Dimensionality Reduction
Linear Method: PCA I

M. R. Hasan
What We Will Cover

• Subspace Estimation: Dimensionality Reduction
• Feature selection vs. feature extraction
• Linear method vs. nonlinear method
• Unsupervised technique: Principle Component Analysis (PCA)
• PCA: Applications
• PCA: Limitations
Readings

• Alpaydin: 6.1, 6.3, 6.6, 6.8
• Geron: Chapter 8
• Jupyter notebook:
Motivating Question: Can we find a small number of features that accurately capture the relevant properties of the data?

For example: we can describe the trajectory of a tennis ball by its velocity, diameter, and mass.
Motivation

• The problem of finding a small set of features is known as **subspace estimation**.

• A popular technique for subspace estimation is the **dimensionality reduction**.
In dimensionality reduction, we intend to **project the data** into a *low-dimensional* subspace.

3D Dataset

Intrinsic lower dimension is 2D

Projected to 2D
Dimensionality Reduction

- Depending on the type of the data, they may live in one of the two types of sub-spaces in lower dimension:
  - Linear sub-space
  - Nonlinear sub-space
Dimensionality Reduction

Although following dataset is 2D, in lower 1D it lives on a **linear** sub-space.

Although following dataset is 3D, in lower 2D it lives on a **nonlinear** sub-space.
Dimensionality Reduction

• Thus, depending on the type of the lower-dimensional sub-space, we will discuss **two types** of dimensionality reduction techniques.
  - Linear methods
  - Nonlinear methods
Dimensionality Reduction

• First, let’s present the **main intuition** behind dimensionality reduction.
Motivation

How much information is needed to discriminate the person in these pictures?
• Input data may have **thousands or millions of dimensions**!
• For example, images may have hundreds of thousands of pixels.
Dimensionality Reduction

• Learning becomes **expensive or inefficient**!
• One solution is to **decrease the complexity** of the data.
• But how?
Dimensionality Reduction

• Represent **data with fewer dimensions**!
• Our assumption is that the essence or core content of a data **does not span along all dimensions**.

Identity is retained in **lower dimension**!
Dimensionality Reduction

- Data (x) may live in a **high-dimensional space** $\mathbb{R}^d$.
- But it really occupies a **small “lower dimension”** subspace.
- Consider the following example.
- Although **x is 2D**, it occupies (is alive) only **one direction**.

Therefore, if we could **project the data onto one dimension** along the **largest variance**, the learning becomes easier.
Dimensionality Reduction

• Our goal is to discover the dimensions that matter most!
The aim of dimensionality reduction is to **preserve as much of the significant structure** of the high-dimensional data as possible in the **low-dimensional map**.

It identifies the **combination of features** (principal components, or directions in the feature space) that **account for the most variance** in data.
Dimensionality Reduction

• **Benefits** of dimensionality reduction:
  - Learning becomes *easier* because of *fewer* parameters.
  - Enables *visualization* by projecting data in lower dimension.
  - We can discover "*intrinsic dimensionality*” of data.
  - Discover that for some problems, the high dimensional data that is *truly lower dimensional*. 
Dimensionality Reduction

- There are two main methods for reducing dimensionality:
  - Feature selection
  - Feature extraction
In feature selection, we are interested in finding $k$ of the $d$ dimensions that give us the most information and we discard the other $(d - k)$ dimensions.

- Example: most of the variation is captured in the $x_1$ dimension, so discard $x_2$. 
In feature extraction, we are interested in finding a new set of $k$ dimensions that are combinations of the original $d$ dimensions.

Example: most of the variation is neither captured in the $x_1$ dimension nor in the $x_2$ dimension.

Combine $(x_1, x_2)$ to create $(z_1, z_2)$, then discover $z_1$ where most of the variation happens.
Dimensionality Reduction

- The feature extraction methods may be **supervised or unsupervised** depending on whether or not they use the output information.
- The best known and most widely used **feature extraction** methods are
  - Principal Components Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
Both PCA & LDA are linear projection methods.  
- PCA is unsupervised.  
- LDA is supervised.

