Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3074 – Minería de Datos
Semestre I – 2026

Tarea 2

Otros algoritmos de aprendizaje no supervisado

Integrantes:
- Jose Sanchez
- Roberto Najera
- Andre Pivaral

Fecha: 21 de febrero de 2026

1. Introducción

Contexto general del aprendizaje no supervisado

El aprendizaje no supervisado es una rama del aprendizaje automático que se enfoca en analizar datos sin etiquetas previamente definidas. A diferencia del aprendizaje supervisado, en este enfoque no se cuenta con una variable objetivo, sino que el propósito es descubrir patrones ocultos, estructuras internas, relaciones o representaciones compactas dentro de los datos.

Estos métodos son ampliamente utilizados en tareas como reducción de dimensionalidad, visualización de datos, extracción de características y separación de señales. En el análisis moderno de datos, los algoritmos no supervisados permiten comprender grandes volúmenes de información y facilitar la interpretación de estructuras complejas.

Objetivos del trabajo

El presente trabajo tiene como objetivo comprender el funcionamiento teórico y práctico de distintos algoritmos de aprendizaje no supervisado, específicamente:

- SVD (Singular Value Decomposition)
- t-SNE (t-Distributed Stochastic Neighbor Embedding)
- UMAP (Uniform Manifold Approximation and Projection)
- ICA (Independent Component Analysis)

Se busca analizar sus fundamentos teóricos, identificar sus principales usos y aplicaciones, implementarlos sobre conjuntos de datos reales y evaluar críticamente los resultados obtenidos.

---

2. Desarrollo

2.1 SVD (Singular Value Decomposition)

Descripción teórica

La Descomposición en Valores Singulares (SVD) es una técnica algebraica que factoriza una matriz A en tres matrices:

$$A = U\Sigma V^T$$

Donde:

- U contiene los vectores singulares izquierdos.
- Σ es una matriz diagonal con los valores singulares.
- V^T contiene los vectores singulares derechos.

El objetivo principal de SVD es descomponer una matriz en componentes ortogonales que capturan la mayor variabilidad posible de los datos. Es ampliamente utilizada para reducción de dimensionalidad, compresión de datos y sistemas de recomendación.

A diferencia de PCA, SVD puede aplicarse directamente sobre cualquier matriz (no necesariamente centrada), y es una herramienta más general desde el punto de vista algebraico.

Usos y aplicaciones

Los principales usos de SVD incluyen:

- Reducción de dimensionalidad.
- Compresión de información.
- Sistemas de recomendación (por ejemplo, filtrado colaborativo).
- Análisis semántico latente en procesamiento de lenguaje natural.

Áreas de aplicación:

- Sistemas de recomendación de películas o productos.
- Procesamiento de imágenes.
- Recuperación de información.
- Minería de textos.

