17. Matrix algebra (contd.), Singular Value Decomposition, and Principal Component Analysis

M.A.Z. Chowdhury and M.A. Oehlschlaeger

Department of Mechanical, Aerospace and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York

chowdm@rpi.edu and oehlsm@rpi.edu

MANE 4962 and 6962

Announcements and Agenda for Today

- ☞ HW 5 and quiz 5 is due Today
- ☞ No office hours on Wednesday, March 22, (GM Week)
- ☞ Office hours on Friday, March 24, 3-5 PM
- ☞ SVM
- ☞ k-means
- ☞ PCA and SVD
- ☞ Dimensionality reduction
- ☞ Creating improved features (if new features are better, your model will improve)

Applications of SVD

Origin: Linear Algebra

- **Image compression**
- **Collaborative filtering**
- **Data/text mining and natural language processing**
- **Denoising**
- **Many other applications in numerical linear algebra**

Applications of PCA

Origin: Linear Algebra

- **Data visualization**
- **Noise reduction**
- **Feature extraction**
- **Image compression**
- **Finance**
- **Genetics**
- **Neuroscience**
- **Chemometrics**
- **Face recognition**
- **Recommender systems**
- **Text mining**
- **Speech recognition**

Concept of Decomposition

- Matrices transform one vector to another. Tensors transform matrices and vectors.
- In ML, high dimensional vectors, matrices, and tensors are very common, for example, images were represented as matrices.
- Similar to prime factorization in number theory. Decomposition tells us what makes a matrix.
- Decomposition provides a smaller set of representative numbers Example : Eigenvalues, Singular Values

Matrix Determinant and Volume

The determinant of a square matrix $A \in \mathbb{R}^{n \times n}$ is a function $\det : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$, and is denoted $|A|$ or $\det(A)$.

- Given a matrix

$$A = \begin{bmatrix} - a_1^T - \\ - a_2^T - \\ \vdots \\ - a_n^T - \end{bmatrix},$$

- Laplace expansion : For a $n \times n$ matrix, A we can write

$$det(A) = \sum_{j=1}^{n} (-1)^{n+j} A_{ij} det(A_{ij})$$

  A_{ij} are the submatrices.

- Physically represents volume formed by column vectors
- A square matrix, A have an inverse A^{-1} if and only if $det(A) \neq 0$
- If $det(A) \neq 0$ then columns of A has to be linearly independent.
- If $det(A) = 0$, it is a singular matrix.

The Determinant: Properties

Algebraically, the determinant satisfies the following three properties:

1. The determinant of the identity is 1, $|I| = 1$. (Geometrically, the volume of a unit hypercube is 1).
2. Given a matrix $A \in \mathbb{R}^{n \times n}$, if we multiply a single row in A by a scalar $t \in \mathbb{R}$, then the determinant of the new matrix is $t|A|$. (Geometrically, multiplying one of the sides of the set S by a factor t causes the volume to increase by a factor t.)
3. If we exchange any two rows a_i^T and a_j^T of A, then the determinant of the new matrix is $-|A|$, for example
4. For $A \in \mathbb{R}^{n \times n}$, $|A| = |A^T|$.
5. For $A, B \in \mathbb{R}^{n \times n}$, $|AB| = |A| \cdot |B|$.
6. For $A \in \mathbb{R}^{n \times n}$, $|A| = 0$ if and only if A is singular (i.e., non-invertible). (If A is singular then it does not have full rank, and hence its columns are linearly dependent. In this case, the set S corresponds to a "flat sheet" within the n-dimensional space and hence has zero volume.)
7. For $A \in \mathbb{R}^{n \times n}$ and A non-singular, $|A^{-1}| = 1/|A|$.

