

Principal Component Analysis vs Factor Analysis

MA2003B - Application of Multivariate Methods in Data Science

MA2003B Course Team

Tecnológico de Monterrey

2025-09-29

Introducción al Análisis de Componentes Principales y Análisis Factorial

¿Por qué Reducción de Dimensionalidad?

La reducción de dimensionalidad es una técnica fundamental en el análisis multivariado que busca:

- Manejar datos de alta dimensionalidad de manera efectiva
- Reducir la complejidad computacional
- Eliminar ruido y redundancia
- Mejorar la interpretabilidad de los modelos
- Permitir la visualización de datos

Dos Enfoques Principales

- **Análisis de Componentes Principales (PCA):** Enfoque basado en datos, maximización de varianza
- **Análisis Factorial (FA):** Enfoque basado en modelo, identificación de constructos latentes

Análisis de Componentes Principales (PCA)

¿Qué es PCA?

El PCA es una técnica de reducción de dimensionalidad que transforma variables correlacionadas en componentes principales no correlacionadas, maximizando la varianza a lo largo de cada nuevo eje y ordenando los componentes por varianza explicada.

Fundamento Matemático

Dado una matriz de datos X con n observaciones y p variables:

1. Centrar los datos: $X_{\text{centered}} = X - \bar{X}$
2. Calcular la matriz de covarianza: $S = \left(\frac{1}{n-1}\right) X_{\text{centered}}^T X_{\text{centered}}$
3. Encontrar valores propios λ_i y vectores propios v_i de S
4. Componentes principales: $PC_i = X_{\text{centered}} v_i$

Ejemplo de PCA

```
import numpy as np
from sklearn.decomposition import PCA
```

```
# Matriz de datos simple 3x2
X = np.array([[5, 3],
              [3, 1],
              [1, 3]])
```

```
# Aplicar PCA
```

```
pca = PCA()
X_transformed = pca.fit_transform(X)

# Resultados
eigenvalues = pca.explained_variance_
variance_ratio = pca.explained_variance_ratio_

print(f"Valores propios: {eigenvalues}")
print(f"PC1 explica {variance_ratio[0]:.1%} de la varianza")
```

Resultados Clave

- PC1 explica la mayor varianza (valor propio más alto)
- Los componentes están incorrelacionados por construcción
- Los datos originales pueden reconstruirse a partir de los componentes

Criterios de Retención de Componentes

- **Criterio de Kaiser:** Conservar componentes con $\lambda_i > 1$
- **Gráfico de sedimentación (scree plot):** Buscar el “codo” en el gráfico de valores propios
- **Varianza acumulada:** Retener suficientes componentes para varianza deseada (ej. 80%)
- **Análisis paralelo:** Comparar con valores propios de datos aleatorios

Análisis Factorial

¿Qué es el Análisis Factorial?

El análisis factorial asume que las variables observadas son combinaciones lineales de factores comunes (constructos latentes compartidos) y factores únicos (varianza específica de cada variable).

El Modelo Factorial Común

Para cada variable observada x_j :

$$x_j = \mu_j + \sum_{i=1}^m \lambda_{ji} f_i + \varepsilon_j$$

Donde:

- λ_{ji} : Carga factorial (correlación entre x_j y f_i)
- f_i : Factor común (variable latente)
- ε_j : Factor único (término de error)

Ejemplo de Análisis Factorial

```
import numpy as np
from factor_analyzer import FactorAnalyzer

# Matriz de correlación
R = np.array([[1.00, 0.60, 0.48],
              [0.60, 1.00, 0.72],
              [0.48, 0.72, 1.00]])
```

```
# Realizar Análisis Factorial
fa = FactorAnalyzer(n_factors=1, rotation=None)
fa.fit(R)

# Resultados
loadings = fa.loadings_
communalities = fa.get_communalities()
uniqueness = fa.get_uniquenesses()

print(f"Cargas factoriales:\n{loadings}")
print(f"Comunalidades: {communalities}")
print(f"Unicidades: {uniqueness}")
```

Conceptos Clave

- **Cargas:** Correlaciones entre variables y factores
- **Comunalidades:** Varianza explicada por factores comunes
- **Unicidades:** Varianza específica de cada variable

Rotación Factorial

Método	Descripción	Asunción	Caso de Uso
Varimax	Rotación ortogonal	Factores no correlacionados	Estructura simple
Promax	Rotación oblicua	Factores pueden correlacionarse	Modelos realistas
Quartimax	Simplificar variables	Equilibrar factores	Uso general
Oblimin	Rotación oblicua	Correlación flexible	Datos complejos

¿Por qué Rotar?

