

Dimensionality Reduction (Contd.)

Piyush Rai

Introduction to Machine Learning (CS771A)

October 9, 2018



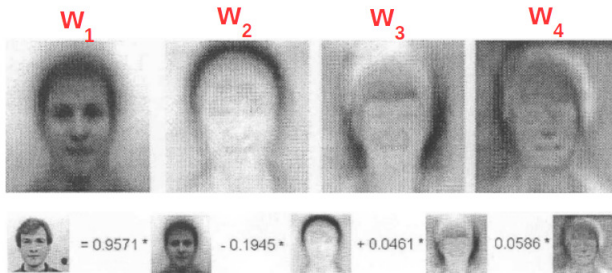
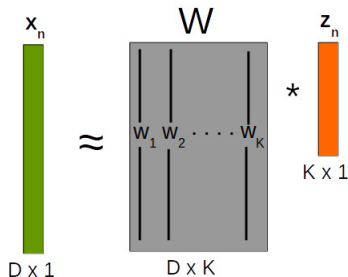
Announcements

- Quiz graded and scores sent
- Homework 3 out. Due on Oct 31, 11:59pm. Please start early.
- We will finish homework 1 and 2 grading soon
- Start thinking about your course project (if not working on it already)



Recap: Dimensionality Reduction - The Compression View

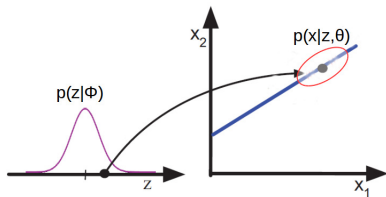
$$\mathbf{x}_n \approx \sum_{k=1}^K z_{nk} \mathbf{w}_k$$
$$\mathbf{x}_n \approx \mathbf{W} \mathbf{z}_n$$



Recap: Probabilistic PCA

A probabilistic model that maps a low-dim \mathbf{z} via a linear mapping to generate a high-dim \mathbf{x}

$$\begin{aligned}\mathbf{z}_n &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K) \\ \mathbf{x}_n | \mathbf{z}_n &\sim \mathcal{N}(\mathbf{W}\mathbf{z}_n, \sigma^2 \mathbf{I}_D)\end{aligned}$$



$$p(\mathbf{x}_n) = \underbrace{\mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}_D)}_{\text{Low-rank Gaussian as } \sigma^2 \rightarrow 0}$$

Many improvements possible (non-Gaussian distributions, nonlinear mappings, etc)



Recap: Such Models Can Learn to Generate Real-Looking Data..

- Learn the model parameters from training data $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, e.g., using MLE
- Generate a random \mathbf{z} from $p(\mathbf{z})$ and a random new sample \mathbf{x} conditioned on that \mathbf{z} using $p(\mathbf{x}|\mathbf{z})$



(a) Training data



(b) Random samples

Learning the PPCA Model

- One way: Maximize (conditional) log-likelihood $\sum_{n=1}^N \log p(\mathbf{x}_n | \mathbf{z}_n)$, or minimize its negative

$$\begin{aligned}\mathcal{L}(\mathbf{Z}, \mathbf{W}, \sigma^2) &= \frac{1}{2\sigma^2} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{W}\mathbf{z}_n\|^2 + \frac{ND}{2} \log(2\pi\sigma^2) \\ &= \frac{1}{2\sigma^2} \|\mathbf{X} - \mathbf{Z}\mathbf{W}^\top\|_F^2 + \frac{ND}{2} \log(2\pi\sigma^2)\end{aligned}$$

- For known σ^2 , learning PPCA boils down to solve

$$\{\hat{\mathbf{Z}}, \hat{\mathbf{W}}\} = \arg \min_{\mathbf{Z}, \mathbf{W}} \|\mathbf{X} - \mathbf{Z}\mathbf{W}^\top\|_F^2$$

- Similar to doing **matrix factorization** of \mathbf{X} by minimizing the reconstruction error
- Can solve it using ALT-OPT (\mathbf{Z} given \mathbf{W} , and \mathbf{W} given \mathbf{Z})
- Another (better) way: will be to do a proper MLE on $\log p(\mathbf{x}_n)$



MLE for PPCA

- Doing MLE for PPCA requires maximizing

$$\log p(\mathbf{X}|\Theta) = -\frac{N}{2}(D \log 2\pi + \log |\mathbf{C}| + \text{trace}(\mathbf{C}^{-1}\mathbf{S}))$$

where \mathbf{S} is the data covariance matrix, $\mathbf{C}^{-1} = \sigma^{-1}\mathbf{I} - \sigma^{-1}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^\top$ and $\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}$

