
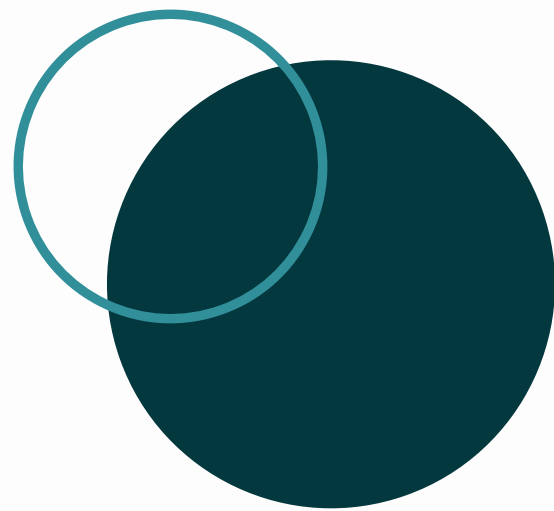


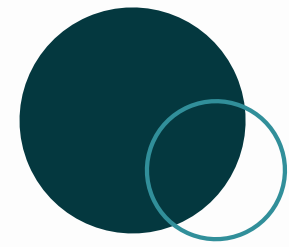


PRINCIPAL COMPONENT ANALYSIS



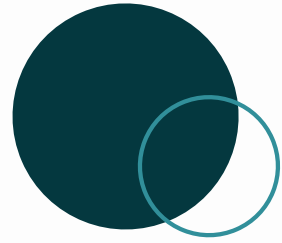
CHIRANTH R-PES1UG20CS116
SAI SUSHAEN-PES1UG20CS117
SHIVDATTA-PES1UG20CS100
ARUN KUMAR-PES1UG20CS076





Introduction

- > The central idea of principal component analysis (PCA) is to **reduce the dimensionality** of a data set consisting of a large number of interrelated variables, while **retaining as much as possible of the variation** present in the data set
- > Its goal is to extract the important information from the statistical data to represent it as a set of **new orthogonal variables** called principal components
- > It is determined by **eigenvectors and eigenvalues**



Goals of PCA

- **Extract the most important information from the data table.**
- **Compress the size of the data set by keeping only this important information**
- **Compress the data, by reducing the number of dimensions, without much loss of information**
- **This technique used in image compression**

COVARIANCE MATRIX

- Covariance is always measured between 2 dimensions.
- If we calculate the covariance between one dimension and itself, we will get the variance

