Week 7: Dimensionality Reduction & Clustering

Complete Theoretical Foundations

October 22, 2025

Abstract

This document provides comprehensive theoretical foundations for dimensionality reduction and clustering techniques applied to Finance analytics. We cover Principal Component Analysis (PCA), Factor Analysis, and K-Means Clustering with both rigorous mathematical formulations and intuitive explanations with low-to-moderate mathematical depth. Applications focus on portfolio construction, risk decomposition, and/or market segmentation in DeFi ecosystems.

Contents

Part I
Mathematical Theory with Proofs

1 Principal Component Analysis (PCA)

1.1 Mathematical Framework

Definition 1.1 (Data Matrix)**.** Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a data matrix where n is the number of observations (e.g., time periods) and p is the number of features (e.g., stock returns). Each row $\mathbf{x}_i \in \mathbb{R}^p$ represents observation i, and each column represents a feature.

Definition 1.2 (Centered Data Matrix)**.** The centered data matrix $\mathbf{X}_c$ is obtained by subtracting the column means:

$$\mathbf{X}_c = \mathbf{X} - \mathbf{1}_n \boldsymbol{\mu}^T \tag{1}$$

where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i = \frac{1}{n}\mathbf{X}^T\mathbf{1}_n$ is the mean vector and $\mathbf{1}_n \in \mathbb{R}^n$ is a vector of ones.

Definition 1.3 (Sample Covariance Matrix)**.** The sample covariance matrix is defined as:

$$\boldsymbol{\Sigma} = \frac{1}{n-1}\mathbf{X}_c^T\mathbf{X}_c = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \in \mathbb{R}^{p \times p} \tag{2}$$

where $\boldsymbol{\Sigma}_{jk} = \mathrm{Cov}(X_j, X_k)$ represents the covariance between features j and k.

Theorem 1.1 (Properties of Covariance Matrix)**.** The covariance matrix $\boldsymbol{\Sigma}$ has the following properties:

(i) Symmetry: $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$

(ii) Positive semi-definiteness: $\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w} \geq 0$ for all $\mathbf{w} \in \mathbb{R}^p$

(iii) Real eigenvalues and orthogonal eigenvectors

1.2 Eigenvalue Decomposition

Definition 1.4 (Eigenvalue Problem)**.** The principal components are found by solving the eigenvalue problem:

$$\boldsymbol{\Sigma}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{3}$$

where λ_i is the i-th eigenvalue and $\mathbf{v}_i$ is the corresponding eigenvector.

By convention, eigenvalues are ordered: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$.

Theorem 1.2 (Spectral Decomposition)**.** The covariance matrix can be decomposed as:

$$\boldsymbol{\Sigma} = \sum_{i=1}^{p} \lambda_i \mathbf{v}_i \mathbf{v}_i^T = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T \tag{4}$$

where $\mathbf{V} = [\mathbf{v}_1 \mid \mathbf{v}_2 \mid \cdots \mid \mathbf{v}_p]$ is the matrix of eigenvectors and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ is the diagonal matrix of eigenvalues.

1.3 Principal Component Transformation

Definition 1.5 (Principal Component Scores)**.** The principal component scores are obtained by projecting the centered data onto the principal components:

$$\mathbf{Z} = \mathbf{X}_c \mathbf{V}_k \in \mathbb{R}^{n \times k} \tag{5}$$

where $\mathbf{V}_k = [\mathbf{v}_1 \mid \mathbf{v}_2 \mid \cdots \mid \mathbf{v}_k]$ contains the first k eigenvectors.

The i-th principal component score for observation t is:

$$z_{ti} = \mathbf{x}_t^T \mathbf{v}_i = \sum_{j=1}^{p} x_{tj} v_{ji} \tag{6}$$

Theorem 1.3 (Orthogonality of Principal Components)**.** Principal components are mutually orthogonal:

$$\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{7}$$

Proof. Let $\mathbf{v}_i$ and $\mathbf{v}_j$ be eigenvectors corresponding to distinct eigenvalues $\lambda_i \neq \lambda_j$. Then:

$$\begin{aligned} \boldsymbol{\Sigma}\mathbf{v}_i &= \lambda_i \mathbf{v}_i \\ \boldsymbol{\Sigma}\mathbf{v}_j &= \lambda_j \mathbf{v}_j \end{aligned}$$

Multiplying the first equation by $\mathbf{v}_j^T$ from the left:

$$\mathbf{v}_j^T \boldsymbol{\Sigma} \mathbf{v}_i = \lambda_i \mathbf{v}_j^T \mathbf{v}_i$$

Since $\boldsymbol{\Sigma}$ is symmetric, $\mathbf{v}_j^T \boldsymbol{\Sigma} \mathbf{v}_i = \mathbf{v}_i^T \boldsymbol{\Sigma} \mathbf{v}_j$. Similarly, multiplying the second equation by $\mathbf{v}_i^T$:

$$\mathbf{v}_i^T \boldsymbol{\Sigma} \mathbf{v}_j = \lambda_j \mathbf{v}_i^T \mathbf{v}_j$$

Therefore:

$$\lambda_i \mathbf{v}_j^T \mathbf{v}_i = \lambda_j \mathbf{v}_i^T \mathbf{v}_j \implies (\lambda_i - \lambda_j) \mathbf{v}_i^T \mathbf{v}_j = 0$$

Since $\lambda_i \neq \lambda_j$, we have $\mathbf{v}_i^T \mathbf{v}_j = 0$. □

1.4 Variance Analysis

Theorem 1.4 (Variance of Principal Components)**.** The variance of the i-th principal component is equal to the i-th eigenvalue:

$$\mathrm{Var}(Z_i) = \mathbf{v}_i^T \boldsymbol{\Sigma} \mathbf{v}_i = \lambda_i \tag{8}$$

Proof. By definition, the variance of Z_i is:

$$\begin{aligned} \mathrm{Var}(Z_i) &= \mathrm{Var}(\mathbf{X}_c \mathbf{v}_i) \\ &= \mathbb{E}[(\mathbf{X}_c \mathbf{v}_i)^T (\mathbf{X}_c \mathbf{v}_i)] \\ &= \mathbb{E}[\mathbf{v}_i^T \mathbf{X}_c^T \mathbf{X}_c \mathbf{v}_i] \\ &= \mathbf{v}_i^T \mathbb{E}[\mathbf{X}_c^T \mathbf{X}_c] \mathbf{v}_i \\ &= \mathbf{v}_i^T (n-1) \boldsymbol{\Sigma} \mathbf{v}_i \\ &= (n-1) \mathbf{v}_i^T \boldsymbol{\Sigma} \mathbf{v}_i \end{aligned}$$

