Machine Learning

Lecture 8: Factor Analysis and Principle Component Analysis

Feng Li

fli@sdu.edu.cn

https://funglee.github.io

School of Computer Science and Technology
Shandong University

Fall 2018

Higher Dimension But Less Data

- Consider a case with $n \gg m$
  - The given training data span only a low-dimensional subspace of $\mathbb{R}^n$
- Model the data as Gaussian and estimate the mean and covariance using MLE

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

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

- Σ may be singular such that Σ^{-1} does not exist and $1/|\Sigma|^{1/2} = 1/0$

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Marginals and Conditionals of Gaussians

- Consider a vector-valued random variable

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

  where $x_1 \in \mathbb{R}^r$, $x_2 \in \mathbb{R}^s$ and $x \in \mathbb{R}^{r+s}$
- x follows a Gaussian distribution $x \sim \mathcal{N}(\mu, \Sigma)$

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

  where $\mu_1 \in \mathbb{R}^r$, $\mu_2 \in \mathbb{R}^s$, $\Sigma_{11} \in \mathbb{R}^{r \times r}$, $\Sigma_{12} \in \mathbb{R}^{r \times s}$, $\Sigma_{21} \in \mathbb{R}^{s \times r}$, and $\Sigma_{22} \in \mathbb{R}^{s \times s}$
- Also, $\Sigma_{12} = \Sigma_{21}^T$ due to the symmetry of Σ

Marginals and Conditionals of Gaussians (Contd.)

- x_1 and x_2 are jointly multivariate Gaussian
- What is the marginal distribution of x_1
  - $E[x_1] = \mu_1$
  - $Cov(x_1) = \Sigma_{11}$
- Since the marginal distribution of Gaussian are themselves Gaussian, we have

$$x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$$

- Conditional multivariate Gaussian distribution $x_1 \mid x_2 \sim \mathcal{N}(\mu_{1|2}, \Sigma_{1|2})$

$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$$
$$\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

Factor Analysis Model

- $x = \mu + \Lambda z + \varepsilon$
  - $x \in \mathbb{R}^n$, $\mu \in \mathbb{R}^n$, $\Lambda \in \mathbb{R}^{n \times k}$, $z \in \mathbb{R}^k$, $\varepsilon \in \mathcal{R}^n$
  - Λ is the factor loading matrix
  - $z \sim \mathcal{N}(0, I)$ (zero-mean independent normals, with unit variance)
  - $\varepsilon \sim \mathcal{N}(0, \Psi)$ where Ψ is a diagonal matrix (the observed variables are independent given the factors)
- How do we get the training data $\{x^{(i)}\}_i$?
  - Generate $\{z^{(i)}\}_i$ according to a multivariate Gaussian distribution $\mathcal{N}(0, I)$
  - Map $\{z^{(i)}\}_i$ into a n-dimensional affine space by Λ and μ
  - Generate $\{x^{(i)}\}_i$ by sampling the above affine space with noise ε
- Equivalently,

$$z \sim \mathcal{N}(0, I)$$
$$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$$

Factor Analysis Model (Contd.)

- z and x have a joint Gaussian distribution

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}(\mu_{zx}, \Sigma)$$

- Question: How to calculate μ_{zx} and Σ?
- Since $E[z] = 0$, we have

$$E[x] = E[\mu + \Lambda z + \epsilon] = \mu + \Lambda E[z] + E[\epsilon] = \mu$$

and then

$$\mu_{zx} = \begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}$$

Factor Analysis Model (Contd.)

- Since $z \sim \mathcal{N}(0, I)$, we have

$$\begin{aligned}
\Sigma_{zz} &= E[(z - E[z])(z - E[z])^T] = Cov(z) = I \\
\Sigma_{zx} &= E[(z - E[z])(x - E[x])^T] = E[z(\mu + \Lambda z + \epsilon - \mu)^T] = \Lambda^T \\
\Sigma_{xx} &= E[(\mu + \Lambda z + \epsilon - \mu)(\mu + \Lambda z + \epsilon - \mu)^T] = \Lambda\Lambda^T + \Psi
\end{aligned}$$

- Putting everything together, we therefore have

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$$

- Then, $x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$
- Log-likelihood function

$$\ell(\mu, \Lambda, \Psi) = \log \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2} |\Sigma_{xx}|^{1/2}} \exp\left(-\frac{1}{2} (x^{(i)} - \mu)^T \Sigma_{xx}^{-1} (x^{(i)} - \mu) \right)$$

EM Algorithm Review

- Repeat the following step until convergence
  - (E-step) For each i, set

$$Q_i(z^{(i)}) := p(z^{(i)} \mid x^{(i)}; \theta)$$

  - (M-step) set

$$\theta := \arg\max_{\theta} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

EM Algorithm for Factor Analysis

- Recall that if

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$$

  we then have

$$x_1 | x_2 \sim \mathcal{N}(\mu_{1|2}, \Sigma_{1|2})$$

  where

$$\begin{aligned} \mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\ \Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned}$$

EM Algorithm for Factor Analysis (Contd.)

