Linear Dimensionality Reduction

Piyush Rai

CS5350/6350: Machine Learning

October 20, 2011

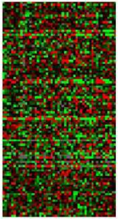
High-Dimensional Datasets Abound..

face images

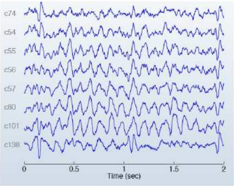
Zambian President Levy Mwanawasa has won a second term in office in an election his challenger Michael Sata accused him of rigging, official results showed on Monday.

According to media reports, a pair of hackers said on Saturday that the Firefox Web browser, commonly perceived as the safer and more customizable alternative to market leader Internet Explorer, is critically flawed. A presentation on the flaw was shown during the ToorCon hacker conference in San Diego.

documents



gene expression data



MEG readings

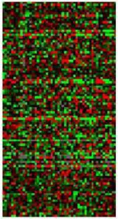
High-Dimensional Datasets Abound..

face images

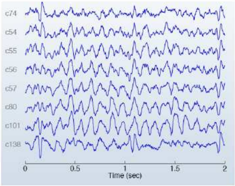
Zambian President Levy Mwanawasa has won a second term in office in an election his challenger Michael Sata accused him of rigging, official results showed on Monday.

According to media reports, a pair of hackers said on Saturday that the Firefox Web browser, commonly perceived as the safer and more customizable alternative to market leader Internet Explorer, is critically flawed. A presentation on the flaw was shown during the ToorCon hacker conference in San Diego.

documents



gene expression data



MEG readings

Goal: Find a low-dimensional, yet useful representation of the data

Why Dimensionality Reduction?

Why Dimensionality Reduction?

- Insights into the low-dimensional structures in the data (visualization)

Why Dimensionality Reduction?

- Insights into the low-dimensional structures in the data (visualization)
- Fewer dimensions ⇒ Less chances of overfitting ⇒ Better generalization

Why Dimensionality Reduction?

- Insights into the low-dimensional structures in the data (visualization)
- Fewer dimensions $\Rightarrow$ Less chances of overfitting $\Rightarrow$ Better generalization
- Speeding up learning algorithms
  - Most algorithms scale badly with increasing data dimensionality

Why Dimensionality Reduction?

- Insights into the low-dimensional structures in the data (visualization)
- Fewer dimensions $\Rightarrow$ Less chances of overfitting $\Rightarrow$ Better generalization
- Speeding up learning algorithms
  - Most algorithms scale badly with increasing data dimensionality
- Less storage requirements (data compression)

Why Dimensionality Reduction?

- Insights into the low-dimensional structures in the data (visualization)
- Fewer dimensions $\Rightarrow$ Less chances of overfitting $\Rightarrow$ Better generalization
- Speeding up learning algorithms
  - Most algorithms scale badly with increasing data dimensionality
- Less storage requirements (data compression)
- Note: Dimensionality Reduction is different from Feature Selection
  - .. although the goals are kind of the same

Why Dimensionality Reduction?

- Insights into the low-dimensional structures in the data (visualization)
- Fewer dimensions ⇒ Less chances of overfitting ⇒ Better generalization
- Speeding up learning algorithms
  - Most algorithms scale badly with increasing data dimensionality
- Less storage requirements (data compression)
- Note: Dimensionality Reduction is different from Feature Selection
  - .. although the goals are kind of the same
- Dimensionality reduction is more like "Feature Extraction"
  - Constructing a small set of new features from the original features

Linear Dimensionality Reduction

- Based on the idea of doing a linear projection of the data

Linear Dimensionality Reduction

- Based on the idea of doing a linear projection of the data
- Works well if the data lies close to a linear subspace

Linear Dimensionality Reduction

- Based on the idea of doing a linear projection of the data
- Works well if the data lies close to a linear subspace
- Consider a high dimensional example $\mathbf{x} \in \mathbb{R}^D$
- We want to project it down to a K-dimensional vector $\mathbf{z}$ ($K \ll D$)

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

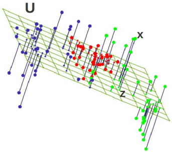
- $\mathbf{z} \in \mathbb{R}^K$ is the projection
- $\mathbf{U}$ is the $D \times K$ projection matrix (defining K **projection directions**)