Since $\boldsymbol{\Sigma}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ and $\|\mathbf{v}_i\| = 1$:

$$\mathrm{Var}(Z_i) = (n-1) \lambda_i \mathbf{v}_i^T \mathbf{v}_i = (n-1) \lambda_i$$

In the sample covariance definition, we divide by $n-1$, so $\mathrm{Var}(Z_i) = \lambda_i$. □

Theorem 1.5 (Total Variance Preservation)**.** PCA preserves the total variance:

$$\sum_{i=1}^{p} \text{Var}(Z_i) = \sum_{i=1}^{p} \lambda_i = \text{tr}(\mathbf{\Sigma}) = \sum_{j=1}^{p} \text{Var}(X_j) \tag{9}$$

Proof. By the spectral theorem, $\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$. Taking the trace:

$$\begin{aligned} \text{tr}(\mathbf{\Sigma}) &= \text{tr}(\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T) \\ &= \text{tr}(\mathbf{\Lambda}\mathbf{V}^T\mathbf{V}) \quad \text{(cyclic property)} \\ &= \text{tr}(\mathbf{\Lambda}) \quad (\text{since } \mathbf{V}^T\mathbf{V} = \mathbf{I}) \\ &= \sum_{i=1}^{p} \lambda_i \end{aligned}$$

Also, $\text{tr}(\mathbf{\Sigma}) = \sum_{j=1}^{p} \mathbf{\Sigma}_{jj} = \sum_{j=1}^{p} \text{Var}(X_j)$. □

Definition 1.6 (Proportion of Variance Explained)**.** The proportion of variance explained by the i-th principal component is:

$$\rho_i = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \tag{10}$$

The cumulative proportion explained by the first k components is:

$$R_k^2 = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{p} \lambda_j} \tag{11}$$

1.5 Optimization Perspective

Theorem 1.6 (Variance Maximization)**.** The first principal component $\mathbf{v}_1$ solves the constrained optimization problem:

$$\mathbf{v}_1 = \text{argmax}_{\mathbf{w} \in \mathbb{R}^p} \mathbf{w}^T \mathbf{\Sigma} \mathbf{w} \quad \text{subject to} \quad \mathbf{w}^T\mathbf{w} = 1 \tag{12}$$

Proof. Form the Lagrangian:

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T \mathbf{\Sigma} \mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1) \tag{13}$$

Taking the derivative with respect to $\mathbf{w}$ and setting to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 2\mathbf{\Sigma}\mathbf{w} - 2\lambda\mathbf{w} = 0 \implies \mathbf{\Sigma}\mathbf{w} = \lambda\mathbf{w} \tag{14}$$

This is the eigenvalue equation. To maximize $\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} = \lambda\mathbf{w}^T\mathbf{w} = \lambda$, we choose the largest eigenvalue λ_1 with corresponding eigenvector $\mathbf{v}_1$. □

Theorem 1.7 (Sequential Optimization)**.** The k-th principal component $\mathbf{v}_k$ solves:

$$\mathbf{v}_k = \text{argmax}_{\mathbf{w} \in \mathbb{R}^p} \mathbf{w}^T \mathbf{\Sigma} \mathbf{w} \quad \text{subject to} \quad \mathbf{w}^T\mathbf{w} = 1, \quad \mathbf{w}^T\mathbf{v}_i = 0 \text{ for } i < k \tag{15}$$

1.6 Reconstruction and Approximation

Definition 1.7 (Data Reconstruction)**.** The reconstruction of $\mathbf{X}_c$ using k principal components is:

$$\hat{\mathbf{X}}_c = \mathbf{Z}\mathbf{V}_k^T = \mathbf{X}_c\mathbf{V}_k\mathbf{V}_k^T \tag{16}$$

Theorem 1.8 (Eckart-Young Theorem)**.** The best rank-k approximation of $\mathbf{X}_c$ (in Frobenius norm) is given by PCA:

$$\min_{\text{rank}(\mathbf{A})=k} \|\mathbf{X}_c - \mathbf{A}\|_F^2 = \|\mathbf{X}_c - \mathbf{X}_c\mathbf{V}_k\mathbf{V}_k^T\|_F^2 = \sum_{i=k+1}^{p} \lambda_i \tag{17}$$

Definition 1.8 (Mean Squared Reconstruction Error)**.**

$$\text{MSE}_k = \frac{1}{np}\|\mathbf{X}_c - \hat{\mathbf{X}}_c\|_F^2 = \frac{1}{np}\sum_{i=k+1}^{p} \lambda_i \tag{18}$$

1.7 Factor Loadings

Definition 1.9 (Factor Loadings)**.** The loading of feature j on principal component i is:

$$\ell_{ji} = \text{Cor}(X_j, Z_i) = \frac{\text{Cov}(X_j, Z_i)}{\sqrt{\text{Var}(X_j)\text{Var}(Z_i)}} \tag{19}$$

For standardized data (where $\text{Var}(X_j) = 1$):

$$\ell_{ji} = \frac{v_{ji}\sqrt{\lambda_i}}{1} = v_{ji}\sqrt{\lambda_i} \tag{20}$$

Key Point

Loadings with large absolute values indicate which original features contribute most to each principal component, facilitating interpretation.