$$\text{Var}(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

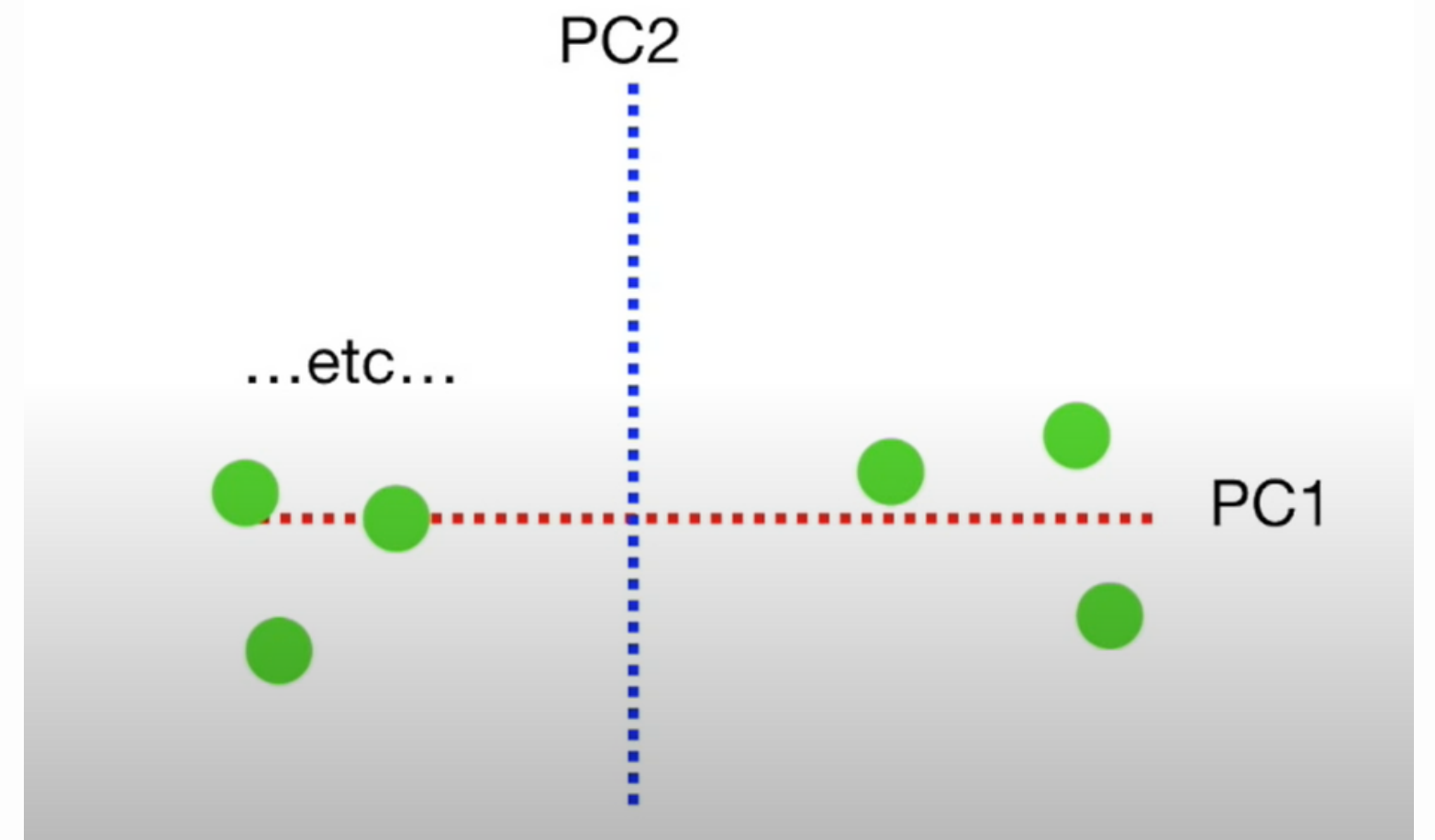
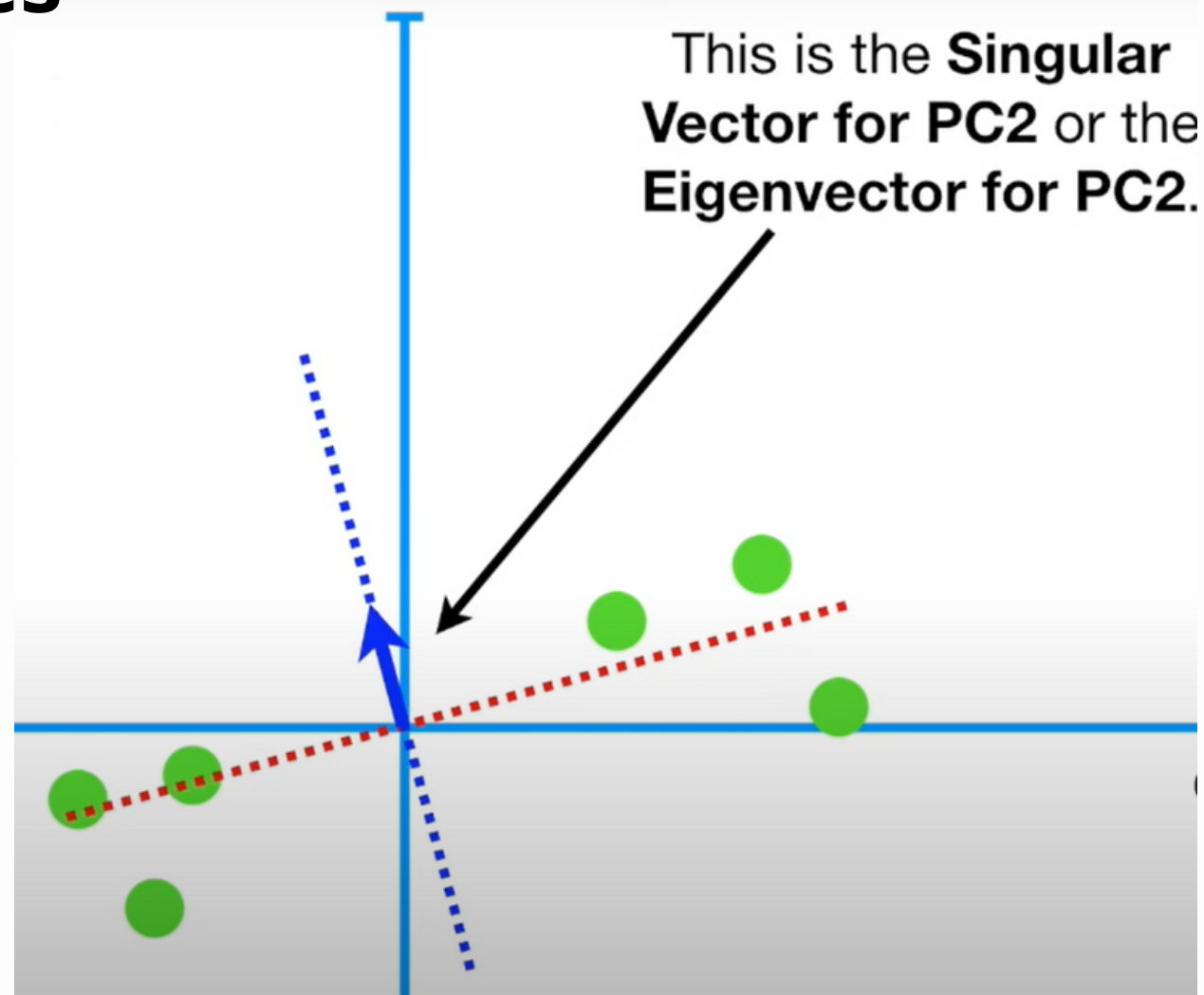
$$\text{Cov}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