- Since

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$$

  we have

$$z^{(i)} | x^{(i)}; \mu, \Lambda, \Psi \sim \mathcal{N}(\mu_{z^{(i)}|x^{(i)}}, \Sigma_{z^{(i)}|x^{(i)}})$$

  where

$$\begin{aligned} \mu_{z^{(i)}|x^{(i)}} &= \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}(x^{(i)} - \mu) \\ \Sigma_{z^{(i)}|x^{(i)}} &= I - \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}\Lambda \end{aligned}$$

- Calculate $Q_i(z^{(i)})$ in the E-step

$$Q_i(z^{(i)}) = \frac{\exp\left(-\frac{1}{2}(z^{(i)} - \mu_{z^{(i)}|x^{(i)}})^T \Sigma^{-1}_{z^{(i)}|x^{(i)}} (z^{(i)} - \mu_{z^{(i)}|x^{(i)}})\right)}{(2\pi)^{k/2} |\Sigma_{z^{(i)}|x^{(i)}}|^{1/2}}$$

EM Algorithm for Factor Analysis (Contd.)

- In M-step, we maximize

$$\sum_{i=1}^{m} \int_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \mu, \Lambda, \Psi)}{Q_i(z^{(i)})} dz^{(i)}$$

  with respect to μ, Λ, and Ψ

- Results are as follows

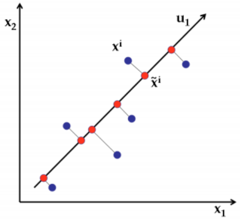
$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

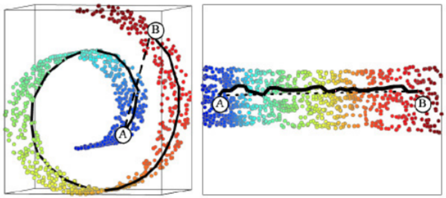
$$\Lambda = \left(\sum_{i=1}^{m} (x^{(i)} - \mu) \mu_{z^{(i)}|x^{(i)}}^{T} \right) \left(\sum_{i=1}^{m} \mu_{z^{(i)}|x^{(i)}} \mu_{z^{(i)}|x^{(i)}}^{T} + \Sigma_{z^{(i)}|x^{(i)}} \right)^{-1}$$

$$\Phi = diag(\frac{1}{m} \sum_{i=1}^{m} x^{(i)} {x^{(i)}}^{T} - x^{(i)} \mu_{z^{(i)}|x^{(i)}}^{T} \Lambda^{T} - \Lambda \mu_{z^{(i)}|x^{(i)}} {x^{(i)}}^{T} +$$
$$\Lambda(\mu_{z^{(i)}|x^{(i)}} \mu_{z^{(i)}|x^{(i)}}^{T} + \Sigma_{z^{(i)}|x^{(i)}}) \Lambda^{T})$$

Dimensionality Reduction

- Usually considered an unsupervised learning method
- Used for learning the low-dimensional structures in the data

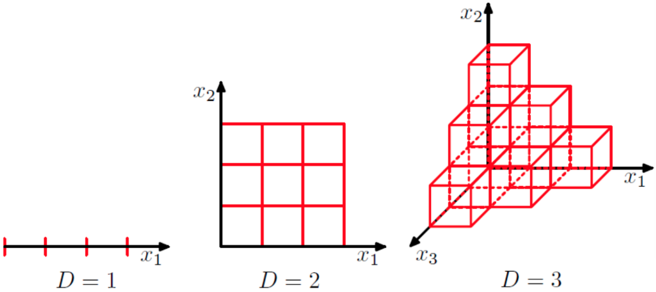




- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,
  - Documents using using topic vectors instead of bag-of-words vectors
  - Images using their constituent parts (faces - eigenfaces)
- Can be used for speeding up learning algorithms

