Aprendizaje Profundo (2024-2025)
Máster en Robótica e Inteligencia Artificial
Universidad de León

Práctica 4

Sheila Martínez Gómez
Alejandro Mayorga Caro
Ángel Morales Romero

15 de diciembre de 2024

Índice

1. Abstract

En esta práctica se implementan y comparan diferentes técnicas de reducción de dimensionalidad y clasificación utilizando un conjunto de datos sintético en $\mathbb{R}^{10}$. El objetivo es evaluar el rendimiento de las siguientes técnicas:

- Sammon Mapping
- PCA (Principal Component Analysis)
- LDA (Linear Discriminant Analysis)
- t-SNE (t-distributed Stochastic Neighbor Embedding)
- Redes Neuronales Encoder-Decoder

Además, se analiza el rendimiento de un clasificador basado en un Encoder-Decoder, congelando el modelo.

2. Generación del Dataset

El conjunto de datos consta de 3 grupos o esferas generadas aleatoriamente en un espacio de 2 dimensiones ($\mathbb{R}^2$), y giradas en un espacio de 10 dimensiones ($\mathbb{R}^{10}$). Las clases se etiquetarán como Clase 0, Clase 1 y Clase 2. Este conjunto fue generado con el siguiente código:

```
def generate_spheres(samples_per_class, dim=10):
    np.random.seed(0)
    centers = [np.zeros((1, dim)), np.ones((1, dim)) * 2, np.ones((1, dim)) * 4]
    data, labels = [], []
    for idx, center in enumerate(centers):
        sphere = np.random.normal(size=(samples_per_class, dim)) + center
        data.append(sphere)
        labels.extend([idx] * samples_per_class)
    return np.vstack(data), np.array(labels)
```

El conjunto generado tiene un total de 600 muestras, distribuidas equitativamente entre las tres clases.

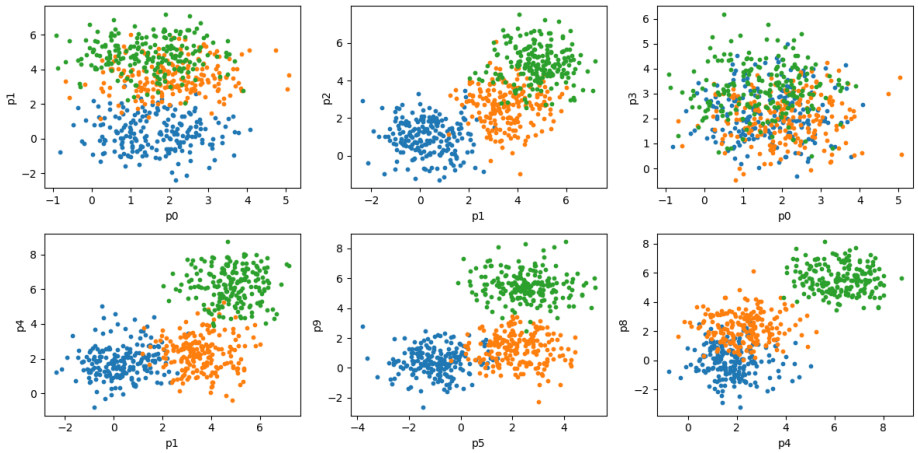

Figura 2.1: Proyección del conjunto de datos original en diferentes dimensiones.

3. Métodos de Reducción de Dimensionalidad

En esta sección se muestran los diferentes métodos de reducción de dimensionalidad que se han utilizado, con el objetivo de simplificar los datos, mejorar el rendimiento del modelo a aplicar y simplificar la visualización del *dataset*.
Los métodos aplicados para reducir las dimensiones de los datos son los siguientes:

3.1. Sammon Mapping

El Sammon Mapping es un método no lineal que minimiza las distorsiones en las distancias entre puntos al proyectarlos a un espacio de menor dimensión. La implementación utiliza la biblioteca de ‘sklearn‘:

```
from sklearn.manifold import MDS

sammon = MDS(n_components=2, metric=True)
X_sammon = sammon.fit_transform(X)
```