Determinants of Small Matrices

Equations for determinants of small matrices up to size 3×3 are fairly common, and it is good to know them:

$$\left|\begin{bmatrix} a_{11} \end{bmatrix}\right| : \quad |A| = a_{11}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} : \quad |A| = a_{11}a_{22} - a_{12}a_{21}$$

$$\left|\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right| : \quad |A| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31}$$

$$A = \left[\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right]$$

$$\det A = \det \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21} \tag{1}$$

Eigenvalues and Eigenvectors

- Extremely useful for square symmetric matrices. Used for other matrices as well
- Every real matrix can be thought of as a combination of rotation and stretching
- Eigenvectors for a matrix are those special vectors that only stretch under the action of the matrix
- Eigen values are the factor by which eigenvectors stretch.
- Given a square matrix $A \in \mathbb{R}^{n\times n}$, we say that $\lambda \in \mathrm{C}$ and $\underline{v} \in \mathrm{C}^n$ is an eigenvalue of A and $\underline{v} \in \mathrm{C}^n$ is the corresponding eigenvector if

$$A\underline{v} = \lambda\underline{v}, \quad \underline{v} \neq 0$$

- This definition means that multiplying A by the vector $\underline{v}$ results in a new vector that points in the same direction as $\underline{v}$, but scaled by a factor .

Eigenvalues and Eigenvectors

- We can rewrite the equation above to state that (λ, x) is an eigenvalue-eigenvector pair of A if,

$$(\lambda I - A)x = 0 \text{ and } x \neq 0$$

.

- But $(\lambda I - A)x = 0$ has a non-zero solution to x if and only if $(\lambda I - A)$ has a non-empty nullspace, which is only the case if $(\lambda I - A)$ is singular, i.e.,

$$|(\lambda I - A)| = 0$$

We can now use the previous definition of the determinant to expand this expression $|(\lambda I - A)|$ into a (very large) polynomial in λ, where λ will have degree n. It's often called the characteristic polynomial of the matrix A.

Eigenvalues and Eigenvectors

- Consider, A has N linearly independent eigenvectors $\{\underline{v_1}, \underline{v_2}, \ldots, \underline{v_N}\}$
- Concatenate all the vectors (as columns) and make a eigenvector matrix

$$V = \begin{bmatrix} | & | & | & | \\ \underline{v_1} & \underline{v_2} & \ldots & \underline{v_N} \\ | & | & | & | \end{bmatrix}$$

- If we concatenate the corresponding eigenvalues into a diagonal matrix,

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

- Then, the eigen decomposition or eigen factorization of A is

$$A = V \Lambda V^{-1}$$

- Note: Eigen decomposition may not be unique.

Properties of eigenvalues and eigenvectors

☞ The trace of a matrix A is equal to the sum of its eigenvalues, i.e.,

$$\mathrm{tr}(A) = \sum_{i=1}^{n} \lambda_i .$$

☞ The determinant of A is equal to the product of its eigenvalues, i.e.,

$$\det(A) = \prod_{i=1}^{n} \lambda_i .$$

☞ The rank of A is equal to the number of non-zero eigenvalues of A.

☞ Suppose A is non-singular with eigenvalue λ and an associated eigenvector x. Then $1/\lambda$ is an eigenvalue of A^{-1} with an associated eigenvector x, i.e.,

$$A^{-1}\underline{v} = \frac{1}{\lambda}\underline{v}.$$

☞ The eigenvalues of a diagonal matrix $D = \mathrm{diag}(d_1, \ldots, d_n)$ are exactly the diagonal entries $d_1, \ldots, d_n$.

Physical Significance of Eigen Decomposition

- ☞ All matrices can be thought of as rotating and stretching vectors.
- ☞ Eigenvectors have pure stretch.
- ☞ Real symmetric matrices have real eigenvectors and values.

$$A = Q\Lambda Q^{-1} = Q\Lambda Q^{T}$$

- ☞ $Q = Q^T$ is orthogonal and a rotation matrix.
- ☞ $A\underline{v} = Q\Lambda Q^T\underline{v}$, The term, $\Lambda Q^T\underline{v}$ is important.
- ☞ Identity matrix is a special matrix where every column is an eignevector.

$$I = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

Eigenvalues and Eigenvectors of Symmetric Matrices

Throughout this section, let's assume that A is a symmetric real matrix (i.e., $A^\top = A$). We have the following properties:

1. All eigenvalues of A are real numbers. We denote them by $\lambda_1, \ldots, \lambda_n$.
2. There exists a set of eigenvectors $u_1, \ldots, u_n$ such that (i) for all i, u_i is an eigenvector with eigenvalue λ_i and (ii) $u_1, \ldots, u_n$ are unit vectors and orthogonal to each other.

