CSC 311: Introduction to Machine Learning
Tutorial - Matrix Completion, Auto-Encoders & PCA

TA: Vahid Balazadeh
Instructors: Michael Zhang and Chandra Gummaluru

University of Toronto
Today

- Review of PCA
- Matrix completion
- Nonlinear dimensionality reduction (Auto-Encoders)
- Exercise
PCA - Projection onto a subspace

- How to project onto a $K$-dimensional subspace?
  - **Idea:** choose an orthonormal basis \( \{ \mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K \} \) for \( S \) (i.e. all unit vectors and orthogonal to each other)
  - Project onto each unit vector individually (as in previous slide), and sum together the projections.

- Mathematically, the projection is given as:

\[
\text{Proj}_S(\mathbf{x}) = \sum_{i=1}^{K} z_i \mathbf{u}_i \quad \text{where} \quad z_i = \mathbf{x}^\top \mathbf{u}_i.
\]

- In vector form:

\[
\text{Proj}_S(\mathbf{x}) = \mathbf{Uz} \quad \text{where} \quad \mathbf{z} = \mathbf{U}^\top \mathbf{x}
\]
PCA - Projection onto a subspace

- We assumed the subspace passes through 0.
- In mathematical terminology, the “subspaces” we want to project onto are really affine spaces, and can have an arbitrary origin $\hat{\mu}$.

$$\tilde{x} = Uz + \hat{\mu} = z_1u_1 + z_2u_2 + \hat{\mu}$$

In machine learning, $\tilde{x}$ is also called the reconstruction of $x$.
- $z$ is its representation, or code.
PCA - Learning a Subspace

- How to choose a good subspace $S$?
  - Origin $\hat{\mu}$ is the empirical mean of the data
  - Need to choose a $D \times K$ matrix $U$ with orthonormal columns.

- Two criteria:
  - Minimize the reconstruction error:
    $$\min_U \frac{1}{N} \sum_{i=1}^{N} \| \mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)} \|^2$$
  - Maximize the variance of reconstructions: Find a subspace where data has the most variability.
    $$\max_U \frac{1}{N} \sum_i \| \tilde{\mathbf{x}}^{(i)} - \hat{\mu} \|^2$$
  - These two criteria are equivalent! I.e., we’ll show
    $$\frac{1}{N} \sum_{i=1}^{N} \| \mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)} \|^2 = \text{const} - \frac{1}{N} \sum_i \| \tilde{\mathbf{x}}^{(i)} - \hat{\mu} \|^2$$
Choosing a subspace to maximize the projected variance, or minimize the reconstruction error, is called principal component analysis (PCA).

- Consider the empirical covariance matrix:

  \[ \hat{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \hat{\mu})(x^{(i)} - \hat{\mu})^\top \]

- Recall: \( \hat{\Sigma} \) is symmetric and positive semidefinite.

- The optimal PCA subspace is spanned by the top \( K \) eigenvectors of \( \hat{\Sigma} \).
  - More precisely, choose the first \( K \) of any orthonormal eigenbasis for \( \hat{\Sigma} \).

- These eigenvectors are called principal components, analogous to the principal axes of an ellipse.
Recap:

- Dimensionality reduction aims to find a low-dimensional representation of the data.
- PCA projects the data onto a subspace which maximizes the projected variance, or equivalently, minimizes the reconstruction error.
- The optimal subspace is given by the top eigenvectors of the empirical covariance matrix.
Two more interpretations of PCA, which have interesting generalizations.

1. Matrix factorization
2. Autoencoder
The Netflix problem

Movie recommendation: Users watch movies and rate them out of 5★.