- Mejorar la interpretabilidad de las cargas factoriales
- Lograr “estructura simple” (variables que cargan altamente en pocos factores)
- Diferentes rotaciones pueden revelar diferentes interpretaciones sustantivas

Comparación: PCA vs Análisis Factorial

Diferencias Principales

Aspecto	PCA	Análisis Factorial
Objetivo	Maximización de varianza	Identificación de constructos latentes
Modelo	Basado en datos	Basado en teoría
Componentes/Factores	Toda la varianza	Solo varianza común
Rotación	No típicamente usada	Esencial para interpretación
Asunciones	Mínimas	Normalidad multivariada

Aspecto	PCA	Análisis Factorial
Estimación	Descomposición en valores propios	Máxima verosimilitud/FAF
Salida	Componentes principales	Cargas factoriales

¿Cuándo Usar Cada Método?

Usar PCA cuando:

- La reducción de datos es el objetivo principal
- No existe un modelo teórico
- Toda la varianza es de interés
- La predicción es el objetivo
- Se necesita visualización de datos

Usar Análisis Factorial cuando:

- Se buscan identificar constructos latentes
- El análisis es guiado por teoría
- Se desarrolla modelo de medición
- Se busca entender relaciones entre variables
- Se realiza validación/desarrollo de escalas

Flujo de Análisis Completo

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from factor_analyzer import FactorAnalyzer
from factor_analyzer.factor_analyzer import calculate_kmo,
calculate_bartlett_sphericity

# 1. Cargar y preparar datos
X = np.random.randn(100, 5) # Tus datos aquí

# 2. Estandarizar
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. Verificar idoneidad para AF
kmo_all, kmo_model = calculate_kmo(X_scaled)
chi_square_value, p_value = calculate_bartlett_sphericity(X_scaled)

print(f"KMO: {kmo_model:.3f} (>0.6 es bueno)")
print(f"Prueba de Bartlett p-valor: {p_value:.3f} (<0.05 es bueno)")

# 4. Determinar número de factores
pca = PCA()
pca.fit(X_scaled)
```

```
eigenvalues = pca.explained_variance_
n_factors = sum(eigenvalues > 1) # Criterio de Kaiser

# 5. Realizar Análisis Factorial
fa = FactorAnalyzer(n_factors=n_factors, rotation='varimax')
fa.fit(X_scaled)

# 6. Comparar resultados
loadings = fa.loadings_
communalities = fa.get_communalities()
variance_explained = fa.get_factor_variance()
```

Aplicaciones y Mejores Prácticas

Aplicaciones en el Mundo Real

Finanzas

- Factores de riesgo en mercados bursátiles
- Optimización de portafolios
- Modelos de calificación crediticia

Salud

- Encuestas de satisfacción del paciente
- Agrupamiento de síntomas
- Medidas de calidad de vida

Mercadotecnia

- Segmentación de clientes
- Estudios de percepción de marca
- Análisis de atributos de producto

Ciencias Sociales

- Evaluación de personalidad
- Medición de actitudes
- Pruebas educativas

Mejores Prácticas

Preparación de Datos

- Asegurar tamaño de muestra adecuado (5-10 observaciones por variable)
- Verificar normalidad multivariada
- Manejar datos faltantes apropiadamente
- Considerar estandarización de variables

Selección de Modelo

- Usar pruebas KMO y Bartlett para idoneidad del AF
- Comparar múltiples criterios de retención de factores
- Considerar tanto rotaciones ortogonales como oblicuas
- Validar resultados con validación cruzada

Interpretación

- Enfocarse en el significado sustantivo de los factores
- Usar cargas factoriales > 0.3 para interpretación
- Considerar correlaciones entre factores en rotaciones oblicuas
- Validar con criterios externos cuando sea posible

Conclusión

Puntos Clave

- **PCA y Análisis Factorial** sirven propósitos diferentes pero complementarios
- Elegir método basado en objetivos de investigación y características de los datos
- Siempre validar asunciones e interpretar resultados sustantivamente
- El software moderno facilita la implementación

Próximos Pasos

- Practicar con conjuntos de datos reales
- Comparar PCA y AF en los mismos datos
- Explorar técnicas avanzadas (AF confirmatorio, modelado de ecuaciones estructurales)
- Aplicar a sus propias preguntas de investigación

Referencias

Referencias Principales

- **Fabrigar, L. R., & Wegener, D. T.** (2011). Exploratory Factor Analysis. Oxford University Press.
- **Hair, J. F., et al.** (2019). Multivariate Data Analysis. Cengage Learning.
- **Jolliffe, I. T.** (2002). Principal Component Analysis. Springer.
- **Tabachnick, B. G., & Fidell, L. S.** (2013). Using Multivariate Statistics. Pearson.

Recursos de Software

- Python: scikit-learn, factor-analyzer, statsmodels
- R: psych, FactoMineR, lavaan
- SPSS: Módulo de Análisis Factorial
- SAS: PROC FACTOR

Recursos en Línea

- Grupo de Consultoría Estadística UCLA
- Canal de YouTube StatQuest
- Artículos de Towards Data Science

Introduction to Dimensionality Reduction

Why Dimensionality Reduction?

- Handle high-dimensional data effectively
- Reduce computational complexity
- Remove noise and redundancy
- Improve model interpretability
- Enable data visualization

Two Main Approaches

- **Principal Component Analysis (PCA)**: Data-driven, variance maximization
- **Factor Analysis (FA)**: Model-based, latent variable identification

Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

What is PCA?

PCA is a dimensionality reduction technique that:

- Transforms correlated variables into uncorrelated principal components
- Maximizes variance along each new axis
- Orders components by explained variance

Mathematical Foundation

Given data matrix \mathbf{X} with n observations and p variables:

1. Center the data: $\mathbf{X}_{\text{centered}} = \mathbf{X} - \overline{\mathbf{X}}$
2. Compute covariance matrix: $\mathbf{S} = \left(\frac{1}{n-1}\right)\mathbf{X}_{\text{centered}}^T \mathbf{X}_{\text{centered}}$
3. Find eigenvalues λ_i and eigenvectors \mathbf{v}_i of \mathbf{S}
4. Principal components: $\mathbf{PC}_i = \mathbf{X}_{\text{centered}} \mathbf{v}_i$

PCA Example