Dimensionality Reduction

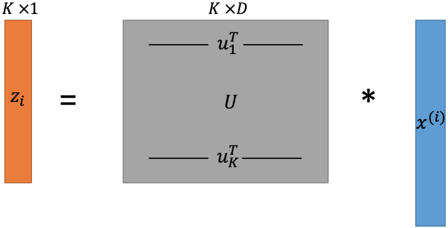
- Exponentially large # of examples required to "fill up" high-dim spaces



- Fewer dimensions ⇒ Less chances of overfitting ⇒ Better generalization
- Dimensionality reduction is a way to beat the curse of dimensionality

Liear Dimensionality Reduction

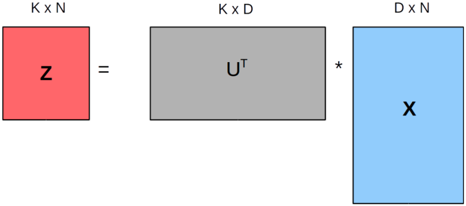
- A projection matrix $U = [u_1 u_2 \cdots u_K]$ of size $D \times K$ defines K linear projection direction
- Use U to transform $x^{(i)} \in \mathbb{R}^D$ into $z^{(i)} \in \mathbb{R}^K$



- $z^{(i)} = U^T x^{(i)} = [u_1^T x^{(i)},\ u_2^T x^{(i)},\ \cdots u_K^T x^{(i)}]^T$ is a K-dim projection of $x^{(i)}$
  - $z^{(i)} \in \mathbb{R}^K$ is also called low-dimensional "embeding" of $x^{(i)} \in \mathbb{R}^D$

Liear Dimensionality Reduction

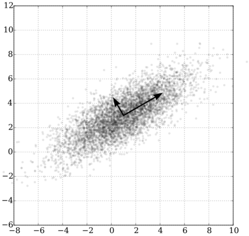
- $X = [x^{(1)}\ x^{(2)} \cdots x^{(N)}]$ is $D \times N$ matrix deoting all the N data points
- $Z = [z^{(1)}\ z^{(2)} \cdots z^{(N)}]$ is $K \times N$ matrix denoting embeddings of the data points
- With this notation, the figure on previous slide can be re-drawn as



- How do we learn the "best" projection matrix U?
- What criteria should we optimize for when learning U
- Principle Component Analysis (PCA) is an algorithm for doing this

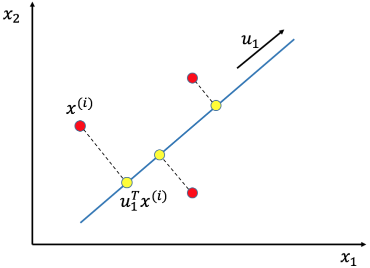
Principle Component Analysis (PCA)

- PCA is a technique widely used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization
- Two commonly used definitions
  - Learning projection directions that capture maximum variance in data
  - Learning projection directions that result in smallest reconstruction error
- Can also be seen as changing the basis in which the data is represented (and transforming the features such that new features become decorrelated)



Variance Captured by Projections

- Consider $x^{(i)} \in \mathbb{R}^D$ on a one-dim subspace defined by $u_1 \in \mathbb{R}^D$
- Projection of $x^{(i)}$ along a one-dim subspace $u_1 = u_1^T x^{(i)}$



- Mean of projections of all the data

$$\frac{1}{N}\sum_{i=1}^{N} u_1^T x^{(i)} = u_1^T \frac{1}{N}\sum_{i=1}^{N} x^{(i)} = u_1^T \mu$$

Variance Captured by Projections

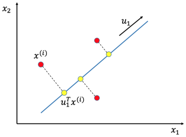
- Variance of the projected data

$$\frac{1}{N}\sum_{i=1}^{N}(u_1^T x^{(i)} - u_1^T \mu)^2 = \frac{1}{N}\sum_{i=1}^{N}[u_1^T(x^{(i)} - \mu)]^2 = u_1^T S u_1$$

- S is the $D \times D$ data covariance matrix

$$S = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

  - Variance of the projected data ("spread" of the yellow points)
  - If data already centered at $\mu = 0$, then $S = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}(x^{(i)})^T$



Optimization Problem

- We want u_1 s.t. the variance of the projected data is maximized

$$\max_{u_1} u_1^T S u_1$$

- To prevent trivial solution (max var. $=$ infinite), assume $\|u_1\| = 1 = u_1^T u_1$
- The method of Lagrange multipliers

$$\mathcal{L}(u_1, \lambda_1) = u_1^T S u_1 + \lambda_1(1 - u_1^T u_1)$$

where λ_1 is a Lagrange multiplier

  - If u_1^* is the optimal solution for the original constrained problem, then there exists λ_1^* such that (u_1^*, λ_1^*) is a stationary point for the Lagrange function (stationary points are those points where the partial derivatives of $\mathcal{L}$ are zero).

