

SVD para clasificación de imágenes

Teoría de Análisis Numérico

Daniel Chacón, Luis Rubiano, Martín Cárdenas
Octubre 21 de 2022

Agenda

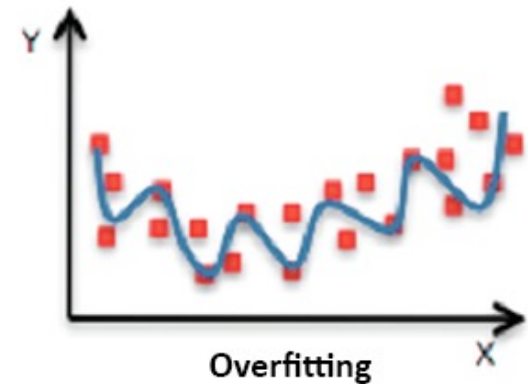
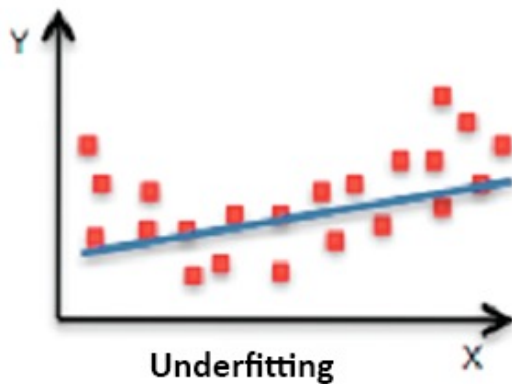
- 1) Motivación
- 2) Meta
- 3) Base de Datos
- 4) Algoritmo de Clasificación
- 5) Aproximación de rango k mediante SVD
- 6) Experimentos
- 7) Bibliografía

Motivación

- La factorización en SVD ha tenido resultados prometedores en tareas tales como reducir el número de parámetros de las redes neuronales.
- Ha permitido reducir uno de los problemas más comunes en Machine Learning: **el Overfitting**.
- También, se han incorporado capas a las redes convolucionales, en donde se descomponen matrices de covarianza de los datos por medio del SVD: facilitando **la clasificación de imágenes**
 - i) Mejorando las métricas de *exactitud*
 - ii) Reduciendo el tiempo de cómputo.

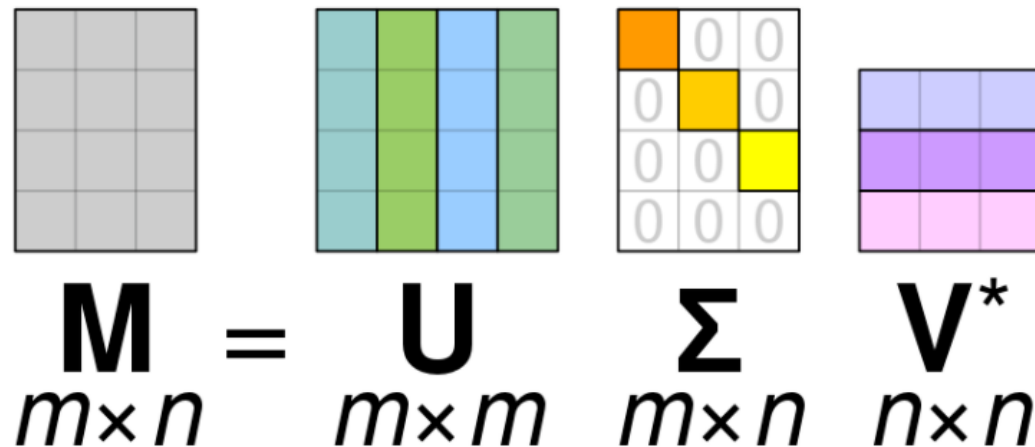
¿Qué es el *Overfitting*?

- El modelo no es capaz de generalizar adecuadamente a datos con los que no fue entrenado.



Meta del proyecto

- Estudiar el **espacio de representación** de imágenes mediante sus descomposiciones en valores singulares (SVD) para optimizar algoritmos de clasificación y **prevenir el overfitting**.