1.8 Finance Application: Portfolio Risk Decomposition

DeFi Application

In DeFi portfolio management, let $\mathbf{r}_t \in \mathbb{R}^p$ be the vector of token log-returns at time t. The portfolio return is:

$$r_{pt} = \mathbf{w}^T\mathbf{r}_t \tag{21}$$

where $\mathbf{w} \in \mathbb{R}^p$ is the portfolio weight vector with $\sum_{i=1}^{p} w_i = 1$.
Using PCA on the return covariance matrix, we can decompose portfolio variance as:

$$\text{Var}(r_{pt}) = \mathbf{w}^T\mathbf{\Sigma}_r\mathbf{w} = \sum_{i=1}^{k} \alpha_i^2\lambda_i + \epsilon \tag{22}$$

where $\alpha_i = \mathbf{w}^T\mathbf{v}_i$ is the portfolio loading on PC i, and ϵ represents residual variance from components $k+1$ to p.
This decomposition allows us to:

- Identify major sources of portfolio risk
- Construct hedging strategies based on principal factors
- Monitor factor exposure over time
- Optimize diversification across risk factors

2 Factor Analysis

2.1 Statistical Model

Definition 2.1 (Factor Model)**.** Factor Analysis postulates that observed variables are linear combinations of unobserved (latent) common factors plus unique errors:

$$\mathbf{X} = \mathbf{\Lambda F} + \boldsymbol{\epsilon} \tag{23}$$

where:

- $\mathbf{X} \in \mathbb{R}^{n \times p}$: observed variables (centered)
- $\mathbf{\Lambda} \in \mathbb{R}^{p \times m}$: factor loadings matrix
- $\mathbf{F} \in \mathbb{R}^{n \times m}$: latent common factors with $m < p$
- $\boldsymbol{\epsilon} \in \mathbb{R}^{n \times p}$: unique factors (idiosyncratic errors)

In component form for observation i:

$$x_{ij} = \sum_{k=1}^{m} \lambda_{jk} f_{ik} + \epsilon_{ij}, \quad j = 1, \ldots, p \tag{24}$$

2.2 Distributional Assumptions

The common factors are standardized:

$$\mathbb{E}[\mathbf{F}] = \mathbf{0}, \quad \text{Cov}(\mathbf{F}) = \mathbf{I}_m \tag{25}$$

Under normality: $\mathbf{f}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m)$

The unique factors have zero mean and diagonal covariance:

$$\mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0}, \quad \text{Cov}(\boldsymbol{\epsilon}) = \Psi = \text{diag}(\psi_1, \psi_2, \ldots, \psi_p) \tag{26}$$

Under normality: $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \Psi)$

Common factors and unique factors are uncorrelated:

$$\text{Cov}(\mathbf{F}, \boldsymbol{\epsilon}) = \mathbf{0} \tag{27}$$

2.3 Covariance Structure

Theorem 2.1 (Implied Covariance)**.** Under the distributional assumptions above, the covariance matrix of $\mathbf{X}$ is:

$$\mathbf{\Sigma} = \text{Cov}(\mathbf{X}) = \mathbf{\Lambda\Lambda}^T + \Psi \tag{28}$$

Proof. Starting from the factor model $\mathbf{X} = \mathbf{\Lambda F} + \boldsymbol{\epsilon}$:

$$\begin{aligned}
\mathbf{\Sigma} &= \text{Cov}(\mathbf{X}) \\
&= \text{Cov}(\mathbf{\Lambda F} + \boldsymbol{\epsilon}) \\
&= \text{Cov}(\mathbf{\Lambda F}) + \text{Cov}(\boldsymbol{\epsilon}) + 2\text{Cov}(\mathbf{\Lambda F}, \boldsymbol{\epsilon}) \\
&= \mathbf{\Lambda}\text{Cov}(\mathbf{F})\mathbf{\Lambda}^T + \Psi + \mathbf{0} \\
&= \mathbf{\Lambda I}_m \mathbf{\Lambda}^T + \Psi \\
&= \mathbf{\Lambda\Lambda}^T + \Psi
\end{aligned}$$

□

2.4 Communality and Uniqueness

Definition 2.2 (Communality)**.** The communality h_j^2 of variable j is the proportion of its variance explained by common factors:

$$h_j^2 = \sum_{k=1}^{m} \lambda_{jk}^2 \tag{29}$$

Definition 2.3 (Uniqueness)**.** The uniqueness u_j of variable j is the proportion of its variance explained by the unique factor:

$$u_j = \psi_j = \text{Var}(X_j) - h_j^2 \tag{30}$$

For standardized variables ($\text{Var}(X_j) = 1$):

$$u_j = 1 - h_j^2 \tag{31}$$

Theorem 2.2 (Variance Decomposition)**.** For variable j, the total variance decomposes as:

$$\text{Var}(X_j) = h_j^2 + \psi_j = \sum_{k=1}^{m} \lambda_{jk}^2 + \psi_j \tag{32}$$

Intuition

Communality represents the "shared" variance that variable j has with other variables through common factors. **Uniqueness** represents the "specific" variance unique to variable j. High communality ($h_j^2 \approx 1$) means the variable is well-explained by common factors; low communality means it has substantial idiosyncratic behavior.

2.5 Factor Scores

Given observations $\mathbf{X}$, we estimate latent factor scores $\hat{\mathbf{F}}$.

Definition 2.4 (Regression Factor Scores)**.** The regression method estimates:

$$\hat{\mathbf{f}}_i = (\mathbf{\Lambda}^T \Psi^{-1} \mathbf{\Lambda})^{-1} \mathbf{\Lambda}^T \Psi^{-1} \mathbf{x}_i \tag{33}$$

Definition 2.5 (Bartlett Factor Scores)**.** Bartlett's method (weighted least squares):

$$\hat{\mathbf{f}}_i = (\mathbf{\Lambda}^T \Psi^{-1} \mathbf{\Lambda})^{-1} \mathbf{\Lambda}^T \Psi^{-1} \mathbf{x}_i \tag{34}$$

2.6 Factor Rotation

Theorem 2.3 (Rotation Invariance)**.** If $(\mathbf{\Lambda}, \mathbf{F})$ satisfies the factor model, then so does $(\mathbf{\Lambda T}, \mathbf{T}^{-1}\mathbf{F})$ for any non-singular matrix $\mathbf{T}$.

Proof.

$$\begin{aligned} \mathbf{X} &= \mathbf{\Lambda F} + \boldsymbol{\epsilon} \\ &= \mathbf{\Lambda T T}^{-1}\mathbf{F} + \boldsymbol{\epsilon} \\ &= \tilde{\mathbf{\Lambda}}\tilde{\mathbf{F}} + \boldsymbol{\epsilon} \end{aligned}$$

where $\tilde{\mathbf{\Lambda}} = \mathbf{\Lambda T}$ and $\tilde{\mathbf{F}} = \mathbf{T}^{-1}\mathbf{F}$.

