Principal Component Analysis

Melina Kopischkie

MSDS Statistical Methods 5000

December 6, 2024

Abstract

Reducing the dimensionality of datasets is imperative to utilizing data science and machine learning techniques. It reduces the time and computational complexity while maintaining a certain percentage of the variance. In this paper, I will detail the definition of Principal Component Analysis (PCA), the mathematics behind it, the Python implementation, and a discussion of the results.

1 Introduction

Principal Component Analysis (PCA) is a dimensionality reduction technique that can be utilized by data scientists' and others in a multitude of fields including machine learning, finance, and healthcare. Essentially, the data is transformed into a new coordinate system where the features that contribute most to the variance in the dataset can be easily identified. [1] In this paper, I apply PCA to an algorithmic trading dataset so I can reduce the dimensions and prep the dataset for additional machine learning or data science projects. Additionally, I explain the math behind this application in detail and close with discussions about how PCA can be used in the real world.

2 Methods

The main four steps of Principal Component Analysis can be summarized in the following way:

1. Calculate the covariance matrix of the selected dataset
2. Calculate the eigenvalues of the covariance matrix
3. Calculate the corresponding eigenvectors
4. The eigenvector with the largest eigenvalue represents the direction of the most variance, indicating that it is the first principal component. The eigenvector with the second highest eigenvalue is the second principal component, and so on.

I will complete PCA by hand using the dataset below.

x_1	x_2
3	9
7	6
2	7
5	5
8	3

Covariance Matrix

The covariance is "a measure of how changes in one variable are associated with changes in a second variable. Specifically, it's a measure of the degree to which two variables are linearly associated." [2] So, a covariance matrix shows the covariance between multiple variables, allowing us to understand how the variables relate to each other. Covariance is most beneficial when the data is normalized. PCA is sensitive to the scale of different attributes and it could disproportionately represent the variance of a certain attribute. To normalize, we subtract the given value by the mean value of that particular attribute, as seen below:

x_1	x_2
3	9
7	6
2	7
5	5
8	3
$\mu = 5$	$\mu = 6$

Table 1: Original Data

x_1	x_2
3 - 5	9 - 6
7 - 5	6 - 6
2 - 5	7 - 6
5 - 5	5 - 6
8 - 5	3 - 6

Table 2: Data Minus Mean

x_1	x_2
-2	3
2	0
-3	1
0	-1
3	-3

Table 3: Normalized Data

To construct a covariance matrix, we need to turn our normalized data from Table 4 into a matrix and find its transpose.

$$\text{Normalized Data} = \begin{bmatrix} -2 & 3 \\ 2 & 0 \\ -3 & 1 \\ 0 & -1 \\ 3 & -3 \end{bmatrix} \qquad \text{Transposed Data} = \begin{bmatrix} -2 & 2 & -3 & 0 & 3 \\ 3 & 0 & 1 & -1 & -3 \end{bmatrix}$$

To obtain the covariance matrix, we can use the following formula,

$$C = \frac{X^T * X}{m - 1} \tag{1}$$

where X^T is the transposed matrix, X is the matrix containing the normalized data, and m is the number of samples, or rows, in the dataset. Using our data, we can compute the covariance matrix as follows:

$$\mathrm{C} = \left(\begin{bmatrix} -2 & 2 & -3 & 0 & 3 \\ 3 & 0 & 1 & -1 & -3 \end{bmatrix} \begin{bmatrix} -2 & 3 \\ 2 & 0 \\ -3 & 1 \\ 0 & -1 \\ 3 & -3 \end{bmatrix} \right) * \frac{1}{4}$$

Performing matrix multiplication and dividing by a constant, we get the covariance matrix

$$\mathrm{C} = \begin{bmatrix} 6.5 & -4.5 \\ -4.5 & 5 \end{bmatrix} \tag{2}$$

The diagonals of the matrix represent the variance within that particular attribute. The other two values, in this case -4.5, represent the covariance between different attributes.

Eigenvalues

An eigenvalue quantifies how much a linear transformation is stretched or shrunk. We calculate them by using the following equation:

$$|C - \lambda I| = 0 \tag{3}$$

where I is the identity matrix, or $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and λ is the eigenvalue we are solving for. Performing this computation, we find

$$\begin{bmatrix} 6.5 & -4.5 \\ -4.5 & 5 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 6.5 - \lambda & -4.5 \\ -4.5 & 5 - \lambda \end{bmatrix}$$

