Dimension Reduction using PCA and SVD
Plan of Class

- Starting the machine Learning part of the course.
- Based on Linear Algebra.
- If your linear algebra is rusty, check out the pages on “Resources/Linear Algebra”
- This class will all be theory.
- Next class will be on doing PCA in Spark.
- HW3 will open on friday, be due the following friday.
Dimensionality reduction

Why reduce the number of features in a data set?

1. It reduces storage and computation time.
2. High-dimensional data often has a lot of redundancy.
3. Remove noisy or irrelevant features.

Example: are all the pixels in an image equally informative?

$28 \times 28 = 784$ pixels. A vector $\vec{x} \in \mathbb{R}^{784}$

If we were to choose a few pixels to discard, which would be the prime candidates?

Those with lowest variance...
Eliminating low variance coordinates

Example: MNIST. What fraction of the total variance is contained in the 100 (or 200, or 300) coordinates with lowest variance?

We can easily drop 300-400 pixels...
Can we eliminate more?
Yes! By using features that are combinations of pixels instead of single pixels.
Suppose $X$ has mean $\mu_X$ and $Y$ has mean $\mu_Y$.

- **Covariance**

\[
\text{cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] = \mathbb{E}[XY] - \mu_X\mu_Y
\]

Maximized when $X = Y$, in which case it is $\text{var}(X)$.
In general, it is at most $\text{std}(X)\text{std}(Y)$. 

**Covariance (a quick review)**
Covariance: example 1

\[ \text{cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] = \mathbb{E}[XY] - \mu_X \mu_Y \]

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$\text{Pr}(x, y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-1$</td>
<td>$-1$</td>
<td>$1/3$</td>
</tr>
<tr>
<td>$-1$</td>
<td>$1$</td>
<td>$1/6$</td>
</tr>
<tr>
<td>$1$</td>
<td>$-1$</td>
<td>$1/3$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
<td>$1/6$</td>
</tr>
</tbody>
</table>

$\mu_X = 0$

$\mu_Y = \frac{-1}{3}$

$\text{var}(X) = 1$

$\text{var}(Y) = \frac{8}{9}$

$\text{cov}(X, Y) = 0$

In this case, $X, Y$ are independent. Independent variables always have zero covariance.
Covariance: example 2

\[ \text{cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] = \mathbb{E}[XY] - \mu_X \mu_Y \]

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$\text{Pr}(x, y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-10</td>
<td>1/6</td>
</tr>
<tr>
<td>-1</td>
<td>10</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td>-10</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>1/6</td>
</tr>
</tbody>
</table>

$\mu_X = 0$

$\mu_Y = 0$

$\text{var}(X) = 1$

$\text{var}(Y) = 100$

$\text{cov}(X, Y) = -10/3$

In this case, $X$ and $Y$ are negatively correlated.
Example: MNIST

approximate a digit from class $j$ as the class average plus $k$ corrections:

$$\tilde{x} \approx \mu_j + \sum_{i=1}^{k} a_i \vec{v}_{j,i}$$

- $\mu_j \in \mathbb{R}^{784}$ class mean vector
- $\vec{v}_{j,1}, \ldots, \vec{v}_{j,k}$ are the principal directions.
The effect of correlation

Suppose we wanted just one feature for the following data.

This is the direction of maximum variance.
Two types of projection

Projection onto $\mathbb{R}$:

Projection onto a 1-d line in $\mathbb{R}^2$: 

```math
\mathbb{R}^2
```
Projection: formally

What is the projection of \( x \in \mathbb{R}^p \) onto direction \( u \in \mathbb{R}^p \) (where \( \|u\| = 1 \))? 

As a one-dimensional value:

\[
x \cdot u = u \cdot x = u^T x = \sum_{i=1}^{p} u_i x_i.
\]

As a \( p \)-dimensional vector:

\[
(x \cdot u)u = uu^T x
\]

“Move \( x \cdot u \) units in direction \( u \)”

What is the projection of \( x = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \) onto the following directions?