Linear Dimensionality Reduction

- Based on the idea of doing a linear projection of the data
- Works well if the data lies close to a linear subspace
- Consider a high dimensional example $\mathbf{x} \in \mathbb{R}^D$
- We want to project it down to a K-dimensional vector $\mathbf{z}$ ($K \ll D$)

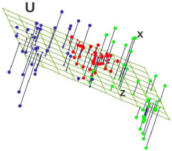
$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

- $\mathbf{z} \in \mathbb{R}^K$ is the projection
- $\mathbf{U}$ is the $D \times K$ projection matrix (defining K **projection directions**)
- Can also think of $\mathbf{U}$ as defining a K-dimensional subspace



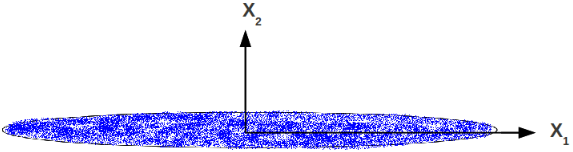
Linear Dimensionality Reduction

- Based on the idea of doing a linear projection of the data
- Works well if the data lies close to a linear subspace
- Consider a high dimensional example $\mathbf{x} \in \mathbb{R}^D$
- We want to project it down to a K-dimensional vector $\mathbf{z}$ ($K \ll D$)
$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$
- $\mathbf{z} \in \mathbb{R}^K$ is the projection
- $\mathbf{U}$ is the $D \times K$ projection matrix (defining K **projection directions**)
- Can also think of $\mathbf{U}$ as defining a K-dimensional subspace



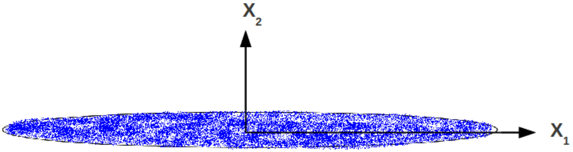
- Different methods differ in how $\mathbf{U}$ is defined/learned
- The differences depend on what properties of data we want to capture

Dimensionality Reduction: A Simple Illustration



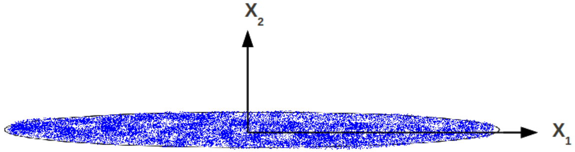
- Consider this 2 dimensional data
- Each example **x** has 2 features $\{x_1, x_2\}$

Dimensionality Reduction: A Simple Illustration



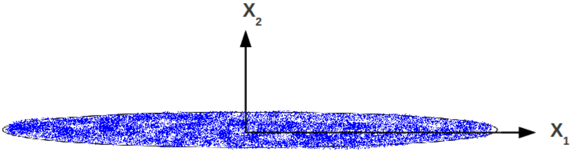
- Consider this 2 dimensional data
- Each example $\mathbf{x}$ has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{x_1\}$

Dimensionality Reduction: A Simple Illustration



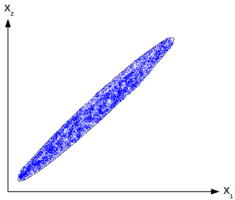
- Consider this 2 dimensional data
- Each example $\mathbf{x}$ has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{x_1\}$
- Are we losing much information by throwing away x_2?

Dimensionality Reduction: A Simple Illustration



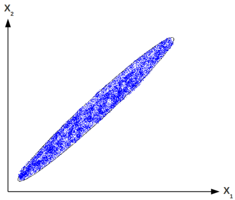
- Consider this 2 dimensional data
- Each example $\mathbf{x}$ has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{x_1\}$
- Are we losing much information by throwing away x_2?
- No. Most of the data spread is along x_1 (very little variance along x_2)

Dimensionality Reduction: A Simple Illustration



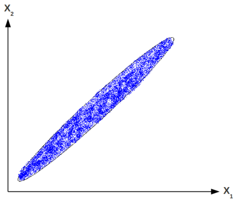
- Consider this 2 dimensional data
- Each example **x** has 2 features $\{x_1, x_2\}$

Dimensionality Reduction: A Simple Illustration



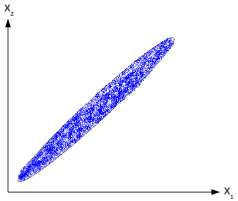
- Consider this 2 dimensional data
- Each example **x** has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example **x** now becomes 1-dimensional $\mathbf{x} = \{x_1\}$

Dimensionality Reduction: A Simple Illustration



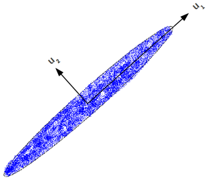
- Consider this 2 dimensional data
- Each example $\mathbf{x}$ has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{x_1\}$
- Are we losing much information by throwing away x_2?