Linear projection methods work when features of the dataset maintains a linear relationship in lower dimension.
Dimensionality Reduction

- The linear methods of dimensionality reduction are flexible, fast, and easily interpretable.
- For example, PCA works well when the data lies in a linear sub-space.
- However, this may not hold in many applications when there are nonlinear relationships within the data in lower dimension.
Dimensionality Reduction

- For example, the following dataset is 3D.
- However, notice that it is created by *folding* a 2D plane.
- Thus the **intrinsic lower dimension** is 2D.

The lower-dimension subspace is **nonlinear**!
Dimensionality Reduction

**Nonlinear Methods**

- One approach to resolve this issue is to use a **nonlinear** dimensionality reduction technique.
- It is known as **Manifold Learning**.
- In Manifold Learning we assume that the data of interest lie on an **embedded nonlinear manifold** within the higher-dimensional space.
Dimensionality Reduction

• **Nonlinear Methods**
  
  • More generally, a \( d \)-dimensional manifold is a part of an \( n \)-dimensional space (where \( d < n \)) that locally resembles a \( d \)-dimensional hyperplane.

The Swiss roll is an example of a **2D manifold**.

It locally resembles a **2D plane**, but it is rolled in the 3D.
Dimensionality Reduction

- **Manifold Learning Algorithms**
  - There are many algorithms for implementing Manifold Learning.
    - Locally Linear Embedding (LLE)
    - Isometric feature mapping (Isomap)
    - Multi-dimensional Scaling (MDS)
    - t-distributed Stochastic Neighbor Embedding (t-SNE)
  - For information on other algorithms and their scikit-learn implementation: [https://scikit-learn.org/stable/modules/manifold.html](https://scikit-learn.org/stable/modules/manifold.html)
Dimensionality Reduction

• For an empirical understanding of both the linear and nonlinear methods of dimensionality reduction (including Manifold Learning), see the notebooks in the Github repository:

Dimensionality Reduction

- We will discuss two dimensionality reduction techniques.
  - Linear method: PCA
  - Nonlinear method: t-SNE
Principle Component Analysis (PCA)
Dimensionality Reduction

- PCA was formulated by mathematical statistician Harold Hotelling in 1933 and published in the Journal of Educational Psychology.

Hotelling was trying to solve the problem of multiple indicators of educational ability (e.g., reading speed, arithmetic speed) to **discover a single general construct** (known as “general ability”).
Dimensionality Reduction

- We will use **PCA** to project data into lower dimension.
- In projection methods, we are interested in finding a **mapping from the inputs** in the original $d$-dimensional space to a new $(k < d)$-dimensional space, with **minimum loss of information**.

![Dimensionality Reduction Diagram](image)
Motivation for PCA

• Assume we are given a class of students with grades on five tasks:
  - Mid 1
  - Mid 2
  - Assignment 1
  - Assignment 2
  - Final Exam

• We want to rank these students.
• The features are 5 dimensional.

• Our ranking task would be trivial if we could “create” a feature that captures the variance in the grades.
Motivation for PCA

- Hence, we want to find a technique for projecting the 5 dimensional features onto one dimension with maximum variance.
- In other words, we want to extract a feature (not to choose from the existing 5).
- PCA can do this for us.
Principle Component Analysis (PCA)

• What is the **main idea** of PCA?
• Let’s consider a **2D data distribution**.
• We want to project it to 1D.

Thus, we want to find a **feature** that can explain most of the variance of the data.

Example: instead of using mid 1 and mid 2 scores, create a **single metric** that can explain the variance.
The problem is that the two features could be correlated. I.e., it has non-zero covariance. It means that both contribute to the variance of the data. In other words, we need both features to explain the variation.

