PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA is probably the most common "dimension reduction" tool in data analysis. As an example, we use the Netflix Challenge, a dataset of ratings, on a 1-5 scale, of about 18,000 films by 500,000 people. Here's how this data table might look like:

<table>
<thead>
<tr>
<th>Person 1 (Alice)</th>
<th>Film 1 (Ace Ventura)</th>
<th>Film 2 (The Alamo)</th>
<th>Film 3 (Avatar)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 person 1</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1 person 2 (Bob)</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

\( M = 500,000 \) rows \( N = 18,000 \) columns.

This is a huge matrix, call it \( A \), with entries \( a_{ij} = i = \text{person index} \) \( j = \text{film index} \).

To explain PCA, let's imagine everyone rated every movie: \( A \) is entirely known.

The idea behind PCA is that Alice's high rating for Avatar \( (a_{13} = 5) \) is controlled by an unknown number of "latent" factors that both shape Alice's taste (e.g., she likes sci-fi, dislikes violence), and describe films (Avatar is futuristic but non-violent).

Bob's taste differs (he likes sci-fi, but more so violence), explaining his higher rating for The Alamo over Avatar. The reaction of people to films is dominated, in this model, by their reaction to factors ("violence", "futuristic", etc.).

PCA extracts these factors, ranking them most to least important.

Let's plot Alice's ratings as a point in 3D, "ratings space":

\( (x, y, z) = (4, 2, 5) \)

In fact there's 18,000 dimensions, but we can only sketch the first 3!
PCA extracts the crude geometry of this point cloud: the 1st principal component (eigenvector \( v_1 \)) is the cloud's longest axis, i.e., the factor explaining the most variance in ratings. The 2nd P.C. is the direction \( v_2 \), transversal (at right angles) to the 1st, of most remaining variance, and so on. 

In this way, the gross shape of the point cloud may be captured by a few directions of spread, even though it lives in a huge \((18000)\) dimension space. 

- Mean subtraction: you'll notice the P.C. axes are drawn from the "center of mass" of the cloud. This is because, e.g., Avatar may receive a higher average rating than other films, which is not a "latent" effect. Thus, before PCA is done, the average of each column is subtracted from the data matrix \( A \), which shifts the origin to be at the \( \Theta \) symbol. Since Alice may be universally more generous than Bob, the row means are also subtracted.

- In matrix language: 

\[
A \approx U \Sigma V^T
\]

This is an (approximate) singular value decomposition (SVD) of \( A \). The number of factors is \( K \) (in the above example, \( K=2 \)).

Another way to write this is: 

\[
a_{ij} = \sum_{k=1}^{K} \sigma_k u_{ik} v_{jk}, \quad \text{for all } i=1...M, \quad j=1...N.
\]

In practice, the SVD of \( A \) is taken (if not too huge), or eigenvectors of \( A^TA \) are found by iteration.
Afternotes on PCA:

- Replacing the film (column) label by space, and the person (row) label by time, PCA becomes a way to extract, from measurements such as temperature recorded over space and time, the dominant "modes" controlling temperature. This is very common in geophysical & climate analysis, where it is called "empirical orthogonal functions" (EOF).

Each mode (each term \( u_{ik}v_{jk} \) in \( a_{ij} \)) is by definition "separable," i.e., a product of a function of space only (\( v_{jk} \)) and a function of time only (\( u_{ik} \)). To illustrate,

\[ u_{ik} \]
\[ \downarrow \]
\[ (ori) \]
\[ \downarrow \]
\[ t \]
\[ \text{time} \]

\[ v_{jk} \]
\[ \rightarrow \]
\[ x \]
\[ \text{(ori)} \]
\[ \rightarrow \]
\[ \text{space} \]

\[ \rightarrow \]
\[ X \]
\[ \text{EOF} \]

\[ \rightarrow \]
\[ t \]
\[ \text{EOF} \]

Note: no single EOF mode can capture this:

Example: "traveling wave" of temperature, a feature EOF cannot easily decompose.

- In fact, only 1% of the Netflix data matrix entries were known (99% of films were unrated by the average person). This is why the Task (a low-rank "matrix completion" problem) was a challenge, more than usual PCA. The "training set" of known entries was still huge (100 million entries). The $1M prize was given in 2009 for the algorithm which first improved prediction accuracy, on a hidden "test set" of 3 million entries, by 10%. Latent factor (PCA-based) models played a huge role in the successful algorithms, and continue to do so in "collaborative filtering" (recommendation systems online).