- Assuming both \mathbf{W} and σ^2 as unknowns, their MLE solution is given by

$$\begin{aligned}\mathbf{W}_{ML} &= \mathbf{U}_K(\mathbf{L}_K - \sigma_{ML}^2\mathbf{I})^{1/2}\mathbf{R} \\ \sigma_{ML}^2 &= \frac{1}{D-K} \sum_{k=K+1}^D \lambda_k\end{aligned}$$

where \mathbf{U}_K is $D \times K$ matrix of **top K eigvecs** of \mathbf{S} , \mathbf{L}_K : $K \times K$ diagonal matrix of **top K eigvals** $\lambda_1, \dots, \lambda_K$, \mathbf{R} is a $K \times K$ **arbitrary rotation matrix** (equivalent to PCA for $\mathbf{R} = \mathbf{I}$ and $\sigma^2 \rightarrow 0$)

- Need to do **eigen-decomposition** of $D \times D$ data covariance matrix \mathbf{S} . EXPENSIVE!!!
- Also, can't do MLE like this if each \mathbf{x}_n has some **missing entries**

[†] Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)



MLE for PPCA using EM

- Using EM for MLE for PPCA has several benefits
 - No need to do expensive eigen-decomposition
 - Works even when \mathbf{x}_n may have some missing entries (HW3 has a problem related to this)
- EM does MLE by maximizing the expected CLL

$$\{\mathbf{W}, \sigma^2\} = \arg \max_{\mathbf{W}, \sigma^2} \mathbb{E}_{p(\mathbf{Z}|\mathbf{X}, \mathbf{W}, \sigma^2)} [\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$$

- This is done by iterating between the following two steps
 - E Step: For $n = 1, \dots, N$, infer the **posterior** $p(\mathbf{z}_n|\mathbf{x}_n)$ given current estimate of $\Theta = (\mathbf{W}, \sigma^2)$

$$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n, \sigma^2 \mathbf{M}^{-1}) \quad (\text{where } \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I}_K)$$

- M Step: Maximize the **expected CLL** $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. \mathbf{W}, σ^2



MLE for PPCA using EM

- The expected complete data log-likelihood $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$

$$= - \sum_{n=1}^N \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \|\mathbf{x}_n\|^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^\top \mathbf{W}^\top \mathbf{x}_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \mathbf{W}^\top \mathbf{W}) + \frac{1}{2} \text{tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top]) \right\}$$

- Taking the derivative of $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. \mathbf{W} and setting to zero

$$\mathbf{W} = \left[\sum_{n=1}^N \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^\top \right] \left[\sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \right]^{-1}$$

- To compute \mathbf{W} , we need two posterior expectations $\mathbb{E}[\mathbf{z}_n]$ and $\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top]$
- These can be easily obtained from the posterior $p(\mathbf{z}_n|\mathbf{x}_n)$ computed in E step

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) &= \mathcal{N}(\mathbf{M}^{-1} \mathbf{W}^\top \mathbf{x}_n, \sigma^2 \mathbf{M}^{-1}) \quad \text{where } \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I}_K \\ \mathbb{E}[\mathbf{z}_n] &= \mathbf{M}^{-1} \mathbf{W}^\top \mathbf{x}_n \\ \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] &= \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^\top + \text{cov}(\mathbf{z}_n) = \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^\top + \sigma^2 \mathbf{M}^{-1} \end{aligned}$$

- Note: The noise variance σ^2 can also be estimated (take deriv., set to zero..)



The Full EM Algorithm for PPCA

- Specify K , initialize \mathbf{W} and σ^2 randomly. Also **center the data** ($\mathbf{x}_n = \mathbf{x}_n - \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$)
- E step:** For each n , compute $p(\mathbf{z}_n|\mathbf{x}_n)$ using current \mathbf{W} and σ^2 . Compute exp. for the M step

$$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n, \sigma^2 \mathbf{M}^{-1}) \quad \text{where } \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I}_K$$

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n$$

$$\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] = \text{cov}(\mathbf{z}_n) + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top = \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \sigma^2 \mathbf{M}^{-1}$$