Direction of Maximum Variance

- Taking the derivative w.r.t. u_1 and setting to zero gives

$$Su_1 = \lambda_1 u_1$$

- Thus u_1 is an eigenvector of S (with corresponding eigenvalue λ_1)
- But which of S's eigenvectors it is?
- Note that since $u_1^T u_1 = 1$, the variance of projected data is

$$u_1^T S u_1 = \lambda_1$$

- Var. is maximized when u_1 is the top eigenvector with largest eigenvalue
- The top eigenvector u_1 is also known as the first Principle Component (PC)
- Other directions can also be found likewise (with each being orthogonal to all previous ones) using the eigendecomposition of S (this is PCA)

Steps in Principle Component Analysis

- Center the data (subtract the mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$ from each data point)
- Compute the covariance matrix

$$S = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} {x^{(i)}}^T = \frac{1}{N} X X^T$$

- Do an eigendecomposition of the covariance matrix S
- Take first K leading eigenvectors $\{u_l\}_{l=1,\cdots,K}$ with eigenvalues $\{\lambda_l\}_{l=1,\cdots,K}$
- The final K dim. projection of data is given by

$$Z = U^T X$$

where U is $D \times K$ and Z is $K \times N$

PCA as Minimizing the Reconstruction Error

- Assume complete orthonormal basis vector $u_1, u_2, \cdots, u_D$, each $u_l \in \mathbb{R}^N$
- We can represent each data point $x^{(i)} \in \mathbb{R}^D$ exactly using the new basis

$$x^{(i)} = \sum_{l=1}^{D} z_l^{(i)} u_l$$

$$\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_D^{(i)} \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & \cdots & u_D \end{bmatrix} * \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \\ \vdots \\ z_D^{(i)} \end{bmatrix}$$

- Denoting $z^{(i)} = [z_1^{(i)} \cdots z_D^{(i)}]^T$, $U = [u_1 \cdots u_D]$, and using $U^T U = I$

$$x^{(i)} = U z^{(i)} \quad \text{and} \quad z^{(i)} = U^T x^{(i)}$$

- Also note that each component of vector $z^{(i)}$ is $z_l^{(i)} = u_l^T x^{(i)}$

Reconstruction of Data from Projections

- Reconstruction of $x^{(i)}$ from $z^{(i)}$ will be exact if we use all D basis vectors
- Will be approximate if we only use $K < D$ basis vectors:

$$x^{(i)} \approx \sum_{l=1}^{K} z_l^{(i)} u_l$$

- Let's use $K = 1$ basis vector. Then, the one-dim embedding of $x^{(i)}$ is

$$z^{(i)} = u_1^T x^{(i)} \quad (z^{(i)} \in \mathbb{R})$$

- We can now try to "reconstruct" $x^{(i)}$ from its embedding $z^{(i)}$ as follows

$$\tilde{x}^{(i)} = u_1 z^{(i)} = u_1 u_1^T x^{(i)}$$

- Total error or "loss" in reconstructing all the data points

$$\ell(u_1) = \sum_{i=1}^{N} \|x^{(i)} - \tilde{x}^{(i)}\|^2 = \sum_{i=1}^{N} \|x^{(i)} - u_1 u_1^T x^{(i)}\|^2$$

Direction with Best Reconstruction

- We want to find u_1 that minimize the reconstruction error

$$\ell(u_1) = \sum_{i=1}^{N} \|x^{(i)} - u_1 u_1^T x^{(i)}\|^2 = \sum_{i=1}^{N} \left(-u_1^T x^{(i)} (x^{(i)})^T u_1 + (x^{(i)})^T x^{(i)} \right)$$

  by using $u_1^T u_1 = 1$

- Minimizing the error of reconstructing all the data points is equivalent to

$$\max_{u_1 : \|u_1\|^2 = 1} u_1^T \left(\sum_{n=1}^{N} x^{(i)} (x^{(i)})^T \right) u_1 = \max_{u_1 : \|u_1\|^2 = 1} u_1^T S u_1$$

  where S is the covariance matrix of the data (which are assumed to be centered)

- It is the same objective that we had when we maximized the variance

Thanks!

Q & A