$$\begin{bmatrix} \text{cov}(x,y) & \text{cov}(y,z) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{bmatrix}$$

- PCA is obtained from the eigen-decomposition of a covariance or a correlation matrix

Eigenvectors and eigenvalues

- All the eigenvectors of a matrix are perpendicular, i.e. at right angles to each other, no matter how many dimensions you have
- This is important because it means that you can express the data in terms of these perpendicular eigenvectors, instead of expressing them in terms of the x and y axes



METHODOLOGY

Get data

| Variable X | Variable Y |
|------------|------------|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

Find variance

| Deviation from Mean for X | Deviation from Mean for Y |
|------------------------------|------------------------------|
| 0.69 | 0.49 |
| -1.31 | -1.21 |
| 0.39 | 0.99 |
| 0.09 | 0.29 |
| 1.29 | 1.09 |
| 0.49 | 0.79 |
| 0.19 | -0.31 |
| -0.81 | -0.81 |
| -0.31 | -0.31 |
| -0.71 | -1.01 |

Covariance matrix

$$\text{COV} = \begin{bmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.716555558 \end{bmatrix}$$

Eigenvector

$$\text{Eigenvalue} = \begin{bmatrix} 0.0490833989 \\ 1.28402771 \end{bmatrix}$$

$$\text{Eigenvector} = \begin{bmatrix} -0.735178656 & -0.677873399 \\ 0.677873399 & -0.735178656 \end{bmatrix}$$

Choosing Components and Forming a Feature Vector

- The eigenvector with the highest eigenvalue is the principle component of the data set
- The eigenvector with the largest eigenvalue was the one that pointed down the middle of the data
- Once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance.
- We choose only the first eigenvectors, then the final data set will have fewer dimensions