The covariance is preserved:

$$\tilde{\mathbf{\Lambda}}\tilde{\mathbf{\Lambda}}^T + \Psi = \mathbf{\Lambda T T}^T \mathbf{\Lambda}^T + \Psi = \mathbf{\Lambda\Lambda}^T + \Psi = \mathbf{\Sigma}$$

□

Definition 2.6 (Varimax Rotation)**.** Varimax rotation (orthogonal) maximizes the variance of squared loadings:

$$\max_{\mathbf{T}^T\mathbf{T}=\mathbf{I}} \sum_{k=1}^{m} \left[\frac{1}{p} \sum_{j=1}^{p} \tilde{\lambda}_{jk}^4 - \left(\frac{1}{p} \sum_{j=1}^{p} \tilde{\lambda}_{jk}^2 \right)^2 \right] \tag{35}$$

where $\tilde{\mathbf{\Lambda}} = \mathbf{\Lambda}\mathbf{T}$.

Key Point

Rotation makes factors more interpretable by producing "simple structure"—each variable loads highly on one factor and weakly on others.

2.7 Maximum Likelihood Estimation

Definition 2.7 (Likelihood Function)**.** Under normality assumptions, the log-likelihood for factor analysis is:

$$\ell(\mathbf{\Lambda}, \Psi) = -\frac{np}{2}\log(2\pi) - \frac{n}{2}\log|\mathbf{\Sigma}| - \frac{1}{2}\sum_{i=1}^{n} \mathbf{x}_i^T \mathbf{\Sigma}^{-1} \mathbf{x}_i \tag{36}$$

where $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^T + \Psi$.

The maximum likelihood estimates $(\hat{\mathbf{\Lambda}}, \hat{\Psi})$ are found by numerical optimization (e.g., EM algorithm).

2.8 Application: Multi-Factor Risk Model

DeFi Application

Consider a universe of p DeFi tokens with returns $\mathbf{r}_t$. Factor analysis identifies m latent risk factors:

Return Decomposition:

$$r_{it} = \alpha_i + \sum_{k=1}^{m} \beta_{ik} f_{kt} + \epsilon_{it} \tag{37}$$

where:

- r_{it}: return of token i at time t
- α_i: token-specific intercept (alpha)
- f_{kt}: common factor k (e.g., market, liquidity, protocol risk)
- $\beta_{ik} = \lambda_{ik}$: factor loading (sensitivity to factor k)
- ϵ_{it}: idiosyncratic return

Portfolio Risk Decomposition:

$$\text{Var}(r_{pt}) = \sum_{k=1}^{m} \beta_{pk}^2 \text{Var}(f_{kt}) + \text{Var}(\epsilon_{pt}) \tag{38}$$

where $\beta_{pk} = \sum_{i=1}^{p} w_i \beta_{ik}$ is the portfolio loading on factor k.

Applications:

- **Risk Attribution:** Decompose portfolio risk into factor contributions
- **Hedging:** Neutralize exposure to unwanted factors
- **Alpha Generation:** Identify tokens with high idiosyncratic returns
- **Diversification:** Construct portfolios with low factor concentration

3 K-Means Clustering

3.1 Problem Formulation

Definition 3.1 (Clustering Problem)**.** Given a dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^p$, partition the data into K non-overlapping clusters $\{C_1, \ldots, C_K\}$ such that observations within each cluster are similar.

Definition 3.2 (Within-Cluster Sum of Squares (WCSS))**.** The within-cluster sum of squares for cluster C_k with centroid $\boldsymbol{\mu}_k$ is:

$$\text{WCSS}_k = \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 \tag{39}$$

where $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$ is the cluster centroid.

Definition 3.3 (K-Means Objective)**.** K-Means minimizes the total within-cluster sum of squares:

$$J(\{C_k\}, \{\boldsymbol{\mu}_k\}) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 \tag{40}$$

Equivalently:

$$\min_{\{C_k\}_{k=1}^K, \{\boldsymbol{\mu}_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 \tag{41}$$

3.2 Lloyd's Algorithm

Algorithm 1 K-Means Clustering (Lloyd's Algorithm)

Input: Data $\{\mathbf{x}_i\}_{i=1}^n$, number of clusters K
Initialize: Randomly select K centroids $\{\boldsymbol{\mu}_k^{(0)}\}_{k=1}^K$
$t \leftarrow 0$
repeat
  Assignment Step:
  for $i = 1$ to n **do**
    $C_k^{(t+1)} \leftarrow \{\mathbf{x}_i : k = \text{argmin}_j \|\mathbf{x}_i - \boldsymbol{\mu}_j^{(t)}\|_2\}$
  end for
  Update Step:
  for $k = 1$ to K **do**
    $\boldsymbol{\mu}_k^{(t+1)} \leftarrow \frac{1}{|C_k^{(t+1)}|} \sum_{\mathbf{x}_i \in C_k^{(t+1)}} \mathbf{x}_i$
  end for
  $t \leftarrow t + 1$
until convergence (no change in assignments or $|J^{(t)} - J^{(t-1)}| < \epsilon$)
Output: Clusters $\{C_k\}_{k=1}^K$, centroids $\{\boldsymbol{\mu}_k\}_{k=1}^K$

3.3 Convergence Properties

Theorem 3.1 (Monotonic Decrease)**.** Lloyd's algorithm guarantees that the objective function decreases monotonically:

$$J^{(t+1)} \leq J^{(t)} \tag{42}$$

for all iterations t.

Proof. Consider iteration t:

- **Assignment step:** Each point is assigned to its nearest centroid, which minimizes distance. Therefore, $J^{(t+1)} \leq J^{(t)}$ after assignment.
- **Update step:** The centroid $\boldsymbol{\mu}_k$ is the mean of points in C_k, which minimizes $\sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$ by the least squares property. Therefore, J further decreases or stays constant.

Since J is lower bounded (by zero) and decreases monotonically, the algorithm must converge. □

Theorem 3.2 (Finite Convergence)**.** Lloyd's algorithm converges to a local minimum in a finite number of iterations.

Proof. There are only finitely many possible partitions of n points into K clusters. Since J decreases strictly at each iteration (unless at a local minimum) and cannot revisit the same partition (as that would require J to increase), the algorithm must terminate in finite time. □

Remark 3.1. Lloyd's algorithm is guaranteed to converge, but only to a **local minimum**, not necessarily the global minimum. Multiple random initializations are recommended.