Representing Symmetric Matrices

Let U be a orthonormal matrix that contains u_i's as columns:

$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{bmatrix}$$

Let $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ be the diagonal matrix that contains $\lambda_1, \dots, \lambda_n$.

$$AU = \begin{bmatrix} | & | & & | \\ Au_1 & Au_2 & \dots & Au_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \lambda_1 u_1 & \lambda_2 u_2 & \dots & \lambda_n u_n \\ | & | & & | \end{bmatrix} = U \operatorname{diag}(\lambda_1, \dots, \lambda_n) = U\Lambda$$

Orthonormal matrix U satisfies that $UU^T = I$, we can diagonalize matrix A:

$$A = AUU^T = U\Lambda U^T$$

Representing a Vector in Another Basis

- Any orthonormal matrix

$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{bmatrix}$$

defines a new basis of $\mathbb{R}^n$.

- For any vector $x \in \mathbb{R}^n$, it can be represented as a linear combination of $u_1, \dots, u_n$ with coefficients $x_1, \dots, x_n$:

$$x = x_1 u_1 + \cdots + x_n u_n = U\hat{x}$$

- Indeed, such $\hat{x}$ uniquely exists.

$$x = U\hat{x} \iff U^T x = \hat{x}$$

In other words, the vector $\hat{x} = U^T x$ can serve as another representation of the vector x with respect to the basis defined by U.

Singular Value Decomposition

- Generalization of factorization or decomposition to nonsquare matrices.
- Factorizing matrices based on stretch and rotation
- If A is a $m \times n$ matrix, then

$$A = UDV^T$$

$$\underset{m\times m}{\mathrm{A}} = \underset{m\times m}{U} \quad \underset{m\times n}{D} \quad \underset{n\times n}{V^T}$$

- U and V are orthogonal and D is diagonal.
- Elements of U are eigenvectors of AA^T , called left-singular vectors.
- Elements of V are eigenvectors of A^TA , called right-singular vectors.
- Non-zero Elements of D comes from $\sqrt{A^TA}$ are called singular values. Sometimes the symbol S is used instead of D.
- Common PCA algorithms uses SVD to obtain the principal components since the input matrix does not have to be square.

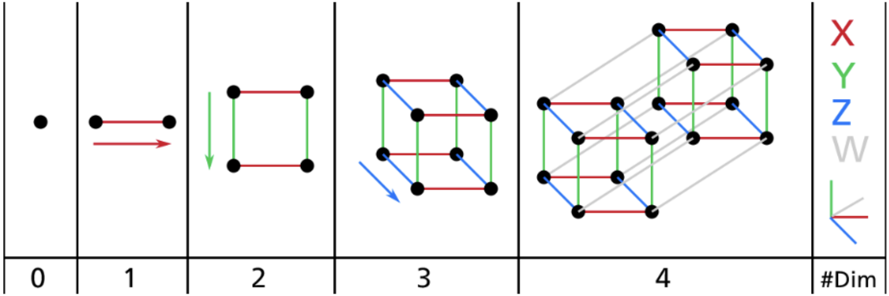
Covariance Matrix

- If $\mathbf{x} \in \mathbb{R}^n$ is a vector, it is often useful to know pairwise covariances between all pairs of components.
- Think of the x being the components of an input vector such as pixels of an image.
- Define: $\mathrm{Cov}[\mathbf{x}, \mathbf{x}]_{(i,j)} = \mathrm{Cov}[\mathbf{x}_i, \mathbf{x}_j]$.

$$\mathrm{Cov}[\mathbf{x}, \mathbf{x}] = \begin{bmatrix} \mathrm{Cov}[\mathbf{x}_1, \mathbf{x}_1] & \mathrm{Cov}[\mathbf{x}_1, \mathbf{x}_2] & \cdots & \mathrm{Cov}[\mathbf{x}_1, \mathbf{x}_n] \\ \mathrm{Cov}[\mathbf{x}_2, \mathbf{x}_1] & \mathrm{Cov}[\mathbf{x}_2, \mathbf{x}_2] & \cdots & \mathrm{Cov}[\mathbf{x}_2, \mathbf{x}_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[\mathbf{x}_n, \mathbf{x}_1] & \mathrm{Cov}[\mathbf{x}_n, \mathbf{x}_2] & \cdots & \mathrm{Cov}[\mathbf{x}_n, \mathbf{x}_n] \end{bmatrix} \in \mathbb{R}^{n \times n}$$

- Note that the diagonal elements are simply the variances of individual components, that is, $\mathrm{Cov}[\mathbf{x}_i, \mathbf{x}_i] = \mathrm{Var}[\mathbf{x}_i]$.