The diagram illustrates the SVD decomposition of a matrix M into three matrices: U , Σ , and V^* . Each matrix is represented by a grid of colored squares. M is a 4x4 grid of gray squares. U is a 4x4 grid with four vertical columns of different colors: teal, green, blue, and green. Σ is a 4x4 grid with a diagonal of colored squares (orange, yellow, yellow, and white) and zeros elsewhere. V^* is a 4x4 grid with four horizontal rows of different colors: light blue, purple, purple, and pink.

$$\begin{matrix} \text{4x4} & & \text{4x4} & & \text{4x4} & & \text{4x4} \\ \mathbf{M} & = & \mathbf{U} & & \mathbf{\Sigma} & & \mathbf{V}^* \\ m \times n & & m \times m & & m \times n & & n \times n \end{matrix}$$

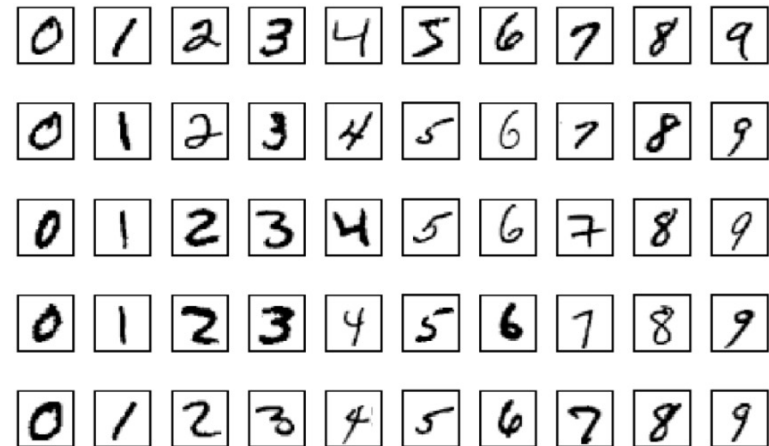
Datos

- **MNIST dataset**

- Modified National Institute of Standards and Technology database por sus siglas en inglés.
- Creada en 1998. Consiste de dígitos escritos a mano por estudiantes y trabajadores del censo de Estados Unidos.

Each datapoint is a 8x8 image of a digit.

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64



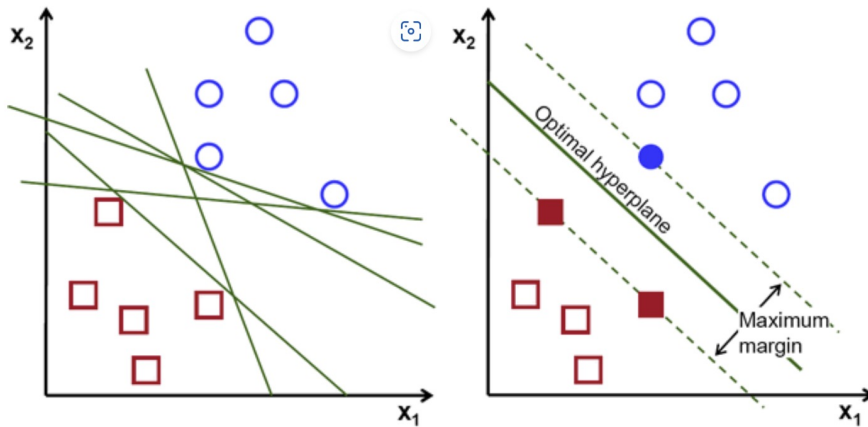
Algoritmo de clasificación

- **Support Vector Machine (SVM)**

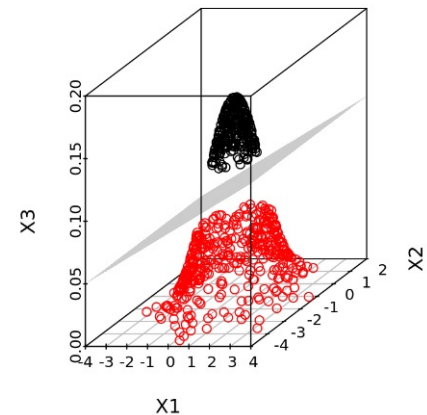
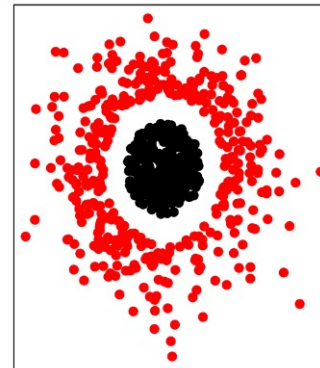
En Machine Learning, el SVM es un algoritmo que analiza los datos para la clasificación.