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>⭐️</td>
<td>Thor</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>⭐️</td>
<td>Chained</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>⭐️</td>
<td>Frozen</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Chained</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Bambi</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Titanic</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Goodfellas</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Dumbo</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Twilight</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Frozen</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
<tr>
<td>🐰</td>
<td>Tangled</td>
<td>⭐⭐⭐⭐⭐</td>
</tr>
</tbody>
</table>

Because users only rate a few items, one would like to infer their preference for unrated items.
PCA as Matrix Factorization

- Recall PCA: each input vector $\mathbf{x}^{(i)} \in \mathbb{R}^D$ is approximated as $\hat{\mu} + \mathbf{Uz}^{(i)}$,

$$\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)} = \hat{\mu} + \mathbf{Uz}^{(i)}$$

where $\hat{\mu} = \frac{1}{n} \sum_i \mathbf{x}^{(i)}$ is the data mean, $\mathbf{U} \in \mathbb{R}^{D \times K}$ is the orthogonal basis for the principal subspace, and $\mathbf{z}^{(i)} \in \mathbb{R}^K$ is the code vector, and $\tilde{\mathbf{x}}^{(i)} \in \mathbb{R}^D$ is $\mathbf{x}^{(i)}$'s reconstruction or approximation.

- Assume for simplicity that the data is centered: $\hat{\mu} = 0$. Then, the approximation looks like

$$\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)} = \mathbf{Uz}^{(i)}.$$
PCA as Matrix Factorization

- PCA (on centered data): input vector $x^{(i)}$ is approximated as $Uz^{(i)}$
  \[ x^{(i)} \approx Uz^{(i)} \]

- Write this in matrix form, we have $X \approx ZU^\top$ where $X$ and $Z$ are matrices with one row per data point

  \[
  X = \begin{bmatrix}
  [x^{(1)}]^\top \\
  [x^{(2)}]^\top \\
  \vdots \\
  [x^{(N)}]^\top
  \end{bmatrix} \in \mathbb{R}^{N \times D}
  \quad \text{and} \quad
  Z = \begin{bmatrix}
  [z^{(1)}]^\top \\
  [z^{(2)}]^\top \\
  \vdots \\
  [z^{(N)}]^\top
  \end{bmatrix} \in \mathbb{R}^{N \times K}
  \]

- Can write the squared reconstruction error as
  \[
  \sum_{i=1}^{N} \| x^{(i)} - Uz^{(i)} \|^2 = \| X - ZU^\top \|^2_F,
  \]

- $\| \cdot \|_F$ denotes the Frobenius norm:
  \[
  \| Y \|^2_F = \| Y^\top \|^2_F = \sum_{i,j} y_{ij}^2 = \sum_i \| y^{(i)} \|^2.
  \]
So PCA is approximating $\mathbf{X} \approx \mathbf{ZU}^\top$, or equivalently $\mathbf{X}^\top \approx \mathbf{UZ}^\top$.

Based on the sizes of the matrices, this is a rank-$K$ approximation.

Since $\mathbf{U}$ was chosen to minimize reconstruction error, this is the *optimal* rank-$K$ approximation, in terms of error $\|\mathbf{X}^\top - \mathbf{UZ}^\top\|_F^2$. 
Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization to a matrix $X$.
- Can we generalize this to the case where $X$ is only partially observed?
  - A sparse $1000 \times 1000$ matrix with 50,000 observations (only 5% observed).
  - A rank 5 approximation requires only 10,000 parameters, so it’s reasonable to fit this.
  - Unfortunately, no closed form solution.
The Netflix problem

Movie recommendation: Users watch movies and rate them as good or bad.

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>🎥</td>
<td>Thor</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🎥</td>
<td>Chained</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🎥</td>
<td>Frozen</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🐏</td>
<td>Chained</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🐏</td>
<td>Bambi</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🎥</td>
<td>Titanic</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🎥</td>
<td>Goodfellas</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>🎥</td>
<td>Dumbo</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>😱</td>
<td>Twilight</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>😱</td>
<td>Frozen</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
<tr>
<td>😞</td>
<td>Tangled</td>
<td>★ ★ ★ ★ ★ ★</td>
</tr>
</tbody>
</table>
Matrix completion problem: Transform the table into a $N$ users by $M$ movies matrix $\mathbf{R}$

- **Data**: Users rate some movies. $\mathbf{R}_{\text{user,movie}}$. Very sparse
- **Task**: Predict missing entries, i.e. how a user would rate a movie they haven’t previously rated
- **Evaluation Metric**: Squared error (used by Netflix Competition). Is this a reasonable metric?
In our current setting, latent factor models attempt to explain the ratings by characterizing both movies and users on a number of factors $K$ inferred from the ratings patterns.