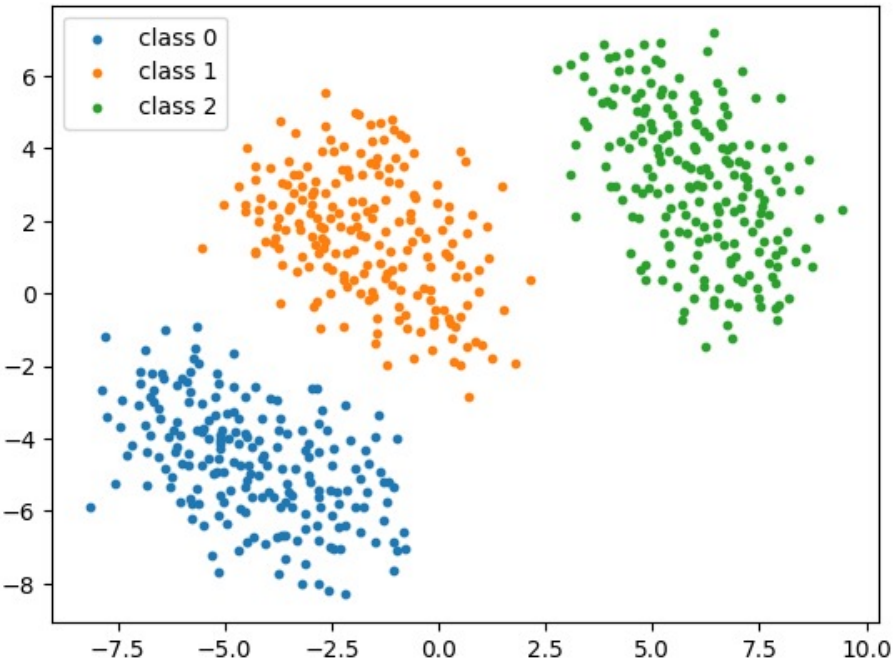

Figura 3.1: Proyección de los datos en 2D usando Sammon Mapping.

3.2. PCA

El análisis de componentes principales busca las direcciones de máxima varianza en los datos. En este caso, se seleccionaron las dos primeras componentes principales:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

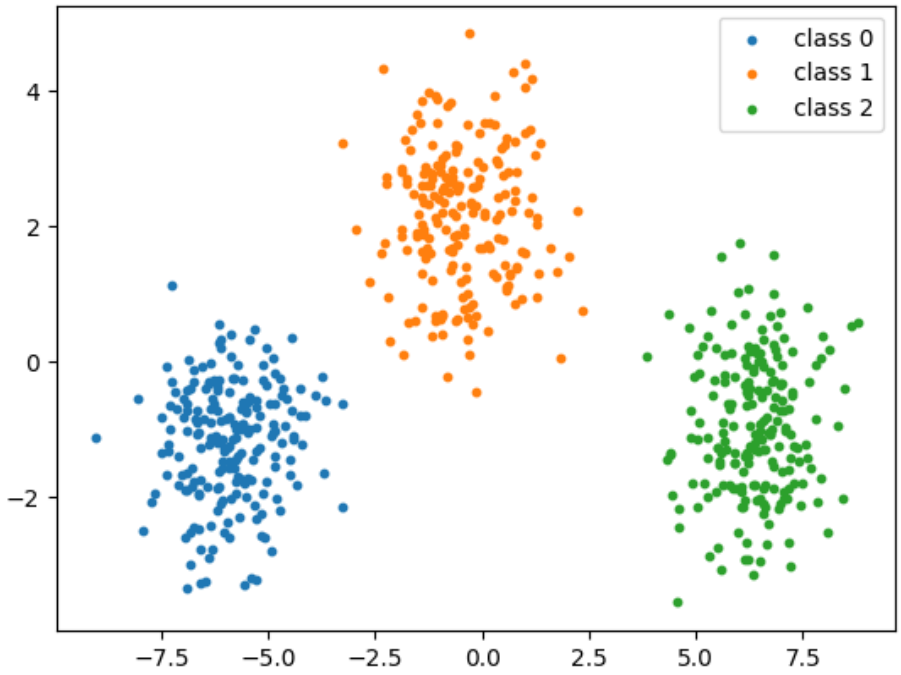

Figura 3.2: Proyección de los datos en 2D usando PCA.

