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 $\mathcal{S}$ (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:

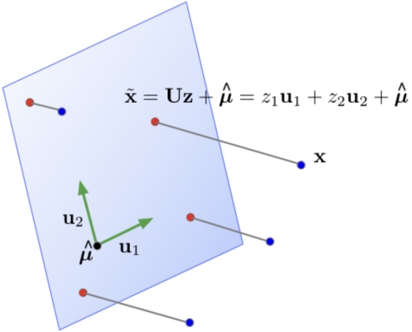
$$\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \sum_{i=1}^{K} z_i \mathbf{u}_i \quad \text{where} \quad z_i = \mathbf{x}^\top \mathbf{u}_i.$$

- In vector form:

$$\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \mathbf{U}\mathbf{z} \quad \text{where} \quad \mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

PCA - Projection onto a subspace

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



$$\mathbf{z} = \mathbf{U}^\top(\mathbf{x} - \hat{\boldsymbol{\mu}})$$

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

PCA - Learning a Subspace

- How to choose a good subspace $\mathcal{S}$?
  - Origin $\hat{\boldsymbol{\mu}}$ is the empirical mean of the data
  - Need to choose a $D \times K$ matrix $\mathbf{U}$ with orthonormal columns.
- Two criteria:
  - Minimize the reconstruction error:

$$\min_{\mathbf{U}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2$$

  - Maximize the variance of reconstructions: Find a subspace where data has the most variability.

$$\max_{\mathbf{U}} \frac{1}{N} \sum_{i} \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2$$

  - These two criteria are equivalent! I.e., we'll show

$$\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2 = \text{const} - \frac{1}{N} \sum_{i} \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2$$

Principal Component Analysis

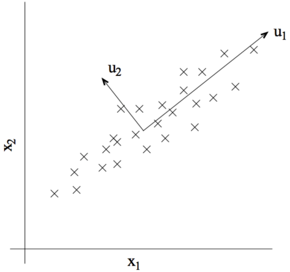
Choosing a subspace to maximize the projected variance, or minimize the reconstruction error, is called principal component analysis (PCA).

- Consider the empirical covariance matrix:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top$$

- Recall: $\hat{\boldsymbol{\Sigma}}$ is symmetric and positive semidefinite.

- The optimal PCA subspace is spanned by the top K eigenvectors of $\hat{\boldsymbol{\Sigma}}$.
  - More precisely, choose the first K of any orthonormal eigenbasis for $\hat{\boldsymbol{\Sigma}}$.
- These eigenvectors are called principal components, analogous to the principal axes of an ellipse.



Recap

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.

Next

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★.

User	Movie	Rating
	Thor	★ ☆ ☆ ☆ ☆
	Chained	★ ★ ☆ ☆ ☆
	Frozen	★ ★ ★ ☆ ☆
	Chained	★ ★ ★ ★ ☆
	Bambi	★ ★ ★ ★ ★
	Titanic	★ ★ ★ ☆ ☆
	Goodfellas	★ ★ ★ ★ ★
	Dumbo	★ ★ ★ ★ ★
	Twilight	★ ★ ☆ ☆ ☆
	Frozen	★ ★ ★ ★ ★
	Tangled	★ ☆ ☆ ☆ ☆

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 $\hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{z}^{(i)}$,

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

  where $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_i \mathbf{x}^{(i)}$ 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: $\hat{\boldsymbol{\mu}} = 0$. Then, the approximation looks like

$$\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)} = \mathbf{U}\mathbf{z}^{(i)}.$$

