Linear Dimensionality Reduction: Principal Component Analysis

Piyush Rai

Machine Learning (CS771A)

Sept 2, 2016

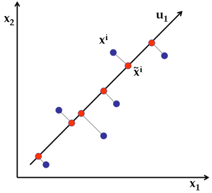
Dimensionality Reduction

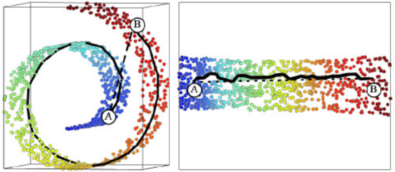
Dimensionality Reduction

- Usually considered an unsupervised learning method

Dimensionality Reduction

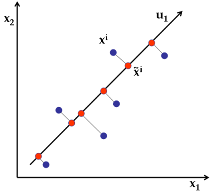
- Usually considered an unsupervised learning method
- Used for learning the low-dimensional structures in the data

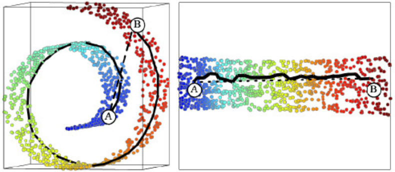




Dimensionality Reduction

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

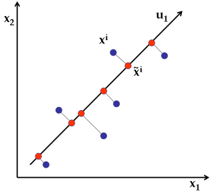


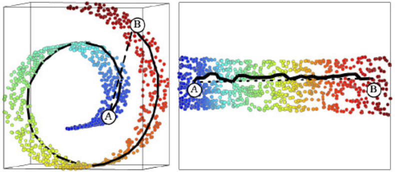


- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,

Dimensionality Reduction

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

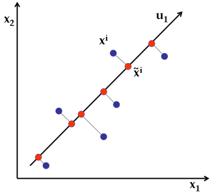


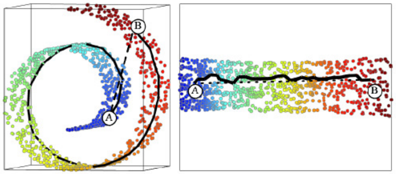


- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,
  - Documents using using topic vectors instead of bag-of-words vectors

Dimensionality Reduction

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

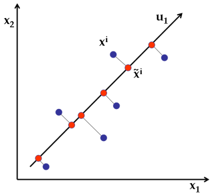


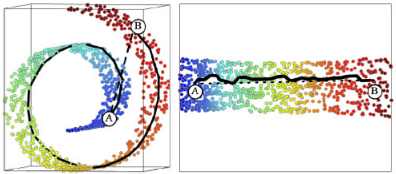


- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,
    - Documents using using topic vectors instead of bag-of-words vectors
    - Images using their constituent parts (faces - eigenfaces)

Dimensionality Reduction

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

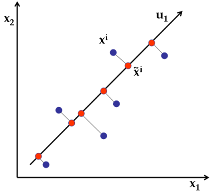


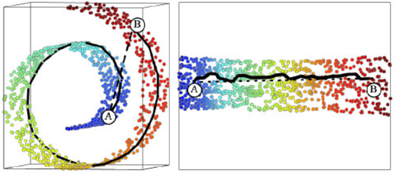


- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,
  - Documents using using topic vectors instead of bag-of-words vectors
  - Images using their constituent parts (faces - eigenfaces)
- Can be used for speeding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

Dimensionality Reduction

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

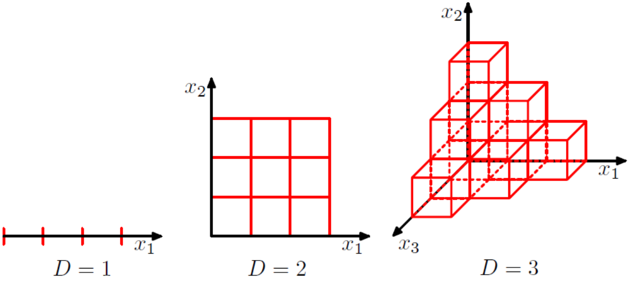




- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,
  - Documents using using topic vectors instead of bag-of-words vectors
  - Images using their constituent parts (faces - eigenfaces)
- Can be used for speeding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

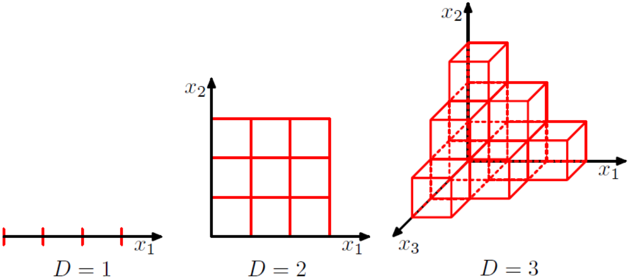
Curse of Dimensionality

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



Curse of Dimensionality

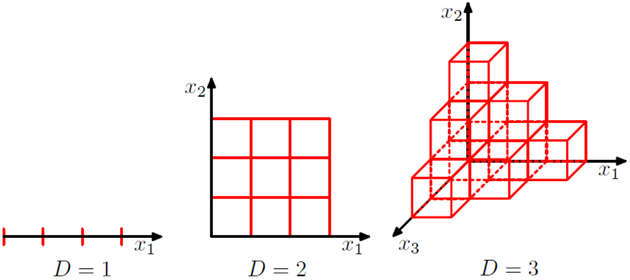
- Exponentially large # of examples required to “fill up” high-dim spaces



- Fewer dimensions $\Rightarrow$ Less chances of overfitting $\Rightarrow$ Better generalization