Dimensionality Reduction: A Simple Illustration



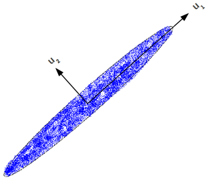
- Consider this 2 dimensional data
- Each example $\mathbf{x}$ has 2 features $\{x_1, x_2\}$
- Consider ignoring the feature x_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{x_1\}$
- Are we losing much information by throwing away x_2?
- Yes. The data has substantial variance along both features (i.e., both axes)

Dimensionality Reduction: A Simple Illustration



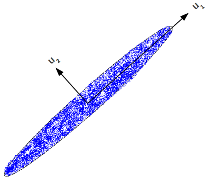
- Now consider a change of axes (the co-ordinate system)

Dimensionality Reduction: A Simple Illustration



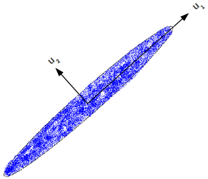
- Now consider a change of axes (the co-ordinate system)
- Each example $\mathbf{x}$ has 2 features $\{u_1, u_2\}$

Dimensionality Reduction: A Simple Illustration



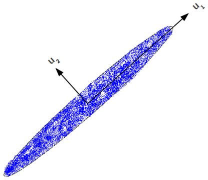
- Now consider a change of axes (the co-ordinate system)
- Each example $\mathbf{x}$ has 2 features $\{u_1, u_2\}$
- Consider ignoring the feature u_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{u_1\}$

Dimensionality Reduction: A Simple Illustration



- Now consider a change of axes (the co-ordinate system)
- Each example $\mathbf{x}$ has 2 features $\{u_1, u_2\}$
- Consider ignoring the feature u_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{u_1\}$
- Are we losing much information by throwing away u_2?

Dimensionality Reduction: A Simple Illustration



- Now consider a change of axes (the co-ordinate system)
- Each example $\mathbf{x}$ has 2 features $\{u_1, u_2\}$
- Consider ignoring the feature u_2 for each example
- Each 2-dimensional example $\mathbf{x}$ now becomes 1-dimensional $\mathbf{x} = \{u_1\}$
- Are we losing much information by throwing away u_2?
- No. Most of the data spread is along u_1 (very little variance along u_2)

Principal Component Analysis (PCA)

- Used when we want projections capturing maximum variance directions

Principal Component Analysis (PCA)

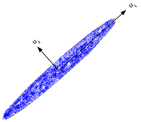
- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data

Principal Component Analysis (PCA)

- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data
- Principal Components (PC): Directions of maximum variability in the data

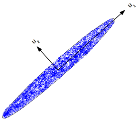
Principal Component Analysis (PCA)

- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data
- Principal Components (PC): Directions of maximum variability in the data
- Roughly speaking, PCA does a change of axes that represent the data



Principal Component Analysis (PCA)

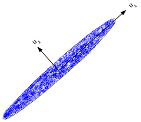
- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data
- Principal Components (PC): Directions of maximum variability in the data
- Roughly speaking, PCA does a change of axes that represent the data



- First PC: Direction of the highest variability
- Second PC: Direction of next highest variability (**orthogonal** to the first PC)

Principal Component Analysis (PCA)

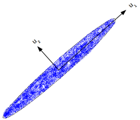
- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data
- Principal Components (PC): Directions of maximum variability in the data
- Roughly speaking, PCA does a change of axes that represent the data



- First PC: Direction of the highest variability
- Second PC: Direction of next highest variability (**orthogonal** to the first PC)
- Subsequent PCs: Other directions of highest variability (in decreasing order)

Principal Component Analysis (PCA)

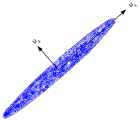
- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data
- Principal Components (PC): Directions of maximum variability in the data
- Roughly speaking, PCA does a change of axes that represent the data



- First PC: Direction of the highest variability
- Second PC: Direction of next highest variability (**orthogonal** to the first PC)
- Subsequent PCs: Other directions of highest variability (in decreasing order)
- Note: All principal components are orthogonal to each other