- The coordinate direction \( e_1 \)? Answer: 2
- The direction \( \begin{pmatrix} 1 \\ -1 \end{pmatrix} \)? Answer: \(-1/\sqrt{2}\)
A notation that allows a simple representation of multiple projections

A vector $\vec{v} \in \mathbb{R}^d$ can be represented, in matrix notation, as

- A column vector:
  \[
  \vec{v} = \begin{pmatrix}
  v_1 \\
  v_2 \\
  \vdots \\
  v_d
  \end{pmatrix}
  \]

- A row vector:
  \[
  \vec{v}^T = (v_1 \ v_2 \ \cdots \ v_d)
  \]
By convention an inner product is represented by a **row** vector followed by a **column** vector:

\[
\begin{pmatrix}
  u_1 & u_2 & \cdots & u_d
\end{pmatrix}
\begin{pmatrix}
  v_1 \\
  v_2 \\
  \vdots \\
  v_d
\end{pmatrix}
= \sum_{i=1}^{d} u_i v_i
\]

While a **column** vector followed by a **row** vector represents an outer product which is a matrix:

\[
\begin{pmatrix}
  v_1 \\
  v_2 \\
  \vdots \\
  v_n
\end{pmatrix}
\begin{pmatrix}
  u_1 & u_2 & \cdots & u_m
\end{pmatrix}
= \begin{pmatrix}
  u_1 v_1 & u_2 v_1 & \cdots & u_m v_1 \\
  \vdots & \vdots & \ddots & \vdots \\
  u_1 v_n & u_2 v_n & \cdots & u_m v_n
\end{pmatrix}
\]
Projection onto multiple directions

Want to project \( x \in \mathbb{R}^p \) into the \( k \)-dimensional subspace defined by vectors \( u_1, \ldots, u_k \in \mathbb{R}^p \).

This is easiest when the \( u_i \)'s are **orthonormal**:  
- They each have length one.
- They are at right angles to each other: \( u_i \cdot u_j = 0 \) whenever \( i \neq j \)

Then the projection, as a \( k \)-dimensional vector, is

\[
(x \cdot u_1, x \cdot u_2, \ldots, x \cdot u_k) = \left( \begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_k \end{array} \right) \left( \begin{array}{c} x \\ \end{array} \right)
\]

As a \( p \)-dimensional vector, the projection is

\[
(x \cdot u_1)u_1 + (x \cdot u_2)u_2 + \cdots + (x \cdot u_k)u_k = UU^T x.
\]
Projection onto multiple directions: example

Suppose data are in $\mathbb{R}^4$ and we want to project onto the first two coordinates.

Take vectors $u_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, $u_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ (notice: orthonormal)

Then write $U^T = \begin{pmatrix} u_1 & u_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

The projection of $x \in \mathbb{R}^4$, as a 2-d vector, is

$$U^T x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The projection of $x$ as a 4-d vector is

$$UU^T x = \begin{pmatrix} x_1 \\ x_2 \\ 0 \\ 0 \end{pmatrix}$$

But we'll generally project along non-coordinate directions.
The best single direction

Suppose we need to map our data $x \in \mathbb{R}^p$ into just one dimension:

$$x \mapsto u \cdot x \quad \text{for some unit direction } u \in \mathbb{R}^p$$

What is the direction $u$ of maximum variance?

**Theorem:** Let $\Sigma$ be the $p \times p$ covariance matrix of $X$. The variance of $X$ in direction $u$ is given by $u^T \Sigma u$.

- Suppose the mean of $X$ is $\mu \in \mathbb{R}^p$. The projection $u^T X$ has mean
  $$\mathbb{E}(u^T X) = u^T \mathbb{E}X = u^T \mu.$$

- The variance of $u^T X$ is
  $$\text{var}(u^T X) = \mathbb{E}(u^T X - u^T \mu)^2 = \mathbb{E}(u^T (X - \mu)(X - \mu)^T u)$$
  $$= u^T \mathbb{E}(X - \mu)(X - \mu)^T u = u^T \Sigma u.$$

Another theorem: $u^T \Sigma u$ is maximized by setting $u$ to the first *eigenvector* of $\Sigma$. The maximum value is the corresponding *eigenvalue*. 
Best single direction: example

This direction is the **first eigenvector** of the $2 \times 2$ covariance matrix of the data.
The best \( k \)-dimensional projection

Let \( \Sigma \) be the \( p \times p \) covariance matrix of \( X \). Its \textit{eigendecomposition} can be computed in \( O(p^3) \) time and consists of:

- real \textit{eigenvalues} \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \)
- corresponding \textit{eigenvectors} \( u_1, \ldots, u_p \in \mathbb{R}^p \) that are orthonormal: that is, each \( u_i \) has unit length and \( u_i \cdot u_j = 0 \) whenever \( i \neq j \).

\textbf{Theorem:} Suppose we want to map data \( X \in \mathbb{R}^p \) to just \( k \) dimensions, while capturing as much of the variance of \( X \) as possible. The best choice of projection is:

\[ x \mapsto (u_1 \cdot x, u_2 \cdot x, \ldots, u_k \cdot x), \]

where \( u_i \) are the eigenvectors described above.