```
import numpy as np
from sklearn.decomposition import PCA

# Simple 3x2 data matrix
X = np.array([[5, 3],
              [3, 1],
              [1, 3]])

# Apply PCA
pca = PCA()
X_transformed = pca.fit_transform(X)

# Results
eigenvalues = pca.explained_variance_
variance_ratio = pca.explained_variance_ratio_

print(f"Eigenvalues: {eigenvalues}")
print(f"PC1 explains {variance_ratio[0]:.1%} of variance")
```

Key Results

- PC1 explains most variance (higher eigenvalue)
- Components are uncorrelated by construction
- Original data can be reconstructed from components

Component Retention in PCA

Scree plot would be displayed here

Retention Criteria

- **Kaiser criterion:** Keep components with $\lambda_i > 1$
- **Scree plot:** Look for “elbow” in eigenvalue plot
- **Cumulative variance:** Retain enough components for desired variance (e.g., 80%)
- **Parallel analysis:** Compare with random data eigenvalues

Factor Analysis

Factor Analysis

What is Factor Analysis?

Factor Analysis assumes observed variables are linear combinations of:

- **Common factors:** Shared underlying constructs
- **Unique factors:** Variable-specific variance

The Common Factor Model

For each observed variable x_j :

$$x_j = \mu_j + \sum_{i=1}^m \lambda_{ji} f_i + \varepsilon_j$$

Where:

- λ_{ji} : Factor loading (correlation between x_j and f_i)
- f_i : Common factor (latent variable)
- ε_j : Unique factor (error term)

Factor Analysis Example

```
import numpy as np
from factor_analyzer import FactorAnalyzer

# Correlation matrix
R = np.array([[1.00, 0.60, 0.48],
              [0.60, 1.00, 0.72],
              [0.48, 0.72, 1.00]])

# Perform Factor Analysis
fa = FactorAnalyzer(n_factors=1, rotation=None)
fa.fit(R)

# Results
loadings = fa.loadings_
communalities = fa.get_communalities()
uniqueness = fa.get_uniquenesses()

print(f"Factor loadings:\\n{loadings}")
print(f"Communalities: {communalities}")
print(f"Uniqueness: {uniqueness}")
```

Key Concepts

- **Loadings:** Correlations between variables and factors
- **Communalities:** Variance explained by common factors
- **Uniqueness:** Variable-specific variance

Factor Rotation

Method	Description	Assumption	Use Case
Varimax	Orthogonal rotation	Factors uncorrelated	Simple structure
Promax	Oblique rotation	Factors may correlate	Realistic models
Quartimax	Simplify variables	Balance factors	General use
Oblimin	Oblique rotation	Flexible correlation	Complex data

Why Rotate?

- Improve interpretability of factor loadings
- Achieve “simple structure” (variables load highly on few factors)
- Different rotations may reveal different substantive interpretations

PCA vs Factor Analysis

PCA vs Factor Analysis

Key Differences

Aspect	PCA	Factor Analysis
Goal	Variance maximization	Latent construct identification
Model	Data-driven	Theory-driven
Components/Factors	All variance	Common variance only
Rotation	Not typically used	Essential for interpretation
Assumptions	Minimal	Multivariate normality
Estimation	Eigenvalue decomposition	Maximum likelihood/PAF
Output	Principal components	Factor loadings

When to Use Each Method

Use PCA when:

- Data reduction is primary goal
- No theoretical model exists
- All variance is of interest
- Prediction is the objective
- Data visualization needed

Use Factor Analysis when:

- Identifying latent constructs
- Theory-driven analysis
- Measurement model development
- Understanding variable relationships
- Scale development/validation

Complete Analysis Workflow

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from factor_analyzer import FactorAnalyzer
from factor_analyzer.factor_analyzer import calculate_kmo,
calculate_bartlett_sphericity

# 1. Load and prepare data
X = np.random.randn(100, 5) # Your data here

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

# 3. Check suitability for FA
kmo_all, kmo_model = calculate_kmo(X_scaled)
chi_square_value, p_value = calculate_bartlett_sphericity(X_scaled)

print(f"KMO: {kmo_model:.3f} (>0.6 is good)")
print(f"Bartlett's test p-value: {p_value:.3f} (<0.05 is good)")

# 4. Determine number of factors
pca = PCA()
pca.fit(X_scaled)
eigenvalues = pca.explained_variance_
n_factors = sum(eigenvalues > 1) # Kaiser criterion

# 5. Perform Factor Analysis
fa = FactorAnalyzer(n_factors=n_factors, rotation='varimax')
fa.fit(X_scaled)

# 6. Compare results
loadings = fa.loadings_
communalities = fa.get_communalities()
variance_explained = fa.get_factor_variance()
```