Curse of Dimensionality

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



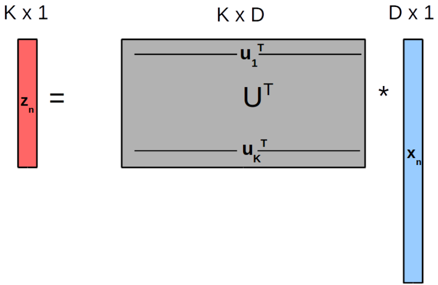
- Fewer dimensions ⇒ Less chances of overfitting ⇒ Better generalization
- Dimensionality reduction is a way to beat the curse of dimensionality

Linear Dimensionality Reduction

- A projection matrix $\mathbf{U} = [\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \ldots \ \boldsymbol{u}_K]$ of size $D \times K$ defines K linear projection directions, each $\boldsymbol{u}_k \in \mathbb{R}^D$, for the D dim. data (assume $K < D$)

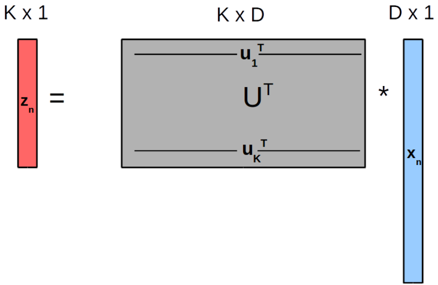
Linear Dimensionality Reduction

- A projection matrix $\mathbf{U} = [\boldsymbol{u}_1 \; \boldsymbol{u}_2 \; \dots \; \boldsymbol{u}_K]$ of size $D \times K$ defines K linear projection directions, each $\boldsymbol{u}_k \in \mathbb{R}^D$, for the D dim. data (assume $K < D$)
- Can use $\mathbf{U}$ to transform $\boldsymbol{x}_n \in \mathbb{R}^D$ into $\boldsymbol{z}_n \in \mathbb{R}^K$ as shown below



Linear Dimensionality Reduction

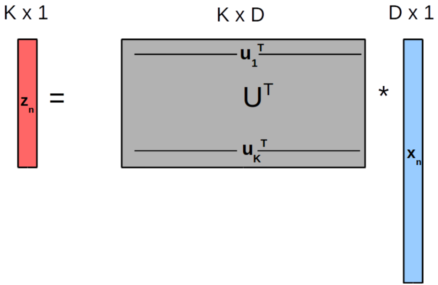
- A projection matrix $\mathbf{U} = [\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \dots \ \boldsymbol{u}_K]$ of size $D \times K$ defines K linear projection directions, each $\boldsymbol{u}_k \in \mathbb{R}^D$, for the D dim. data (assume $K < D$)
- Can use $\mathbf{U}$ to transform $\boldsymbol{x}_n \in \mathbb{R}^D$ into $\boldsymbol{z}_n \in \mathbb{R}^K$ as shown below



- Note that $\boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n = [\boldsymbol{u}_1^\top \boldsymbol{x}_n \ \boldsymbol{u}_2^\top \boldsymbol{x}_n \ \dots \ \boldsymbol{u}_K^\top \boldsymbol{x}_n]$ is a K-dim projection of $\boldsymbol{x}_n$

Linear Dimensionality Reduction

- A projection matrix $\mathbf{U} = [\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \dots \ \boldsymbol{u}_K]$ of size $D \times K$ defines K linear projection directions, each $\boldsymbol{u}_k \in \mathbb{R}^D$, for the D dim. data (assume $K < D$)
- Can use $\mathbf{U}$ to transform $\boldsymbol{x}_n \in \mathbb{R}^D$ into $\boldsymbol{z}_n \in \mathbb{R}^K$ as shown below



- Note that $\boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n = [\boldsymbol{u}_1^\top \boldsymbol{x}_n \ \boldsymbol{u}_2^\top \boldsymbol{x}_n \ \dots \ \boldsymbol{u}_K^\top \boldsymbol{x}_n]$ is a K-dim projection of $\boldsymbol{x}_n$
  - $\boldsymbol{z}_n \in \mathbb{R}^K$ is also called low-dimensional "embedding" of $\boldsymbol{x}_n \in \mathbb{R}^D$

Linear Dimensionality Reduction

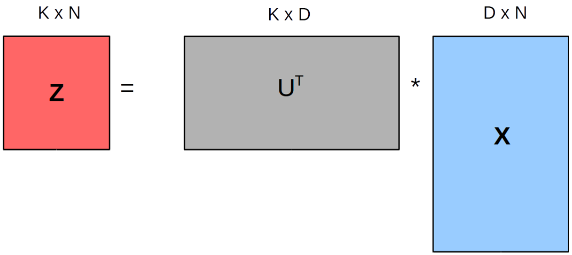
- $\mathbf{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \ldots \ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points

Linear Dimensionality Reduction

- $\mathbf{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \ldots \ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points
- $\mathbf{Z} = [\boldsymbol{z}_1 \ \boldsymbol{z}_2 \ \ldots \ \boldsymbol{z}_N]$ is $K \times N$ matrix denoting **embeddings** of data points

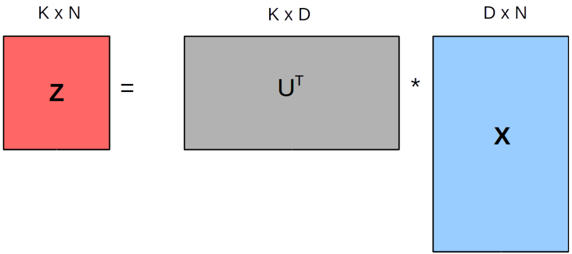
Linear Dimensionality Reduction

- $\mathbf{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \ldots \ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points
- $\mathbf{Z} = [\boldsymbol{z}_1 \ \boldsymbol{z}_2 \ \ldots \ \boldsymbol{z}_N]$ is $K \times N$ matrix denoting **embeddings** of data points
- With this notation, the figure on previous slide can be re-drawn as below