Projecting the data in this way is \textit{principal component analysis} (PCA).
Example: MNIST

Contrast coordinate projections with PCA:
MNIST: image reconstruction

Reconstruct this original image from its PCA projection to $k$ dimensions.

$k = 200$

$k = 150$

$k = 100$

$k = 50$

Q: What are these reconstructions exactly?
A: Image $x$ is reconstructed as $UU^Tx$, where $U$ is a $p \times k$ matrix whose columns are the top $k$ eigenvectors of $\Sigma$. 
What are eigenvalues and eigenvectors?

There are several steps to understanding these.

1. Any matrix $M$ defines a function (or transformation) $x \mapsto Mx$.
2. If $M$ is a $p \times q$ matrix, then this transformation maps vector $x \in \mathbb{R}^q$ to vector $Mx \in \mathbb{R}^p$.
3. We call it a **linear transformation** because $M(x + x') = Mx + Mx'$.
4. We’d like to understand the nature of these transformations. The easiest case is when $M$ is **diagonal**:

$$
\begin{pmatrix}
2 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 10
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
= 
\begin{pmatrix}
2x_1 \\
-x_2 \\
10x_3
\end{pmatrix}
$$

In this case, $M$ simply scales each coordinate separately.

5. What about more general matrices that are symmetric but not necessarily diagonal? They also just scale coordinates separately, but in a **different coordinate system**.
Eigenvalue and eigenvector: definition

Let $M$ be a $p \times p$ matrix. We say $u \in \mathbb{R}^p$ is an eigenvector if $M$ maps $u$ onto the same direction, that is,

$$Mu = \lambda u$$

for some scaling constant $\lambda$. This $\lambda$ is the eigenvalue associated with $u$.

Question: What are the eigenvectors and eigenvalues of:

$$M = \begin{pmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

Answer: Eigenvectors $e_1, e_2, e_3$, with corresponding eigenvalues 2, $-1, 10$.

Notice that these eigenvectors form an orthonormal basis.
Eigenvectors of a real symmetric matrix

Theorem. Let $M$ be any real symmetric $p \times p$ matrix. Then $M$ has

- $p$ eigenvalues $\lambda_1, \ldots, \lambda_p$
- corresponding eigenvectors $u_1, \ldots, u_p \in \mathbb{R}^p$ that are orthonormal

We can think of $u_1, \ldots, u_p$ as being the axes of the natural coordinate system for understanding $M$.

Example: consider the matrix

$$M = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

It has eigenvectors

$$u_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad u_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

and corresponding eigenvalues $\lambda_1 = 4$ and $\lambda_2 = 2$. (Check)
Spectral decomposition

**Theorem.** Let $M$ be any real symmetric $p \times p$ matrix. Then $M$ has

- $p$ eigenvalues $\lambda_1, \ldots, \lambda_p$
- corresponding eigenvectors $u_1, \ldots, u_p \in \mathbb{R}^p$ that are orthonormal

**Spectral decomposition:** Here is another way to write $M$:

$$M = U \Lambda U^T$$

Thus $Mx = U \Lambda U^T x$, which can be interpreted as follows:

- $U^T$ rewrites $x$ in the $\{u_i\}$ coordinate system
- $\Lambda$ is a simple coordinate scaling in that basis
- $U$ then sends the scaled vector back into the usual coordinate basis
Spectral decomposition: example

Apply spectral decomposition to the matrix $M$ we saw earlier:

$M = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$

$M \begin{pmatrix} 1 \\ 2 \end{pmatrix} = ???$

$= U\Lambda U^T \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$= U\Lambda \frac{1}{\sqrt{2}} \begin{pmatrix} 3 \\ 1 \end{pmatrix}$

$= U \frac{1}{\sqrt{2}} \begin{pmatrix} 12 \\ 2 \end{pmatrix}$

$= \begin{pmatrix} 5 \\ 7 \end{pmatrix}$
Principal component analysis: recap

Consider data vectors $X \in \mathbb{R}^p$.

- The covariance matrix $\Sigma$ is a $p \times p$ symmetric matrix.
- Get eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, eigenvectors $u_1, \ldots, u_p$.
- $u_1, \ldots, u_p$ is an alternative basis in which to represent the data.
- The variance of $X$ in direction $u_i$ is $\lambda_i$.
- To project to $k$ dimensions while losing as little as possible of the overall variance, use $x \mapsto (x \cdot u_1, \ldots, x \cdot u_k)$.