Principal Component Analysis (PCA)

- Used when we want projections capturing maximum variance directions
- Based on identifying the **Principal Components** in the data
- Principal Components (PC): Directions of maximum variability in the data
- Roughly speaking, PCA does a change of axes that represent the data



- First PC: Direction of the highest variability
- Second PC: Direction of next highest variability (**orthogonal** to the first PC)
- Subsequent PCs: Other directions of highest variability (in decreasing order)
- Note: All principal components are orthogonal to each other
- PCA: Take top K PC's and project the data along those

PCA: Finding the Principal Components

- Given: N examples $\mathbf{x}_1, \ldots, \mathbf{x}_N$, each example $\mathbf{x}_n \in \mathbb{R}^D$
- Goal: Project the data from D dimensions to K dimensions ($K < D$)
- Want to capture the maximum possible variance in the projected data

PCA: Finding the Principal Components

- Given: N examples $\mathbf{x}_1, \dots, \mathbf{x}_N$, each example $\mathbf{x}_n \in \mathbb{R}^D$
- Goal: Project the data from D dimensions to K dimensions ($K < D$)
- Want to capture the maximum possible variance in the projected data
- Let $\mathbf{u}_1, \dots, \mathbf{u}_D$ be the principal components, assumed to be:
  - Orthogonal: $\mathbf{u}_i^\top \mathbf{u}_j = 0$ if $i \neq j$, Orthonormal: $\mathbf{u}_i^\top \mathbf{u}_i = 1$

PCA: Finding the Principal Components

- Given: N examples $\mathbf{x}_1, \ldots, \mathbf{x}_N$, each example $\mathbf{x}_n \in \mathbb{R}^D$
- Goal: Project the data from D dimensions to K dimensions ($K < D$)
- Want to capture the maximum possible variance in the projected data
- Let $\mathbf{u}_1, \ldots, \mathbf{u}_D$ be the principal components, assumed to be:
  - Orthogonal: $\mathbf{u}_i^\top \mathbf{u}_j = 0$ if $i \neq j$, Orthonormal: $\mathbf{u}_i^\top \mathbf{u}_i = 1$
- Each principal component is a vector of size $D \times 1$

PCA: Finding the Principal Components

- Given: N examples $\mathbf{x}_1, \dots, \mathbf{x}_N$, each example $\mathbf{x}_n \in \mathbb{R}^D$
- Goal: Project the data from D dimensions to K dimensions ($K < D$)
- Want to capture the maximum possible variance in the projected data
- Let $\mathbf{u}_1, \dots, \mathbf{u}_D$ be the principal components, assumed to be:
  - Orthogonal: $\mathbf{u}_i^\top \mathbf{u}_j = 0$ if $i \neq j$, Orthonormal: $\mathbf{u}_i^\top \mathbf{u}_i = 1$
- Each principal component is a vector of size $D \times 1$
- We want only the first K principal components

PCA: Finding the Principal Components

- Projection of a data point $\mathbf{x}_n$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \mathbf{x}_n$

PCA: Finding the Principal Components

- Projection of a data point $\mathbf{x}_n$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \mathbf{x}_n$
- Projection of the mean $\bar{\mathbf{x}}$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \bar{\mathbf{x}}$ (where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$)

PCA: Finding the Principal Components

- Projection of a data point $\mathbf{x}_n$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \mathbf{x}_n$
- Projection of the mean $\bar{\mathbf{x}}$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \bar{\mathbf{x}}$ (where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$)
- Variance of the projected data (along projection direction $\mathbf{u}_1$):

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

where $\mathbf{S}$ is the data covariance matrix defined as

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

PCA: Finding the Principal Components

- Projection of a data point $\mathbf{x}_n$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \mathbf{x}_n$
- Projection of the mean $\bar{\mathbf{x}}$ along $\mathbf{u}_1$: $\mathbf{u}_1^\top \bar{\mathbf{x}}$ (where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$)
- Variance of the projected data (along projection direction $\mathbf{u}_1$):

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

where $\mathbf{S}$ is the data covariance matrix defined as

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

- Want to have $\mathbf{u}_1$ that maximizes the projected data variance $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1$
  - Subject to the constraint: $\mathbf{u}_1^\top \mathbf{u}_1 = 1$
  - We will introduce a Lagrange multiplier λ_1 for this constraint

