Dimensionality Reduction and Principal Component Analysis

- To visualize our data, e.g., if the dimensions can be reduced to 2 or 3.
- To save computer memory/disk space if the data are large.
- To reduce execution time.
- To visualize our data, e.g., if the dimensions can be reduced to 2 or 3.

Dimensionality Reduction results in an approximation of the original data

Before running any ML algorithm on our data, we may want to reduce the number of features:

- To save computer memory/disk space if the data are large.
- To reduce execution time.
- To visualize our data, e.g., if the dimensions can be reduced to 2 or 3.

Dimensionality Reduction results in an approximation of the original data

Projecting Data - 2D to 1D
Why such projections are effective

- In many datasets, some of the features are correlated.
- Correlated features can often be well approximated by a smaller number of new features.
- For example, consider the problem of predicting housing prices. Some of the features may be the square footage of the house, number of bedrooms, number of bathrooms, and lot size. These features are likely correlated.
Suppose we want to reduce data from \( d \) dimensions to \( k \) dimensions, where \( d > k \).

Principal Component Analysis (PCA) finds \( k \) vectors onto which to project the data so that the projection errors are minimized.

In other words, PCA finds the \textit{principal components}, which offer the best approximation.

**PCA ≠ Linear Regression**

Given \( n \) data points, each with \( d \) features, i.e., an \( n \times d \) matrix \( X \):

- Preprocessing: perform feature scaling
- Compute covariance matrix \( \Sigma = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T = \frac{1}{n}X^TX \)
- Calculate eigenvalues and corresponding eigenvectors of covariance matrix \( \Sigma \) via singular value decomposition
- This yields a new basis of \( d \) vectors as well as the variance along each of the \( d \) axes
- Retain the \( k \) vectors with the largest corresponding variance and project the data onto this \( k \)-dimensional subspace
PCA Example

```
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
Z = pca.fit_transform(X)
print(Z.shape)
print(pca.explained_variance_ratio_)
```

`X` is 30×2 matrix

(30, 1)
[ 0.5815958 ]

58% of variance explained by first principal component

Shape of compressed data Z

```
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
Z = pca.fit_transform(X)
print(Z.shape)
print(pca.explained_variance_ratio_)
```

`X` is 30×2 matrix

(30, 1)
[ 0.9945036 ]

99% of variance explained by first principal component

Shape of compressed data Z
**PCA with sklearn**

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
Z = pca.fit_transform(X)
print(Z.shape)
print(pca.explained_variance_ratio_)
```

X is 30×2 matrix

100% of variance explained by first principal component

Shape of compressed data Z: \([30, 1]\)

**Iris Data**

For 50 instances of each type of iris, we have four features:

- sepal length
- sepal width
- petal length
- petal width

**Feature Scaling**

```python
from sklearn import preprocessing
X_scaled = preprocessing.scale(X)
```

X_scaled is 150×4 matrix

**PCA: 4D to 2D**

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
Z = pca.fit_transform(X_scaled)
print(Z.shape)
print(pca.explained_variance_ratio_)
```

Z is 150×2 matrix

95% of variance explained by first two principal components

Shape of compressed data Z: \([150, 2]\)
from sklearn.decomposition import PCA
pca = PCA(n_components=200)
eigenfaces = pca.components_
print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.sum())

95% of variance explained by first 200 principal components
57% of variance explained by first 10 principal components

How to choose $k$

- We want to lose as little information as possible, i.e., we want the proportion of variance that is retained to be as large as possible.
- Typically, $k$ is chosen so that 95% or 99% of the variance in the data is retained.
- If there are many correlated features in the data, often a high percentage of the variance can be retained while using a small number of features, i.e., $k$ much less than $d$. 
PCA Summary

- Prior to running a ML algorithm, PCA can be used to reduce the number of dimensions in the data. This is helpful, e.g., to speed up execution of the ML algorithm.
- Since datasets often have many correlated features, PCA is effective in reducing the number of features while retaining most of the variance in the data.
- Before performing PCA, feature scaling is critical.
- Principal components aren’t easily interpreted features. We’re not using a subset of the original features.

Overview