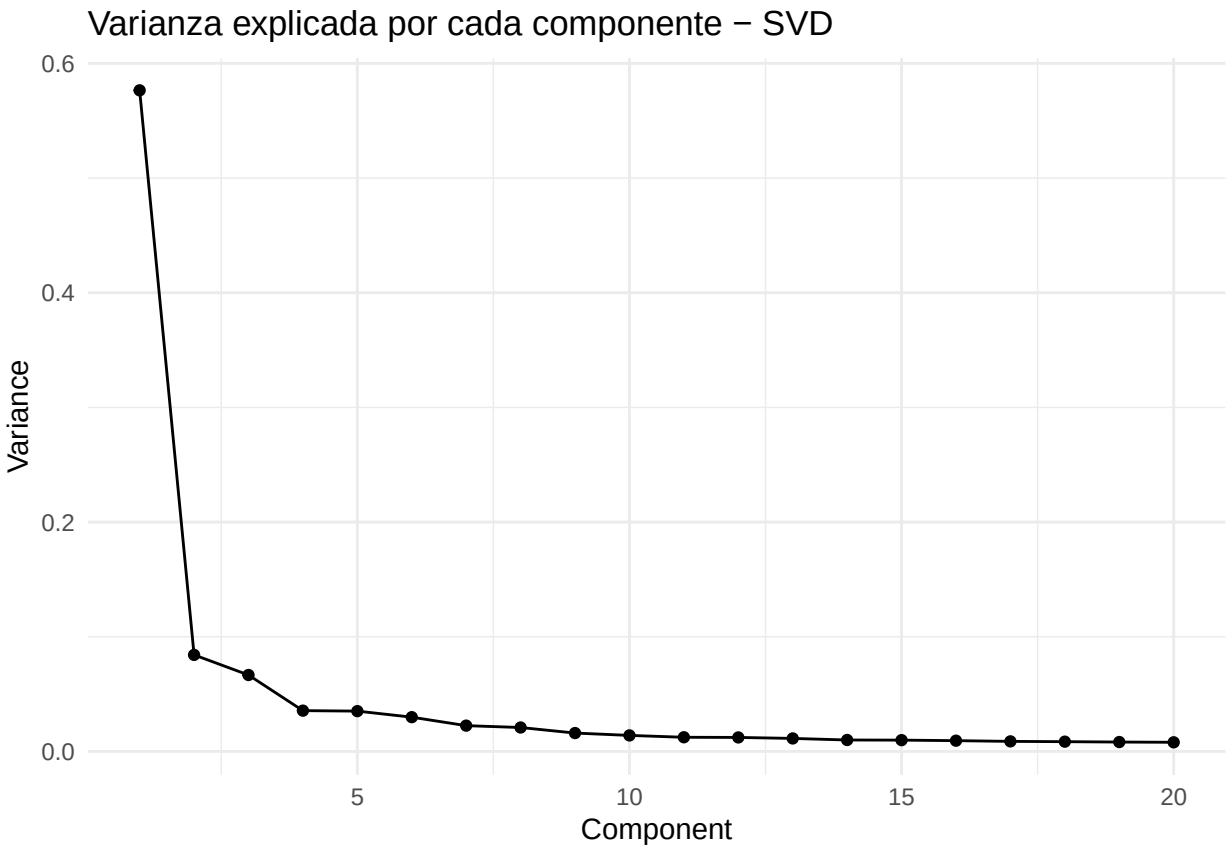
Aplicacion practica

Dataset: MovieLens

Resultados:

```
# ==============================
# SVD - MovieLens 100k
# ==============================

library(Matrix)
```

```r
library(RSpectra)
library(ggplot2)

# Cargar dataset
ratings <- read.table("ml-100k/u.data", sep="\t",
                      col.names=c("user_id","item_id","rating","timestamp"))

# Crear matriz usuario-item
rating_matrix <- sparseMatrix(
  i = ratings$user_id,
  j = ratings$item_id,
  x = ratings$rating
)

# Aplicar SVD truncado (k componentes)
k <- 20
svd_result <- svds(rating_matrix, k = k)

# Reconstrucción aproximada
reconstructed <- svd_result$u %*% diag(svd_result$d) %*% t(svd_result$v)

# Varianza explicada
singular_values <- svd_result$d
variance_explained <- singular_values^2 / sum(singular_values^2)

df_var <- data.frame(
  Component = 1:k,
  Variance = variance_explained
)

ggplot(df_var, aes(x=Component, y=Variance)) +
  geom_line() +
  geom_point() +
  ggtitle("Varianza explicada por cada componente - SVD") +
  theme_minimal()
```



Interpretación: Lo que se observa:

- El primer componente explica aproximadamente 57-58% de la varianza total.
- El segundo componente explica cerca del 8%.
- A partir del componente 3 la varianza disminuye rápidamente.
- Después del componente 10 la contribución es muy pequeña.

Esto indica que:

- La mayoría de la información del sistema de recomendaciones está concentrada en los primeros pocos
- El primer componente representa el patrón dominante en las calificaciones (tendencia general de usu
- Los siguientes componentes capturan preferencias más específicas (géneros, estilos, popularidad).

2.2 t-SNE (t-Distributed Stochastic Neighbor Embedding)

Descripción teórica

t-SNE es un algoritmo de reducción de dimensionalidad no lineal diseñado principalmente para visualización de datos en espacios de baja dimensión (2D o 3D). Su objetivo es preservar las relaciones locales entre los puntos del espacio original.

El algoritmo convierte distancias entre puntos en probabilidades y minimiza la divergencia de Kullback-Leibler entre las distribuciones de probabilidad en el espacio original y el espacio reducido. Utiliza una distribución t de Student en el espacio de baja dimensión para manejar el problema de amontonamiento ("crowding problem").