PCA: Finding the Principal Components

- Objective function: $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$
- Taking derivative w.r.t. $\mathbf{u}_1$ and setting it to zero gives:

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

PCA: Finding the Principal Components

- Objective function: $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$
- Taking derivative w.r.t. $\mathbf{u}_1$ and setting it to zero gives:

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

- This is the eigenvalue equation
  - $\mathbf{u}_1$ must be *an* eigenvector of $\mathbf{S}$ (and λ_1 the corresponding eigenvalue)

PCA: Finding the Principal Components

- Objective function: $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$
- Taking derivative w.r.t. $\mathbf{u}_1$ and setting it to zero gives:

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

- This is the eigenvalue equation
  - $\mathbf{u}_1$ must be *an* eigenvector of $\mathbf{S}$ (and λ_1 the corresponding eigenvalue)
- But there are multiple eigenvectors of $\mathbf{S}$. Which one is $\mathbf{u}_1$?

PCA: Finding the Principal Components

- Objective function: $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$
- Taking derivative w.r.t. $\mathbf{u}_1$ and setting it to zero gives:

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

- This is the eigenvalue equation
  - $\mathbf{u}_1$ must be *an* eigenvector of $\mathbf{S}$ (and λ_1 the corresponding eigenvalue)
- But there are multiple eigenvectors of $\mathbf{S}$. Which one is $\mathbf{u}_1$?
- Consider $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 = \mathbf{u}_1^\top \lambda_1 \mathbf{u}_1 = \lambda_1$ (using $\mathbf{u}_1^\top \mathbf{u}_1 = 1$)
- We know that the projected data variance $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 = \lambda_1$ is maximum
  - Thus λ_1 should be the largest eigenvalue
  - Thus $\mathbf{u}_1$ is the first (top) eigenvector of $\mathbf{S}$ (with eigenvalue λ_1)
    $\Rightarrow$ the first principal component (direction of highest variance in the data)

PCA: Finding the Principal Components

- Objective function: $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$
- Taking derivative w.r.t. $\mathbf{u}_1$ and setting it to zero gives:

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

- This is the eigenvalue equation
  - $\mathbf{u}_1$ must be *an* eigenvector of $\mathbf{S}$ (and λ_1 the corresponding eigenvalue)
- But there are multiple eigenvectors of $\mathbf{S}$. Which one is $\mathbf{u}_1$?
- Consider $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 = \mathbf{u}_1^\top \lambda_1 \mathbf{u}_1 = \lambda_1$ (using $\mathbf{u}_1^\top \mathbf{u}_1 = 1$)
- We know that the projected data variance $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 = \lambda_1$ is maximum
  - Thus λ_1 should be the largest eigenvalue
  - Thus $\mathbf{u}_1$ is the first (top) eigenvector of $\mathbf{S}$ (with eigenvalue λ_1)
    $\Rightarrow$ the first principal component (direction of highest variance in the data)
- Subsequent PC's are given by the subsequent eigenvectors of $\mathbf{S}$

PCA: The Algorithm

- Compute the mean of the data

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

- Compute the sample covariance matrix (using the mean subtracted data)

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

- Do the eigenvalue decomposition of the $D \times D$ matrix $\mathbf{S}$
- Take the top K eigenvectors (corresponding to the top K eigenvalues)
- Call these $\mathbf{u}_1, \ldots, \mathbf{u}_K$ (s.t. $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{K-1} \geq \lambda_K$)
- $\mathbf{U} = [\mathbf{u}_1 \ \mathbf{u}_2 \ \ldots \ \mathbf{u}_K]$ is the projection matrix of size $D \times K$

PCA: The Algorithm

- Compute the mean of the data

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

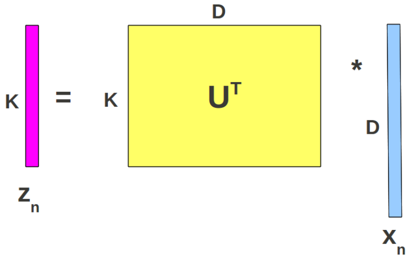
- Compute the sample covariance matrix (using the mean subtracted data)