Tiene como objetivo encontrar un hiperplano en un espacio N-dimensional que de forma distinta clasifica cada punto de los datos

El Hiperplano óptimo es el de mayor margen (distancia)



Possible hyperplanes



Algoritmo de clasificación

- **Support Vector Machine (SVM)**

Fórmula objetivo

$$\underset{\mathbf{w}, b}{\text{Minimizar}} \Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$$

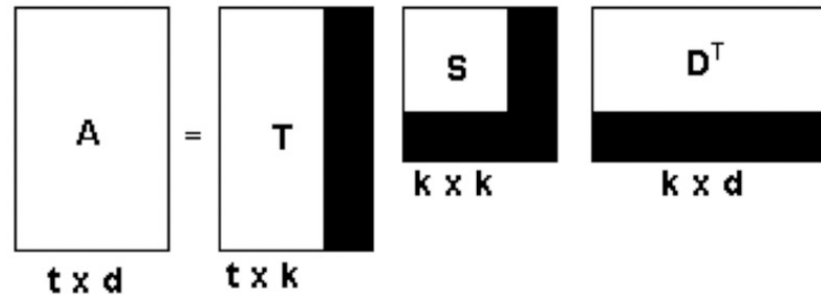
$$\text{Sujeto a: } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$\underset{\mathbf{w}, b, \xi}{\text{Minimizar}} \Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i$$

$$\text{Sujeto a: } y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$$

SVM PRIMAL

Aproximación de rango k mediante SVD

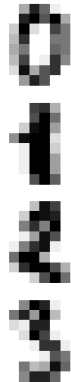


- La aproximación de rango k mediante SVD es la mejor aproximación de rango k a la matriz A .

$$\|A - A_k\|_{Fb} = \min\{\|A - B\|_{Fb} : \text{rank}(B) \leq k\}$$

Experimentos

Datos



Matriz U



Matriz V*

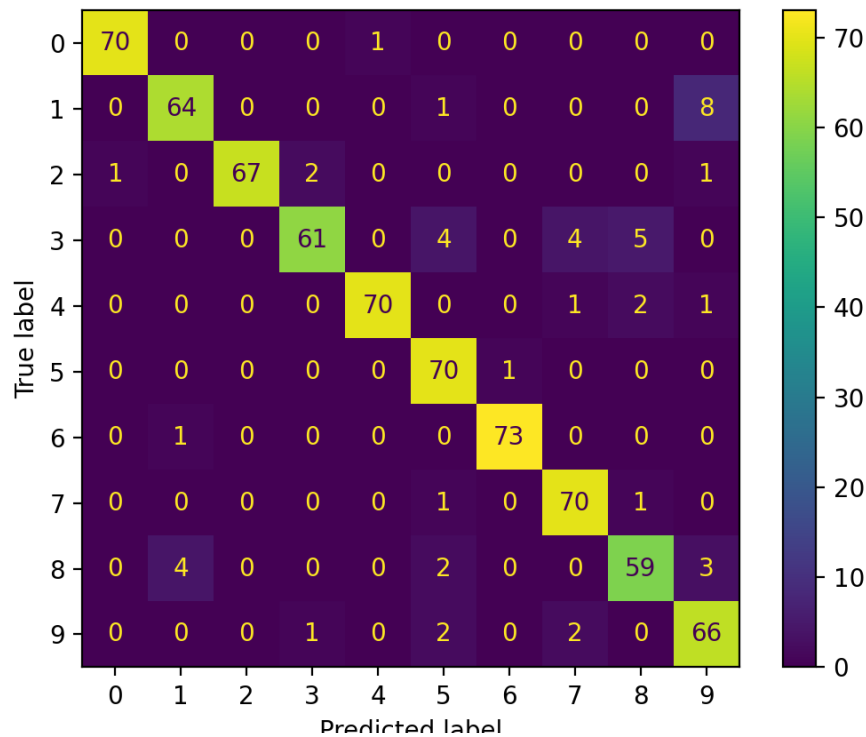


```
1 import matplotlib.pyplot as plt
2 from sklearn import datasets, svm, metrics
3 from sklearn.model_selection import train_test_split
4 import numpy as np
5 import time
6
7 digits = datasets.load_digits() # MNIST
8 fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(10, 3))
9
10 for ax, image, label in zip(axes, digits.images, digits.target):
11     ax[0].imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
12     ax[0].set_axis_off()
13     u, s, vh = np.linalg.svd(image, full_matrices=True)
14     ax[1].imshow(u, cmap=plt.cm.gray_r, interpolation="nearest")
15     ax[1].set_axis_off()
16     ax[2].imshow(vh, cmap=plt.cm.gray_r, interpolation="nearest")
17     ax[2].set_axis_off()
18 plt.show()
```