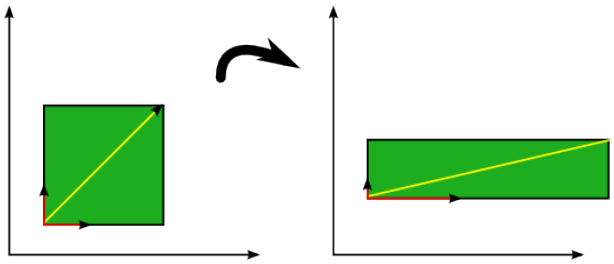
To solve for λ, we can take the determinant to find the resulting characteristic equation:

$$\lambda^2 - 11.5\lambda + 12.25 \tag{4}$$

We can use the quadratic formula to solve for the roots which gives the two eigenvalues $\lambda_1 = 10.3$ and $\lambda_2 = 1.19$

Eigenvectors

An eigenvector represents a direction of the linear transformation. It can be thought of as a skewer that holds the linear transformation together. The eigenvector is a constant and the eigenvalue adjusts the magnitude of the eigenvector to align with the transformed data points. The image below helps illustrate eigenvalues and eigenvectors in a linear transformation.



To obtain the eigenvectors from the eigenvalues, we can use the following equation:

$$C\vec{X} = \lambda\vec{X} \tag{5}$$

Also written as:

$$(C - \lambda I)\vec{X} = 0 \tag{6}$$

I will be using the later equation to continue the provided example. We perform the sample computation for both eigenvalues. For $\lambda_1 = 10.3$:

$$\left(\begin{bmatrix} 6.5 & -4.5 \\ -4.5 & 5 \end{bmatrix} - 10.3\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \cdot \vec{X} = \begin{bmatrix} -3.8 & -4.5 \\ -4.5 & -5.3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Using matrix multiplication, we obtain a system of equations that we can solve.

$$\begin{cases} -3.8x - 4.5y = 0 \\ -4.5x - 5.3y = 0 \end{cases} \tag{7}$$

We find that $x = -1.18y$, so we obtain the eigenvector:

$$\vec{v_1} = \begin{bmatrix} -1.18 \\ 1 \end{bmatrix}$$

For $\lambda_2 = 1.19$:

$$\left(\begin{bmatrix} 6.5 & -4.5 \\ -4.5 & 5 \end{bmatrix} - 10.3\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \cdot \vec{X} = \begin{bmatrix} 5.31 & -4.5 \\ -4.5 & 3.81 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Rewriting as a system of equations,

$$\begin{cases} 5.31x - 4.5y = 0 \\ -4.5x + 3.81y = 0 \end{cases} \tag{8}$$

We find that $x = 0.85y$, so we obtain the eigenvector:

$$\vec{v_2} = \begin{bmatrix} 0.85 \\ 1 \end{bmatrix}$$

When using Principal Component Analysis, it's important to normalize all the data and vectors, so we normalize both eigenvectors by dividing each component by the length of the vector.

The length of $\vec{v_1}$ is calculated as:

$$L_1 = \sqrt{(-1.18)^2 + (1)^2} = 1.55$$

And the length of $\vec{v_2}$ as:

$$L_2 = \sqrt{(0.85)^2 + (1)^2} = 1.31$$

So, the normalized eigenvectors are:

$$\vec{v_1} = \begin{bmatrix} \frac{-1.18}{1.55} \\ \frac{1}{1.55} \end{bmatrix} = \begin{bmatrix} -0.761 \\ 0.645 \end{bmatrix} \quad \text{and} \quad \vec{v_2} = \begin{bmatrix} \frac{0.85}{1.31} \\ \frac{1}{1.31} \end{bmatrix} = \begin{bmatrix} 0.649 \\ 0.763 \end{bmatrix}$$

Labeling Principal Components

The eigenvalues specifically indicate the variance captured by each principal component. Therefore, our first principal component will have the greatest variance and the largest eigenvalue. The second principal component will have the second largest variance and the second largest eigenvalue, and so on. In this example, we started with two attributes and obtained two principal components that we can rank by their eigenvalues. Therefore, the first principal component has an eigenvalue of $\lambda_1 = 10.3$ and the second component has an eigenvalue of $\lambda_2 = 1.19$. The corresponding eigenvectors indicate the direction the old data is mapped in the linear transformation.

3 Results

The initial dataset contained 10 attributes and 50 samples. The data was preprocessed and cleaned. Special characters such as percentage and dollar signs were dropped and the corresponding attributes were scaled appropriately. The majority of the data were object types, so I converted those features to numeric types. Finally, I was ready to set up PCA on my dataset. The following snippet of code is where I split up the data between the target variable and the rest of the features. Then, as I mentioned earlier, I scale the data to normalize each point.

```
X = df.drop(['Price'], axis = 1)
y = df['Price']

x = StandardScaler().fit_transform(X)
```

Then, the computation of PCA could be conducted using the code below.

```
pca = PCA()

principalComponents = pca.fit_transform(x)

principalDf = pd.DataFrame(data = principalComponents)

finalDf = pd.concat([principalDf, y], axis = 1)
```