Given with the example set of data, and the fact that here will be 2 eigenvectors, that means there will be two choices. A feature vector can be form with either of the eigenvectors-

$$\begin{bmatrix} 0.677873399 & -0.735178656 \\ -0.735178656 & -0.677873399 \end{bmatrix}$$

or, the smaller one can be chosen to leave out, less significant component and only have a single column vector-

$$\begin{bmatrix} 0.677873399 \\ -0.735178656 \end{bmatrix}$$



```
# Standardize the data
from sklearn.preprocessing import StandardScaler#sclearn-scikit-classification, regression, clustering, dimensionality reduction.
X = train.values
X_std = StandardScaler().fit_transform(X)

# Calculating Eigenvectors and eigenvalues of Covariance matrix
mean_vec = np.mean(X_std, axis=0)#find the mean
cov_mat = np.cov(X_std.T)#find the covariance matrix
eig_vals, eig_vecs = np.linalg.eig(cov_mat)#we can find the eigen vector

# Create a list of (eigenvalue, eigenvector) tuples
eig_pairs = [ (np.abs(eig_vals[i]),eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the eigenvalue, eigenvector pair from high to low
eig_pairs.sort(key = lambda x: x[0], reverse= True)

# Calculation of Explained Variance from the eigenvalues
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)] # Individual explained variance
cum_var_exp = np.cumsum(var_exp) # Cumulative explained variance
#shows the discrepancy between a model and actual data.
```

[7]:

```
# Invoke SKlearn's PCA method
n_components = 30
pca = PCA(n_components=n_components).fit(train.values)

eigenvalues = pca.components_.reshape(n_components, 28, 28)

# Extracting the PCA components ( eigenvalues )
#eigenvalues = pca.components_.reshape(n_components, 28, 28)
eigenvalues = pca.components_
```