Experimentos

Línea Base

Confusion Matrix



A

Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155

```

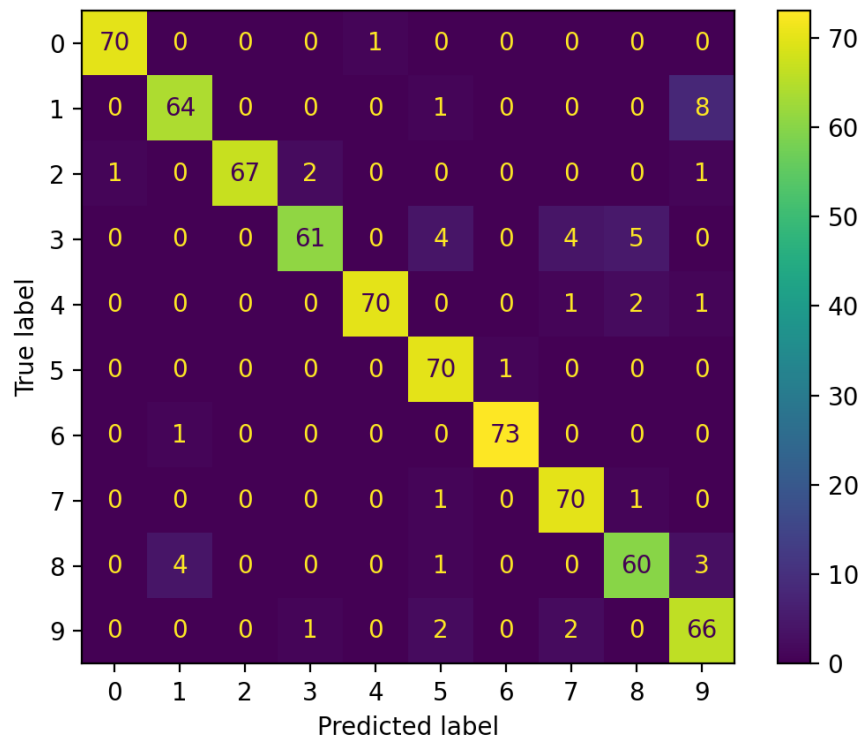
27 # Split data into 60% train and 40% test subsets
28 X_train, X_test, y_train, y_test = train_test_split(data, digits.target, test_size=0.4, shuffle=False)
29 # Train the linear model on the training subset