- M step:** Re-estimate \mathbf{W} and σ^2

$$\mathbf{W}_{new} = \left[\sum_{n=1}^N \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^\top \right] \left[\sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \right]^{-1}$$

$$\sigma_{new}^2 = \frac{1}{ND} \sum_{n=1}^N \left\{ \|\mathbf{x}_n\|^2 - 2\mathbb{E}[\mathbf{z}_n]^\top \mathbf{W}_{new}^\top \mathbf{x}_n + \text{tr} \left(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \mathbf{W}_{new}^\top \mathbf{W}_{new} \right) \right\}$$

- Set $\mathbf{W} = \mathbf{W}_{new}$ and $\sigma^2 = \sigma_{new}^2$. If not converged (monitor $p(\mathbf{X}|\Theta)$), go back to E step
- Note:** For $\sigma^2 = 0$, this EM algorithm can also be used to **efficiently** solve standard PCA (note that this EM algorithm doesn't require any eigen-decomposition)
- Missing entries of \mathbf{x}_n can be estimated in the E step as $p(\mathbf{x}_n^{miss}|\mathbf{x}_n^{obs})$



Why center the data before doing PPCA?

- The PPCA model, for each \mathbf{x}_n , $n = 1, \dots, N$, can also be written as

$$\mathbf{x}_n = \mu + \mathbf{W}\mathbf{z}_n + \epsilon_n \quad \text{where} \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_D)$$

- The marginal distribution is

$$p(\mathbf{x}_n) = \mathcal{N}(\mu, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}_D)$$

- The MLE of μ is simply $\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$
- So we can simply subtract μ from each observation and assume

$$\mathbf{x}_n = \mathbf{W}\mathbf{z}_n + \epsilon_n$$

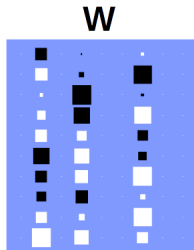
... and apply PPCA without μ



How to Set “K”?

- Several options to select the “best” K , e.g.,
 - Look at AIC/BIC criteria ($\text{NLL} + KD$ or $\text{NLL} + K \log D$) and pick the one with smallest K
 - Use [sparsity inducing priors](#) on \mathbf{W} and/or \mathbf{z}_n (set K to some large value; the unnecessary columns of \mathbf{W} will “turn off” automatically as they will be shrunk to zero during inference)

Using sparsity-inducing Prior
(e.g., [Automatic Relevance
Determination](#)) on \mathbf{W}



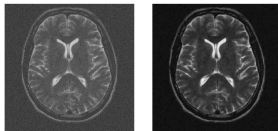
Effect: Only few columns
of \mathbf{W} will have entries with
significant magnitudes

- Compute the [marginal likelihood](#) (or its approximation) for each K and choose the best model
- [Nonparametric Bayesian methods](#) (allow K to grow with data)



Some Applications of PCA/PPCA

- Compression/dimensionality reduction is a natural application (use \mathbf{z}_n instead of \mathbf{x}_n)
- Also used for learning low-dim. “good” features \mathbf{z}_n from high-dim noisy features \mathbf{x}_n
 - Note that this is different from feature selection (\mathbf{z}_n is a transformed version of \mathbf{x}_n , not a subset)
- Learning the noise variance enables “image denoising”: $\mathbf{x}_n = \mathbf{W}\mathbf{z}_n + \epsilon_n$; $\mathbf{W}\mathbf{z}_n$ is the “clean” part

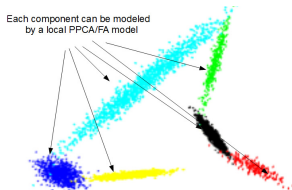


- Ability to fill-in missing data enables “image inpainting” (left: image with 80% missing data, middle: reconstructed, right: original)



Mixture of PPCA

- May be appropriate if data also exists in clusters (suppose $M > 1$ clusters)
- Data in each cluster (say m) can have its own “local” PPCA model defined by $\{\mu_m, \mathbf{W}_m, \sigma_m^2\}$
- Can use M such PPCA models $\{\mu_m, \mathbf{W}_m, \sigma_m^2\}_{m=1}^M$ (one per cluster) for the entire data



- Mixtures of PPCA can be seen as playing several roles
 - Jointly learning clustering and dimensionality reduction
 - Nonlinear dimensionality reduction
 - A flexible probability density model: Mixture of **low-rank** Gaussians



Mixture of PPCA

- For mixture of PPCA, the generative story for each observation \mathbf{x}_n is as follows