What is the covariance of the projected data?
What are the dimensions along which personalities differ?

- **Lexical hypothesis:** most important personality characteristics have become encoded in natural language.

- Allport and Odbert (1936): sat down with the English dictionary and extracted all terms that could be used to distinguish one person’s behavior from another’s. Roughly 18000 words, of which 4500 could be described as personality traits.

- Step: group these words into (approximate) synonyms. This is done by manual clustering. e.g. Norman (1967):

<table>
<thead>
<tr>
<th>Spirit</th>
<th>Jolly, merry, witty, lively, peppy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talkativeness</td>
<td>Talkative, articulate, verbose, gossipy</td>
</tr>
<tr>
<td>Sociability</td>
<td>Companionable, social, outgoing</td>
</tr>
<tr>
<td>Spontaneity</td>
<td>Impulsive, carefree, playful, zany</td>
</tr>
<tr>
<td>Boisterousness</td>
<td>Mischiefous, rowdy, loud, prankish</td>
</tr>
<tr>
<td>Adventure</td>
<td>Brave, ventures, fearless, reckless</td>
</tr>
<tr>
<td>Energy</td>
<td>Active, assertive, dominant, energetic</td>
</tr>
<tr>
<td>Conceit</td>
<td>Boastful, conceited, egotistical</td>
</tr>
<tr>
<td>Vanity</td>
<td>Affected, vain, chic, dapper, jaunty</td>
</tr>
<tr>
<td>Indiscretion</td>
<td>Noisy, snoopy, indiscreet, meddlesome</td>
</tr>
<tr>
<td>Sensuality</td>
<td>Sexy, passionate, sensual, flirtatious</td>
</tr>
</tbody>
</table>

- Data collection: Ask a variety of subjects to what extent each of these words describes them.
Personality assessment: the data

Matrix of data (1 = strongly disagree, 5 = strongly agree)

<table>
<thead>
<tr>
<th></th>
<th>shy</th>
<th>merry</th>
<th>tense</th>
<th>boastful</th>
<th>forgiving</th>
<th>quiet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Person 2</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Person 3</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How to extract important directions?

- Treat each column as a data point, find tight clusters
- Treat each row as a data point, apply PCA
- Other ideas: factor analysis, independent component analysis, ...

Many of these yield similar results
What does PCA accomplish?

Example: suppose two traits (generosity, trust) are highly correlated, to the point where each person either answers “1” to both or “5” to both.

This single PCA dimension entirely accounts for the two traits.
## The “Big Five” taxonomy