When to Use Each Method

Use PCA when:

- Data reduction is primary goal
- No theoretical model exists
- All variance is of interest
- Prediction is the objective
- Data visualization needed

Use Factor Analysis when:

- Identifying latent constructs
- Theory-driven analysis
- Measurement model development
- Understanding variable relationships
- Scale development/validation

Complete Analysis Workflow

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from factor_analyzer import FactorAnalyzer
from factor_analyzer.factor_analyzer import calculate_kmo,
calculate_bartlett_sphericity

# 1. Load and prepare data
X = np.random.randn(100, 5) # Your data here

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

# 3. Check suitability for FA
kmo_all, kmo_model = calculate_kmo(X_scaled)
chi_square_value, p_value = calculate_bartlett_sphericity(X_scaled)

print(f"KMO: {kmo_model:.3f} (>0.6 is good)")
print(f"Bartlett's test p-value: {p_value:.3f} (<0.05 is good)")

# 4. Determine number of factors
pca = PCA()
pca.fit(X_scaled)
eigenvalues = pca.explained_variance_
n_factors = sum(eigenvalues > 1) # Kaiser criterion

# 5. Perform Factor Analysis
fa = FactorAnalyzer(n_factors=n_factors, rotation='varimax')
fa.fit(X_scaled)

# 6. Compare results
loadings = fa.loadings_
communalities = fa.get_communalities()
variance_explained = fa.get_factor_variance()
```

Applications and Examples

Real-World Applications

Finance

- Stock market risk factors
- Portfolio optimization
- Credit scoring models

Healthcare

- Patient satisfaction surveys
- Symptom clustering
- Quality of life measures

Marketing

- Customer segmentation
- Brand perception studies
- Product attribute analysis

Social Science

- Personality assessment
- Attitude measurement
- Educational testing

Best Practices

Data Preparation

- Ensure adequate sample size (5-10 observations per variable)
- Check for multivariate normality
- Handle missing data appropriately
- Consider variable standardization

Model Selection

- Use KMO and Bartlett's test for FA suitability
- Compare multiple factor retention criteria
- Consider both orthogonal and oblique rotations
- Validate results with cross-validation

Interpretation

- Focus on substantive meaning of factors
- Use factor loadings > 0.3 for interpretation
- Consider factor correlations in oblique rotations
- Validate with external criteria when possible

Conclusion

Key Takeaways

- **PCA** and **Factor Analysis** serve different but complementary purposes
- Choose method based on research objectives and data characteristics
- Always validate assumptions and interpret results substantively
- Modern software makes implementation straightforward

Next Steps

- Practice with real datasets
- Compare PCA and FA on same data
- Explore advanced techniques (confirmatory FA, structural equation modeling)
- Apply to your own research questions

Questions?

References

Key References

- **Fabrigar, L. R., & Wegener, D. T.** (2011). Exploratory Factor Analysis. Oxford University Press.
- **Hair, J. F., et al.** (2019). Multivariate Data Analysis. Cengage Learning.
- **Jolliffe, I. T.** (2002). Principal Component Analysis. Springer.
- **Tabachnick, B. G., & Fidell, L. S.** (2013). Using Multivariate Statistics. Pearson.

Software Resources

- Python: scikit-learn, factor-analyzer, statsmodels
- R: psych, FactoMineR, lavaan
- SPSS: Factor Analysis module
- SAS: PROC FACTOR

Online Resources

- UCLA Statistical Consulting Group
- StatQuest YouTube channel
- Towards Data Science articles