3.3. LDA

El análisis discriminante lineal maximiza la separación entre clases al proyectar los datos a un espacio de menor dimensión:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=2)
X_lda = lda.fit_transform(X, y)
```

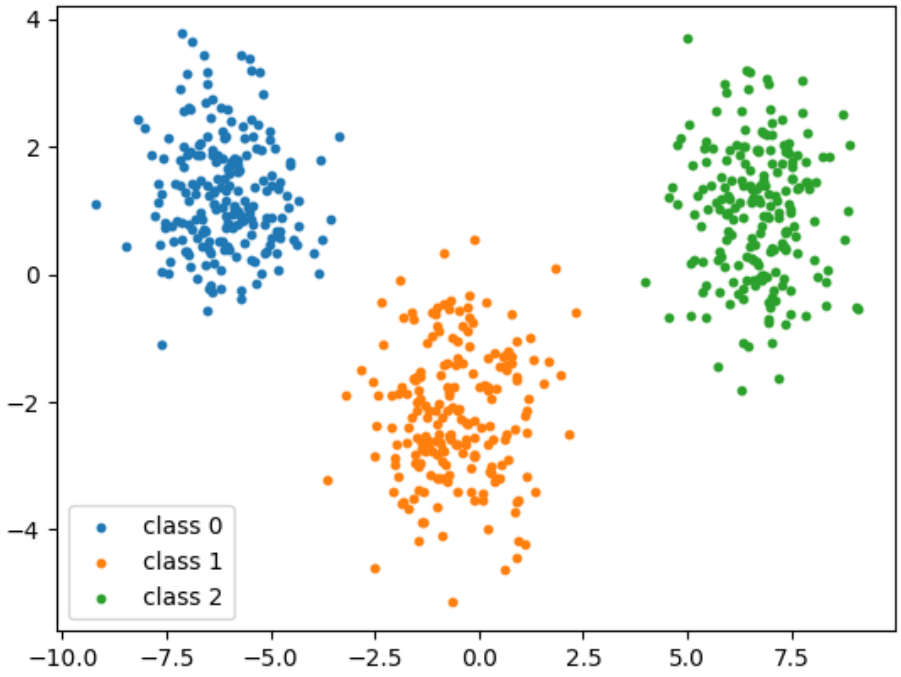

Figura 3.3: Proyección de los datos en 2D usando LDA.

3.4. t-SNE

El t-SNE (*t-Distributed Stochastic Neighbor Embedding*) preserva las relaciones locales en los datos, útil para visualización en 2D ya que permite la división gráfica de conjuntos difícilmente separables mediante fronteras lineales. Es importante señalar que este método tiene una componente estocástica, que controlaremos mediante la selección de un *random_state* para permitir la replicabilidad.

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X)
```