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

- Do the eigenvalue decomposition of the $D \times D$ matrix $\mathbf{S}$
- Take the top K eigenvectors (corresponding to the top K eigenvalues)
- Call these $\mathbf{u}_1, \ldots, \mathbf{u}_K$ (s.t. $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{K-1} \geq \lambda_K$)
- $\mathbf{U} = [\mathbf{u}_1 \ \mathbf{u}_2 \ \ldots \ \mathbf{u}_K]$ is the projection matrix of size $D \times K$
- Projection of each example $\mathbf{x}_n$ is computed as $\mathbf{z}_n = \mathbf{U}^\top \mathbf{x}_n$
  - $\mathbf{z}_n$ is a $K \times 1$ vector (also called the embedding of $\mathbf{x}_n$)

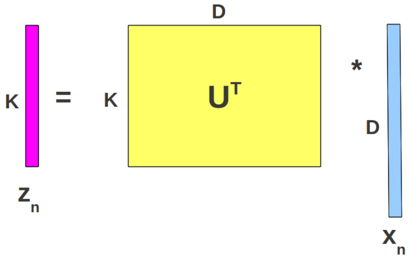
PCA: Pictorially

- For a single example $\mathbf{x}_n$:

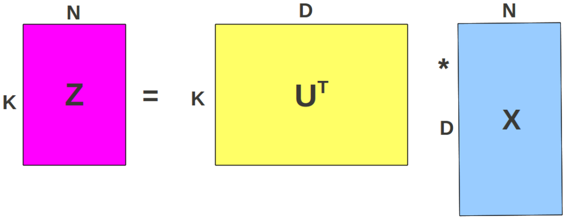


PCA: Pictorially

- For a single example $\mathbf{x}_n$:

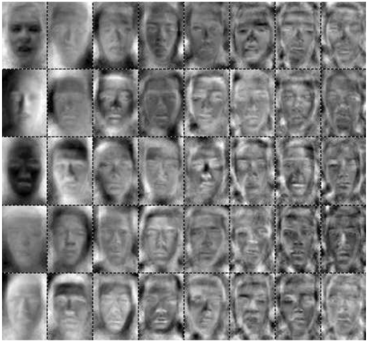


- For a set of N examples:



PCA Example: Eigenfaces

- Principal Components learned using a face image dataset



PCA: Approximate Reconstruction

- Given the principal components $\mathbf{u}_1, \ldots, \mathbf{u}_K$, the PCA approximation of an example $\mathbf{x}_n$ is:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{K} (\mathbf{x}_n^\top \mathbf{u}_i)\mathbf{u}_i = \sum_{i=1}^{K} z_{ni}\mathbf{u}_i$$

where $\mathbf{z}_n = [z_{n1}, \ldots, z_{nK}]$ is the low-dimensional projection of $\mathbf{x}_n$

PCA: Approximate Reconstruction

- Given the principal components $\mathbf{u}_1, \ldots, \mathbf{u}_K$, the PCA approximation of an example $\mathbf{x}_n$ is:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{K} (\mathbf{x}_n^\top \mathbf{u}_i)\mathbf{u}_i = \sum_{i=1}^{K} z_{ni}\mathbf{u}_i$$

  where $\mathbf{z}_n = [z_{n1}, \ldots, z_{nK}]$ is the low-dimensional projection of $\mathbf{x}_n$
- This gives us a way of compressing data
- To compress a dataset $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, all we need is the set of $K \ll D$ principal components, and the projections $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$ of each example

PCA: Approximate Reconstruction

- Given the principal components $\mathbf{u}_1, \ldots, \mathbf{u}_K$, the PCA approximation of an example $\mathbf{x}_n$ is:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{K} (\mathbf{x}_n^\top \mathbf{u}_i)\mathbf{u}_i = \sum_{i=1}^{K} z_{ni}\mathbf{u}_i$$

where $\mathbf{z}_n = [z_{n1}, \ldots, z_{nK}]$ is the low-dimensional projection of $\mathbf{x}_n$

- This gives us a way of compressing data
- To compress a dataset $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, all we need is the set of $K \ll D$ principal components, and the projections $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$ of each example

PCA for Very High Dimensional Data

- In many cases, $N < D$
- Recall: PCA requires eigen-decomposition of $D \times D$ covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ (assuming centered data, and $\mathbf{X}$ being $D \times N$)
- Eigen-decomposition can be expensive if D is very large