Linear Dimensionality Reduction

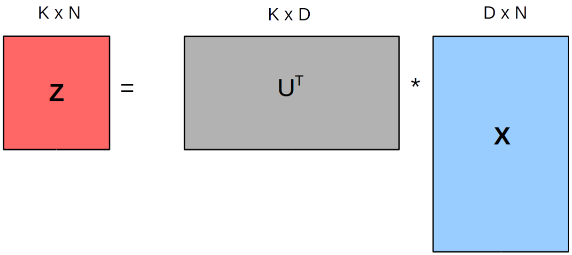
- $\mathbf{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \ldots \ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points
- $\mathbf{Z} = [\boldsymbol{z}_1 \ \boldsymbol{z}_2 \ \ldots \ \boldsymbol{z}_N]$ is $K \times N$ matrix denoting **embeddings** of data points
- With this notation, the figure on previous slide can be re-drawn as below



- How do we learn the "best" projection matrix $\mathbf{U}$?

Linear Dimensionality Reduction

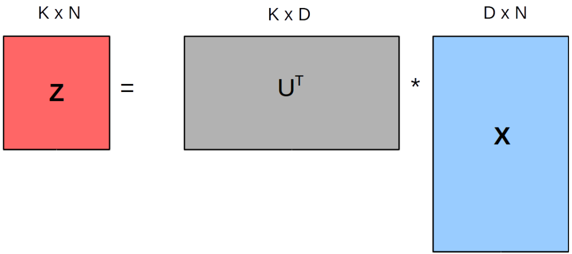
- $\mathbf{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \ldots \ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points
- $\mathbf{Z} = [\boldsymbol{z}_1 \ \boldsymbol{z}_2 \ \ldots \ \boldsymbol{z}_N]$ is $K \times N$ matrix denoting **embeddings** of data points
- With this notation, the figure on previous slide can be re-drawn as below



- How do we learn the "best" projection matrix $\mathbf{U}$?
- What criteria should we optimize for when learning $\mathbf{U}$?

Linear Dimensionality Reduction

- $\mathbf{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \ldots\ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points
- $\mathbf{Z} = [\boldsymbol{z}_1\ \boldsymbol{z}_2\ \ldots\ \boldsymbol{z}_N]$ is $K \times N$ matrix denoting **embeddings** of data points
- With this notation, the figure on previous slide can be re-drawn as below



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

Principal Component Analysis

Principal Component Analysis (PCA)

- A classic linear dim. reduction method (Pearson, 1901; Hotelling, 1930)

Principal Component Analysis (PCA)

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

Principal Component Analysis (PCA)

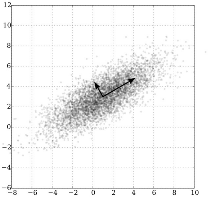
- A classic linear dim. reduction method (Pearson, 1901; Hotelling, 1930)
- Can be seen as
  - Learning projection directions that capture maximum variance in data

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

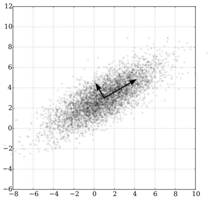
Principal Component Analysis (PCA)

- 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)



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)

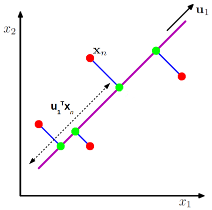


- Also related to other classic methods, e.g., Factor Analysis (Spearman, 1904)

PCA as Maximizing Variance

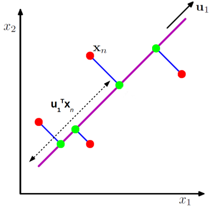
Variance Captured by Projections

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



Variance Captured by Projections

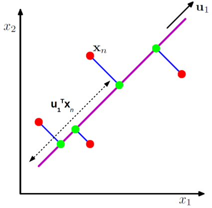
- Consider projecting $\boldsymbol{x}_n \in \mathbb{R}^D$ on a one-dim subspace defined by $\boldsymbol{u}_1 \in \mathbb{R}^D$
- Projection/embedding of $\boldsymbol{x}_n$ along a one-dim subspace $\boldsymbol{u}_1 = \boldsymbol{u}_1^\top \boldsymbol{x}_n$ (location of the green point along the purple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



- Mean of projections of all the data: $\frac{1}{N}\sum_{n=1}^N \boldsymbol{u}_1^\top \boldsymbol{x}_n = \boldsymbol{u}_1^\top(\frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n) = \boldsymbol{u}_1^\top \boldsymbol{\mu}$

Variance Captured by Projections

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

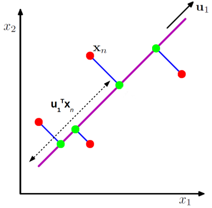


- Mean of projections of all the data: $\frac{1}{N}\sum_{n=1}^N \boldsymbol{u}_1^\top \boldsymbol{x}_n = \boldsymbol{u}_1^\top(\frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n) = \boldsymbol{u}_1^\top \boldsymbol{\mu}$
- Variance of the projected data ("spread" of the green points)