- Generate its cluster id as

$$\mathbf{c}_n \sim \text{multinoulli}(\pi_1, \dots, \pi_M)$$

- Generate latent variable $\mathbf{z}_n \in \mathbb{R}^K$ as

$$\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$$

- Generate observation $\mathbf{x}_n \in \mathbb{R}^D$ from the \mathbf{c}_n^{th} PPCA/FA model

$$\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{c}_n} + \mathbf{W}_{\mathbf{c}_n} \mathbf{z}_n, \sigma_{\mathbf{c}_n}^2 \mathbf{I}_D)$$

- Each PPCA model has its separate mean $\boldsymbol{\mu}_{\mathbf{c}_n}$ (not needed when $M = 1$ if data is centered)
- **Exercise:** What will be the **marginal distribution** of \mathbf{x}_n , i.e.. $p(\mathbf{x}_n|\Theta)$?
- **Exercise:** Use EM in this model to learn the parameters and latent variables

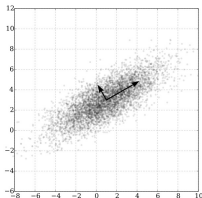


(Classic) Principal Component Analysis



Principal Component Analysis (PCA)

- A classic linear dim. reduction method (Pearson, 1901; Hotelling, 1930)
- Can be seen as
 - 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**)



- Also related to other classic methods, e.g., **Factor Analysis** (Spearman, 1904)

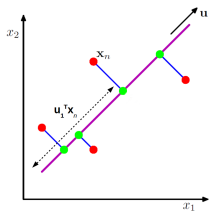


PCA as Maximizing Variance



Variance Captured by Projections

- Consider projecting $\mathbf{x}_n \in \mathbb{R}^D$ on a **one-dim subspace** (basically, a line) defined by $\mathbf{u}_1 \in \mathbb{R}^D$
- Projection/embedding of \mathbf{x}_n along a one-dim subspace $\mathbf{u}_1 = \mathbf{u}_1^\top \mathbf{x}_n$ (location of the green point along the purple line representing \mathbf{u}_1)



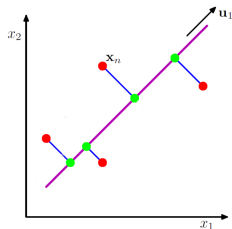
- Mean of projections of all the data: $\frac{1}{N} \sum_{n=1}^N \mathbf{u}_1^\top \mathbf{x}_n = \mathbf{u}_1^\top (\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n) = \mathbf{u}_1^\top \boldsymbol{\mu}$
- Variance** of the projected data (“spread” of the green points)

$$\frac{1}{N} \sum_{n=1}^N (\mathbf{u}_1^\top \mathbf{x}_n - \mathbf{u}_1^\top \boldsymbol{\mu})^2 = \frac{1}{N} \sum_{n=1}^N \{\mathbf{u}_1^\top (\mathbf{x}_n - \boldsymbol{\mu})\}^2 = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1$$

- S** is the $D \times D$ **data covariance matrix**: $\mathbf{s} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^\top$. If data already centered ($\boldsymbol{\mu} = 0$) then $\mathbf{s} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top = \frac{1}{N} \mathbf{X}^\top \mathbf{X}$



Direction of Maximum Variance



- We want \mathbf{u}_1 s.t. the variance of the projected data is maximized

$$\arg \max_{\mathbf{u}_1} \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1$$

- To prevent trivial solution (max var. = infinite), assume $\|\mathbf{u}_1\| = 1 = \mathbf{u}_1^\top \mathbf{u}_1$
- We will find \mathbf{u}_1 by solving the following constrained opt. problem

$$\arg \max_{\mathbf{u}_1} \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^\top \mathbf{u}_1)$$

where λ_1 is a Lagrange multiplier



Direction of Maximum Variance

- The objective function: $\arg \max_{\mathbf{u}_1} \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^\top \mathbf{u}_1)$

- Taking the derivative w.r.t. \mathbf{u}_1 and setting to zero gives

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

- Thus \mathbf{u}_1 is an eigenvector of \mathbf{S} (with corresponding eigenvalue λ_1)

- But which of \mathbf{S} 's (D possible) eigenvectors it is?