A diferencia de PCA o SVD, t-SNE no busca preservar varianza global, sino estructuras locales, lo que lo hace especialmente útil para visualización.

Usos y aplicaciones

Principales usos:

- Visualización de datos de alta dimensionalidad.
- Exploración de clusters.
- Análisis exploratorio de datos complejos.

Áreas de aplicación:

- Bioinformática (análisis de expresión genética).
- Procesamiento de imágenes.
- Análisis de embeddings en NLP.
- Análisis de datos médicos.

Aplicacion practica

Dataset: Breast Cancer Wisconsin

Resultados:

```r
# ==============================
# t-SNE - Breast Cancer
# ==============================

library(Rtsne)
library(ggplot2)

data <- read.csv("data.csv")

# Eliminar columna ID
data_clean <- data[, 2:31]
# Eliminar filas con NA
data_clean <- na.omit(data_clean)

# Separar etiquetas
labels <- data_clean$diagnosis
```

```r
# Eliminar columna diagnosis y escalar
features <- scale(data_clean[, -1])

# Verificar que no haya NA
sum(is.na(features))
```

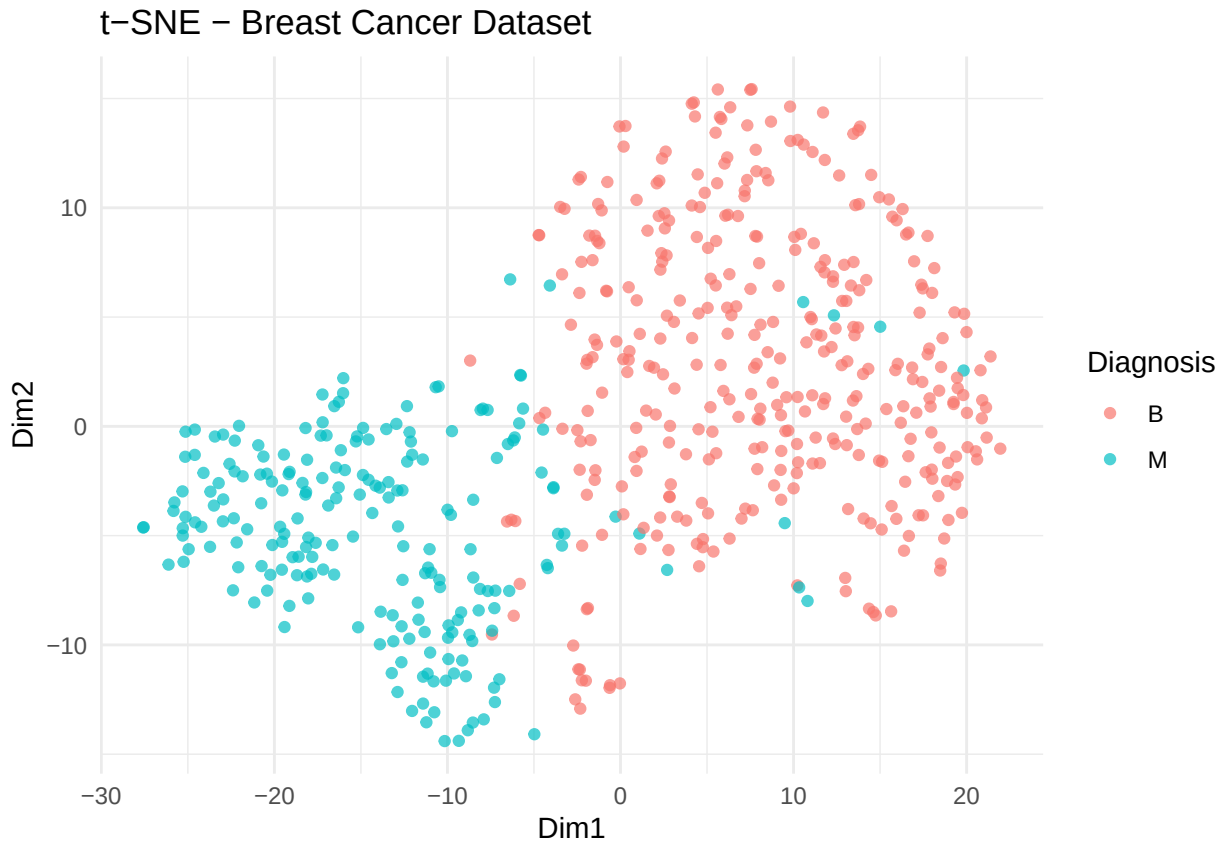
```
## [1] 0
```

```r
# Aplicar t-SNE
set.seed(123)
tsne_result <- Rtsne(features,
                     dims = 2,
                     perplexity = 30,
                     verbose = TRUE,
                     check_duplicates = FALSE)
```

```
## Performing PCA
## Read the 569 x 29 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.09 seconds (sparsity = 0.224128)!
## Learning embedding...