3.4 Distance Metrics

Definition 3.4 (Euclidean Distance)**.** The standard distance metric in K-Means:

$$d_{\text{Euclidean}}(\mathbf{x}_i, \boldsymbol{\mu}_k) = \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2 = \sqrt{\sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2} \tag{43}$$

Definition 3.5 (Manhattan Distance)**.**

$$d_{\text{Manhattan}}(\mathbf{x}_i, \boldsymbol{\mu}_k) = \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_1 = \sum_{j=1}^{p} |x_{ij} - \mu_{kj}| \tag{44}$$

Definition 3.6 (Mahalanobis Distance)**.** Accounts for correlations between features:

$$d_{\text{Mahalanobis}}(\mathbf{x}_i, \boldsymbol{\mu}_k) = \sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k)} \tag{45}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix.

3.5 Evaluation Metrics

Definition 3.7 (Total Sum of Squares (TSS))**.**

$$\text{TSS} = \sum_{i=1}^{n} \|\mathbf{x}_i - \bar{\mathbf{x}}\|_2^2 \tag{46}$$

where $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$ is the overall mean.

Definition 3.8 (Between-Cluster Sum of Squares (BCSS))**.**

$$\text{BCSS} = \sum_{k=1}^{K} |C_k| \|\boldsymbol{\mu}_k - \bar{\mathbf{x}}\|_2^2 \tag{47}$$

Theorem 3.3 (Variance Decomposition)**.**

$$\text{TSS} = \text{WCSS} + \text{BCSS} \tag{48}$$

Definition 3.9 (Silhouette Score)**.** For observation i in cluster C_k, define:

- $a_i = \frac{1}{|C_k|-1} \sum_{\mathbf{x}_j \in C_k, j \neq i} \|\mathbf{x}_i - \mathbf{x}_j\|$: average distance to other points in same cluster
- $b_i = \min_{l \neq k} \frac{1}{|C_l|} \sum_{\mathbf{x}_j \in C_l} \|\mathbf{x}_i - \mathbf{x}_j\|$: average distance to points in nearest different cluster

The silhouette score for observation i is:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1] \tag{49}$$

Key Point

Silhouette Score Interpretation:

- $s_i \approx 1$: Well clustered (far from other clusters, close to own)
- $s_i \approx 0$: On decision boundary between clusters
- $s_i \approx -1$: Likely misclassified

Average silhouette score: $\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s_i$

Definition 3.10 (Davies-Bouldin Index)**.** Measures average similarity between each cluster and its most similar cluster:

$$\text{DB} = \frac{1}{K} \sum_{k=1}^{K} \max_{l \neq k} \left(\frac{s_k + s_l}{d_{kl}} \right) \tag{50}$$

where:

- $s_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|$: average distance within cluster k
- $d_{kl} = \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l\|$: distance between centroids

Lower DB index indicates better clustering.

3.6 Choosing Optimal K

Definition 3.11 (Elbow Method)**.** Plot WCSS versus K and identify the "elbow" point where the rate of decrease sharply changes.

Definition 3.12 (Gap Statistic)**.** Compare WCSS to its expectation under null reference distribution:

$$\text{Gap}(K) = \mathbb{E}_{\text{null}}[\log(\text{WCSS}_K)] - \log(\text{WCSS}_K) \tag{51}$$

Choose K that maximizes the gap statistic.

3.7 K-Means++: Improved Initialization

Algorithm 2 K-Means++ Initialization

1: Choose first centroid $\boldsymbol{\mu}_1$ uniformly at random from $\{\mathbf{x}_i\}$
2: **for** $k = 2$ to K **do**
3: **for** each data point $\mathbf{x}_i$ **do**
4: Compute $D(\mathbf{x}_i)^2 = \min_{j<k} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$
5: **end for**
6: Choose $\boldsymbol{\mu}_k$ from $\{\mathbf{x}_i\}$ with probability $\propto D(\mathbf{x}_i)^2$
7: **end for**
8: **Output:** Initial centroids $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$

Theorem 3.4 (K-Means++ Guarantee)**.** K-Means++ initialization guarantees that the expected objective value is $O(\log K)$ times the optimal objective.

3.8 Application: Token Segmentation

DeFi Application

Problem: Segment DeFi tokens into homogeneous risk-return groups for portfolio construction.
Features: For each token i, compute:

- x_{i1}: Average daily return
- x_{i2}: Volatility (standard deviation of returns)
- x_{i3}: Trading volume (liquidity proxy)
- x_{i4}: Market capitalization
- x_{i5}: Total Value Locked (TVL)
- x_{i6}: Correlation with ETH
- x_{i7}: Maximum drawdown
- x_{i8}: Sharpe ratio

Procedure:

1. Standardize features: $\tilde{x}_{ij} = (x_{ij} - \bar{x}_j)/\sigma_j$
2. Apply K-Means with $K = 4$ clusters
3. Interpret cluster characteristics
4. Construct diversified portfolio

Example Clusters:

- **Cluster 1 (Blue Chips):** Low volatility, high liquidity, large market cap (e.g., WBTC, WETH, USDC)
- **Cluster 2 (Growth DeFi):** Medium risk, established protocols (e.g., UNI, AAVE, COMP)
- **Cluster 3 (Speculative):** High volatility, emerging projects
- **Cluster 4 (Illiquid):** Low liquidity, high risk, small market cap

Portfolio Strategy:

$$\mathbf{w}^* = \text{argmax}_{\mathbf{w}} \left[\mathbb{E}[\mathbf{w}^T \mathbf{r}] - \frac{\lambda}{2} \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} \right] \quad \text{subject to} \quad \sum_{i \in C_k} w_i \leq u_k \quad \forall k \tag{52}$$

where u_k is the maximum exposure to cluster k (e.g., $u_1 = 0.5$, $u_2 = 0.3$, $u_3 = 0.15$, $u_4 = 0.05$).

