Dimensionality Reduction
Linear Method: PCA II

M. R. Hasan
What We Will Cover

• Principle Component Analysis (PCA) by Singular Value Decomposition (SVD)
• Eigendecomposition vs SVD
• PCA: Implementation Issues
• PCA: Applications
• PCA: Limitations
Readings

• Alpaydin: 6.1, 6.3, 6.6, 6.8
• Geron: Chapter 8
• Jupyter notebook:
A Sophisticated Technique for Implementing PCA
A Sophisticated Technique for Implementing PCA

• To reduce computational complexity we can apply a linear algebra trick.

• It is called **Singular Value Decomposition (SVD)**.
Singular Value Decomposition (SVD)

• Using the SVD we can **decompose a matrix of arbitrary size** \((N \times d)\).

• The SVD provides a **better way to factorize** a matrix, into singular vectors and singular values.

The SVD allows us to discover some of the **same kind of information** as the eigendecomposition.
Singular Value Decomposition (SVD)

• SVD works on a non-symmetric matrix.
• For example, when X is a \( N \times d \) dimensional matrix
• SVD of X:

\[
X = U S V^T
\]

• U & V are orthonormal, and S is diagonal.
• The decomposition of X is much faster than finding eigenvectors of \( X^T X \).
• The idea is to derive the eigenvalue system using SVD.
Singular Value Decomposition (SVD)

- After applying the SVD on $X$, we compute
- $X^TX = (U S V^T)^T U S V^T$
  
  $= V S^T U^T U S V^T$
  
  $= V S^T S V^T$
  
  $= V S^2 V^T \ [S^2 \text{ is diagonal}]$

\[ X = U S V^T \]
Singular Value Decomposition (SVD)

- We can derive the eigensystem:
- \( X^TX = V S^2 V^T \)
- Multiply by \( V^T \) from left.
  \[ V^T (X^TX) = V^T S^2 V^T \]
- \( V^T \) is the eigenvector of \( X^TX \)
- \( S_{ii} \) is an eigenvalue of \( X^TX \)
- So now we have to find the eigenvectors \( V^T \) of the covariance matrix \( X^TX \)

Recall that our goal is to find the eigenvectors of the covariance matrix \( X^TX \)
Singular Value Decomposition (SVD)

- We derived the eigensystem: $V^T (X^T X) = S^2 V^T$
- SVD of $X$: $X = U S V^T$
- Hence, simply by computing the SVD of $X$, we can extract the eigenvectors $V^T$ and eigenvalues $S$.

From this we calculate $S_{ii}$ which are eigenvalues of $X^T X$. 
Singular Value Decomposition (SVD)

• Finding the eigenvectors of the covariance matrix ($X^TX$) using SVD is much faster.

• The complexity is: $O(N^2d)$ when $N < d$

• This technique allows us to avoid constructing the large $X^TX$ matrix (d x d).
Truncated SVD

• In truncated SVD, we can ask for top largest eigenvectors.
Eigendecomposition vs. SVD
Eigendecomposition vs. SVD

• Let’s **compare** eigendecomposition with the SVD.
• The singular value decomposition is similar to eigendecomposition, except this time we will write $X$ as a **product of three matrices**:

$$X = U S V^T$$
Eigendecomposition vs. SVD

- $X = U S V^T$
- Suppose that $X$ is an $m \times n$ matrix.
- Then $U$ is defined to be an $m \times n$ (or $m \times m$) matrix,
- $S$ to be an $n \times n$ ($m \times n$) matrix, and
- $V$ to be an $n \times n$ matrix.
Eigendecomposition vs. SVD

- $X = U S V^T$
- Each of these matrices is defined to have a **special structure**.
- The matrices $U$ and $V$ are both defined to be **orthogonal** matrices.
- The matrix $S$ is defined to be a **diagonal matrix**.
- Note that $S$ is **not necessarily square**.
Eigendecomposition vs. SVD

- $X = U S V^T$
- The elements along the diagonal of S are known as the **singular values of the matrix X**.
- The columns of U are known as the **left-singular vectors**.
- The columns of V are known as the **right-singular vectors**.
Eigendecomposition vs. SVD

- $X = U S V^T$
- We can actually interpret the singular value decomposition of $X$ in terms of the eigendecomposition of functions of $X$.
  - The left-singular vectors of $X$ are the eigenvectors of $XX^T$.
  - The right-singular vectors of $X$ are the eigenvectors of $X^TX$.
  - The non-zero singular values of $X$ are the square roots of the eigenvalues of $X^TX$.
  - The same is true for $XX^T$. 
Eigendecomposition vs. SVD