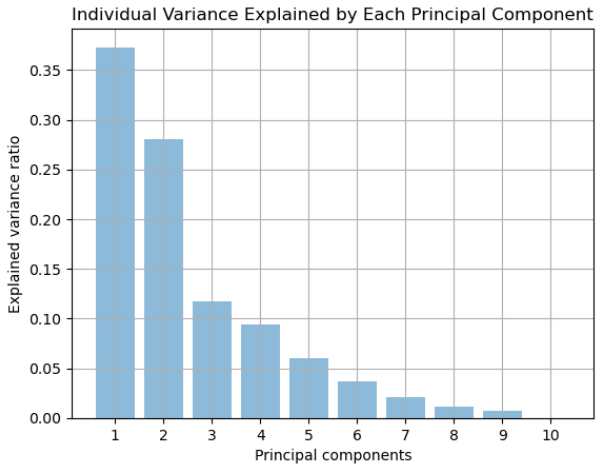
I used the following code to understand how each principal component contributes to the variance and constructed my first plot.

```
explained_variance = pca.explained_variance_ratio_
```

```
print(explained_variance)
```

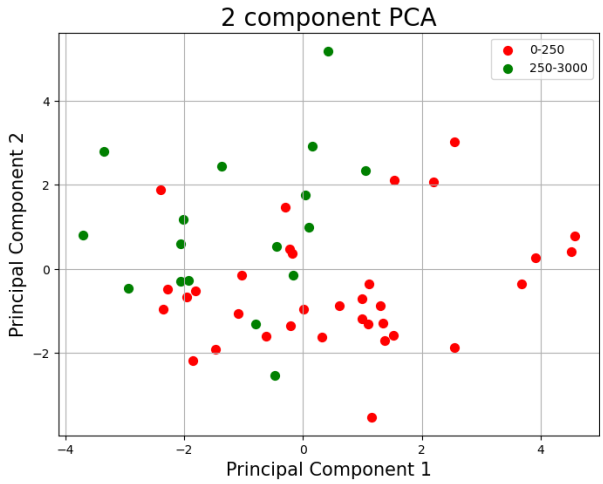
```
[3.72935782e-01 2.80349912e-01 1.17576150e-01 9.35483671e-02
 5.99984246e-02 3.66815364e-02 2.10004007e-02 1.09643879e-02
 6.94502526e-03 1.36691264e-08]
```

```
cumulative_variance = np.cumsum(explained_variance)

cumulative_variance_df = pd.DataFrame({
    'Principal Component': [f'PC{i+1}' for i in range(len(cumulative_variance))],
    'Cumulative Variance': cumulative_variance
})

cumulative_variance_df.round(4)
```

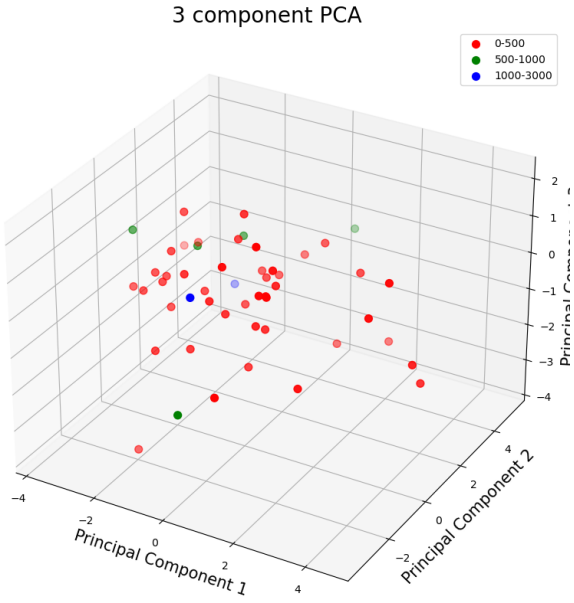


Then, I printed out my two-component and three-component graphs matching the principal components with the corresponding price, indicated in the legend.

```
bins = [0, 250, 3000]  # Define bin edges
labels = ['0-250', '250-3000']  # Define bin labels
finalDf['Price_bins'] = pd.cut(df['Price'], bins=bins, labels=labels, include_lowest=True)

pc2 = principalDf.iloc[:, :2]
pc2.columns = ['principal component 1', 'principal component 2']

fig = plt.figure(figsize = (8,6))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

className = ['0-250', '250-3000']

class_ = ['0-250', '250-3000']
colors = ['r', 'g']
for class_, color in zip(class_,colors):
    indicesToKeep = finalDf['Price_bins'] == class_
    ax.scatter(  pc2.loc[indicesToKeep, 'principal component 1']
               , pc2.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(className)
ax.grid()
```