- Note that since $\mathbf{u}_1^\top \mathbf{u}_1 = 1$, the variance of projected data is

$$\mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 = \lambda_1$$

- Var. is maximized when \mathbf{u}_1 is the (top) eigenvector with largest eigenvalue

- The top eigenvector \mathbf{u}_1 is also known as the first Principal Component (PC)

- Other directions can also be found likewise (with each being orthogonal to all previous ones) using the eigendecomposition of \mathbf{S} (this is PCA)



Principal Component Analysis

- Center the data (subtract the mean $\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$ from each data point)
- Compute the covariance matrix \mathbf{S} using the centered data as

$$\mathbf{S} = \frac{1}{N} \mathbf{X}^\top \mathbf{X} \quad (\text{note: } \mathbf{X} \text{ assumed } D \times N \text{ here})$$

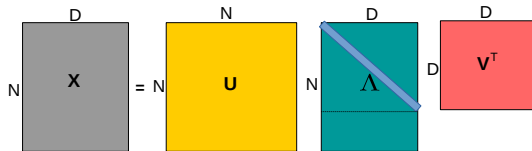
- Do an eigendecomposition of the covariance matrix \mathbf{S}
- Take first K leading eigenvectors $\{\mathbf{u}_k\}_{k=1}^K$ with eigenvalues $\{\lambda_k\}_{k=1}^K$
- The final K dim. projection/embedding of data is given by

$$\mathbf{Z} = \mathbf{X}\mathbf{U}$$

where $\mathbf{U} = [\mathbf{u}_1 \ \dots \ \mathbf{u}_K]$ is $D \times K$ and embedding matrix \mathbf{Z} is $K \times N$



Singular Value Decomposition (SVD)



- We can represent any matrix \mathbf{X} of size $N \times D$ using SVD as $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$
- $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_N]$ is $N \times N$, each $\mathbf{u}_n \in \mathbb{R}^N$ a **left singular vector** of \mathbf{X}
 - \mathbf{U} is orthonormal: $\mathbf{u}_n^T \mathbf{u}_{n'} = 0$ for $n \neq n'$, and $\mathbf{u}_n^T \mathbf{u}_n = 1 \Rightarrow \mathbf{U}\mathbf{U}^T = \mathbf{I}_N$
- $\mathbf{\Lambda}$ is $N \times D$ with only $\min(N, D)$ diagonal entries (all positive) - **singular values** (decreasing order)
- $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_D]$ is $D \times D$, each $\mathbf{v}_d \in \mathbb{R}^D$, a **right singular vector** of \mathbf{X}
 - \mathbf{V} is orthonormal: $\mathbf{v}_d^T \mathbf{v}_{d'} = 0$ for $d \neq d'$, and $\mathbf{v}_d^T \mathbf{v}_d = 1 \Rightarrow \mathbf{V}\mathbf{V}^T = \mathbf{I}_D$
- Note: If \mathbf{X} is symmetric then it is known as eigenvalue decomposition (and $\mathbf{U} = \mathbf{V}$ in that case)



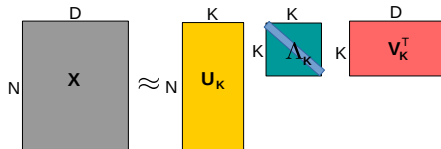
Low-Rank Approximation via SVD

- Can also expand the SVD expression as

$$\mathbf{X} = \sum_{k=1}^{\min(N,D)} \lambda_k \mathbf{u}_k \mathbf{v}_k^\top$$

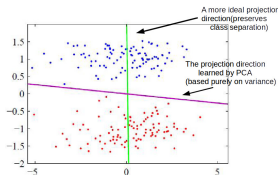
- Can write a rank- K approximation of \mathbf{X} (where $K \ll \min(N, D)$) as

$$\mathbf{X} \approx \hat{\mathbf{X}} = \sum_{k=1}^K \lambda_k \mathbf{u}_k \mathbf{v}_k^\top = \mathbf{U}_K \Lambda_K \mathbf{V}_K^\top$$



PCA/PPCA: Limitations and Extensions

- A linear projection method
 - Won't work well if data can't be approximated by a linear subspace
 - But PCA/PPCA can be kernelized (Kernel PCA or Gaussian Process Latent Variable Models)
- Variance based projection directions can sometimes be suboptimal (e.g., if we want to preserve class separation, e.g., when doing classification)



- PCA relies on eigendecomposition of an $D \times D$ covariance matrix
 - Can be slow if done naïvely. Takes $O(D^3)$ time
 - Many faster methods exist (e.g., Power Method)
 - Note: PPCA doesn't suffer from this issue (EM can be very efficient!)



Next Class

- How to compute singular vectors (SVD) - power method
- Nonlinear Dimensionality Reduction
- Supervised Dimensionality Reduction
- Dimensionality Reduction for Visualization