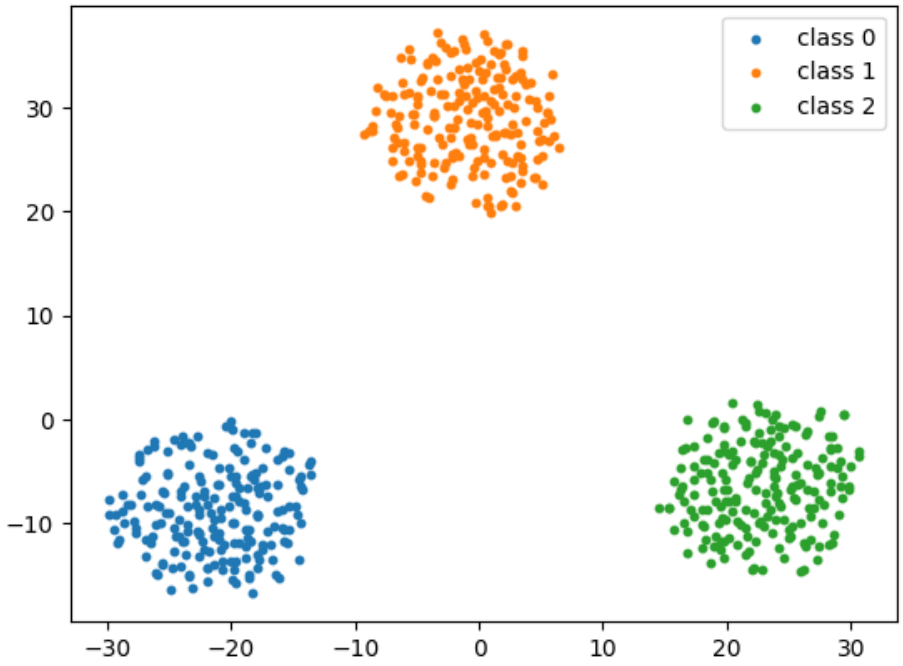

Figura 3.4: Proyección de los datos en 2D usando t-SNE.

4. Red Neuronal Encoder-Decoder

Se utilizó una red neuronal Encoder-Decoder para aprender una representación comprimida de los datos. El modelo fue definido como:

```
def build_encoder_decoder(input_dim=10, encoded_dim=2):
    input_layer = layers.Input(shape=(input_dim,), name='Input')

    # Encoder
    encoded = layers.Dense(8, activation='relu', name='Dense8_encoder')(input_layer)
    encoded = layers.Dense(6, activation='relu', name='Dense6_encoder')(encoded)
    encoded = layers.Dense(4, activation='relu', name='Dense4_encoder')(encoded)
    encoded = layers.Dense(encoded_dim, activation='linear', name=f'Dense{encoded_dim}')
                                                                        (encoded)

    # Decoder
    decoded = layers.Dense(4, activation='relu', name='Dense4_decoder')(encoded)
    decoded = layers.Dense(6, activation='relu', name='Dense6_decoder')(decoded)
    decoded = layers.Dense(8, activation='relu', name='Dense8_decoder')(decoded)
    decoded = layers.Dense(input_dim, activation='linear', name='Output')(decoded)

    # Modelo completo
    autoencoder = models.Model(input_layer, decoded, name='Autoencoder')

    # Encoder separado para reutilización
    encoder = models.Model(input_layer, encoded, name='Encoder')
```

```
    return autoencoder, encoder
```

Posteriormente se dividirán los datos en tres conjuntos (entrenamiento, validación y test).

```
xtrn, xtst, ytrn, ytst = train_test_split(X, y, test_size=0.2,
                                          stratify=y, random_state=42)
xtrn, xval, ytrn, yval = train_test_split(xtrn, ytrn, test_size=0.1,
                                          stratify=ytrn, random_state=42)
```

Este modelo fue entrenado durante 150 épocas, con un valor de *batch_ size* de xtrn.shape[0]//4 y utilizando *Early-stopping*.
Esto otorga unos resultados positivos en pérdidas, como se puede ver en la siguiente gráfica:

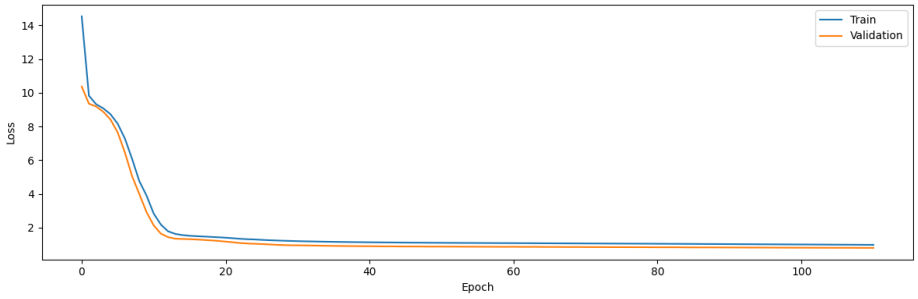

Figura 4.1: Gráfica de la pérdida del modelo encoder-decoder.