- \( X = U S V^T \)
- The singular values \( \sigma_1, \ldots, \sigma_r \) of an \( m \times n \) matrix \( X \) are the positive square roots, \( \sigma_i = \sqrt{\lambda_i} > 0 \), of the nonzero eigenvalues of the associated matrix \( K = X^TX \).
Eigendecomposition vs. SVD

- $X = U S V^T$
- The SVD allows us to discover some of the same kind of information as the eigendecomposition.
- However, the SVD is more generally applicable.
- Every real matrix has a singular value decomposition, but the same is not true of the eigenvalue decomposition.
Eigendecomposition vs. SVD

- Recall that the eigendecomposition involves analyzing a matrix $X$ to *discover a matrix $V$ of eigenvectors* and a vector of eigenvalues $\lambda$ such that we can rewrite $X$ as
  - $X = V \text{diag}(\lambda)V^{-1}$
- The singular value decomposition is *similar*, except this time we will write $X$ as a *product of three matrices*:
  - $X = U S V^T$
PCA: Implementation Issues
PCA: Implementation Issues

• Let’s discuss some issues for implementing PCA.

• For example, **how many eigenvectors** should we choose?
PCA: Implementation Issues

• In practice even if all eigenvalues are greater than 0, we understand that some eigenvalues have little contribution to variance and may be discarded.

• Then, we take into account the leading k components that explain more than, for example, 90 percent, of the variance.

• When $\lambda_i$ are sorted in descending order, the proportion of variance explained by the k principal components is:

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$
PCA: Implementation Issues

- If the dimensions are highly correlated, there will be a small number of eigenvectors with large eigenvalues and $k$ will be much smaller than $d$ and a large reduction in dimensionality may be attained.

- This is typically the case in many image and speech processing tasks where nearby inputs (in space or time) are highly correlated.

- If the dimensions are not correlated, $k$ will be as large as $d$ and there is no gain through PCA.

\[
\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}
\]
PCA: Implementation Issues

- **Scree graph** is the plot of variance explained as a function of the number of eigenvectors kept.
- By visually analyzing it, one can also **decide on** $k$.
- At the “elbow,” adding another eigenvector **does not significantly increase** the variance explained.
Another possibility is to ignore the eigenvectors whose eigenvalues are less than the average input variance.

Given that $\Sigma_i \lambda_i = \Sigma s_i^2$

The average eigenvalue is equal to the average input variance.

When we keep only the eigenvectors with eigenvalues greater than the average eigenvalue, we keep only those that have variance higher than the average input variance.
PCA: Applications
PCA: Applications

- We will discuss two applications of PCA.
  - Visual analysis
  - Eigenfaces
If the first two principal components explain a large percentage of the variance, we can do visual analysis.

We can plot the data in this two dimensional space and search visually for structure, groups, outliers, normality, and so forth.
PCA for Visual Analysis

- This plot gives a better **pictorial description** of the sample than a plot **of any two of the original variables**.
- By looking at the dimensions of the principal components, we can also try to **recover meaningful underlying variables** that describe the data.
Eigenfaces

• Dimensionality reduction usually becomes important when the number of features is not negligible compared to the number of training samples.

• As an example, suppose we would like to perform face recognition, i.e., determine the identity of the person depicted in an image, based on a training dataset of labeled face images.
One approach might be to treat the brightness of each pixel of the image as a feature.

If the input images are of size 32×32 pixels, this means that the feature vector contains 1024 feature values.

Classifying a new face image can then be done by calculating the Euclidean distance between this 1024-dimensional vector, and the feature vectors of the people in our training dataset.

The smallest distance then tells us which person we are looking at.
Eigenfaces

- However, operating in a 1024-dimensional space becomes problematic if we only have a few hundred training samples.
- We could use PCA to reduce the dimensionality of the feature space by calculating the eigenvectors of the covariance matrix of the set of 1024-dimensional feature vectors.
- Then projecting each feature vector onto the largest eigenvectors.
Eigenfaces

• These eigenvectors can also be displayed as images.

• These can be seen as templates for important features.