We cannot determine if one feature contributes more to the variance.
To find the **most discriminating feature** of the two, we could **de-correlate the features** such that their covariance vanishes. These are **new features** created by combining the old features.
This can be done by projecting the data along a new set of axes that are not correlated.
In other words, change the bases!
Principle Component Analysis (PCA)

- Consider a 3D vector (green arrow).
- It is represented by a basis \((x, y, z)\) in the 3D vector space.
- The basis is a combination of vectors.
A vector space can have several distinct sets of basis vectors. This is because we can form any vector in the spanning space as a linear combination of these vectors. This allows us to change bases.

Example: A vector is represented by two different set of basis (purple and red arrows).
PCA is accomplished by the change of basis such that features are de-correlated.

Combining the current bases we create new bases such the features are de-correlated.
Principle Component Analysis (PCA)

- In the projected data (new bases), we see that the horizontal axis accounts for most of the variance of the data.
- Thus, we have created (extracted) a new feature (along the horizontal axis) that contributes most to the variance.
- From the 2D feature space, we extracted a 1D feature.

![Diagram of Principle Component Analysis](image)
In summary, by representing by de-correlated bases, we create a new **Data Model** (new covariance matrix)!

This Data Model may **enable visualization** in $\leq 2D$. 
How does the PCA work?
Principle Component Analysis (PCA)

- In the following figure, feature is two dimensional: $x_1$ and $x_2$
- But $(x_1-x_2)$ are correlated (none of them alone explain the variance).
- Observe, that the sample variance is the largest along $z_1$.
- But $z_1$ is not aligned with any of the feature dimension ($x_1$ or $x_2$).
Principle Component Analysis (PCA)

• So, we don’t know which feature contributes more towards the variance of the data.

To de-correlate \((x_1-x_2)\), we could rotate the data such that \(z_1 - z_2\) becomes the new basis.
Principle Component Analysis (PCA)

- PCA does this first by **centering the sample**.
- Then **rotate** the axes \((x_1-x_2)\) to **line up with the directions of highest variance**.
- In short, we **change the basis** from \((x_1-x_2)\) to \((z_1-z_2)\).
Principle Component Analysis (PCA)

- Now \((z_1 - z_2)\) is the new basis and these two new features are de-correlated.
- Our data is still 2D.
- If the variance on \(z_2\) is too small, it can be ignored.
- Then, we have dimensionality reduction from two to one.
How does PCA find the De-Correlated Features \((z_1-z_2)\)?
PCA: De-Correlation of Features

- The PCA de-correlates the features by diagonalizing the data covariance matrix.
PCA: De-Correlation of Features

- In a **diagonal co-variance matrix**, the **non-diagonal** elements are zero.
- Thus, the covariance vanishes!
PCA: De-Correlation of Features

- We will see that after a non-diagonal matrix is diagonalized, its *new bases would be its eigenvectors* and the diagonal matrix will contain its eigenvalues (new variances).
- The diagonalization of a matrix can be interpreted as a *rotation of the axes* to align them with the eigenvectors (new bases).
- This is typically done by a technique called **Eigendecomposition**.
PCA: De-Correlation of Features

- **Eigendecomposition**: a technique of **diagonalizing** a symmetric matrix by decomposing it into a set of **eigenvectors** and **eigenvalues**.

\[
A = XDX^T
\]

**Matrix A is non-diagonal**

\[
\begin{bmatrix}
5 & 4 \\
4 & 6
\end{bmatrix}
\]

**Matrix A is Symmetric**

\[
A^T = A
\]

**Diagonalized Matrix**

\[
\begin{bmatrix}
5 & 0 \\
0 & 1
\end{bmatrix}
\]

**X: eigenvectors**

**D: eigenvalues**
PCA: De-Correlation of Features

- After **diagonalizing** the covariance matrix, its diagonal elements (**variance** components) will be its **eigenvalues** $\lambda$.
- The off-diagonal elements (covariances) will be zero.
PCA: De-Correlation of Features

• Observe that the **eigenvalues equal the variance** components of the covariance matrix.

• The **eigenvectors** corresponding to the eigenvalues are shown in green and magenta.

The eigenvectors give information of the max variance.
PCA: De-Correlation of Features

• Here the largest eigenvector (horizontal axis) is the **extracted feature** that accounts for most of the variance of the data.