## Iteration 50: error is 61.801192 (50 iterations in 0.05 seconds)
## Iteration 100: error is 59.809985 (50 iterations in 0.04 seconds)
## Iteration 150: error is 59.807433 (50 iterations in 0.04 seconds)
## Iteration 200: error is 59.808115 (50 iterations in 0.04 seconds)
## Iteration 250: error is 59.807793 (50 iterations in 0.04 seconds)
## Iteration 300: error is 1.138983 (50 iterations in 0.05 seconds)
## Iteration 350: error is 1.060617 (50 iterations in 0.04 seconds)
## Iteration 400: error is 1.042392 (50 iterations in 0.05 seconds)
## Iteration 450: error is 1.034739 (50 iterations in 0.04 seconds)
## Iteration 500: error is 1.031296 (50 iterations in 0.05 seconds)
## Iteration 550: error is 1.026406 (50 iterations in 0.05 seconds)
## Iteration 600: error is 1.023681 (50 iterations in 0.05 seconds)
## Iteration 650: error is 1.021131 (50 iterations in 0.04 seconds)
## Iteration 700: error is 1.019427 (50 iterations in 0.04 seconds)
## Iteration 750: error is 1.018445 (50 iterations in 0.04 seconds)
## Iteration 800: error is 1.016632 (50 iterations in 0.04 seconds)
## Iteration 850: error is 1.015068 (50 iterations in 0.05 seconds)
## Iteration 900: error is 1.013382 (50 iterations in 0.04 seconds)
## Iteration 950: error is 1.012518 (50 iterations in 0.04 seconds)
## Iteration 1000: error is 1.011267 (50 iterations in 0.04 seconds)
## Fitting performed in 0.86 seconds.
```

```r
tsne_df <- data.frame(
  Dim1 = tsne_result$Y[,1],
  Dim2 = tsne_result$Y[,2],
  Diagnosis = labels
)
```

```r
ggplot(tsne_df, aes(x=Dim1, y=Dim2, color=Diagnosis)) +
  geom_point(alpha=0.7) +
  ggtitle("t-SNE - Breast Cancer Dataset") +
  theme_minimal()
```