```

Experimentos

Experimento 1

Confusion Matrix



A, U, V*

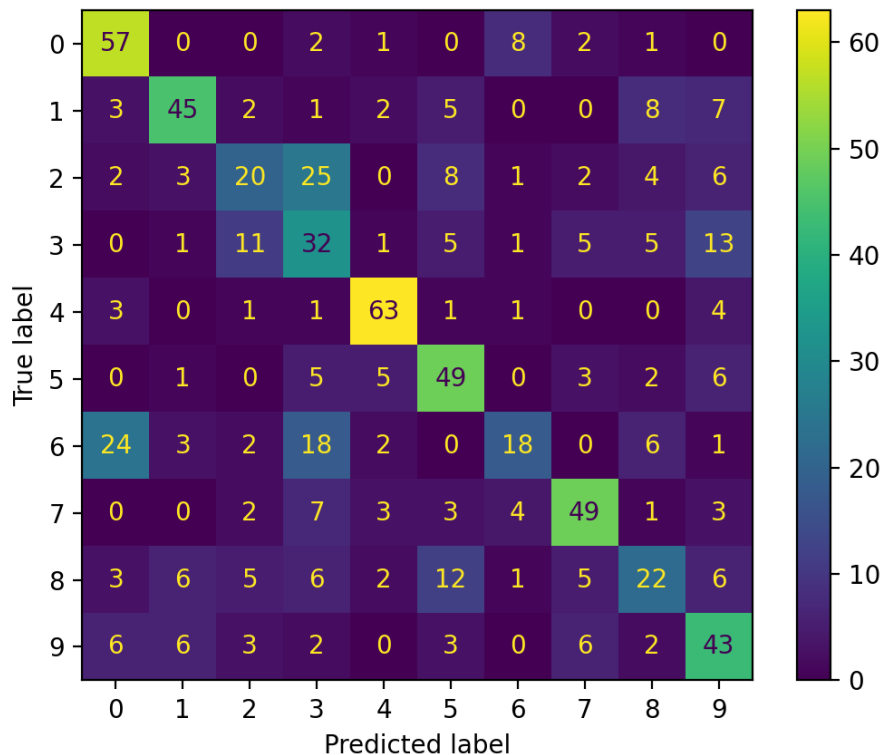
Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155
Experimento 1	0.93	0.94	0.238

```
dataSVD[i] = np.concatenate((data[i],u.reshape(64,),vh.reshape(64,)), axis=0)
```

Experimentos

Experimento 6

Confusion Matrix



$U\Sigma, V^*\Sigma$
(por error)

Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155
Experimento 1	0.93	0.94	0.238

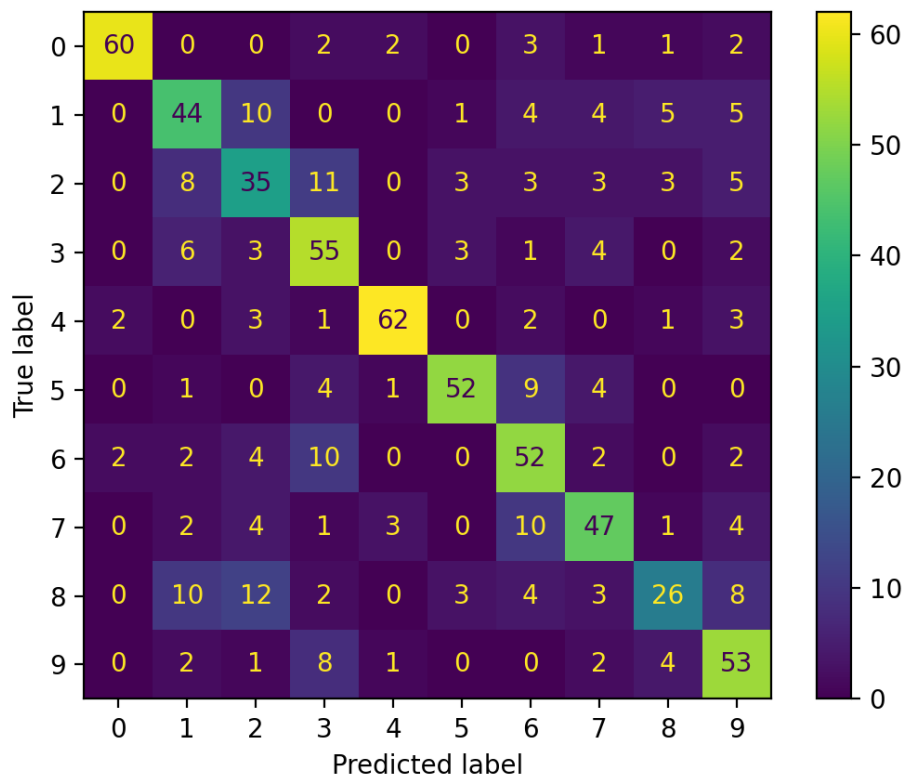
Experimento 6	0.56	0.57	0.267
---------------	------	------	-------

```
dataSVD6[i] = np.concatenate((u@s,vh@s), axis=0)
```