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

Variance Captured by Projections

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

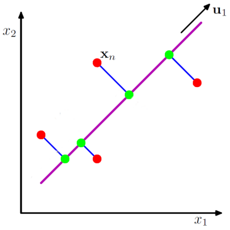


- Mean of projections of all the data: $\frac{1}{N}\sum_{n=1}^N \boldsymbol{u}_1^\top \boldsymbol{x}_n = \boldsymbol{u}_1^\top(\frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n) = \boldsymbol{u}_1^\top \boldsymbol{\mu}$
- Variance of the projected data ("spread" of the green points)

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

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

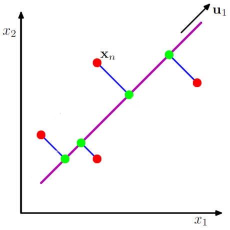
Direction of Maximum Variance



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

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

Direction of Maximum Variance

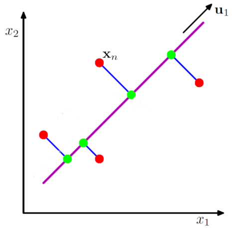


- We want $\boldsymbol{u}_1$ s.t. the variance of the projected data is maximized

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

- To prevent trivial solution (max var. $=$ infinite), assume $||\boldsymbol{u}_1|| = 1 = \boldsymbol{u}_1^\top \boldsymbol{u}_1$

Direction of Maximum Variance



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

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

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

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

where λ_1 is a Lagrange multiplier

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \ \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \; \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

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

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \ \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

- Thus $\boldsymbol{u}_1$ is an eigenvector of $\mathbf{S}$ (with corresponding eigenvalue λ_1)
- But which of $\mathbf{S}$'s (D possible) eigenvectors it is?

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \; \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

- Thus $\boldsymbol{u}_1$ 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 $\boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1$, the variance of projected data is

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

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \ \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

- Thus $\boldsymbol{u}_1$ 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 $\boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1$, the variance of projected data is

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

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

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \; \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

- Thus $\boldsymbol{u}_1$ 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 $\boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1$, the variance of projected data is

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

- Var. is maximized when $\boldsymbol{u}_1$ is the (top) eigenvector with largest eigenvalue
- The top eigenvector $\boldsymbol{u}_1$ is also known as the first Principal Component (PC)

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \; \boldsymbol{u}_1^\top \mathbf{S}\boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

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

- Thus $\boldsymbol{u}_1$ 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 $\boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1$, the variance of projected data is

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

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

- **Steps in Principal Component Analysis**

Principal Component Analysis

- **Steps in Principal Component Analysis**
  - Center the data (subtract the mean $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$ from each data point)

Principal Component Analysis

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

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

Principal Component Analysis

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

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

  - Do an eigendecomposition of the covariance matrix $\mathbf{S}$

Principal Component Analysis

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

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

  - Do an eigendecomposition of the covariance matrix $\mathbf{S}$
  - Take first K leading eigenvectors $\{\boldsymbol{u}_k\}_{k=1}^{K}$ with eigenvalues $\{\lambda_k\}_{k=1}^{K}$

Principal Component Analysis

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

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

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

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

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

Principal Component Analysis

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

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

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

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

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

- A word about notation: If $\mathbf{X}$ is $N \times D$, then $\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$ (needs to be $D \times D$) and the embedding will be computed as $\mathbf{Z} = \mathbf{X}\mathbf{U}$ where $\mathbf{Z}$ is $N \times K$

PCA as Minimizing the Reconstruction Error

Data as Combination of Basis Vectors

- Assume *complete* orthonormal basis vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D$, each $\boldsymbol{u}_d \in \mathbb{R}^D$

Data as Combination of Basis Vectors

- Assume *complete* orthonormal basis vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D$, each $\boldsymbol{u}_d \in \mathbb{R}^D$
- We can represent each data point $\boldsymbol{x}_n \in \mathbb{R}^D$ **exactly** using this new basis

$$\boldsymbol{x}_n = \sum_{k=1}^{D} z_{nk} \boldsymbol{u}_k$$

Data as Combination of Basis Vectors

- Assume *complete* orthonormal basis vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D$, each $\boldsymbol{u}_d \in \mathbb{R}^D$
- We can represent each data point $\boldsymbol{x}_n \in \mathbb{R}^D$ **exactly** using this new basis

$$\boldsymbol{x}_n = \sum_{k=1}^{D} z_{nk} \boldsymbol{u}_k$$

$$\begin{bmatrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & & u_D \\ | & | & & | \end{bmatrix} * \begin{bmatrix} z_{n1} \\ z_{n2} \\ \vdots \\ z_{nD} \end{bmatrix}$$

Data as Combination of Basis Vectors

- Assume *complete* orthonormal basis vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D$, each $\boldsymbol{u}_d \in \mathbb{R}^D$
- We can represent each data point $\boldsymbol{x}_n \in \mathbb{R}^D$ **exactly** using this new basis

$$\boldsymbol{x}_n = \sum_{k=1}^{D} z_{nk} \boldsymbol{u}_k$$

$$\begin{bmatrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & & u_D \\ | & | & & | \end{bmatrix} * \begin{bmatrix} z_{n1} \\ z_{n2} \\ \vdots \\ z_{nD} \end{bmatrix}$$

- Denoting $\boldsymbol{z}_n = [z_{n1} \; z_{n2} \ldots z_{nD}]^\top$, $\mathbf{U} = [\boldsymbol{u}_1 \; \boldsymbol{u}_2 \; \ldots \boldsymbol{u}_D]$, and using $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_D$

$$\boxed{\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n \quad \text{and} \quad \boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n}$$