<table>
<thead>
<tr>
<th></th>
<th>Extraversion</th>
<th>Agreeableness</th>
<th>Conscientiousness</th>
<th>Neuroticism</th>
<th>Openness/Intellect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>.85 Talkative</td>
<td>.85 Sympathetic</td>
<td>.87 Careless</td>
<td>.80 Organized</td>
<td>.74 Commonplace</td>
</tr>
<tr>
<td>Low</td>
<td>.80 Reserved</td>
<td>.85 Kind</td>
<td>.58 Disorderly</td>
<td>.80 Thouugh</td>
<td>.76 Wide interests</td>
</tr>
<tr>
<td>High</td>
<td>.79 Assertive</td>
<td>.85 Appreciative</td>
<td>.53 Frivolous</td>
<td>.78 Plantul</td>
<td>.73 Narrow interests</td>
</tr>
<tr>
<td>Low</td>
<td>.75 Shy</td>
<td>.85 Affectionate</td>
<td>.49 Irresponsible</td>
<td>.71 Moody</td>
<td>.73 Narrow interests</td>
</tr>
<tr>
<td>High</td>
<td>.72 Active</td>
<td>.84 Soft-hearted</td>
<td>.40 Slipsh</td>
<td>.71 Worrying</td>
<td>.68 Insightful</td>
</tr>
<tr>
<td>Low</td>
<td>.71 Silent</td>
<td>.82 Warm</td>
<td>.39 Undependable</td>
<td>.68 Touchy</td>
<td>.64 Curious</td>
</tr>
<tr>
<td>High</td>
<td>.70 Dominant</td>
<td>.81 Generous</td>
<td>.37 Forgful</td>
<td>.64 Fearful</td>
<td>.59 Sophisticated</td>
</tr>
<tr>
<td>Low</td>
<td>.67 Withdrawn</td>
<td>.78 Trusting</td>
<td>.31 Stern*</td>
<td>.63 High-strung</td>
<td>.59 Artistic</td>
</tr>
<tr>
<td>High</td>
<td>.66 Retiring</td>
<td>.77 Helpful</td>
<td>.28 Thankless</td>
<td>.63 Self-pitying</td>
<td>.59 Clever</td>
</tr>
<tr>
<td>.72 Forceful</td>
<td>.72 Unkind</td>
<td>.77 Forgiving</td>
<td>.24 Stingy*</td>
<td>.60 Temperamental</td>
<td>.58 Inventive</td>
</tr>
<tr>
<td>.73 Enthusiastic</td>
<td>.74 Scrupulous</td>
<td>.74 Pleasent</td>
<td>.14 Unemotional*</td>
<td>.59 Unstable</td>
<td>.56 Sharp-witted</td>
</tr>
<tr>
<td>.68 Show-off</td>
<td>.73 Good-natured</td>
<td>.73 Good</td>
<td>.39 Stable*</td>
<td>.58 Self-punishing</td>
<td>.55 Ingenious</td>
</tr>
<tr>
<td>.68 Sociable</td>
<td>.73 Friendly</td>
<td>.72 Cooperative</td>
<td>.26 Cautious*</td>
<td>.54 Despondent</td>
<td>.45 Witty*</td>
</tr>
<tr>
<td>.64 Spankky</td>
<td>.72 Cooperative</td>
<td>.67 Gentle</td>
<td></td>
<td>.51 Emotional</td>
<td>.37 Wise</td>
</tr>
<tr>
<td>.64 Adventurous</td>
<td>.66 Unselfish</td>
<td>.66 Pleading</td>
<td></td>
<td></td>
<td>.33 Logical*</td>
</tr>
<tr>
<td>.62 Noisy</td>
<td>.65 Pratical</td>
<td>.65 Pleading</td>
<td></td>
<td></td>
<td>.29 Civilized*</td>
</tr>
<tr>
<td>.58 Bossy</td>
<td>.66 Unselfish</td>
<td>.66 Pleading</td>
<td></td>
<td></td>
<td>.22 Foresighted*</td>
</tr>
<tr>
<td></td>
<td>.66 Unselfish</td>
<td>.66 Pleading</td>
<td></td>
<td></td>
<td>.21 Polished*</td>
</tr>
<tr>
<td></td>
<td>.59 Unselfish</td>
<td>.59 Unselfish</td>
<td></td>
<td></td>
<td>.20 Dignified*</td>
</tr>
</tbody>
</table>

### Note

These 112 items were selected as initial prototypes for the Big Five because they were assigned to one factor by at least 90% of the judges. The factor loadings, shown for the hypothesized factor, were based on a sample of 140 males and 140 females, each of whom had been described by 10 psychologists serving as observers during an assessment weekend at the Institute of Personality Assessment and Research at the University of California at Berkeley (John, 1990).

*Potentially misclassified items (i.e., loading more highly on a factor different from the one hypothesized in the original prototype definition)

Many applications, such as online match-making.
Singular value decomposition (SVD)

For **symmetric** matrices, such as covariance matrices, we have seen:

- Results about existence of eigenvalues and eigenvectors
- The fact that the eigenvectors form an alternative basis
- The resulting spectral decomposition, which is used in PCA

But what about arbitrary matrices $M \in \mathbb{R}^{p \times q}$?

Any $p \times q$ matrix (say $p \leq q$) has a **singular value decomposition**:

$$M = \underbrace{\begin{pmatrix} u_1 & \cdots & u_p \end{pmatrix}}_{p \times p \text{ matrix } U} \underbrace{\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix}}_{p \times p \text{ matrix } \Lambda} \underbrace{\begin{pmatrix} v_1 \\ \vdots \\ v_p \end{pmatrix}}_{p \times q \text{ matrix } V^T}$$

- $u_1, \ldots, u_p$ are orthonormal vectors in $\mathbb{R}^p$
- $v_1, \ldots, v_p$ are orthonormal vectors in $\mathbb{R}^q$
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$ are **singular values**
Matrix approximation

We can factor any $p \times q$ matrix as $M = UW^T$:

$$M = \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ u_1 & \cdots & u_p \end{pmatrix} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix} \begin{pmatrix} \leftarrow & v_1 \rightarrow \\ \vdots \end{pmatrix} = \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ u_1 & \cdots & u_p \end{pmatrix} \begin{pmatrix} \leftarrow & \sigma_1 v_1 \rightarrow \\ \vdots \end{pmatrix}$$

$$= \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ u_1 & \cdots & u_p \end{pmatrix} \begin{pmatrix} \leftarrow & \sigma_p v_p \rightarrow \end{pmatrix}$$