Deriving the New Data Set

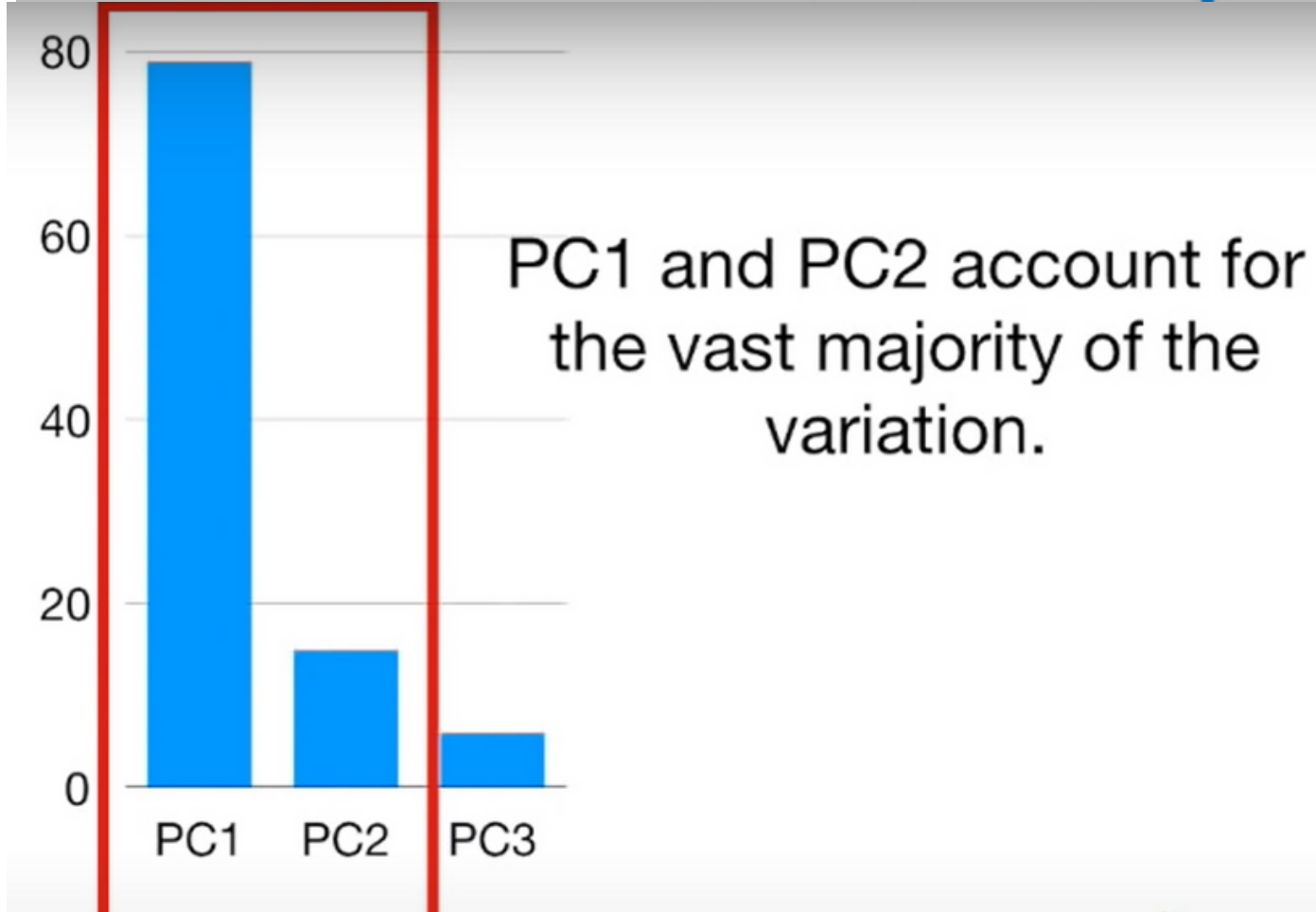
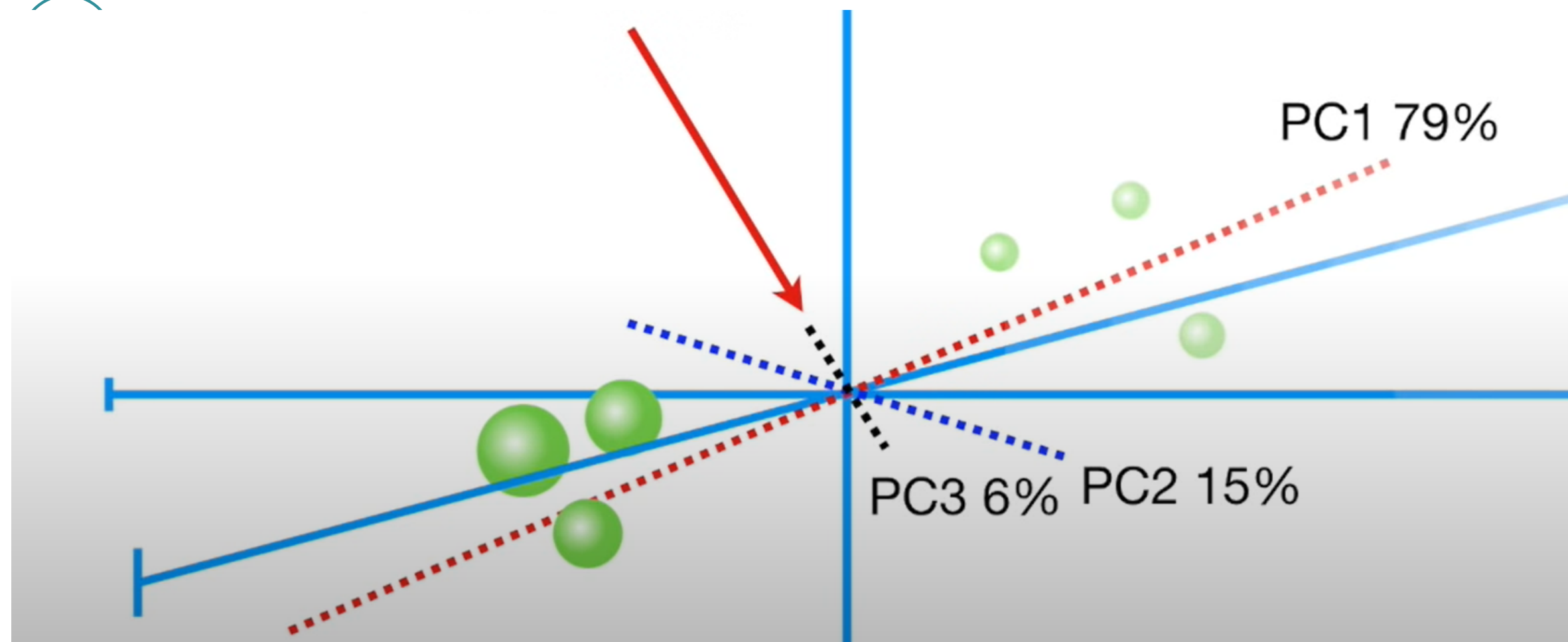
$$\text{Final Data} = \text{Row Feature Vector} \times \text{Row Data Adjust}$$