Data as Combination of Basis Vectors

- Assume *complete* orthonormal basis vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D$, each $\boldsymbol{u}_d \in \mathbb{R}^D$
- We can represent each data point $\boldsymbol{x}_n \in \mathbb{R}^D$ **exactly** using this new basis

$$\boldsymbol{x}_n = \sum_{k=1}^{D} z_{nk} \boldsymbol{u}_k$$

$$\begin{bmatrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_D \\ | & | & & | \end{bmatrix} * \begin{bmatrix} z_{n1} \\ z_{n2} \\ \vdots \\ z_{nD} \end{bmatrix}$$

- Denoting $\boldsymbol{z}_n = [z_{n1} \; z_{n2} \ldots z_{nD}]^\top$, $\mathbf{U} = [\boldsymbol{u}_1 \; \boldsymbol{u}_2 \; \ldots \boldsymbol{u}_D]$, and using $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_D$

$$\boxed{\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n \quad \text{and} \quad \boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n}$$

- Also note that each component of vector $\boldsymbol{z}_n$ is $z_{nk} = \boldsymbol{u}_k^\top \boldsymbol{x}_n$

Reconstruction of Data from Projections

- Reconstruction of $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$ will be exact if we use all the D basis vectors

Reconstruction of Data from Projections

- Reconstruction of $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$ will be exact if we use all the D basis vectors
- Will be approximate if we only use $K < D$ basis vectors: $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$

Reconstruction of Data from Projections

- Reconstruction of $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$ will be exact if we use all the D basis vectors
- Will be approximate if we only use $K < D$ basis vectors: $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$
- Let's use $K = 1$ basis vector. Then the one-dim embedding of $\boldsymbol{x}_n$ is

$$\boldsymbol{z}_n = \boldsymbol{u}_1^\top \boldsymbol{x}_n \qquad \text{(note: this will just be a scalar)}$$

Reconstruction of Data from Projections

- Reconstruction of $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$ will be exact if we use all the D basis vectors
- Will be approximate if we only use $K < D$ basis vectors: $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$
- Let's use $K = 1$ basis vector. Then the one-dim embedding of $\boldsymbol{x}_n$ is

$$\boldsymbol{z}_n = \boldsymbol{u}_1^\top \boldsymbol{x}_n \qquad \text{(note: this will just be a scalar)}$$

- We can now try "reconstructing" $\boldsymbol{x}_n$ from its embedding $\boldsymbol{z}_n$ as follows

$$\tilde{\boldsymbol{x}}_n = \boldsymbol{u}_1 \boldsymbol{z}_n = \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{x}_n$$

Reconstruction of Data from Projections

- Reconstruction of $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$ will be exact if we use all the D basis vectors
- Will be approximate if we only use $K < D$ basis vectors: $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk}\boldsymbol{u}_k$
- Let's use $K = 1$ basis vector. Then the one-dim embedding of $\boldsymbol{x}_n$ is

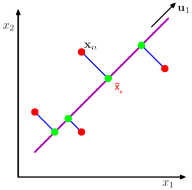
$$\boldsymbol{z}_n = \boldsymbol{u}_1^\top \boldsymbol{x}_n \qquad \text{(note: this will just be a scalar)}$$

- We can now try "reconstructing" $\boldsymbol{x}_n$ from its embedding $\boldsymbol{z}_n$ as follows

$$\tilde{\boldsymbol{x}}_n = \boldsymbol{u}_1 \boldsymbol{z}_n = \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{x}_n$$

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

$$L(\boldsymbol{u}_1) = \sum_{n=1}^{N} ||\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n||^2 = \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{x}_n||^2$$



Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$L(\boldsymbol{u}_1) = \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{x}_n||^2$$

Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$
\begin{aligned}
L(\boldsymbol{u}_1) &= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n||^2 \\
&= \sum_{n=1}^{N} \{\boldsymbol{x}_n^\top \boldsymbol{x}_n + (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n)^\top(\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n) - 2\boldsymbol{x}_n^\top \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n\}
\end{aligned}
$$

Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$
\begin{aligned}
L(\boldsymbol{u}_1) &= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n||^2 \\
&= \sum_{n=1}^{N} \{\boldsymbol{x}_n^\top \boldsymbol{x}_n + (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n)^\top (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n) - 2\boldsymbol{x}_n^\top \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n\} \\
&= \sum_{n=1}^{N} -\boldsymbol{u}_1^\top \boldsymbol{x}_n\boldsymbol{x}_n^\top \boldsymbol{u}_1 \qquad (\text{using } \boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1 \text{ and ignoring constants w.r.t. } \boldsymbol{u}_1)
\end{aligned}
$$

Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$
\begin{aligned}
L(\boldsymbol{u}_1) &= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n||^2 \\
&= \sum_{n=1}^{N} \{\boldsymbol{x}_n^\top \boldsymbol{x}_n + (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n)^\top (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n) - 2\boldsymbol{x}_n^\top \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n\} \\
&= \sum_{n=1}^{N} -\boldsymbol{u}_1^\top \boldsymbol{x}_n\boldsymbol{x}_n^\top \boldsymbol{u}_1 \qquad \text{(using } \boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1 \text{ and ignoring constants w.r.t. } \boldsymbol{u}_1\text{)}
\end{aligned}
$$

- Thus the problem is equivalent to the following maximization

Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$
\begin{aligned}
L(\boldsymbol{u}_1) &= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n||^2 \\
&= \sum_{n=1}^{N} \{\boldsymbol{x}_n^\top \boldsymbol{x}_n + (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n)^\top (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n) - 2\boldsymbol{x}_n^\top \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n\} \\
&= \sum_{n=1}^{N} -\boldsymbol{u}_1^\top \boldsymbol{x}_n\boldsymbol{x}_n^\top \boldsymbol{u}_1 \qquad (\text{using } \boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1 \text{ and ignoring constants w.r.t. } \boldsymbol{u}_1)
\end{aligned}
$$