4 Integration: PCA + K-Means

4.1 Motivation

Combining PCA with K-Means leverages the strengths of both methods:

- **Dimensionality reduction:** PCA reduces feature space from p to $k \ll p$
- **Noise removal:** Small principal components often represent noise
- **Decorrelation:** PCs are uncorrelated, improving K-Means stability
- **Computational efficiency:** Clustering in lower dimension is faster
- **Visualization:** Can plot clusters in 2D/3D PC space

4.2 Theoretical Framework

Definition 4.1 (Two-Stage Procedure)**.**

1. **Stage 1 (PCA):** Project data to principal component space

$$\mathbf{Z} = \mathbf{X}_c \mathbf{V}_k \tag{53}$$

2. **Stage 2 (K-Means):** Cluster in PC space

$$\min_{\{C_j\}} \sum_{j=1}^{K} \sum_{\mathbf{z}_i \in C_j} \|\mathbf{z}_i - \boldsymbol{\mu}_j^z\|_2^2 \tag{54}$$

Theorem 4.1 (Dimension Reduction and Clustering)**.** Under appropriate conditions, clustering in the reduced k-dimensional PC space yields similar results to clustering in the original p-dimensional space when the first k PCs explain most variance.

4.3 Benefits Over Standard K-Means

Proposition 4.2 (Noise Reduction)**.** Let $\mathbf{X} = \mathbf{X}_{\text{signal}} + \mathbf{X}_{\text{noise}}$ where signal lies in a k-dimensional subspace. PCA+K-Means effectively ignores noise in the orthogonal complement, leading to more stable clusters.

Proposition 4.3 (Stability)**.** Clustering in PC space is more stable across different samples because:

- Correlations are removed (PCs are orthogonal)
- Small perturbations in original space have limited effect on dominant PCs
- Reduced dimensionality decreases variance of cluster assignments

4.4 Algorithm

Algorithm 3 Integrated PCA + K-Means

Input: Data $\mathbf{X} \in \mathbb{R}^{n \times p}$, number of PCs k, number of clusters K
PCA Stage:
  Center data: $\mathbf{X}_c = \mathbf{X} - \mathbf{1}_n \boldsymbol{\mu}^T$
  Compute covariance: $\boldsymbol{\Sigma} = \frac{1}{n-1} \mathbf{X}_c^T \mathbf{X}_c$
  Eigendecomposition: $\boldsymbol{\Sigma} = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^T$
  Select k components: $\mathbf{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$
  Transform: $\mathbf{Z} = \mathbf{X}_c \mathbf{V}_k$
K-Means Stage:
  Run K-Means on $\mathbf{Z}$ with K clusters
  Obtain cluster assignments $\{C_j\}_{j=1}^K$
Interpretation:
  Compute cluster centroids in PC space: $\boldsymbol{\mu}_j^z$
  Map back to original space: $\boldsymbol{\mu}_j = \mathbf{V}_k \boldsymbol{\mu}_j^z + \boldsymbol{\mu}$
Output: Clusters $\{C_j\}$, centroids in both spaces

4.5 Interpretation Strategy

After obtaining clusters, interpret them in both spaces:

1. **PC Space Interpretation:**
   - Examine cluster positions along each PC
   - Identify which PCs differentiate clusters
   - Use PC loadings to understand meaning
2. **Original Space Interpretation:**
   - Map centroids back: $\hat{\boldsymbol{\mu}}_j = \mathbf{V}_k \boldsymbol{\mu}_j^z + \boldsymbol{\mu}$
   - Compute average feature values for each cluster
   - Assign business/economic labels

4.6 Complete DeFi Example

Complete Portfolio Analysis Workflow

Dataset: 50 DeFi tokens, 8 features each (returns, volatility, volume, etc.)
Step 1: PCA

- Compute 8 PCs
- First 3 PCs explain 85% of variance:
  - PC1 (45%): "Market risk factor" (loadings on returns, ETH correlation, market cap)
  - PC2 (25%): "Liquidity factor" (loadings on volume, liquidity score)
  - PC3 (15%): "Volatility factor" (loadings on volatility, drawdown)
- Retain $k = 3$ components

Step 2: K-Means in PC Space

- Elbow method suggests $K = 4$ clusters
- Average silhouette score: $\bar{s} = 0.58$ (good separation)

Step 3: Cluster Characterization
Step 4: Portfolio Construction

$$\text{Allocation:} \begin{cases} 50\% & \text{Cluster 0 (stability)} \\ 30\% & \text{Cluster 1 (growth)} \\ 15\% & \text{Cluster 2 (upside)} \\ 5\% & \text{Cluster 3 (lottery)} \end{cases} \tag{55}$$

Risk Management:

- Monitor cluster membership monthly
- Rebalance if tokens migrate clusters
- Hedge factor exposures using PC loadings
- Set stop-losses per cluster risk profile

Part II
Intuitive Explanations (Non-Technical)

5 Understanding PCA Without Math

5.1 The Core Idea

Imagine you're analyzing 50 different cryptocurrencies, each with 8 different measurements (like price change, trading volume, market size, etc.). That's 400 numbers to keep track of! PCA helps you find the hidden patterns that explain most of what's happening, reducing your analysis to just 3-5 key factors.

Real-World Analogy

Describing People: You could measure height, weight, shoe size, arm length, leg length, hand size, etc. But really, there's one main factor: **"body size."** A second factor might be **"proportions"** (tall-thin vs short-wide). These 2 factors explain most variation instead of tracking 10+ measurements.
In DeFi:

- PC1 might represent "overall market sentiment" (affects all tokens similarly)
- PC2 might represent "DeFi vs CeFi" (DeFi tokens move together, opposite to traditional finance)
- PC3 might represent "large-cap vs small-cap" behavior

5.2 How It Works (Simple Steps)

1. **Center your data:** Subtract the average from each measurement so everything has mean = 0
2. **Find directions of maximum spread:**
   - PC1 = direction where data varies the most
   - PC2 = direction with next most variation (perpendicular to PC1)
   - PC3 = next most variation (perpendicular to PC1 and PC2)
3. **Transform your data:** Rotate your data so axes align with these new directions
4. **Keep only the important ones:** Usually first 3-5 components explain 80-90% of everything

5.3 Key Concepts in Plain English

Principal Components: New variables created by combining original features. Like making a smoothie—you combine fruits (original features) to get one drink (PC).

Loadings: Tell you the "recipe"—how much of each original feature goes into each PC. High loading = that feature is important for that component.

Variance Explained: Percentage of information captured. PC1 might explain 40%, PC2 explains 25%, etc. Usually aim for 80-90% total.

Scree Plot: Graph showing importance of each PC. Look for the "elbow"—point where curve flattens. Components after the elbow are mostly noise.