- The new data set has reduced dimensionality, in terms of the vectors that satisfy the original data the most and leaving the unimportant eigenvectors.

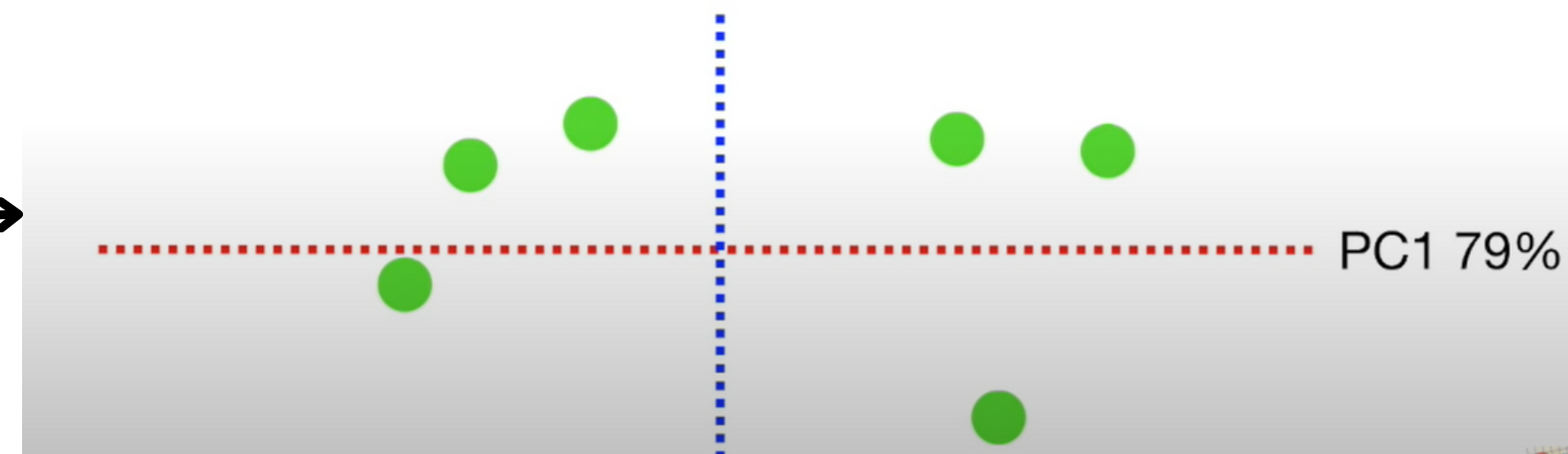
| Transformed Data (Single eigenvector) for Variable X |
|---------------------------------------------------------|
| -0.8279 70186 |
| 1.77758033 |
| -0.992197494 |
| -0.274210416 |
| -1.67580142 |
| -0.912949103 |
| 0.0991094375 |
| 1.14457216 |
| 0.438046137 |
| 1.22382056 |

- Basically we have transformed our data so that is expressed in terms of the patterns between them, where the patterns are the lines that most closely describe the relationships between the data

Example



PC1 and PC2 account for the vast majority of the variation.



Coding

We have worked on the MNIST Dataset for "Dimensionality reduction"