- Thus the problem is equivalent to the following maximization

$$
\arg\max_{\boldsymbol{u}_1:||\boldsymbol{u}_1||^2=1} \boldsymbol{u}_1^\top \left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^\top\right) \boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}_1:||\boldsymbol{u}_1||^2=1} \boldsymbol{u}_1^\top \mathbf{S}\boldsymbol{u}_1
$$

where $\mathbf{S}$ is the covariance matrix of the data (data assumed centered)

Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$
\begin{aligned}
L(\boldsymbol{u}_1) &= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n||^2 \\
&= \sum_{n=1}^{N} \{\boldsymbol{x}_n^\top \boldsymbol{x}_n + (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n)^\top (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n) - 2\boldsymbol{x}_n^\top \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n\} \\
&= \sum_{n=1}^{N} -\boldsymbol{u}_1^\top \boldsymbol{x}_n\boldsymbol{x}_n^\top \boldsymbol{u}_1 \qquad (\text{using } \boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1 \text{ and ignoring constants w.r.t. } \boldsymbol{u}_1)
\end{aligned}
$$

- Thus the problem is equivalent to the following maximization

$$
\underset{\boldsymbol{u}_1:||\boldsymbol{u}_1||^2=1}{\arg\max}\ \boldsymbol{u}_1^\top \left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^\top\right) \boldsymbol{u}_1 = \underset{\boldsymbol{u}_1:||\boldsymbol{u}_1||^2=1}{\arg\max}\ \boldsymbol{u}_1^\top \mathbf{S}\boldsymbol{u}_1
$$

  where $\mathbf{S}$ is the covariance matrix of the data (data assumed centered)

- It's the same objective that we had when we maximized the variance

How many Principal Components to Use?

- Eigenvalue λ_k measures the variance captured by the corresponding PC $\boldsymbol{u}_k$

How many Principal Components to Use?

- Eigenvalue λ_k measures the variance captured by the corresponding PC $\boldsymbol{u}_k$
- The "left-over" variance will therefore be

$$\sum_{k=K+1}^{D} \lambda_k$$

How many Principal Components to Use?

- Eigenvalue λ_k measures the variance captured by the corresponding PC $\boldsymbol{u}_k$
- The "left-over" variance will therefore be

$$\sum_{k=K+1}^{D} \lambda_k$$

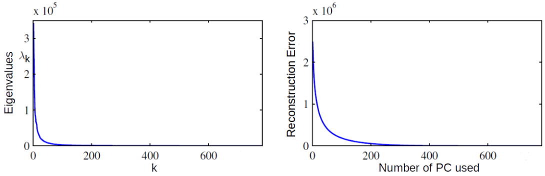
- Can choose K by looking at what fraction of variance is captured by the first K PCs

How many Principal Components to Use?

- Eigenvalue λ_k measures the variance captured by the corresponding PC $\boldsymbol{u}_k$
- The "left-over" variance will therefore be

$$\sum_{k=K+1}^{D} \lambda_k$$

- Can choose K by looking at what fraction of variance is captured by the first K PCs
- Another direct way is to look at the spectrum of the eigenvalues plot, or the plot of reconstruction error vs number of PC

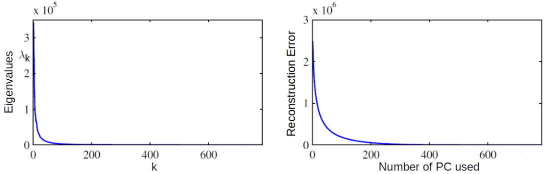


How many Principal Components to Use?

- Eigenvalue λ_k measures the variance captured by the corresponding PC $\boldsymbol{u}_k$
- The "left-over" variance will therefore be

$$\sum_{k=K+1}^{D} \lambda_k$$

- Can choose K by looking at what fraction of variance is captured by the first K PCs
- Another direct way is to look at the spectrum of the eigenvalues plot, or the plot of reconstruction error vs number of PC



- Can also use other criteria such as AIC/BIC (or more advanced probabilistic approaches to PCA using nonparametric Bayesian methods)

PCA as Matrix Factorization

- Note that PCA represents each $\boldsymbol{x}_n$ as $\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n$

PCA as Matrix Factorization

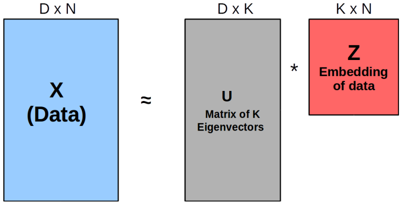
- Note that PCA represents each $\boldsymbol{x}_n$ as $\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n$
- When using only $K < D$ components, $\boldsymbol{x}_n \approx \mathbf{U}\boldsymbol{z}_n$

PCA as Matrix Factorization

- Note that PCA represents each $\boldsymbol{x}_n$ as $\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n$
- When using only $K < D$ components, $\boldsymbol{x}_n \approx \mathbf{U}\boldsymbol{z}_n$
- For all the N data points, we can write the same as

$$\mathbf{X} \approx \mathbf{UZ}$$

where $\mathbf{X}$ is $D \times N$, $\mathbf{U}$ is $D \times K$ and $\mathbf{Z}$ is $K \times N$