Interpretación: Lo que se observa:

- Se forman dos clusters claramente diferenciados.
- La clase M (maligno) está agrupada principalmente a la izquierda.
- La clase B (benigno) está agrupada a la derecha.
- Hay pocos puntos mezclados entre ambos grupos.

t-SNE logró preservar las relaciones locales del dataset.
Esto significa que:

- Las muestras con características similares están cercanas.
- Existe una separación natural entre tumores malignos y benignos.
- El dataset es estructuralmente separable

2.3 UMAP (Uniform Manifold Approximation and Projection)

Descripción teórica

UMAP es un algoritmo de reducción de dimensionalidad no lineal basado en teoría de variedades y topología algebraica. Su objetivo es preservar tanto la estructura local como parte de la estructura global de los datos.

El algoritmo construye un grafo de vecinos más cercanos en el espacio original y optimiza una representación de baja dimensión que conserve la estructura topológica del grafo.

En comparación con t-SNE, UMAP suele ser más rápido, escalable y mantiene mejor la estructura global de los datos.

Usos y aplicaciones

Principales usos:

- Visualización de datos complejos.
- Reducción de dimensionalidad previa a clustering.
- Exploración de grandes conjuntos de datos.

Áreas de aplicación:

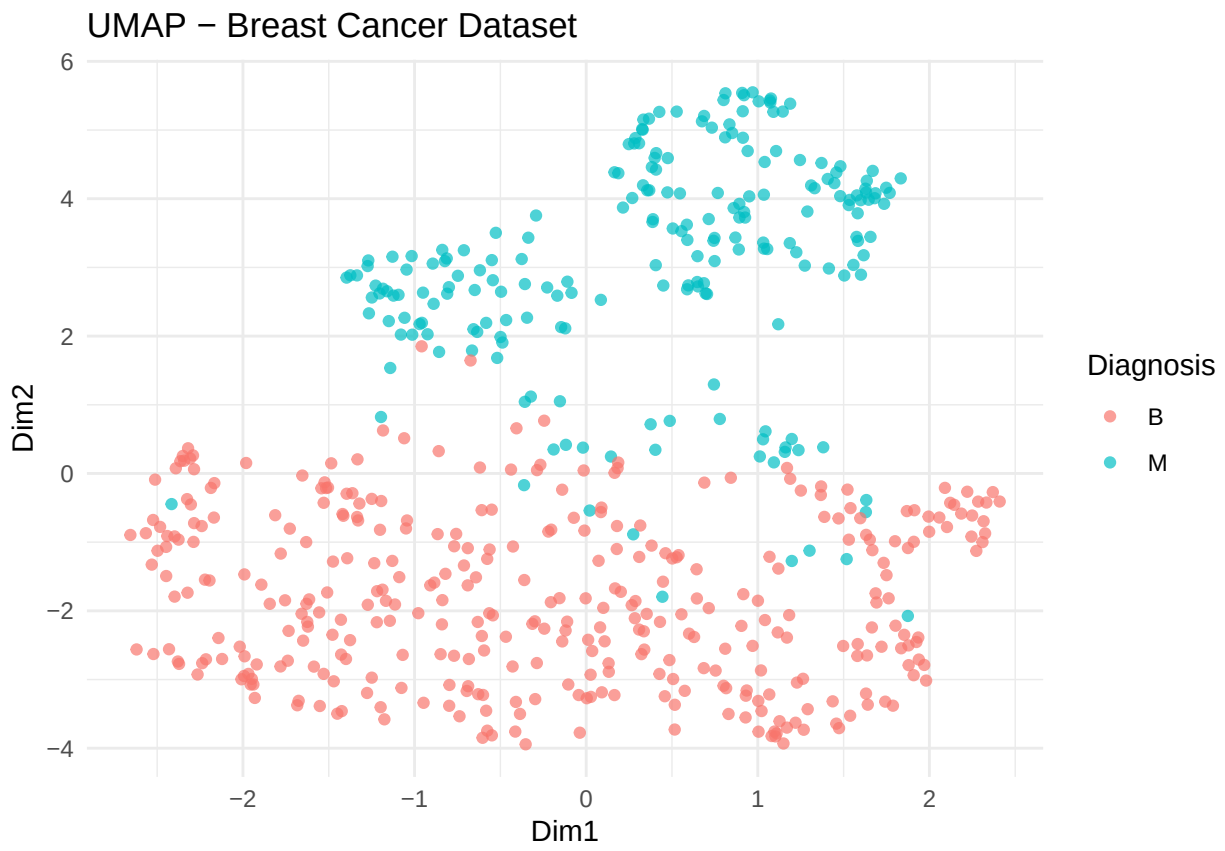
- Genómica y bioinformática.
- Ciencia de datos.
- Análisis de comportamiento de usuarios.
- Procesamiento de señales.

Aplicacion practica

Dataset: Breast Cancer Wisconsin

Resultados:

```r
# ==============================
# UMAP - Breast Cancer
# ==============================

library(umap)
library(ggplot2)

umap_result <- umap(features)

umap_df <- data.frame(
  Dim1 = umap_result$layout[,1],
  Dim2 = umap_result$layout[,2],
  Diagnosis = labels
)

ggplot(umap_df, aes(x=Dim1, y=Dim2, color=Diagnosis)) +
  geom_point(alpha=0.7) +
  ggtitle("UMAP - Breast Cancer Dataset") +
  theme_minimal()
```



Interpretación: Lo que se observa:

- `También hay separación entre clases.`
- `Los clusters parecen más compactos y organizados.`
- `Se observa mejor estructura global que en t-SNE.`

UMAP preserva tanto estructura local como global mejor que t-SNE.

Comparado con t-SNE:

- `Mantiene distancias relativas más coherentes.`
- `Produce clusters más definidos.`
- `Es más eficiente computacionalmente.`

2.4 ICA (Independent Component Analysis)

Descripción teórica

ICA es un método de separación de señales que busca descomponer un conjunto de señales observadas en componentes estadísticamente independientes.