Se realiza la reconstrucción del conjunto de datos para comprobar que todo ha funcionado correctamente:

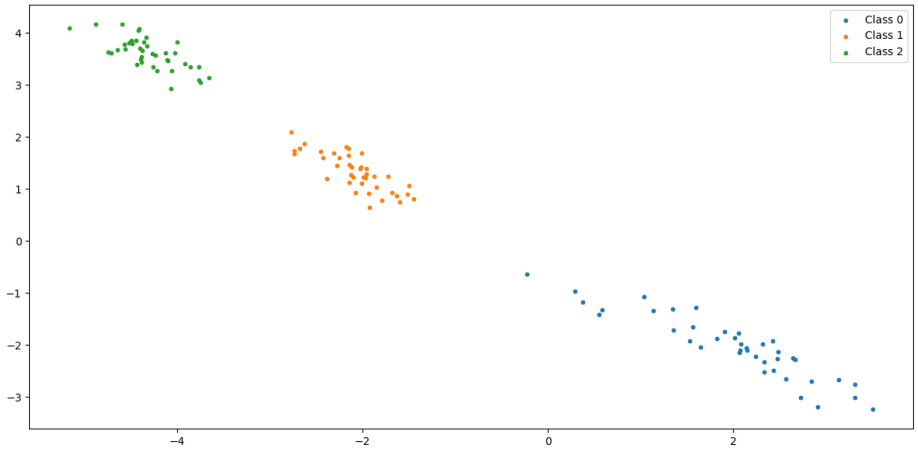

Figura 4.2: Graficación de la reconstrucción.

4.1. Clasificador con Encoder Congelado

Por último, se realizó un clasificador (Fully Connected + Softmax) que se entrenó congelando el *encoder*. Para ello, se creó el modelo secuencial para la clasificación con una capa densa de

activación Softmax que clasificase las 3 clases y se compiló con early stopping. De este modo, tras entrenarlo con 150 épocas, terminó el entrenamiento con los siguientes resultados:

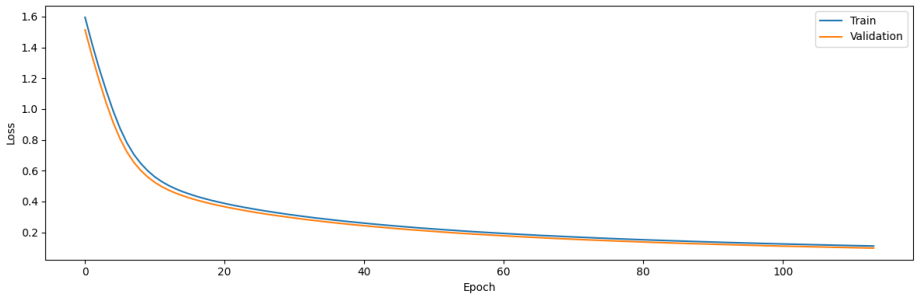

Figura 4.3: Gráfica de la pérdida del modelo de clasificación.

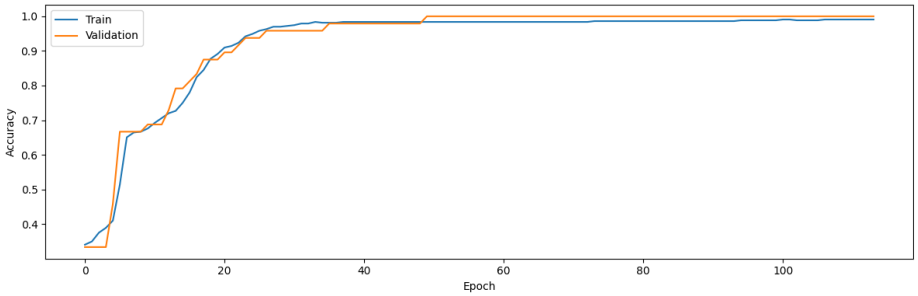

Figura 4.4: Gráfica de la precisión del modelo de clasificación.