• They are typically named “eigenfaces”.
Eigenfaces
Eigenfaces

Top eigenvectors: $u_1, \ldots u_k$

Mean: $\mu$
Eigenfaces

• Since the eigenvector of 2D data is 2-dimensional, and an eigenvector of 3D data is 3-dimensional, the eigenvectors of 1024-dimensional data is 1024-dimensional.

• In other words, we could reshape each of the 1024-dimensional eigenvectors to a 32×32 image for visualization purposes.
Eigenfaces

• Each 1024-dimensional feature vector (and thus each face) can now be projected onto the \( N \) largest \textbf{eigenvectors}, and can be represented as a linear combination of these eigenfaces.

• The \textbf{weights of these linear combinations} determine the identity of the person.
Eigenfaces

• Since the largest eigenvectors represent the largest variance in the data, these eigenfaces describe the most informative image regions (eyes, noise, mouth, etc.).

• By only considering the first $N$ (e.g., $N = 70$) eigenvectors, the dimensionality of the feature space is greatly reduced.
The remaining question is now **how many eigenfaces should be used**, or in the general case; how many eigenvectors should be kept.

Removing too many eigenvectors might remove important information from the feature space, whereas eliminating too few eigenvectors leaves us with the curse of dimensionality.

Regrettably **there is no straight answer** to this problem.
Although cross-validation techniques can be used to obtain an estimate of this hyperparameter, choosing the optimal number of dimensions remains a problem that is mostly solved in an empirical ("trial-and-error") manner.

- Note that it is often useful to check how much (as a percentage) of the variance of the original data is kept while eliminating eigenvectors.
- This is done by dividing the sum of the kept eigenvalues by the sum of all eigenvalues.
PCA: Limitations
PCA: Limitations
Limitations of PCA

- We can give **three limitations** of PCA that are caused by **three assumptions**.
  - Assumption 1: PCA assumes data lies close to a linear subspace.
  - Assumption 2: PCA assumes that the data has a Gaussian distribution.
  - Assumption 3: PCA assumes that the most discriminative information is captured by the largest variance in the feature space.
Assumption 1: Data Lies Close to a Linear Sub-space

- PCA is a classical linear dimension reduction method that **assumes data lies close to a linear sub-space**.
- PCA works only when **data features are linearly dependent** in the lower-dimensional sub-space.
Assumption 1: Data Lies Close to a Linear Subspace

- It does not work when the lower-dimensional feature subspace is nonlinear.
Assumption 2: Data Has a Gaussian Distribution

• PCA assumes that the data has a Gaussian distribution (mean $\mu$, covariance matrix $\Sigma$).

• The shapes of the following datasets are not well described by its principal components.
Assumption 3: Most Discriminative Information is Captured by the Largest Variance

- PCA assumes that the **most discriminative information** is captured by the **largest variance in the feature space**.
- Since the direction of the largest variance encodes the most information this is likely to be true.

Most discriminative information: **largest variance axis** in the feature space.
Assumption 3: Most Discriminative Information is Captured by the Largest Variance

- However, there are cases when the discriminative information actually resides in the directions of the smallest variance.
- In that case PCA could greatly hurt classification performance.
Assumption 3: Most Discriminative Information is Captured by the Largest Variance

• As an example, consider the two cases, where we reduce the 2D feature space to a 1D representation:

Most discriminative information: \textit{largest variance axis} in the feature space.

Most discriminative information: \textit{smallest variance axis} in the feature space.
Assumption 3: Most Discriminative Information is Captured by the Largest Variance

- In the 2nd case, PCA would hurt classification performance because the data becomes **linearly non-separable**.
- This happens when the most discriminative information resides in the **smaller eigenvectors**.

**Most discriminative information**: smallest variance axis in the feature space.

**Most discriminative information**: largest variance axis in the feature space.
Assumption 3: Most Discriminative Information is Captured by the Largest Variance

- If the most discriminative information is contained in the smaller eigenvectors, applying PCA might actually worsen the Curse of Dimensionality.
- Because now a more complicated classification model (e.g. non-linear classifier) is needed to classify the lower dimensional problem.

Most discriminative information: **largest variance axis** in the feature space.

Most discriminative information: **smallest variance axis** in the feature space.
In this case, other dimensionality reduction methods might be of interest, such as **Linear Discriminant Analysis (LDA)**.

LDA tries to find the projection vector that **optimally separates the two classes**.

Assumption 3: Most Discriminative Information is Captured by the Largest Variance

- Most discriminative information: **smallest variance axis** in the feature space.
- Most discriminative information: **largest variance axis** in the feature space.