A concise approximation to $M$: just take the first $k$ columns of $U$ and the first $k$ rows of $W^T$, for $k < p$:

$$\hat{M} = \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ u_1 & \cdots & u_k \end{pmatrix} \begin{pmatrix} \leftarrow & \sigma_1 v_1 \rightarrow \\ \vdots \end{pmatrix} \begin{pmatrix} \leftarrow & \sigma_k v_k \rightarrow \end{pmatrix}$$

$$= \begin{pmatrix} \uparrow & \uparrow & \uparrow \\ u_1 & \cdots & u_k \end{pmatrix} \begin{pmatrix} \leftarrow & \sigma_k v_k \rightarrow \end{pmatrix}$$
Example: topic modeling

Blei (2012):

### Seeking Life’s Bare (Genetic) Necessities

*Cold Spring Harbor, New York— How many genes does an organism need to survive? Last week at the genome meeting here, two genome researchers with radically different approaches presented complementary views of the basic genes needed for life.*

One research team, using computer analyses to compare known *genomes*, concluded that today’s organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn’t be enough.

Although the numbers don’t match precisely, those predictions...
Latent semantic indexing (LSI)

Given a large corpus of \( n \) documents:

- Fix a vocabulary, say of \( V \) words.
- Bag-of-words representation for documents: each document becomes a vector of length \( V \), with one coordinate per word.
- The corpus is an \( n \times V \) matrix, one row per document.

Let's find a concise approximation to this matrix \( M \).
Latent semantic indexing, cont’d

Use SVD to get an approximation to $M$: for small $k$,

$$M \approx \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix} \begin{pmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_k \end{pmatrix}$$

Think of this as a topic model with $k$ topics.

- $\Psi_j$ is a vector of length $V$ describing topic $j$: coefficient $\Psi_{jw}$ is large if word $w$ appears often in that topic.
- Each document is a combination of topics: $\theta_{ij}$ is the weight of topic $j$ in document $i$.

Document $i$ originally represented by $i$th row of $M$, a vector in $\mathbb{R}^V$. Can instead use $\theta_i \in \mathbb{R}^k$, a more concise “semantic” representation.
The rank of a matrix

Suppose we want to approximate a matrix $M$ by a simpler matrix $\hat{M}$. What is a suitable notion of “simple”?

- Let’s say $M$ and $\hat{M}$ are $p \times q$, where $p \leq q$.
- Treat each row of $\hat{M}$ as a data point in $\mathbb{R}^q$.
- We can think of the data as “simple” if it actually lies in a low-dimensional subspace.
- If the rows lie in $k$-dimensional subspace, we say that $\hat{M}$ has rank $k$.

The rank of a matrix is the number of linearly independent rows.

**Low-rank approximation:** given $M \in \mathbb{R}^{p \times q}$ and an integer $k$, find the matrix $\hat{M} \in \mathbb{R}^{p \times q}$ that is the best rank-$k$ approximation to $M$.

That is, find $\hat{M}$ so that

- $\hat{M}$ has rank $\leq k$
- The approximation error $\sum_{i,j} (M_{ij} - \hat{M}_{ij})^2$ is minimized.

We can get $\hat{M}$ directly from the singular value decomposition of $M$. 
Low-rank approximation

Recall: Singular value decomposition of $p \times q$ matrix $M$ (with $p \leq q$):

$$M = \begin{pmatrix}
\uparrow & \uparrow & \cdots & \uparrow \\
u_1 & \cdots & u_p \\
\downarrow & \downarrow & \cdots & \downarrow
\end{pmatrix}
\begin{pmatrix}
\sigma_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \sigma_p
\end{pmatrix}
\begin{pmatrix}
\underbrace{\mathbf{v}_1} & \\
\underbrace{\mathbf{v}_2} & \\
\underbrace{\mathbf{v}_p}
\end{pmatrix}
$$

- $u_1, \ldots, u_p$ is an orthonormal basis of $\mathbb{R}^p$
- $\mathbf{v}_1, \ldots, \mathbf{v}_q$ is an orthonormal basis of $\mathbb{R}^q$
- $\sigma_1 \geq \cdots \geq \sigma_p$ are singular values