PCA for Very High Dimensional Data

- In many cases, $N < D$
- Recall: PCA requires eigen-decomposition of $D \times D$ covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ (assuming centered data, and $\mathbf{X}$ being $D \times N$)
- Eigen-decomposition can be expensive if D is very large
- Fact: If $N < D$, at most $N - 1$ eigenvalues are non-zero
  - The remaining $D - N + 1$ eigenvalues are zero

PCA for Very High Dimensional Data

- In many cases, $N < D$
- Recall: PCA requires eigen-decomposition of $D \times D$ covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ (assuming centered data, and $\mathbf{X}$ being $D \times N$)
- Eigen-decomposition can be expensive if D is very large
- Fact: If $N < D$, at most $N - 1$ eigenvalues are non-zero
  - The remaining $D - N + 1$ eigenvalues are zero
- Fact: $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ has the same $N - 1$ non-zero eigenvalues as that of the $N \times N$ matrix $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$ (for which eigen-decomposition is cheaper if $N < D$)

PCA for Very High Dimensional Data

- In many cases, $N < D$
- Recall: PCA requires eigen-decomposition of $D \times D$ covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ (assuming centered data, and $\mathbf{X}$ being $D \times N$)
- Eigen-decomposition can be expensive if D is very large
- Fact: If $N < D$, at most $N - 1$ eigenvalues are non-zero
  - The remaining $D - N + 1$ eigenvalues are zero
- Fact: $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ has the same $N - 1$ non-zero eigenvalues as that of the $N \times N$ matrix $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$ (for which eigen-decomposition is cheaper if $N < D$)
- The eigenvectors aren't exactly the same (but still related)
- The relationship is $\mathbf{u}_i = \frac{1}{(N\lambda_i)^2}\mathbf{X}\mathbf{v}_i$
- $\{\lambda_i, \mathbf{v}_i\}$ is an eigenvalue-eigenvector pair of the $N \times N$ matrix $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$, and $\mathbf{u}_i$ is the corresponding eigenvector of $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$ (that we want)

Supervised Dimensionality Reduction

- Dimensionality reduction with label information (when the ultimate goal is classification/regression)

Supervised Dimensionality Reduction

- Dimensionality reduction with label information (when the ultimate goal is classification/regression)
- PCA ignores label information even if it is available
  - Only chooses directions of maximum variance

Supervised Dimensionality Reduction

- Dimensionality reduction with label information (when the ultimate goal is classification/regression)
- PCA ignores label information even if it is available
  - Only chooses directions of maximum variance
- Fisher Discriminant Analysis (FDA) takes into account the label information
  - It's also called Linear Discriminant Analysis (LDA)

Supervised Dimensionality Reduction

- Dimensionality reduction with label information (when the ultimate goal is classification/regression)
- PCA ignores label information even if it is available
  - Only chooses directions of maximum variance
- Fisher Discriminant Analysis (FDA) takes into account the label information
  - It's also called Linear Discriminant Analysis (LDA)
- FDA/LDA projects data while preserving class separation

Supervised Dimensionality Reduction

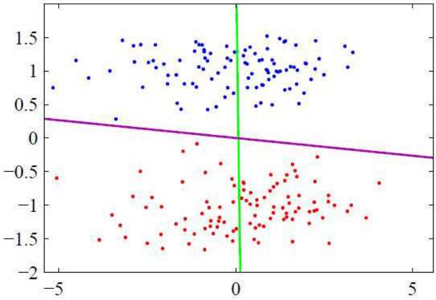
- Dimensionality reduction with label information (when the ultimate goal is classification/regression)
- PCA ignores label information even if it is available
  - Only chooses directions of maximum variance
- Fisher Discriminant Analysis (FDA) takes into account the label information
  - It's also called Linear Discriminant Analysis (LDA)
- FDA/LDA projects data while preserving class separation
  - Examples from same class are put closely together by the projection

Supervised Dimensionality Reduction

- Dimensionality reduction with label information (when the ultimate goal is classification/regression)
- PCA ignores label information even if it is available
  - Only chooses directions of maximum variance
- Fisher Discriminant Analysis (FDA) takes into account the label information
  - It's also called Linear Discriminant Analysis (LDA)
- FDA/LDA projects data while preserving class separation
  - Examples from same class are put closely together by the projection
  - Examples from different classes are placed far apart by the projection

PCA vs FDA/LDA

- PCA: magenta line, FDA: green line



- PCA based projection makes the classes overlap (which is bad)
- LDA/FDA is often better if the final goal is classification