Motivation

Handle High Multicollinearity

Existing Solutions:

1. Variable Selection (stepwise/forward/backward)

Cons: Each time dropping a variable, some information is lost

Visualization of more features

Existing Solutions:

1. Pairwise scatter plots pC2 = (p*(p-1)/2), where p is number of variables

Cons: if p=20, this would mean 190 plots!

There must be a better way of doing this. Goal is to find an algorithm to reduce the number of variables without losing information. i.e. PCA

Usecases

1. **Dimensionality Reduction** without losing information.
2. Easy **Data Visualization and Exploratory Data Analysis**
3. Create **uncorrelated features/variables** that can be an input to a prediction model
4. Uncovering latent variables/themes/concepts
5. **Noise reduction** in dataset

Prerequisite Knowledge

Building Blocks:

1. *The basis of a space:*

Set of linearly independent vectors/directions that span the entire space i.e. Any point in space can be represented as a combination of these vectors.

Ex: Each row of a dataset is a point in the space. Each column is a basis vector (representation of any point in terms of columns).

2. *Basis transformation:*

The process of converting your information from one set of basis to another. (OR) Representing your data in new columns different from original. Often for convenience, efficiency or just for common sense.

Ex: Dropping or Adding a column to the dataset.

Prerequisite Knowledge (cont)

3. *Variance as information:*

Variance = Information

If two variables are highly correlated, they together don't add a lot of information than they do individually. So one of them can be dropped.

In 2D geometry, X and Y axes are dimensions. i (1,0) is a unit vector in X direction, j (0,1) is a unit vector in the Y direction. For point a: ax, ay are the units to move in 'i' and 'j' directions to reach 'a' and also denoted as: ax i + ay j. Any point in 2D space can be represented in term of 'i' and 'j'. The 'i' and 'j' vectors are the 'basis of the space'. 'i' and 'j' are independent i.e. 'i' can't be expressed in terms of 'j' and vice versa

What does it do?

PCA is one of a family of techniques for taking high-dimensional data, and using the dependencies between the variables to represent it in a more tractable, lower-dimensional basis, without losing too much information.

Given p features/variables in a dataset, PCA finds the principal components as

1. a **linear combination** of the original features.
2. the principal components capture **maximum variance** in the dataset.

Mathematical Representation

$$Z_1 = \varphi_{11}X1 + \varphi_{21}X2 + \ldots + \varphi_{p1}Xp$$

The above equation represents the **First Principal Component**. PCA finds φ values such that the variance on Z_1 is maximum. The **Second Principal Component** is found as one that has maximal variance of all linear combinations that are uncorrelated with Z_1. And like this, each additional component is capturing incremental variance. The algorithm calculates 'k' principal components, where k<=p (p is number of variables in dataset).

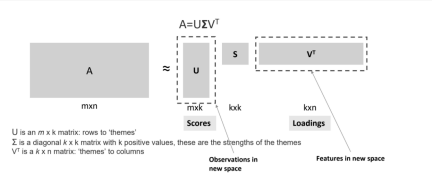
Workings of PCA

1. Find **Principal Components**

a) Using SVD (Singular Value Decomposition)

2. Choose **optimal number** of principal components (k).

Singular Value Decomposition (SVD)



'Decomposition' because it breaks the original data matrix into 3 new matrices

Code

```
from sklearn.decomposition
import PCA
import numpy as np
X = np.arr ay( [[-1, -1], [-2, -
1], [-3, -2], [1, 1], [2, 1],
[3, 2]])
pca = PCA(n_ com pon ents=2)
pca.fit(X)
```