The best rank-$k$ approximation to $M$, for any $k \leq p$, is then

$$\hat{M} = \begin{pmatrix}
\uparrow & \uparrow & \cdots & \uparrow \\
u_1 & \cdots & u_k \\
\downarrow & \downarrow & \cdots & \downarrow
\end{pmatrix}
\begin{pmatrix}
\sigma_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \sigma_k
\end{pmatrix}
\begin{pmatrix}
\underbrace{\mathbf{v}_1} & \\
\underbrace{\mathbf{v}_2} & \\
\underbrace{\mathbf{v}_k}
\end{pmatrix}
$$

$p \times k$  $k \times k$  $k \times q$
Example: Collaborative filtering

Details and images from Koren, Bell, Volinksy (2009).

Recommender systems: matching customers with products.
  - Given: data on prior purchases/interests of users
  - Recommend: further products of interest

Prototypical example: Netflix.

A successful approach: **collaborative filtering**.
  - Model dependencies between different products, and between different users.
  - Can give reasonable recommendations to a relatively new user.

Two strategies for collaborative filtering:
  - Neighborhood methods
  - Latent factor methods
Neighborhood methods

Well-defined dimensions such as depth of character development or quirkiness; or completely uninterpretable dimensions. For users, each factor measures how much the user likes movies that score high on the corresponding movie factor.

Figure 2 illustrates this idea for a simplified example in two dimensions. Consider two hypothetical dimensions characterized as female- versus male-oriented and serious versus escapist. The figure shows where several well-known movies and a few fictitious users might fall on these two dimensions. For this model, a user's predicted rating for a movie, relative to the movie's average rating, would equal the dot product of the movie's and user's locations on the graph. For example, we would expect Gus to love Dumb and Dumber, to hate The Color Purple, and to rate Braveheart about average. Note that some movies—for example, Ocean's 11—and users—for example, Dave—would be characterized as fairly neutral on these two dimensions.

Matrix factorization

Some of the most successful realizations of latent factor models are based on matrix factorization. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a similarity score between the user and the movie. For example, the user-oriented approach evaluates a user's preference for an item based on ratings of "neighboring" items by the same user. A product's neighbors are other products that tend to get similar ratings when rated by the same user. For example, consider the movie Saving Private Ryan. Its neighbors might include war movies, Spielberg movies, and Tom Hanks movies, among others. To predict a particular user's rating for Saving Private Ryan, we would look for the movie's nearest neighbors that this user actually rated. As Figure 1 illustrates, the user-oriented approach identifies like-minded users who can complement each other's ratings.

Collaborative filtering

An alternative to content filtering relies only on past user behavior—for example, previous transactions or product ratings—without requiring the creation of explicit profiles. This approach is known as collaborative filtering, a term coined by the developers of Tapestry, the first recommender system. Collaborative filtering analyzes relationships between users and interdependencies among products to identify new user-item associations. A major appeal of collaborative filtering is that it is domain free, yet it can address data aspects that are often elusive and difficult to profile using content filtering. While generally more accurate than content-based techniques, collaborative filtering suffers from what is called the cold start problem, due to its inability to address the system's new products and users. In this aspect, content filtering is superior.

The two primary areas of collaborative filtering are the neighborhood methods and latent factor models. Neighborhood methods are centered on computing the relationships between items or, alternatively, between users. The item-oriented approach evaluates a user's preference for an item based on ratings of "neighboring" items by the same user. A product's neighbors are other products that tend to get similar ratings when rated by the same user. For example, consider the movie Saving Private Ryan. Its neighbors might include war movies, Spielberg movies, and Tom Hanks movies, among others. To predict a particular user's rating for Saving Private Ryan, we would look for the movie's nearest neighbors that this user actually rated. As Figure 1 illustrates, the user-oriented approach identifies like-minded users who can complement each other's ratings.
Latent factor methods

Matrix factorization models map both users and items to a joint latent factor space of dimensionality \( f \), such that each item \( q \) is associated with a vector \( q \in \mathbb{R}^f \), and each user \( u \) is associated with a vector \( p_\in \mathbb{R}^f \). For a given item \( i \), the user's overall interest in the item's characteristics is measured by the user's rating of item \( \hat{r}_{ui} = p_u^T q_i \). For a given user \( u \), capturing the interaction between user \( u \) and item \( i \) is denoted by \( r_{ui} \), leading to the estimate \( \hat{r}_{ui} = p_u^T q_i \).