PCA as Matrix Factorization

- PCA(on centered data): input vector $\mathbf{x}^{(i)}$ is approximated as $\mathbf{U}\mathbf{z}^{(i)}$

$$\mathbf{x}^{(i)} \approx \mathbf{U}\mathbf{z}^{(i)}$$

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

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

- Can write the squared reconstruction error as

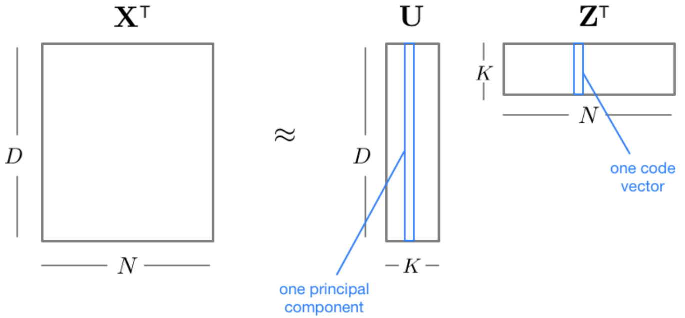
$$\sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \mathbf{U}\mathbf{z}^{(i)}\|^2 = \|\mathbf{X} - \mathbf{Z}\mathbf{U}^\top\|_F^2,$$

- $\|\cdot\|_F$ denotes the Frobenius norm:

$$\|\mathbf{Y}\|_F^2 = \|\mathbf{Y}^\top\|_F^2 = \sum_{i,j} y_{ij}^2 = \sum_i \|\mathbf{y}^{(i)}\|^2.$$

PCA as Matrix Factorization

- So PCA is approximating $\mathbf{X} \approx \mathbf{Z}\mathbf{U}^\top$, or equivalently $\mathbf{X}^\top \approx \mathbf{U}\mathbf{Z}^\top$.



- 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 $||\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top||_F^2$.

Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization to a matrix $\mathbf{X}$.
- Can we generalize this to the case where $\mathbf{X}$ is only partially observed?
  - A sparse 1000×1000 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

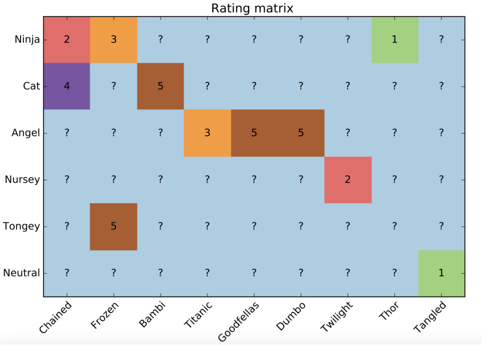
Movie recommendation: Users watch movies and rate them as good or bad.

User	Movie	Rating
	Thor	★ ☆ ☆ ☆ ☆
	Chained	★ ★ ☆ ☆ ☆
	Frozen	★ ★ ★ ☆ ☆
	Chained	★ ★ ★ ★ ☆
	Bambi	★ ★ ★ ★ ★
	Titanic	★ ★ ★ ☆ ☆
	Goodfellas	★ ★ ★ ★ ★
	Dumbo	★ ★ ★ ★ ★
	Twilight	★ ★ ☆ ☆ ☆
	Frozen	★ ★ ★ ★ ★
	Tangled	★ ☆ ☆ ☆ ☆

Because users only rate a few items, one would like to infer their preference for unrated items

Matrix Completion

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?

Matrix Completion

- 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:

$$\mathbf{U} = \begin{bmatrix} — & \mathbf{u}_1^\top & — \\ & \vdots & \\ — & \mathbf{u}_N^\top & — \end{bmatrix} \text{ and } \mathbf{Z}^\top = \begin{bmatrix} | & & | \\ \mathbf{z}_1 & \dots & \mathbf{z}_M \\ | & & | \end{bmatrix}$$

  then: $\mathbf{R} \approx \mathbf{U}\mathbf{Z}^\top$
- This is a matrix factorization problem!

Matrix Completion

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

$$\min_{\mathbf{U},\mathbf{Z}} \|\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top\|_{\mathrm{F}}^2 = \sum_{i,j} (x_{ji} - \mathbf{u}_i^\top \mathbf{z}_j)^2$$

  where $\mathbf{u}_i$ and $\mathbf{z}_i$ 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 $O = \{(n, m) : \text{ entry } (n, m) \text{ of matrix } \mathbf{R} \text{ is observed}\}$
- Using the squared error loss, matrix completion requires solving

$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(i,j)\in O} \left(R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j \right)^2$$