Experimentos

Experimento 2

Confusion Matrix



$$U\Sigma, \Sigma V^*$$

Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155
Experimento 1	0.93	0.94	0.238
Experimento 2	0.68	0.69	0.217

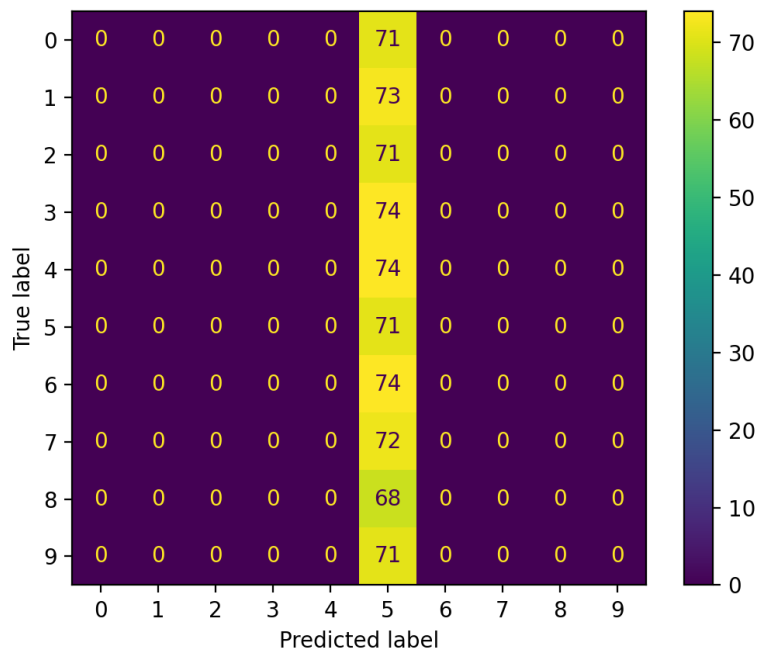
Experimento 6	0.56	0.57	0.267
---------------	------	------	-------

```
dataSVD2[i] = np.concatenate((u@s,s@vh), axis=0)
```

Experimentos

Experimento 3

Confusion Matrix



U, V^*

Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155
Experimento 1	0.93	0.94	0.238
Experimento 2	0.68	0.69	0.217
Experimento 3	0.1	0.01	0.412
			2
			6
Experimento 6	0.56	0.57	0.267

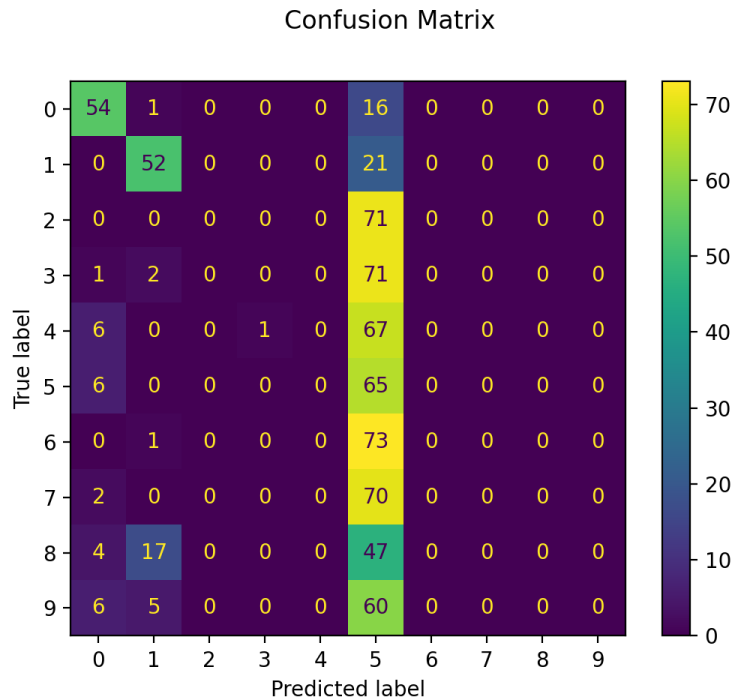
```

103         u, s, vh = np.linalg.svd(image, full_matrices=True)
104         dataSVD3[i] = np.concatenate((u.reshape(64,),vh.reshape(64,)), axis=0)