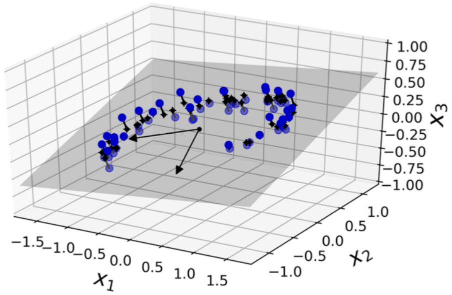
The curse of dimensionality

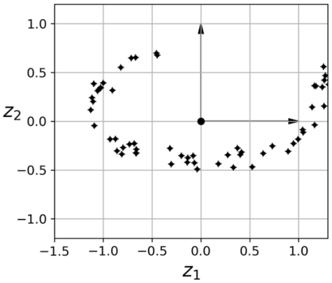


Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow

Intentionally left blank

Notion of Dimensionality Reduction





A 3D dataset lying close to a 2D subspace. By taking projections we have reduced the dimension.

Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow

Intentionally left blank

PCA basics

- Project (linearly) d-dimensional data into k-dimensional space while preserving as much information as possible:
- Example: project space of 784 pixels into 3-dimensions
- Example: project 4-d data into 2-d
- Choose projection with minimum reconstruction error
- `http://setosa.io/ev/principal-component-analysis/`

Reconstruction Error in PCA

- Reconstruction error is the difference between original data and data reconstructed from its lower-dimensional representation after applying PCA
- Goal: find the best lower-dimensional representation preserving as much information as possible by minimizing reconstruction error
- Metrics for measuring reconstruction error:
  - Mean squared error (MSE)
  - Mean absolute error (MAE)
  - Others

  Mean squared error (MSE) is commonly used for reconstruction.

Basic PCA Algorithm

- Compute the covariance matrix:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}_{std}^T\mathbf{X}_{std}$$

  where n is the number of samples, and $\mathbf{C}$ is the covariance matrix.
- Calculate the eigenvalues λ and eigenvectors $\mathbf{V}$ of the covariance matrix $\mathbf{C}$:

$$\mathbf{C}\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

  where λ_i is the i-th eigenvalue and $\mathbf{v}_i$ is the corresponding eigenvector.
- Sort the eigenvalues in descending order and their corresponding eigenvectors:

$$\lambda_{sorted}, \mathbf{V}_{sorted}$$

- Select the top k eigenvectors to form the projection matrix $\mathbf{W}$:

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$$

- Transform the original data matrix $\mathbf{X}$ into the lower-dimensional representation $\mathbf{X}_{reduced}$:

$$\mathbf{X}_{reduced} = \mathbf{X}_{std}\mathbf{W}$$

Basic PCA algorithm with Eigen Decomposition

```python
def my_pca(X, n_components):
    X_std = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
    covariance_matrix = np.cov(X_std.T)
    eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)
    sorted_indices = np.argsort(eigenvalues)[::-1]
    sorted_eigenvectors = eigenvectors[:, sorted_indices]
    top_eigenvectors = sorted_eigenvectors[:, :n_components]
    X_reduced = np.dot(X_std, top_eigenvectors)
    explained_variance = np.sum(eigenvalues[:n_components]) / np.sum(eigenvalues)
    return X_reduced, top_eigenvectors, explained_variance
```