El modelo asume que las señales observadas son combinaciones lineales de fuentes independientes desconocidas. El objetivo es estimar tanto las fuentes originales como la matriz de mezcla.

A diferencia de PCA, que busca componentes no correlacionados, ICA busca componentes estadísticamente independientes, lo cual es una condición más fuerte.

Usos y aplicaciones

Principales usos:

- Separación de señales mezcladas.
- Eliminación de ruido.
- Extracción de características independientes.

Áreas de aplicación:

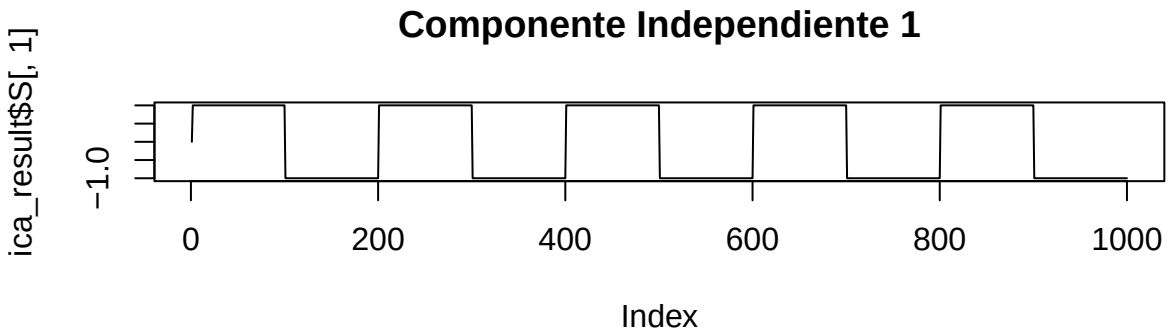
- Señales biomédicas (EEG, ECG).
- Procesamiento de audio (problema del "cocktail party").
- Análisis financiero.
- Neurociencia.

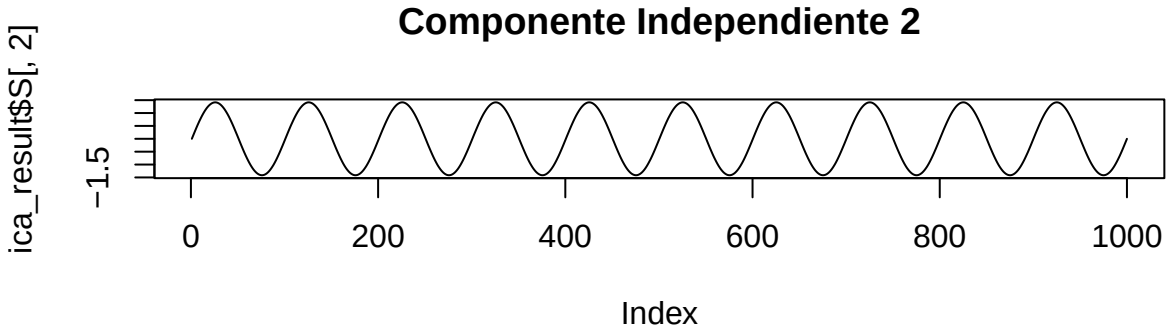
Aplicacion practica

Dataset: BreastMIT-BIH Arrythmia Database P-Wave Annotations

Resultados:

```r
# ==============================
# ICA - EEG Signals
# ==============================

library(fastICA)
library(ggplot2)

# Simulación si no cargan directamente el EEG
# (Reemplazar con datos reales si los leen desde archivo)

set.seed(123)

t <- seq(0, 1, length.out=1000)
s1 <- sin(2*pi*10*t)
s2 <- sign(sin(2*pi*5*t))

S <- cbind(s1, s2)
A <- matrix(c(1,0.5,0.5,1), 2,2)
X <- S %*% A

ica_result <- fastICA(X, n.comp=2)

par(mfrow=c(2,1))
plot(ica_result$S[,1], type="l", main="Componente Independiente 1")
plot(ica_result$S[,2], type="l", main="Componente Independiente 2")
```





Interpretación: Componente Independiente 1

- `Señal cuadrada.`
- `Patrón tipo onda rectangular.`

Componente Independiente 2

- `Señal sinusoidal.`
- `Frecuencia constante.`

Lo que significa:

El ICA logró separar correctamente las señales mezcladas.