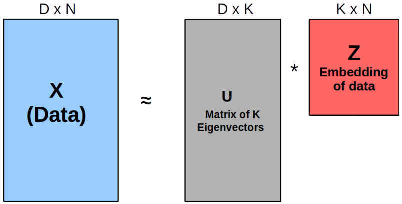


PCA as Matrix Factorization

- Note that PCA represents each $\boldsymbol{x}_n$ as $\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n$
- When using only $K < D$ components, $\boldsymbol{x}_n \approx \mathbf{U}\boldsymbol{z}_n$
- For all the N data points, we can write the same as

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

where $\mathbf{X}$ is $D \times N$, $\mathbf{U}$ is $D \times K$ and $\mathbf{Z}$ is $K \times N$



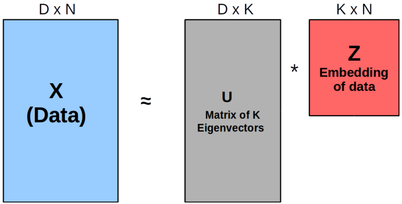
- The above approx. is equivalent to a low-rank matrix factorization of $\mathbf{X}$

PCA as Matrix Factorization

- Note that PCA represents each $\boldsymbol{x}_n$ as $\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n$
- When using only $K < D$ components, $\boldsymbol{x}_n \approx \mathbf{U}\boldsymbol{z}_n$
- For all the N data points, we can write the same as

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

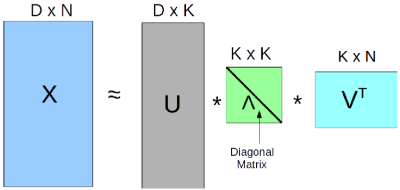
where $\mathbf{X}$ is $D \times N$, $\mathbf{U}$ is $D \times K$ and $\mathbf{Z}$ is $K \times N$



- The above approx. is equivalent to a low-rank matrix factorization of $\mathbf{X}$
  - Also closely related to Singular Value Decomposition (SVD); see next slide

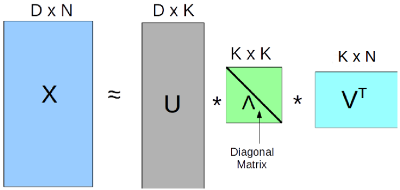
PCA and SVD

- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U} \Lambda \mathbf{V}^\top$



PCA and SVD

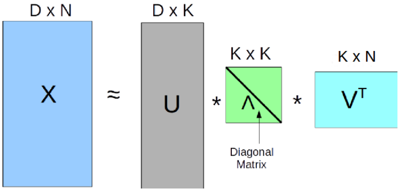
- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$



- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$

PCA and SVD

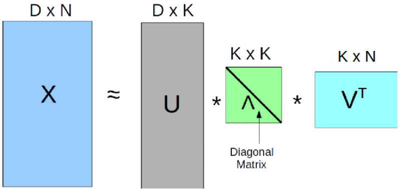
- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$



- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$
- Λ is a $K \times K$ diagonal matrix (with top K singular values)

PCA and SVD

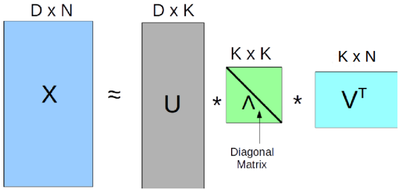
- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$



- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$
- Λ is a $K \times K$ diagonal matrix (with top K singular values)
- $\mathbf{V}$ is $N \times K$ matrix with top K right singular vectors of $\mathbf{X}$

PCA and SVD

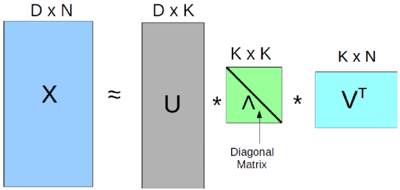
- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$



- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$
- Λ is a $K \times K$ diagonal matrix (with top K singular values)
- $\mathbf{V}$ is $N \times K$ matrix with top K right singular vectors of $\mathbf{X}$
- Rank-K SVD is based on minimizing the reconstruction error

PCA and SVD

- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$

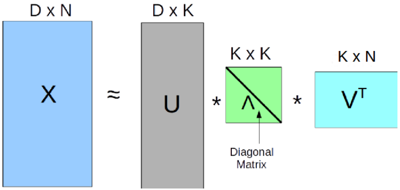


- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$
- Λ is a $K \times K$ diagonal matrix (with top K singular values)
- $\mathbf{V}$ is $N \times K$ matrix with top K right singular vectors of $\mathbf{X}$
- Rank-K SVD is based on minimizing the reconstruction error

$$||\mathbf{X} - \mathbf{U}\Lambda\mathbf{V}^\top||$$

PCA and SVD

- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$



- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$
- Λ is a $K \times K$ diagonal matrix (with top K singular values)
- $\mathbf{V}$ is $N \times K$ matrix with top K right singular vectors of $\mathbf{X}$
- Rank-K SVD is based on minimizing the reconstruction error

$$||\mathbf{X} - \mathbf{U}\Lambda\mathbf{V}^\top||$$

- PCA is equivalent to the best rank-K SVD after centering the data

PCA: Some Comments

- The idea of approximating each data point as a combination of basis vectors

$$\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk}\boldsymbol{u}_k \quad \text{or} \quad \mathbf{X} \approx \mathbf{UZ}$$

  is also popularly known as "Dictionary Learning" in signal/image processing; the learned basis vectors represent the "Dictionary"

PCA: Some Comments

- The idea of approximating each data point as a combination of basis vectors

$$\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k \qquad \text{or} \quad \mathbf{X} \approx \mathbf{UZ}$$

  is also popularly known as "Dictionary Learning" in signal/image processing; the learned basis vectors represent the "Dictionary"