- 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 $\mathbf{U}$. (The case of $\mathbf{Z}$ is exactly symmetric.)
- We can decompose the cost into a sum of independent terms:

$$\sum_{(i,j)\in O} \left(R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j\right)^2 = \sum_i \underbrace{\sum_{j:(i,j)\in O} \left(R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j\right)^2}_{\text{only depends on } \mathbf{u}_i}$$

  This can be minimized independently for each $\mathbf{u}_i$.
- This is a linear regression problem in disguise. Its optimal solution is:

$$\mathbf{u}_i = \left(\sum_{j:(i,j)\in O} \mathbf{z}_j \mathbf{z}_j^\top\right)^{-1} \sum_{j:(i,j)\in O} R_{ij}\mathbf{z}_j$$

Alternating Least Squares

ALS for Matrix Completion problem

1. Initialize $\mathbf{U}$ and $\mathbf{Z}$ randomly
2. repeat until convergence
3. **for** $i = 1, .., N$ **do**
4. $\mathbf{u}_i = \left(\sum_{j:(i,j)\in O} \mathbf{z}_j \mathbf{z}_j^\top\right)^{-1} \sum_{j:(i,j)\in O} R_{ij}\mathbf{z}_j$
5. **for** $j = 1, .., M$ **do**
6. $\mathbf{z}_j = \left(\sum_{i:(i,j)\in O} \mathbf{u}_i \mathbf{u}_i^\top\right)^{-1} \sum_{i:(i,j)\in O} R_{ij}\mathbf{u}_i$

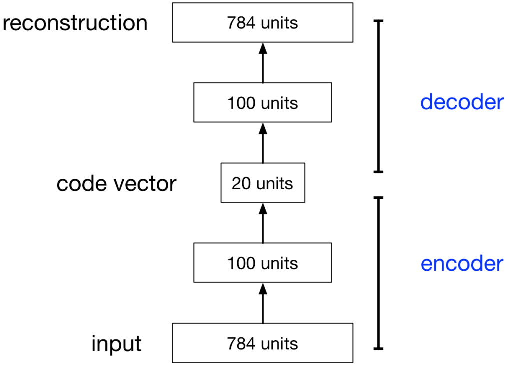
Next

Two more interpretations of PCA, which have interesting generalizations.

1. Matrix factorization
2. **Autoencoder**

Autoencoders

- An autoencoder is a feed-forward neural net whose job is to take an input $\mathbf{x}$ and predict $\mathbf{x}$.
- To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input.



Linear Autoencoders

Why autoencoders?

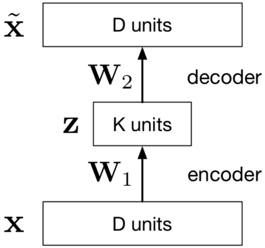
- Map high-dimensional data to two dimensions for visualization
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabled data can be much more plentiful than labeled data

Linear Autoencoders

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

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

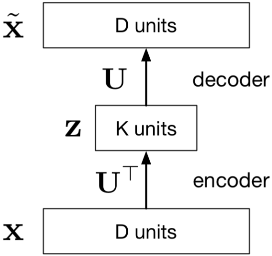
- This network computes $\tilde{\mathbf{x}} = \mathbf{W}_2\mathbf{W}_1\mathbf{x}$, which is a linear function.
- If $K \geq D$, we can choose $\mathbf{W}_2$ and $\mathbf{W}_1$ such that $\mathbf{W}_2\mathbf{W}_1$ is the identity matrix. This isn't very interesting.



- But suppose $K < D$:
  - $\mathbf{W}_1$ maps $\mathbf{x}$ to a K-dimensional space, so it's doing dimensionality reduction.