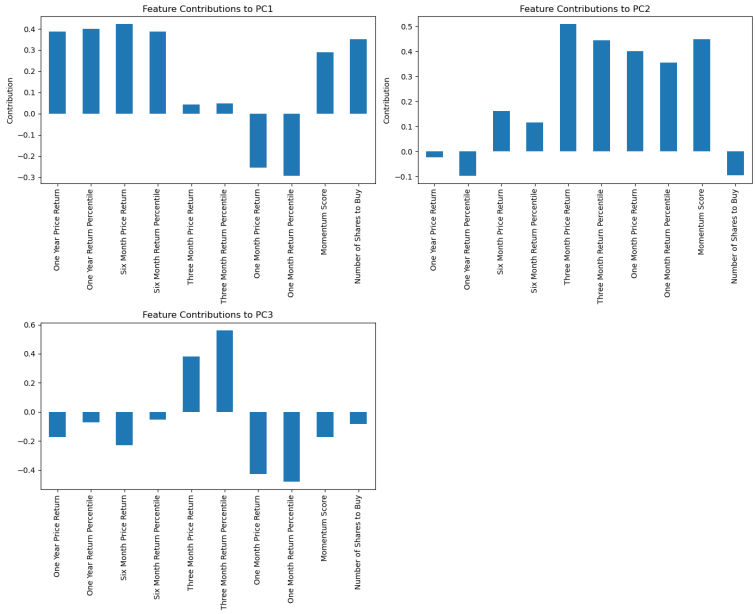



```
bins = [0, 500, 1000, 3000]  # Define bin edges
labels = ['0-500', '500-1000', '1000-3000']  # Define bin labels
finalDf['Price_bins'] = pd.cut(df['Price'], bins=bins, labels=labels, include_lowest=True)

pc3 = principalDf.iloc[:, :3]
pc3.columns = ['principal component 1', 'principal component 2', 'principal component 3']

fig = plt.figure(figsize = (12,10))
ax = fig.add_subplot(1,1,1,  projection='3d')
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_zlabel('Principal Component 3', fontsize = 15)
ax.set_title('3 component PCA', fontsize = 20)

className = ['0-500', '500-1000', '1000-3000']

class_ = ['0-500', '500-1000', '1000-3000']
colors = ['r', 'g', 'b']
for class_, color in zip(class_,colors):
    indicesToKeep = finalDf['Price_bins'] == class_
    ax.scatter(  pc3.loc[indicesToKeep, 'principal component 1']
               , pc3.loc[indicesToKeep, 'principal component 2']
               , pc3.loc[indicesToKeep, 'principal component 3']
               , c = color
               , s = 50)
ax.legend(className)
ax.grid()
```



Finally, the following code I used printed out the contribution ratio of each attribute in the dataset to the first three principal components. This is vital information to further analysis because we see the features that contribute to each principal component which represents the variance

```
components = pd.DataFrame(pca.components_, columns=X.columns)

fig, axes = plt.subplots(2, 2, figsize=(14, 12))
#plt.subplots_adjust(hspace=4)

components.iloc[0].plot(kind='bar', ax=axes[0, 0], title='Feature Contributions to PC1')
axes[0, 0].set_ylabel('Contribution')

components.iloc[1].plot(kind='bar', ax=axes[0,1], title='Feature Contributions to PC2')
axes[0, 1].set_ylabel('Contribution')

components.iloc[2].plot(kind='bar', ax=axes[1, 0], title='Feature Contributions to PC3')
axes[1, 1].set_ylabel('Contribution')

fig.delaxes(axes[1, 1])

plt.tight_layout(rect=[0, 0, 1, 0.95])

plt.show()
```



4 Discussion

Based on the 'Individual Variance Explained by Each Principal Component' graph and the corresponding code, I calculated the cumulative variance for each additional principal component. The results tell me that to maintain 90% of the variance in the dataset, I only need to keep the first five principal components. The reduction cuts my dataset in half which reduces the time and computational power required to complete future data science or machine learning tasks.

The visualizations could continue to be improved by using different price bins for the target variable. There was a large range in the price, making the legend more difficult to read and interpret in the two- and three-component PCA graphs.

Overall, Principal Component Analysis worked effectively to reduce the dimensions of this dataset and should continue to be used in the future. For instance, healthcare data could benefit from dimensionality reduction techniques when trying to find patterns in the source of disease or illness. Larger datasets with more attributes and rows will continue to benefit from the implementation of PCA.

Citations

References

[1] GeeksforGeeks. Mathematical approach to pca. https://www.geeksforgeeks.org/mathematical-approach-to-pca/, n.d. Accessed: 2024-12-06.

[2] Statology. How to read a covariance matrix. https://www.statology.org/how-to-read-covariance-matrix/, n.d. Accessed: 2024-12-06.