We refer to explicit feedback as user feedback as given in response to the presentation of an item, such as clicks, purchases, or ratings and make the rating matrix dense. Usually, explicit feedback comprises a dense matrix, since any single user is likely to have viewed any single item; thus, any single item is likely to have resided in a user's view history, browsing history, search patterns, or even the item's characteristics. This approximation of the user-item interaction is obtained by computing the map \( p \) of each user and map \( q \) of each item to factor vectors \( p \in \mathbb{R}^f \) and \( q \in \mathbb{R}^f \), completing this mapping, it can easily estimate each user's rating of each item and user to factor vectors \( p \) and \( q \), respectively. Upon estimating the user-item interactions as inner products in the factor space, \( r_{ui} = p_u R_i q_i \), such that \( r_{ui} = p_u^T q_i \), we can use Equation 1.

\[ r_{ui} = p_u^T q_i \]

The system learns the model by fitting the previously observed ratings: \( \min \| r - p^T q \|^2 \) to a joint latent factor space of dimensionality \( f \), such that \( \| r \|^2 = \sum_{i=1}^{k} r_{ui}^2 \) and \( \| p \|^2 = \sum_{i=1}^{k} p_{ui}^2 \). After the recommender system computes the map \( p \) and \( q \), it can easily estimate each user's rating of each item and user to factor vectors \( p \) and \( q \), respectively.

\[ \hat{r}_{ui} = p_u^T q_i \]

The major challenge is computing the map \( p \) and \( q \), which is denoted by \( r_{ui} \), leading to the estimate \( \hat{r}_{ui} = p_u^T q_i \). This approximation of the user-item interaction is obtained by computing the map \( p \) of each user and map \( q \) of each item to factor vectors \( p \in \mathbb{R}^f \) and \( q \in \mathbb{R}^f \), respectively. Upon estimating the user-item interactions as inner products in the factor space, \( r_{ui} = p_u^T q_i \), such that \( r_{ui} = p_u^T q_i \), we can use Equation 1.

\[ r_{ui} = p_u^T q_i \]

The system learns the model by fitting the previously observed ratings: \( \min \| r - p^T q \|^2 \) to a joint latent factor space of dimensionality \( f \), such that \( \| r \|^2 = \sum_{i=1}^{k} r_{ui}^2 \) and \( \| p \|^2 = \sum_{i=1}^{k} p_{ui}^2 \). After the recommender system computes the map \( p \) and \( q \), it can easily estimate each user's rating of each item and user to factor vectors \( p \) and \( q \), respectively.

\[ \hat{r}_{ui} = p_u^T q_i \]
The matrix factorization approach

User ratings are assembled in a large matrix $M$:

<table>
<thead>
<tr>
<th></th>
<th>Star Wars</th>
<th>Matrix</th>
<th>Casablanca</th>
<th>Camelot</th>
<th>Godfather</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>User 2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Not rated = 0, otherwise scores 1-5.
- For $n$ users and $p$ movies, this has size $n \times p$.
- Most of the entries are unavailable, and we’d like to predict these.

Idea: Find the best low-rank approximation of $M$, and use it to fill in the missing entries.
User and movie factors

Best rank-$k$ approximation is of the form $M \approx UW^T$:

$$
\begin{pmatrix}
\begin{array}{c}
\text{user 1} \\
\text{user 2} \\
\text{user 3} \\
\vdots \\
\text{user } n
\end{array}
\end{pmatrix}
\approx
\begin{pmatrix}
\begin{array}{c}
\text{u}_1 \\
\text{u}_2 \\
\text{u}_3 \\
\vdots \\
\text{u}_n
\end{array}
\end{pmatrix}
\begin{pmatrix}
\begin{array}{c}
w_1 \\
w_2 \\
\cdots \\
w_p
\end{array}
\end{pmatrix}
$$

Thus user $i$’s rating of movie $j$ is approximated as

$$M_{ij} \approx u_i \cdot w_j$$

This “latent” representation embeds users and movies within the same $k$-dimensional space:

- Represent $i$th user by $u_i \in \mathbb{R}^k$
- Represent $j$th movie by $w_j \in \mathbb{R}^k$
Our winning entries consist of more than 100 different predictor sets, the majority of which are factorization models using some variants of the methods described here. Models using these variants have been validated in public contest forums and are the most popular and successful methods for predicting ratings.

Through our discussions with other top teams and postings on the public contest forum, we know that these are the most popular methods. Hence, our contest entries consist of more than 100 different predictor sets, the majority of which are factorization models using some variants of the methods described here. Models using these variants have been validated in public contest forums and are the most popular and successful methods for predicting ratings.