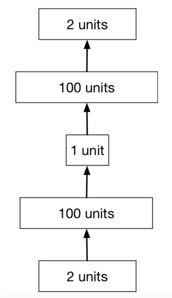
Linear Autoencoders

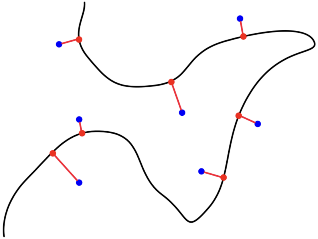
- Observe that the output of the autoencoder must lie in a K-dimensional subspace spanned by the columns of $\mathbf{W}_2$. This is because $\tilde{\mathbf{x}} = \mathbf{W}_2\mathbf{z}$
- We saw that the best possible (min error) K-dimensional linear subspace in terms of reconstruction error is the PCA subspace.
- The autoencoder can achieve this by setting $\mathbf{W}_1 = \mathbf{U}^\top$ and $\mathbf{W}_2 = \mathbf{U}$.
- Therefore, the optimal weights for a linear autoencoder are just the principal components!



Nonlinear Autoencoders

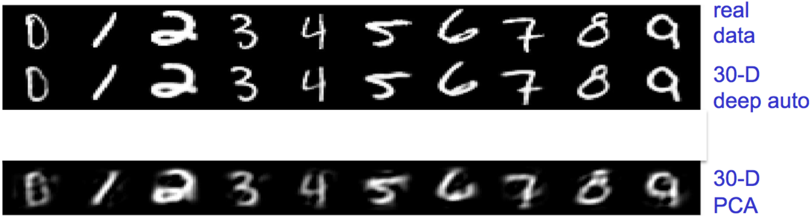
- Deep nonlinear autoencoders learn to project the data, not onto a subspace, but onto a nonlinear manifold
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.





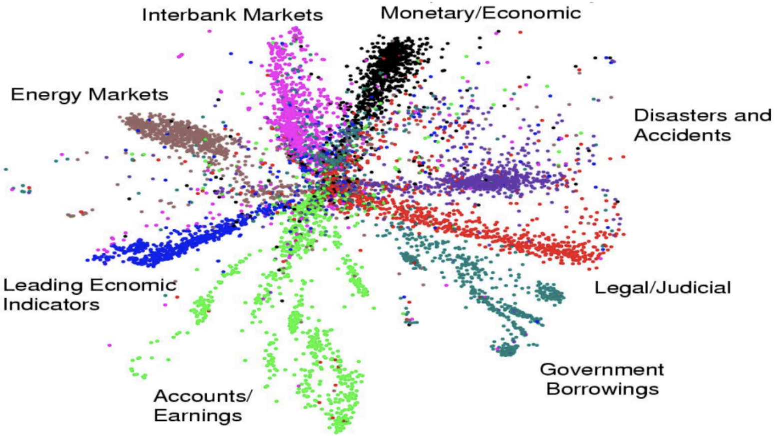
Nonlinear Autoencoders

- 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.



Exercise

Recall that the PCA code vector for a data point $\mathbf{x}$ is given by $\mathbf{z} = \mathbf{U}^\top(\mathbf{x} - \hat{\boldsymbol{\mu}})$. Show that the entries of $\mathbf{z}$ are uncorrelated.

Solution

Recall that the PCA code vector for a data point $\mathbf{x}$ is given by $\mathbf{z} = \mathbf{U}^\top(\mathbf{x} - \hat{\boldsymbol{\mu}})$. Show that the entries of $\mathbf{z}$ are uncorrelated.

$$\begin{aligned}
\mathrm{Cov}(\mathbf{z}) &= \mathbb{E}\left[(\mathbf{z} - \mathbb{E}[\mathbf{z}])(\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top\right] \\
&= \mathbb{E}\left[\mathbf{z}\mathbf{z}^\top\right] \\
&= \mathbf{U}^\top \mathbb{E}\left[(\mathbf{x} - \hat{\boldsymbol{\mu}})(\mathbf{x} - \hat{\boldsymbol{\mu}})^\top\right]\mathbf{U} \\
&= \mathbf{U}^\top \hat{\boldsymbol{\Sigma}} \mathbf{U} \\
&= \mathbf{U}^\top \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top \mathbf{U} \\
&= \begin{pmatrix} \mathbf{I} & 0 \end{pmatrix} \boldsymbol{\Lambda} \begin{pmatrix} \mathbf{I} \\ 0 \end{pmatrix}
\end{aligned}$$

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

Exercise

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

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

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