```

Experimentos

Experimento 4



Σ, U, V^*

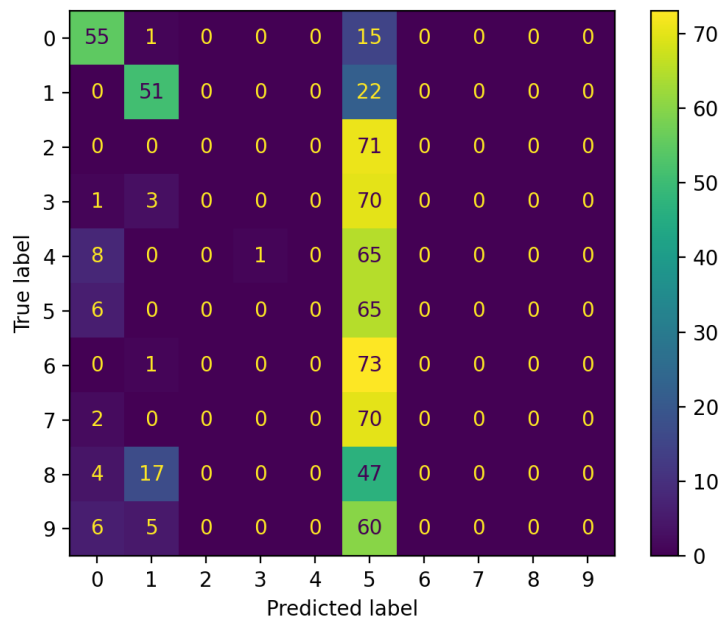
Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155
Experimento 1	0.93	0.94	0.238
Experimento 2	0.68	0.69	0.217
Experimento 3	0.1	0.01	0.412
Experimento 4	0.24	0.15	0.502
Experimento 6	0.56	0.57	0.267

```
dataSVD4[i] = np.concatenate((s,u.reshape(64,),vh.reshape(64,)), axis=0)
```


Experimentos

Experimento 5

Confusion Matrix



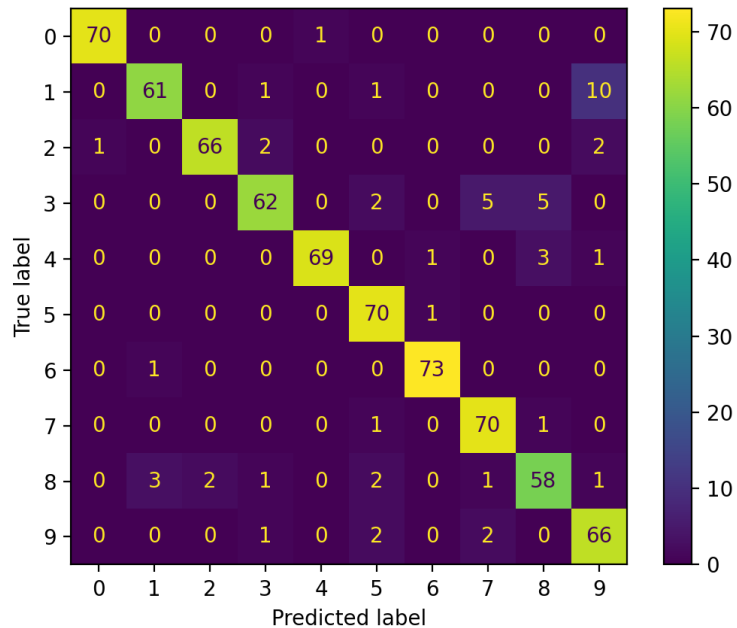
Σ

Experimentos	F1-score	Precision	Tiempo (segundos)
Línea base	0.93	0.93	0.155
Experimento 1	0.93	0.94	0.238
Experimento 2	0.68	0.69	0.217
Experimento 3	0.1	0.01	0.412
Experimento 4	0.24	0.15	0.502
Experimento 5	0.24	0.14	0.266
Experimento 6	0.56	0.57	0.267