- Some examples:
  - Each face in a collection as a combination of a small no of "eigenfaces"

PCA: Some Comments

- The idea of approximating each data point as a combination of basis vectors

$$\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk}\boldsymbol{u}_k \qquad \text{or} \qquad \mathbf{X} \approx \mathbf{UZ}$$

  is also popularly known as "Dictionary Learning" in signal/image processing; the learned basis vectors represent the "Dictionary"

- Some examples:
  - Each face in a collection as a combination of a small no of "eigenfaces"

  - Each document in a collection as a comb. of a small no of "topics"

PCA: Some Comments

- The idea of approximating each data point as a combination of basis vectors

$$\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk}\boldsymbol{u}_k \qquad \text{or} \qquad \mathbf{X} \approx \mathbf{UZ}$$

  is also popularly known as "Dictionary Learning" in signal/image processing; the learned basis vectors represent the "Dictionary"

- Some examples:
  - Each face in a collection as a combination of a small no of "eigenfaces"

  - Each document in a collection as a comb. of a small no of "topics"
  - Each gene-expression sample as a comb. of a small no of "genetic pathways"

PCA: Some Comments

- The idea of approximating each data point as a combination of basis vectors

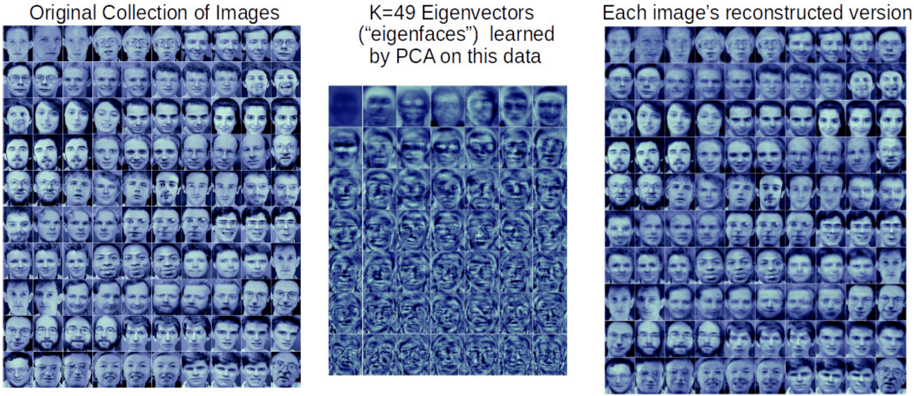
$$\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk}\boldsymbol{u}_k \qquad \text{or} \qquad \mathbf{X} \approx \mathbf{UZ}$$

  is also popularly known as "Dictionary Learning" in signal/image processing; the learned basis vectors represent the "Dictionary"

- Some examples:
  - Each face in a collection as a combination of a small no of "eigenfaces"

  - Each document in a collection as a comb. of a small no of "topics"
  - Each gene-expression sample as a comb. of a small no of "genetic pathways"
  - The "eigenfaces", "topics", "genetic pathways", etc. are the "basis vectors", which can be learned from data using PCA/SVD or other similar methods

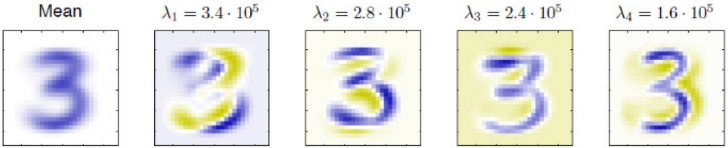
PCA: Example


Original Collection of Images
K=49 Eigenvectors
("eigenfaces") learned
by PCA on this data
Each image's reconstructed version

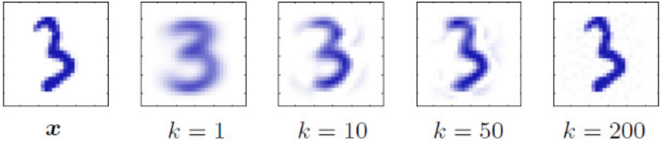
PCA: Example

16×16 pixel images of handwritten 3s (as vectors in $\mathbb{R}^{256}$)

Mean μ and eigenvectors v_1, v_2, v_3, v_4



Reconstructions:



Each input image now represented by just k numbers
(combination weights of each of the k eigenvectors)

PCA: Limitations and Extensions

- A linear projection method

PCA: Limitations and Extensions

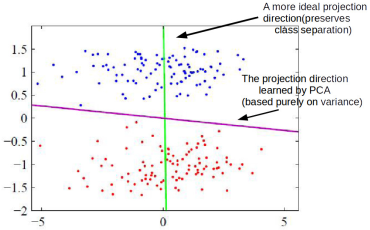
- A linear projection method
  - Won't work well if data can't be approximated by a linear subspace

PCA: Limitations and Extensions

- A linear projection method
  - Won't work well if data can't be approximated by a linear subspace
  - But PCA can be kernelized easily (Kernel PCA)

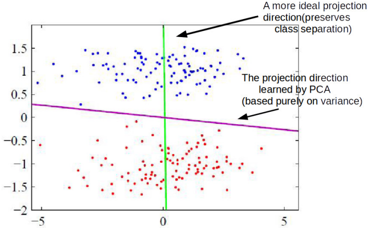
PCA: Limitations and Extensions

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



PCA: Limitations and Extensions

- A linear projection method
  - Won't work well if data can't be approximated by a linear subspace
  - But PCA can be kernelized easily (Kernel PCA)
- 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 exists (e.g., Power Method)