5.4 When to Use PCA

✓ You have many correlated features

✓ Want to visualize high-dimensional data

✓ Need to remove noise

✓ Want to speed up other analyses

× Features are already independent

× Need to keep all original features for interpretation

6 Understanding Factor Analysis Without Math

6.1 The Core Idea

Factor Analysis is like PCA's cousin with a different goal. Instead of just finding patterns, it tries to find **hidden causes** that explain why things are correlated.

Student Test Example

Students take 10 tests. Scores are correlated because of hidden abilities:

- **Factor 1:** "Math ability" (explains algebra, geometry, calculus scores)
- **Factor 2:** "Verbal ability" (explains reading, writing, vocabulary scores)
- **Unique factors:** Test-specific skills or luck

In DeFi:

- **Factor 1:** "Market risk" (affects all tokens)
- **Factor 2:** "Smart contract risk" (affects DeFi protocols)
- **Factor 3:** "Liquidity risk" (affects low-volume tokens)
- **Unique:** Token-specific news, hacks, upgrades

6.2 Key Differences from PCA

PCA	Factor Analysis
Explains ALL variance	Explains only SHARED variance
Components are math constructs	Factors are real hidden causes
First PC always explains most	Factors can be equal importance
Best for data reduction	Best for understanding structure

6.3 Key Concepts in Plain English

Common Factors: Hidden variables affecting multiple tokens (like "market sentiment")

Unique Factors: Specific to each token (like "this protocol got hacked")

Communality: How much of a token's behavior is "typical" (explained by common factors). High = follows the crowd.

Uniqueness: How much is "special" to that token. High = marches to its own drum.

Factor Rotation: Like tilting your head to see a pattern more clearly. Makes factors easier to interpret.

7 Understanding K-Means Without Math

7.1 The Core Idea

K-Means automatically groups similar things together. Like organizing your closet—putting all shirts together, pants together, shoes together—but the algorithm decides what "together" means.

Room Full of People

Imagine forming groups in a crowded room:

1. Pick 3 random people as "leaders"
2. Everyone stands next to their nearest leader
3. Leaders move to center of their group
4. People re-check if they're still closest to their leader
5. Repeat until no one switches groups

That's exactly how K-Means works with data!

7.2 How It Works (Simple Steps)

1. **Choose K** (number of groups you want)
2. **Pick K random starting points** (initial cluster centers)
3. **Assign each point** to nearest center
4. **Move centers** to average of their group
5. **Repeat steps 3-4** until groups stabilize

7.3 Key Concepts in Plain English

Centroid: The "center" or "average point" of a group. Represents typical characteristics of that cluster.

Distance: How we measure similarity. Smaller distance = more similar. Like measuring how far apart two people stand.

Inertia: Measures how "tight" clusters are. Lower = better. Like measuring how close group members stand to their leader.

Silhouette Score: Measures how well points fit their clusters. Range -1 to +1. Higher = better clustering.

7.4 Choosing the Right Number of Groups (K)

Elbow Method: Try K=1,2,3,...,10. Plot "tightness" vs K. Look for "elbow"—where improvement slows dramatically.

Silhouette Method: Try different K values. Pick the one with highest silhouette score.

Domain Knowledge: Use your understanding. In DeFi: maybe K=3 (low/medium/high risk) or K=4 (stable/growth/speculative/meme).

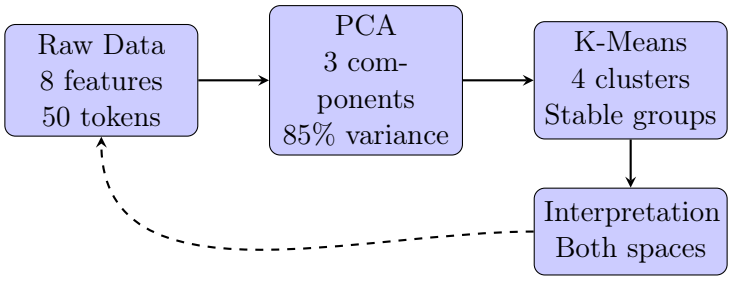
7.5 Practical Tips

- **Standardize first:** Make all features same scale (otherwise big numbers dominate)
- **Run multiple times:** Algorithm can get stuck. Run 10-20 times, keep best result.
- **Validate results:** Check if clusters make business sense
- **Combine with PCA:** Reduce dimensions first, then cluster (more stable!)

8 Putting It All Together: The Complete Workflow

8.1 The Power of Integration

Using PCA before clustering is like cleaning and organizing data before grouping—you get better, more stable results.



8.2 Step-by-Step Guide for Real DeFi Analysis

8.2.1 Phase 1: Data Preparation

1. **Collect Data:** Returns, volume, TVL, market cap, liquidity, sentiment
2. **Clean Data:** Handle missing values, remove extreme outliers
3. **Standardize:** Convert to z-scores (mean=0, std=1) *Critical!*

8.2.2 Phase 2: PCA Analysis

1. **Run PCA:** Extract all components initially
2. **Scree Plot:** Find elbow (usually 3-5 components)
3. **Keep Components:** That explain 80-90% variance
4. **Interpret:** Examine loadings, assign economic meaning

8.2.3 Phase 3: Clustering

1. **Elbow Method:** Test K=2 to 10, find optimal K
2. **Run K-Means:** On PC scores (not original data!)
3. **Validate:** Check silhouette scores, cluster sizes
4. **Interpret:** Map back to original features

8.2.4 Phase 4: Portfolio Construction

1. **Profile Clusters:** Calculate average characteristics
2. **Allocate:** Decide portfolio weight per cluster
3. **Select Tokens:** Choose representatives from each cluster
4. **Monitor:** Track cluster membership changes monthly

9 Common Mistakes to Avoid

1. **Not Standardizing:** Features with large values dominate. *Always standardize!*
2. **Too Many Components:** Keeping all PCs defeats the purpose. Use scree plot.
3. **Ignoring Validation:** Always check silhouette scores and cluster characteristics.
4. **Fixed Clusters:** Tokens move between clusters over time. Retrain regularly.
5. **Over-Interpretation:** Don't force unrealistic meanings on components.
6. **Wrong Distance:** Use standardized Euclidean or consider Mahalanobis.
7. **Outliers:** One extreme token distorts everything. Remove or treat separately.