`dataSVD5[i] = s`

Experimentos

Confusion Matrix

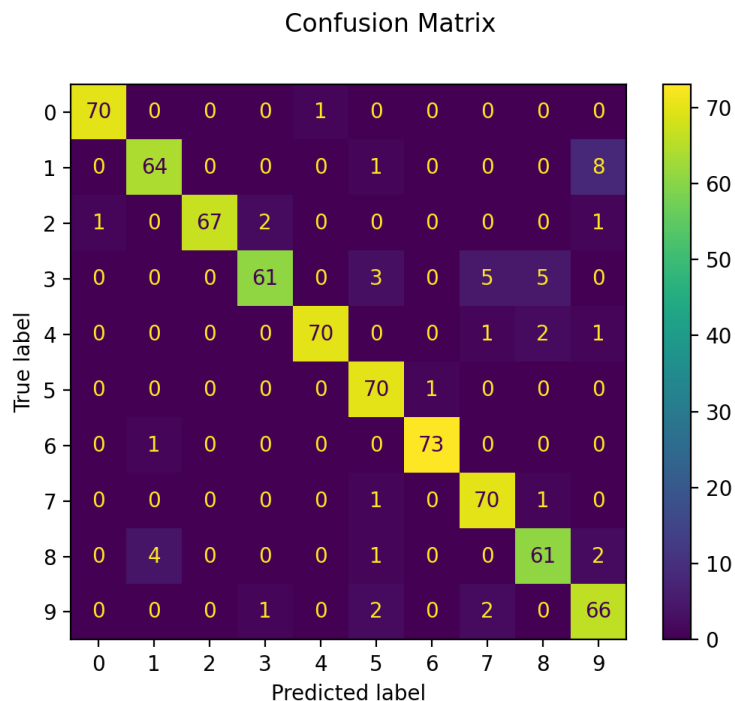


A_3

Experimentos	F1-score	Precision	Tiempo (segundos)
Rank 8	0.93	0.93	0.186
Rank 3	0.92	0.93	0.172

```
u, s, vh = np.linalg.svd(image, full_matrices=True)
sn = np.diag(s[0:n])
dataSVD7[i] = (u[:, :n]@sn@vh[:, :]).reshape(64,)
```

Experimentos



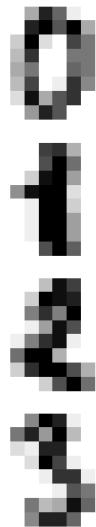
A_5

Experimentos	F1-score	Precision	Tiempo (segundos)
Rank 8	0.93	0.93	0.186
Rank 5	0.93	0.94	0.175
Rank 3	0.92	0.93	0.172

```
u, s, vh = np.linalg.svd(image, full_matrices=True)
sn = np.diag(s[0:n])
dataSVD7[i] = (u[:, :n]@sn@vh[:, :n]).reshape(64,)
```

Experimentos

A



A_1



A_5



Bibliografía

1. Wang, W., Dang, Z., Hu, Y., Fua, P., & Salzmann, M. (2021). Robust differentiable SVD. *IEEE transactions on pattern analysis and machine intelligence*.
2. Yu, B., Xu, Z. B., & Li, C. H. (2008). Latent semantic analysis for text categorization using neural network. *Knowledge-Based Systems*, 21(8), 900-904.
3. Li, C. H., & Park, S. C. (2009). An efficient document classification model using an improved back propagation neural network and singular value decomposition. *Expert Systems with Applications*, 36(2), 3208-3215.
4. Mateen, M., Wen, J., Song, S., & Huang, Z. (2018). Fundus image classification using VGG-19 architecture with PCA and SVD. *Symmetry*, 11(1), 1.
5. Wu, C., Berry, M., Shivakumar, S., & McLarty, J. (1995). Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning*, 21(1), 177-193.
6. Xue, J., Li, J., & Gong, Y. (2013, August). Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspace* (pp. 2365-2369).
7. Xue, J., Li, J., Yu, D., Seltzer, M., & Gong, Y. (2014, May). Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6359-6363). IEEE.
8. Wang, Y., & Zhu, L. (2017, May). Research and implementation of SVD in machine learning. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)* (pp. 471-475). IEEE.
9. Narwaria, M., & Lin, W. (2011). SVD-based quality metric for image and video using machine learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2), 347-364.
10. Shnayderman, A., Gusev, A., & Eskicioglu, A. M. (2006). An SVD-based grayscale image quality measure for local and global assessment. *IEEE transactions on Image Processing*, 15(2), 422-429