5. Resultados

El modelo clasificador logró una precisión del 100 % al predecir las etiquetas del conjunto de prueba. La siguiente gráfica muestra la proyección de las representaciones comprimidas en generadas por el encoder y la clasificación asignada por el modelo:

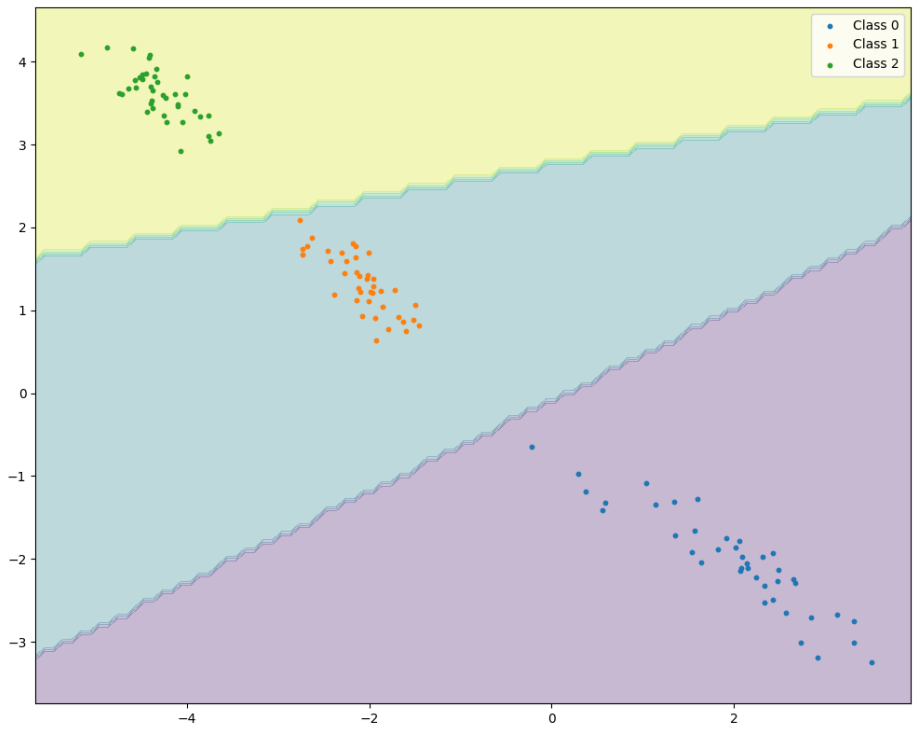

Figura 5.1: Representación de la clasificación del modelo.

Esto confirma la capacidad del pipeline completo para separar efectivamente las clases originales en un espacio reducido, preservando la estructura de los datos.

6. Conclusión

En esta práctica se implementaron diversas técnicas de reducción de dimensionalidad y clasificación, utilizando un conjunto de datos sintético generado en $\mathbb{R}^{10}$. Estas técnicas incluyeron métodos tradicionales como Sammon Mapping, PCA, LDA y t-SNE, así como un enfoque basado en aprendizaje profundo mediante una red neuronal Encoder-Decoder.
Los resultados demostraron que las técnicas implementadas son efectivas para reducir la dimensionalidad del dataset, preservando la separación entre las clases en el espacio proyectado. En particular, el método t-SNE mostró su capacidad para identificar agrupamientos locales no lineales, mientras que LDA maximizó la separación entre clases usando información supervisada. Por otro lado, el Encoder-Decoder logró aprender representaciones comprimidas en $\mathbb{R}^2$, que fueron utilizadas con éxito para clasificar las muestras con una precisión del 100 % al congelar el encoder y añadir una cabeza de clasificación.

Apéndice

El código relacionado con este documento está disponible en el siguiente enlace de GitHub:

github.com/angelleon01/aprendizaje_profundo