That is, we seek representations for movies and users as vectors in $\mathbb{R}^K$ that can ultimately be translated to ratings.

For simplicity, we can associate these factors (i.e. the dimensions of the vectors) with idealized concepts like:
- comedy
- drama
- action
- But also uninterpretable dimensions

Can we use the sparse ratings matrix $\mathbf{R}$ to find these latent factors automatically?
Matrix Completion

- Let the representation of user $i$ in the $K$-dimensional space be $\mathbf{u}_i$ and the representation of movie $j$ be $\mathbf{z}_j$
  - Intuition: maybe the first entry of $\mathbf{u}_i$ says how much the user likes horror films, and the first entry of $\mathbf{z}_j$ says how much movie $j$ is a horror film.

- Assume the rating user $i$ gives to movie $j$ is given by a dot product: $R_{ij} \approx \mathbf{u}_i^\top \mathbf{z}_j$

- In matrix form, if:

  $$
  \mathbf{U} = 
  \begin{bmatrix}
  - & \mathbf{u}_1^\top & - \\
  \vdots & \ddots & \vdots \\
  - & \mathbf{u}_N^\top & - 
  \end{bmatrix}
  \quad \text{and} \quad
  \mathbf{Z}^\top = 
  \begin{bmatrix}
  | & | & | \\
  \mathbf{z}_1 & \cdots & \mathbf{z}_M 
  \end{bmatrix}
  $$

then: $\mathbf{R} \approx \mathbf{UZ}^\top$

- This is a matrix factorization problem!
Matrix Completion

- Recall PCA: To enforce $\mathbf{X}^\top \approx \mathbf{UZ}^\top$, we minimized

  $$
  \min_{\mathbf{U}, \mathbf{Z}} \|\mathbf{X}^\top - \mathbf{UZ}^\top\|_F^2 = \sum_{i,j}(x_{ji} - u_i^\top z_j)^2
  $$

  where $u_i$ and $z_i$ are the $i$-th rows of matrices $\mathbf{U}$ and $\mathbf{Z}$, respectively.

- What’s different about the Netflix problem?
  - Most entries are missing!
  - We only want to count the error for the observed entries.
Matrix Completion

- Let \( O = \{(n, m) : \text{entry } (n, m) \text{ of matrix } R \text{ is observed}\} \)

- Using the squared error loss, matrix completion requires solving

\[
\min_{U, Z} \frac{1}{2} \sum_{(i, j) \in O} \left( R_{i,j} - u_i^T z_j \right)^2
\]

- The objective is non-convex in \( U \) and \( Z \) jointly, and in fact it’s generally NP-hard to minimize the above cost function exactly.

- As a function of either \( U \) or \( Z \) individually, the problem is convex and easy to optimize. We can use coordinate descent, just like with K-means!

**Alternating Least Squares (ALS):** fix \( Z \) and optimize \( U \), followed by fix \( U \) and optimize \( Z \), and so on until convergence.
Alternating Least Squares

- Want to minimize the squared error cost with respect to the factor $U$. (The case of $Z$ is exactly symmetric.)
- We can decompose the cost into a sum of independent terms:

$$
\sum_{(i,j) \in O} (R_{ij} - u_i^T z_j)^2 = \sum_i \sum_{j:(i,j) \in O} (R_{ij} - u_i^T z_j)^2
$$

This can be minimized independently for each $u_i$.
- This is a linear regression problem in disguise. Its optimal solution is:

$$
u_i = \left( \sum_{j:(i,j) \in O} z_j z_j^T \right)^{-1} \sum_{j:(i,j) \in O} R_{ij} z_j$$
Alternating Least Squares

