CS 335: Matrix Factorization and Principal Components Analysis

Dan Sheldon

Matrix Factorization

Movies: $R \approx UV^T$

- R: only some entries observed
- UV^T: lets you fill in missing entries

Unsupservised learning

Data: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^n$

Feature vectors, but **no labels**

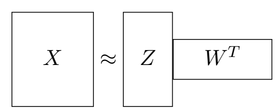
Goal: find patterns in data

Matrix Factorization for Unsupervised Learning

Given: $X \in \mathbb{R}^{m \times n}$ (data matrix, rows are feature vectors)

Find: $Z \in \mathbb{R}^{m \times k}$, $W \in \mathbb{R}^{n \times k}$ such that

$$X \approx ZW^T$$



$$\mathbf{x}^{(i)} \approx z_1^{(i)}\mathbf{w}_1 + z_2^{(i)}\mathbf{w}_2 + \ldots + z_k^{(i)}\mathbf{w}_k$$

Parse on board: $\mathbf{x}^{(i)}, \mathbf{z}^{(i)}, \mathbf{w}_j$

Interpretation 1: Finding a Good Basis

$$\mathbf{x}^{(i)} \approx z_1^{(i)}\mathbf{w}_1 + z_2^{(i)}\mathbf{w}_2 + \ldots + z_k^{(i)}\mathbf{w}_k$$

- Find k "patterns" or *basis elements* $\mathbf{w}_1, \ldots, \mathbf{w}_k \in \mathbb{R}^n$
- Every data vector $\mathbf{x}^{(i)}$ can be well approximated as a weighted sum of basis elements

Practical Tip: "Center" the Data

In practice, the data is usually "centered" by subtracting the mean:

$$\boldsymbol{\mu} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)}$$

$$\mathbf{x}^{(i)} \leftarrow \mathbf{x}^{(i)} - \boldsymbol{\mu}$$

In Python:

```
mu = X.mean(0)
X = X - mu
```

Interpretation 1: Finding a Good Basis

$$\mathbf{x}^{(i)} \approx z_1^{(i)}\mathbf{w}_1 + z_2^{(i)}\mathbf{w}_2 + \ldots + z_k^{(i)}\mathbf{w}_k$$

- Find k "patterns" or *basis elements* $\mathbf{w}_1, \ldots, \mathbf{w}_k \in \mathbb{R}^n$
- Every data vector $\mathbf{x}^{(i)}$ can be well approximated as a weighted sum of basis elements

Demo: digits using mean + one basis element / compression

Interpretation 2: Dimension Reduction

$$\mathbf{x}^{(i)} \approx z_1^{(i)}\mathbf{w}_1 + z_2^{(i)}\mathbf{w}_2 + \ldots + z_k^{(i)}\mathbf{w}_k$$

- Define Φ so that $\mathbf{z}^{(i)} = \Phi(\mathbf{x}^{(i)})$. This is a feature map from n dimensions *down* to k dimensions (no explicit formula yet)
- Φ selected to preserve "as much information as possible" about data vectors. $\mathbf{x}^{(i)}$ can be approximately reconstructed from $\mathbf{z}^{(i)}$ and the basis elements $\mathbf{w}_1, \ldots, \mathbf{w}_k$.
- Practical application: map feature vectors to 2d or 3d space so they can be visualized. Demo: digits plotted in reduced feature space

Learning Problem

Given $X \in \mathbb{R}^{m \times n}$ (feature vectors in rows)

Find:

- $Z \in \mathbb{R}^{m \times k}$ (reduced feature fectors in rows)
- $W \in \mathbb{R}^{n \times k}$ (basis elements in columns)

to minimize

$$J = \sum_i \sum_j (X_{ij} - (ZW^T)_{ij})^2$$

Problem: Non-Uniqueness

While the problem is well defined, it does not have a unique solution.

Example:

- Suppose Z, W minimize J
- Let A be an invertible $k \times k$ matrix. Then

$$ZW^T = \underbrace{ZA}_{Z'}\underbrace{A^{-1}W^T}_{W'^T} = Z'W'^T$$

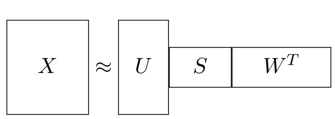
$\implies Z', W'$ also minimize J

Solution: Singular Value Decomposition (SVD)

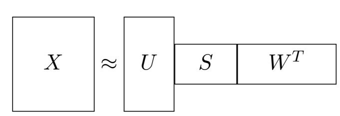
Solve the non-uniqueness problem by imposing additional constraints on the factors

Definition: the (rank-k) **singular value decomposition (SVD)** is the unique factorization of X that minimizes squared error and has the following form:

$$X \approx USW^T$$



. . . continued on next slide



where U and W have *orthonormal columns*:

$$U^TU = I_{k\times k}, \quad W^TW = I_{k\times k}$$

and S is *diagonal*:

$$S = \begin{bmatrix} \sigma_1 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2 & 0 & \ldots & 0 \\ 0 & 0 & \sigma_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \sigma_k \end{bmatrix}$$

with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$.

SVD Properties

- Uniquely defines U, S, V
- Closely related to eigenvalue decomposition of $X^T X$
- Efficient to compute. E.g., in Python

```
U,S,W_T = scipy.sparse.linalg.svds(X, k)
```

Note: does not work when entries of X are missing (i.e., for movie recommendations!)

Summary: Principal Components Analysis

Principal Components Analysis (PCA) is a well-known technique for dimensinality reduction that boils down to the following:

- Step 1: center data
- Step 2: perform SVD to get $X \approx \underbrace{US}_{Z} W^T$
- Step 3: Let $Z = US$, so we have $X \approx ZW^T$

The rows of Z are the reduced feature vectors, and the columns of W are the basis elements or "principal components"

Discusssion

- Briefly discuss alternate view of PCA
  - Linear feature map
  - MATLAB demo
- Uses of PCA
  - Data exploration
  - Run prior to supervised learning