10 Summary and Key Takeaways

Key Takeaways

PCA

- Reduces dimensions while preserving 80-90% of information
- Creates uncorrelated components ordered by importance
- Use when you have many correlated features
- Interpret via loadings to understand economic meaning

Factor Analysis

- Finds hidden causes of correlations
- Separates common factors from unique variance
- Use for risk modeling and causal interpretation
- Communality shows how "typical" each token is

K-Means

- Groups similar observations automatically
- Simple, fast, and interpretable
- Use elbow method and silhouette scores to choose K
- Always standardize data first

Integration

- PCA $\rightarrow$ K-Means gives more stable, interpretable clusters
- Removes noise and decorrelates features
- Interpret in both PC space and original space
- Best practice for production systems

11 Practice Problems

11.1 Problem 1: PCA Interpretation

Given a DeFi dataset with correlation matrix showing that 30 tokens have pairwise correlations ranging from 0.70 to 0.95:

(a) How many principal components would you expect to be significant?

(b) If PC1 explains 65% of variance, what does this suggest about the market?

(c) If all tokens have positive loadings on PC1, how would you interpret this component?

11.2 Problem 2: Factor Analysis

You perform factor analysis on 20 DeFi tokens and obtain:

- Factor 1: High loadings on UNI, SUSHI, CAKE (all DEX tokens)
- Factor 2: High loadings on AAVE, COMP, MKR (all lending)
- Factor 3: Positive loadings on all tokens

Token X has communality = 0.45. What does this mean?

11.3 Problem 3: K-Means Selection

Elbow plot shows:

K	Inertia
2	500
3	300
4	250
5	230
6	220

(a) What K would you choose and why?

(b) Why not always choose the largest K possible?

(c) What other metric would you check?

11.4 Problem 4: Integration Strategy

You have 100 tokens with 12 features each. Design a complete analysis strategy:

(a) How many PCs would you initially extract?

(b) What threshold for cumulative variance?

(c) How would you choose K for clustering?

(d) How would you validate your results?

A Mathematical Notation Summary

Symbol	Meaning
$\mathbf{X} \in \mathbb{R}^{n \times p}$	Data matrix (n observations, p features)
$\mathbf{x}_i \in \mathbb{R}^p$	Observation i (row vector)
$\mathbf{X}_c$	Centered data matrix
$\boldsymbol{\mu} \in \mathbb{R}^p$	Mean vector
$\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$	Covariance matrix
$\mathbf{v}_i \in \mathbb{R}^p$	i-th eigenvector (principal component)
$\lambda_i \in \mathbb{R}$	i-th eigenvalue
$\mathbf{Z} \in \mathbb{R}^{n \times k}$	Principal component scores
$\boldsymbol{\Lambda} \in \mathbb{R}^{p \times m}$	Factor loadings matrix
$\mathbf{F} \in \mathbb{R}^{n \times m}$	Latent factors
$\Psi \in \mathbb{R}^{p \times p}$	Unique variance (diagonal)
h_j^2	Communality of variable j
ψ_j	Uniqueness of variable j
C_k	Cluster k
$\boldsymbol{\mu}_k \in \mathbb{R}^p$	Centroid of cluster k
K	Number of clusters
$\|\cdot\|_2$	Euclidean norm
$\text{tr}(\cdot)$	Trace operator

Table 2: Mathematical notation used throughout the document

B Software Implementation Notes

B.1 Python Libraries

- `sklearn.decomposition.PCA`: Principal Component Analysis
- `sklearn.decomposition.FactorAnalysis`: Factor Analysis
- `sklearn.cluster.KMeans`: K-Means Clustering
- `sklearn.preprocessing.StandardScaler`: Data standardization
- `sklearn.metrics`: Silhouette score, Davies-Bouldin index

B.2 Basic Usage Example

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
```

```
# K-Means
kmeans = KMeans(n_clusters=4, n_init=10, random_state=42)
clusters = kmeans.fit_predict(X_pca)
```

C Further Reading

C.1 Foundational Papers

1. Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". *Philosophical Magazine*, 2(11), 559-572.
2. Hotelling, H. (1933). "Analysis of a Complex of Statistical Variables into Principal Components". *Journal of Educational Psychology*, 24(6), 417-441.
3. MacQueen, J. (1967). "Some Methods for Classification and Analysis of Multivariate Observations". *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1, 281-297.
4. Jolliffe, I. T. (2002). *Principal Component Analysis*, 2nd ed. Springer.
5. Arthur, D., & Vassilvitskii, S. (2007). "k-means++: The Advantages of Careful Seeding". *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027-1035.

C.2 DeFi-Specific Applications

1. Liu, Y., & Tsyvinski, A. (2021). "Risks and Returns of Cryptocurrency". *The Review of Financial Studies*, 34(6), 2689-2727.
2. Borri, N., & Shakhnov, K. (2022). "The Cross-Section of Cryptocurrency Returns". *Review of Asset Pricing Studies*, 12(3), 667-705.
3. Hu, A., Parlour, C. A., & Rajan, U. (2019). "Cryptocurrencies: Stylized Facts on a New Investible Instrument". *Financial Management*, 48(4), 1049-1068.

C.3 Online Resources

- StatQuest: https://statquest.org (Excellent visual explanations)
- Scikit-learn Documentation: https://scikit-learn.org
- Stanford CS229: Machine Learning (PCA lecture notes)
- Coursera: Applied Data Science with Python

D Glossary of Terms

Biplot: Visualization showing both observations and variable loadings in PC space

Centroid: The mean point of a cluster

Communality: Proportion of variance in a variable explained by common factors

Covariance Matrix: Matrix containing pairwise covariances between all features

Eigenvalue: Scalar representing variance explained by a principal component

Eigenvector: Direction in feature space defining a principal component

Elbow Method: Technique for choosing K by plotting inertia vs number of clusters

Factor Loading: Correlation between an observed variable and a latent factor

Inertia: Within-cluster sum of squares; lower is better

Latent Factor: Unobserved variable that influences multiple observed variables

Loading: Weight of an original feature in a principal component

Orthogonal: Perpendicular; uncorrelated for random variables

Scree Plot: Graph of eigenvalues vs component number

Silhouette Score: Measure of how well an observation fits its cluster (-1 to +1)

Standardization: Transforming features to have mean 0 and standard deviation 1

Uniqueness: Proportion of variance in a variable not explained by common factors

Variance Explained: Proportion of total variance captured by components