ALS for Matrix Completion problem

1. Initialize $U$ and $Z$ randomly
2. repeat until convergence
3. for $i = 1, \ldots, N$ do
4. $u_i = \left( \sum_{j: (i,j) \in O} z_j z_j^\top \right)^{-1} \sum_{j: (i,j) \in O} R_{ij} z_j$
5. for $j = 1, \ldots, M$ do
6. $z_j = \left( \sum_{i: (i,j) \in O} u_i u_i^\top \right)^{-1} \sum_{i: (i,j) \in O} R_{ij} u_i$
Two more interpretations of PCA, which have interesting generalizations.

1. Matrix factorization

2. Autoencoder
An autoencoder is a feed-forward neural net whose job is to take an input $x$ and predict $x$.

To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input.
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabeled data can be much more plentiful than labeled data
Linear Autoencoders

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

\[ \mathcal{L}(x, \tilde{x}) = \|x - \tilde{x}\|^2 \]

- This network computes \( \tilde{x} = W_2 W_1 x \), which is a linear function.

- If \( K \geq D \), we can choose \( W_2 \) and \( W_1 \) such that \( W_2 W_1 \) is the identity matrix. This isn’t very interesting.

- But suppose \( K < D \):
  
  - \( W_1 \) maps \( x \) to a \( K \)-dimensional space, so it’s doing dimensionality reduction.
Linear Autoencoders

- Observe that the output of the autoencoder must lie in a $K$-dimensional subspace spanned by the columns of $W_2$. This is because $\tilde{x} = W_2 z$.
- We saw that the best possible (min error) $K$-dimensional linear subspace in terms of reconstruction error is the PCA subspace.
- The autoencoder can achieve this by setting $W_1 = U^T$ and $W_2 = U$.
- Therefore, the optimal weights for a linear autoencoder are just the principal components!
Nonlinear Autoencoders

- Deep nonlinear autoencoders learn to project the data, not onto a subspace, but onto a nonlinear manifold.
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.
Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA).
Nonlinear Autoencoders

Here’s a 2-dimensional autoencoder representation of newsgroup articles. They’re color-coded by topic, but the algorithm wasn’t given the labels.
Recall that the PCA code vector for a data point $x$ is given by $z = \mathbf{U}^\top (x - \mu)$. Show that the entries of $z$ are uncorrelated.
Recall that the PCA code vector for a data point $\mathbf{x}$ is given by $\mathbf{z} = \mathbf{U}^\top (\mathbf{x} - \hat{\mu})$. Show that the entries of $\mathbf{z}$ are uncorrelated.

\[
\text{Cov}(\mathbf{z}) = \mathbb{E}\left[ (\mathbf{z} - \mathbb{E}[\mathbf{z}]) (\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top \right] \\
= \mathbb{E}\left[ \mathbf{zz}^\top \right] \\
= \mathbf{U}^\top \mathbb{E}\left[ (\mathbf{x} - \hat{\mu})(\mathbf{x} - \hat{\mu})^\top \right] \mathbf{U} \\
= \mathbf{U}^\top \hat{\Sigma} \mathbf{U} \\
= \mathbf{U}^\top \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top \mathbf{U} \\
= \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{\Lambda} \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\]

Which is the top $K \times K$ block of $\mathbf{\Lambda}$. Matrix $\mathbf{\Lambda}$ is diagonal $\implies$ Uncorrelated features
Exercise

Consider the following data matrix, representing four samples $X_i \in \mathbb{R}^2$:

$$X = \begin{pmatrix}
4 & 1 \\
2 & 3 \\
5 & 4 \\
1 & 0
\end{pmatrix}$$

- Compute the unit-length principal component directions of $X$, and state which one the PCA algorithm would choose if you request just one principal component.
- Find the best (min reconstruction error) projection of $X$ into a 1-dimensional subspace with the origin of zero.