Basic PCA algorithm with S.V. Decomposition

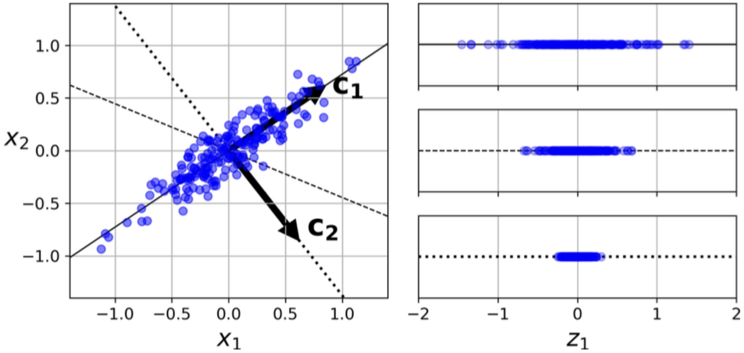
```python
def my_pca(X, n_components):
    X_std = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
    U, S, Vt = np.linalg.svd(X_std, full_matrices=False)
    U_reduced = U[:, :n_components]
    X_reduced = U_reduced * S[:n_components]
    explained_variance = np.sum(S[:n_components]**2) / np.sum(S**2)
    return X_reduced, Vt[:n_components, :], explained_variance
```

Scikit's implementation of SVD based PCA

```python
import numpy as np
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
data = load_iris()
X = data.data
y = data.target
n_components = 2
pca = PCA(n_components=n_components)
X_reduced = pca.fit_transform(X)
print("Reduced dataset (first 5 rows):\n", X_reduced[:5])
explained_variance = np.sum(pca.explained_variance_ratio_)
print("Explained variance:", explained_variance)
print("Top eigenvectors :", pca.components_)
X_reconstructed = pca.inverse_transform(X_reduced)
reconstruction_error = np.mean((X - X_reconstructed) ** 2)
print("Reconstruction error :", reconstruction_error)
```

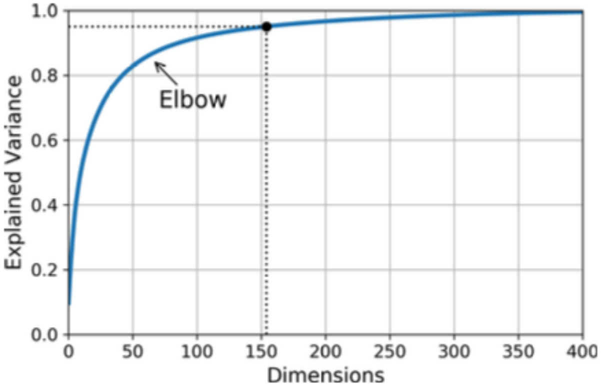
Note: Only two principal components are calculated for these examples.

How PCA preserves variance?



PCA identifies the hyperplane that lies closest to the data, and then it projects the data onto it First, choose the right low-dimensional hyperplane. Projections onto three different 1D hyperlanes are shown. The projection onto the solid line preserves the maximum variance, while the projection onto the dotted line preserves very little variance and the projection onto the dashed line preserves an intermediate amount of variance.

How to choose number of components?



- **Visualization: For data visualization you are limited to picking 2 or 3 components.**
- **Dimensionality reduction: Pick the number of components/dimensions that preserves a significant explained variance.**

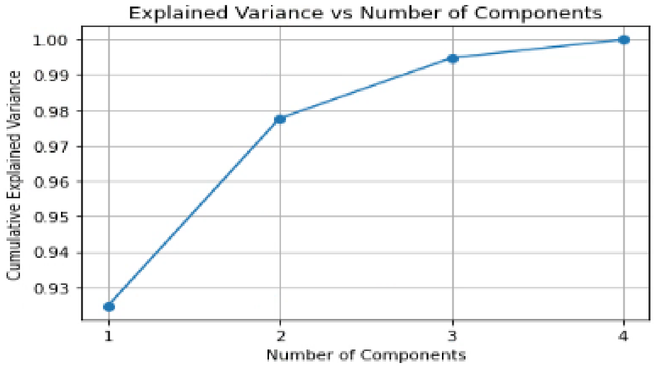
Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow

Plotting explained variance for Iris

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
data = load_iris()
X = data.data
y = data.target
n_components = 4
pca = PCA(n_components=n_components)
X_reduced = pca.fit_transform(X)
explained_variance_ratios = pca.explained_variance_ratio_
cumulative_explained_variance = np.cumsum(explained_variance_ratios)
plt.figure()
plt.plot(range(1, n_components+1), cumulative_explained_variance, marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.xticks(range(1, n_components+1))
plt.title('Explained Variance vs Number of Components')
plt.grid()
plt.show()
```

Iris has only four features so you can't calculate more than four PCs.

Explained variance ratios for Iris



- **Iris data is such that the first component alone captures 93% information in the dataset.**

PCA variants

- **Incremental PCA (IPCA):** An adaptation of PCA that is designed to handle large datasets that cannot fit into memory. It processes the dataset in smaller chunks (or mini-batches) instead of processing the entire dataset at once.
- **Kernel PCA (KPCA):** A non-linear extension of PCA that is particularly useful for datasets that are not linearly separable.
- In standard PCA, the principal components are linear combinations of the original features. Datasets with complex, non-linear relationships won't be captured effectively via regular PCA.
- Kernel PCA applies a kernel function (e.g., Radial Basis Function, Polynomial etc.) to the data points, implicitly mapping them to a higher-dimensional space where the relationships are linear.
- Then, it applies PCA in this transformed space to extract principal components. The main advantage of KPCA is its ability to handle non-linear data and find meaningful patterns that standard PCA might miss.
- You may need to tune hyperparameters to make these methods work better.

Intentionally left blank

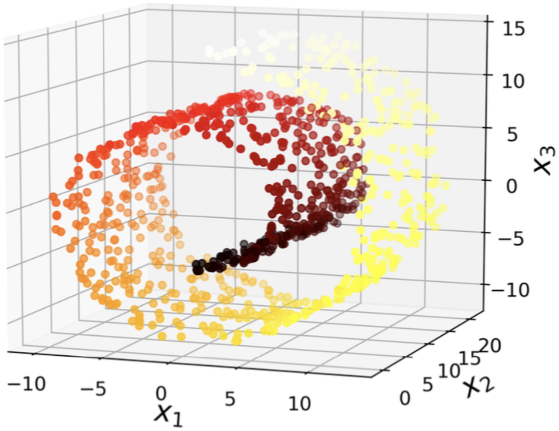
Advantages and Disadvantages of PCA

- **Advantages:**
  - Reduces the dimensionality of the data while preserving the maximum variance using linear transformation. Thus it improves the performance of some machine learning algorithms
  - Easily interpretable since it finds orthogonal axes (principal components) that explain the largest variances in the data.
  - Can be useful in visualizing high-dimensional data. Use pairplot to visualize the PCs or plot 2 or 3 PCs at a time to examine the pattern present in the data.
- **Disadvantages:**
  - Assumes linear relationships between features so may not perform well when the relationships are non-linear.
  - PCA is sensitive to the scaling of the data. Standardizing the data before applying PCA is essential.
  - Unsupervised learning, so we cannot guide the learning process.
  - Will perform poorly on manifolds.

Advantages and Disadvantages of SVD

- **Advantages:**
  - Can be applied directly to any rectangular data matrix, not just square covariance or correlation matrices, making it more versatile.
  - Provides a more numerically stable, faster and efficient solution than PCA for decomposing large and sparse data matrices.
- **Disadvantages:**
  - Like PCA, SVD assumes linear relationships between features and may not perform well when the relationships are non-linear.
  - SVD is computationally expensive for large datasets, although various approximations and optimizations can mitigate this.
  - SVD may not be as easily interpretable as PCA because it decomposes the data matrix directly rather than finding orthogonal axes that explain the largest variances in the data.
  - SVD does not consider the class labels in supervised learning tasks, similar to PCA, thus making it an unsupervised method.

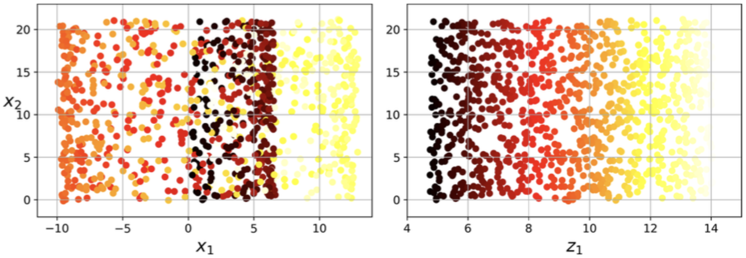
Understanding the limitation of PCA



Simply projecting onto a plane (e.g., by dropping x3) would squash different layers of the Swiss roll together. PCA won't be able to learn manifolds (twists, rolls, etc.) such as these.

Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow

Understanding the limitation of PCA



Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow