoencoders & nonlinear PCA

• 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>00000101</td>
<td>00000100</td>
</tr>
<tr>
<td>00010000</td>
<td>00001000</td>
</tr>
<tr>
<td>00100000</td>
<td>00100000</td>
</tr>
<tr>
<td>01000000</td>
<td>01000000</td>
</tr>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
</tbody>
</table>

• Using the following network:

• With linear hidden units, how do the weights match up to W and H?

Flashback to NN lecture.....
.. vs k-means

**indicators for r clusters**

\[
\begin{pmatrix}
0 & 1 \\
1 & 0 \\
\vdots & \vdots \\
nx & nyn
\end{pmatrix}
\]

**cluster means**

\[
\begin{pmatrix}
a_1 & a_2 & \cdots & a_m \\
b_1 & b_2 & \cdots & b_m
\end{pmatrix}
\]

\sim

**original data set**

\[
\begin{pmatrix}
v_{11} & \cdots \\
\vdots & \ddots \\
v_{ij} & \cdots \\
v_{nm}
\end{pmatrix}
\]