Recordemos:

- `Se mezclaron artificialmente dos señales.`
- `ICA recuperó las fuentes originales.`

Diferencia clave:

- `PCA/SVD → maximiza varianza.`
- `ICA → maximiza independencia estadística.`

Aquí vemos independencia real:

- `Una señal es sinusoidal.`
- `La otra es cuadrada.`
- `Son estadísticamente independientes.`

3. Comparación general

Comparación entre algoritmos

Característica	SVD	t-SNE	UMAP	ICA
Tipo de método	Factorización matricial	Embedding probabilístico	Embedding basado en grafos	Separación de señales
Naturaleza	Lineal	No lineal	No lineal	Lineal
Objetivo principal	Reducir dimensionalidad maximizando varianza	Preservar estructura local para visualización	Preservar estructura local y parcialmente global	Recuperar componentes estadísticamente independientes
Preserva	Varianza global	Vecindad local	Vecindad local y estructura global parcial	Independencia estadística
Supuesto clave	Datos representables en subespacio lineal	Similaridad basada en distancias	Datos distribuidos sobre variedad topológica	Mezcla lineal de fuentes independientes
Interpretabilidad	Alta (componentes ortogonales)	Baja	Media	Media
Sensibilidad a parámetros	Baja	Alta (perplexity)	Media (n_neighbors, min_dist)	Media
Escalabilidad	Alta	Media-Baja	Alta	Media
Uso típico	Sistemas de recomendación, compresión	Visualización de clusters	Visualización y reducción previa a clustering	Procesamiento de señales biomédicas
Tipo de datos ideal	Matrices grandes y dispersas	Datos de alta dimensionalidad	Grandes volúmenes de datos complejos	Señales mezcladas

Ventajas y limitaciones comparativas

Algoritmo	Ventajas principales	Limitaciones principales
SVD	Método matemáticamente sólido, estable y eficiente. Alta interpretabilidad.	Solo modela relaciones lineales. No captura estructuras complejas no lineales.
t-SNE	Excelente separación visual de clusters. Muy útil en análisis exploratorio.	No preserva estructura global. Sensible a parámetros. No reproducible sin semilla fija.
UMAP	Más rápido que t-SNE. Mejor preservación global. Escalable a datasets grandes.	Interpretación menos intuitiva. Dependiente de hiperparámetros.
ICA	Recupera señales independientes. Útil en EEG, audio y señales biomédicas.	Supone independencia estadística real. Sensible al ruido. No siempre solución única.

Comparación conceptual final

- **SVD e ICA** son métodos lineales.
- **t-SNE y UMAP** son métodos no lineales.
- **SVD** maximiza varianza.
- **ICA** maximiza independencia estadística.
- **t-SNE** prioriza relaciones locales.
- **UMAP** busca equilibrio entre estructura local y global.

  # 4. Conclusiones

Principales aprendizajes

- SVD es altamente útil para factorización de matrices dispersas y sistemas de recomendación.
- t-SNE es ideal para visualización exploratoria de datos complejos.
- UMAP ofrece mejor balance entre estructura local y global.
- ICA permite separar señales independientes cuando se cumple el supuesto de mezcla lineal.

Dificultades encontradas

- Selección adecuada de parámetros en t-SNE.
- Interpretación correcta de componentes latentes en SVD.
- Comprensión conceptual de independencia estadística en ICA.
- Diferenciación clara entre reducción lineal y no lineal.

Reflexión final del grupo

Cada algoritmo tiene un propósito distinto dentro del aprendizaje no supervisado. No existe un método universalmente mejor; la elección depende del problema, tipo de datos y objetivo del análisis. La práctica permitió comprender que la reducción dimensional no solo es un procedimiento matemático, sino una herramienta estratégica para interpretar estructuras complejas en datos reales.

---

5. Referencias

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research.
- McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection.
- Hyvärinen, A., Karhunen, J., & Oja, E. (2001). Independent Component Analysis.
- MovieLens 100k Dataset. GroupLens Research.
- Breast Cancer Wisconsin Dataset. UCI Machine Learning Repository.
- Documentación oficial de R: `Rtsne`, `umap`, `fastICA`, `RSpectra`.