• Hence, after diagonalizing the covariance matrix we are able to extract the **most discriminating feature**.
PCA: Implementation via Eigendecomposition
PCA: Implementation

• Now let’s present the **implementation** of PCA.
• In PCA dimensionality reduction technique, our goal is to transform data \( X \in \mathbb{R}^d \) to \( Z \in \mathbb{R}^k \) where \( k < d \)
• We do it in two steps.
  - **Step 1**: project the data using the new bases so that covariance vanishes.
  - **Step 2**: Then, keep the dimensions along which we have large variance.
PCA: Step 1 - KL Transform

• Step 1:
  • This transformation is formally known as **Karhunen-Loeve (KL) transform**.
  
  • First we project the data from $d$ dimension to another $d$ dimension.
  
  • Then we reduce the dimension in step 2.
PCA: Step 1 - KL Transform

- **Step 1:**
- Let’s do this projection: from $X \in \mathbb{R}^d$ to $Z \in \mathbb{R}^d$
- Assume: $E[X] = 0$
- Or **center the data**: $X = X - E[X]$ so that $E[X] = 0$
- Covariance of $X$: $\Sigma_x = E[(X - E[X])(X - E[X])^T]$
  $$= E[XX^T] - E[X]E[X]^T$$
  $$= E[XX^T]$$
- Dimension of $\Sigma_x$: $d \times d$

\[
\text{cov}[x, y] = \mathbb{E}_{x,y} \{x - \mathbb{E}[x]\{y^T - \mathbb{E}[y^T]\} \\
= \mathbb{E}_{x,y} [xy^T] - \mathbb{E}[x]\mathbb{E}[y^T].
\]
PCA: Step 1 - KL Transform

- We want to **transform features** as follows:
  \[ Z = U^T X \]
- Where \( U \) is an **orthonormal matrix**.
- This \( U \) vectors will be used for **basis transformation**.

\[
U_i^T U_j = \begin{cases} 
0 & \text{if } i \neq j \text{ (orthogonal)} \\
1 & \text{if } i = j \text{ (unit norm)}
\end{cases}
\]
PCA: Step 1 - KL Transform

• We need to find \( U \) such that the new feature basis vectors \( Z \in \mathbb{R}^d \) are de-correlated.

• In other words, after transformation its covariance matrix is diagonal \( \Sigma_z \) that contains the eigenvalues over the main diagonal.

\[
\Sigma_z = \begin{bmatrix}
\lambda_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \lambda_d
\end{bmatrix}_{d \times d}
\]

How do we make this transformation?
PCA: Step 1 - KL Transform

- Let's compute the **covariance** of the projected vector $Z$:
  - $\Sigma_z = E[(Z - E[Z])(Z - E[Z])^T]$
  - $= E[ZZ^T]$

- Note: $E[Z] = E[U^TX] = U^TE[X] = 0$
PCA: Step 1 - KL Transform

• $\Sigma_z = E[ZZ^T]$
  
  $= E[(U^T X) (U^T X)^T]$
  
  $= E[U^T XX^T U]$
  
  $= U^T E[XX^T] U$

• Hence, $\Sigma_z = U^T \Sigma_x U$

$Z = U^T X$
Hence, $\Sigma_z = U^T \Sigma_x U$

Now let’s do a little **linear algebra trick**.

Multiply by $U$ (left) and by $U^T$ (right).

$U \Sigma_z U^T = UU^T \Sigma_x UU^T$

$U \Sigma_z U^T = \Sigma_x$  \quad [U^T U = I = U U^T]$

$\Sigma_x = U \Sigma_z U^T$

Hence, we have found the **eigendecomposition** of the **covariance matrix of our sample data**!

This is eigendecomposition of $\Sigma_x$ because we can show that it’s an eigenvector equation of $\Sigma_x$.
PCA: Step 1 - KL Transform

- Eigendecomposition: \( \Sigma_x = U\Sigma_z U^T \)
- Multiplying by U (right):
  \( \Sigma_x U = U\Sigma_z U^T U \)
- \( \Sigma_x U = U\Sigma_z \) \([U^TU = I]\)
- We derived an **eigenvector equation** for the covariance matrix \( \Sigma_x \).
- Hence, \( \Sigma_z \) is a diagonal matrix in which the entries are the **eigenvalues**.
- For an arbitrary eigenvector \( U_i \): \( \Sigma_x U_i = \lambda U_i \)

\[
\Sigma_z = \begin{bmatrix}
\lambda_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \lambda_d
\end{bmatrix}_{d \times d}
\]

The eigenvalues represent the **variance of data** along the projected bases (Z)
PCA: Step 1 - KL Transform

• $\Sigma_x U = U \Sigma_z$
• In other words, we have decomposed the covariance matrix $\Sigma_x$ into its eigenvectors and eigenvalues.
• Its eigenvectors are orthonormal.
• Its eigenvalues represent the variance along each new dimension.
• For an arbitrary eigenvector $U_i$: $\Sigma_x U_i = \lambda U_i$

$$\Sigma_z = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}_{d \times d}$$
PCA: Step 1 - KL Transform

\[ \Sigma_x = E[(X - E[X])(X - E[X])^T] = E[XX^T] \]

• \( \Sigma_x U = U\Sigma_z \)
• Thus, finding the eigenvectors of \( \Sigma_x \) is trivial.
• First, we compute the estimated covariance from the sample.

\[ \hat{\Sigma}_x = \frac{1}{N} \sum \hat{x}_i \hat{x}_i^T \]

Using **matrix notation** for the Data matrix \( X \) (denominator \( N-1 \) is used for an unbiased estimator)

\[ \hat{\Sigma}_x = \frac{X^T \cdot X}{N - 1} \]

Then, solve the eigenvalue equation.
PCA: Step 2

- **Step 1**: project the data using the new bases so that covariance vanishes.
- **Step 2**: Then, keep the dimensions along which we have large variance.

**Step 2:**

- After finding the eigenvectors, we **sort them** from large to small eigenvalues.
- Then, we can only keep the **eigenvectors (k) with largest eigenvalues**.
- These are the **principle axes**.
- This is how the dimension of the features is reduced ($k < d$).

**Summary of PCA:**

Eigendecomposition of $\Sigma_x(X^T.X)$ to find the eigenvectors $U$ corresponding to largest eigenvalues.
Steps: PCA

- Take all training data $X$ (dimension: $N \times d$)
- **Center it**: subtract mean from each row of $X$
- Compute the **covariance matrix**: $\Sigma_x = E[XX^T]$
- Find the eigenvectors & eigenvalues of $\Sigma_x$: $\Sigma_x U = U \Sigma_z$
- Principle components: $k$ eigenvectors with highest eigenvalues.

$$\hat{\Sigma}_x = \frac{X^T X}{N - 1}$$
PCA: Python Implementation

- See the following notebook for understanding the **manual implementation** of the eigendecomposition based PCA using python.
Limitations:
Eigendecomposition based PCA
Limitations of the Eigendecomposition Based PCA

• So far we have discussed the PCA using the **eigendecomposition technique**.
• However, this technique would not work if the sample covariance matrix is **not square**.
• Recall that **rectangular matrices do not have eigenvalues**!

\[
\Sigma_x = E[(X - E[X])(X - E[X])^T] = E[XX^T]
\]
Limitations of the Eigendecomposition Based PCA

- Another issue with the eigendecomposition based PCA is its **computational complexity**.
- Note that the covariance matrix $\Sigma_{d \times d}$ can be very large (10,000+ features)
- Complexity of eigendecomposition is $O(d^3)$.
- Therefore, the eigenvalue computation will be **extremely slow**.

\[
\hat{\Sigma}_x = \frac{1}{N} \sum \vec{x}_i \vec{x}_i^T
\]

Note that we take the dot product of **each feature (d dimension) with itself**, then compute the average of the sum.