Redes neuronales y reducción de dimensionalidad

Ian Gabriel Cañas Fernández. — 1092228
Instituto Tecnológico de Santo Domingo (INTEC)
Santo Domingo, D.N. República Dominicana
Machine Learning — INL367. Sección 01

***Resumen*—En este documento se presentó el proceso de clasificación de automóviles. Se empezó detallando el método de reducción de dimensionalidad por análisis de componentes principales y por análisis de discriminante lineal.**

A continuación, se implementó un autoencoder tanto mediante un método de reducción como por redes neuronales y comparando las distinciones. Finalmente, se ha entrenado una red neuronal para ser utilizada como clasificadora y se ha visto su eficiencia frente a los métodos de reducción de dimensionalidad.

Palabras clave – Red neuronal, reducción de dimensionalidad, autoencoder.

***Abstract*—In this document, the automobile classification process was presented. It began by detailing the method of dimensionality reduction by principal component analysis and by linear discriminant analysis.**

Next, an autoencoder was implemented both by a reduction method and by neural networks and comparing the distinctions. Finally, a neural network has been trained to be used as a classifier and its efficiency compared to dimensionality reduction methods has been verified.

Keywords – Neural network, dimensionality reduction, autoencoder.

I. Introducción

En la presente se procurará codificar los datos de un set que presenta varias características de automóviles. A partir del mismo set, se procurará preparar un clasificador que prediga la cantidad de cilindros que posee cada uno. Todo esto usando como herramientas redes neuronales y métodos de reducción de dimensionalidad.

A. Objetivo general

- Encontrar el mejor método de codificación de datos y de reducción de dimensionalidad para un determinado conjunto de datos.

B. Objetivos específicos

- Implementar una red neuronal para la extracción de características que representen las clases de automóviles: cantidad de cilindros.
- Implementar una red neuronal como autoencoder.
- Implementar LDA y PCA para representar y codificar los datos respectivamente.
- Procurar el mejor método de codificación y clasificación de datos entre las herramientas tenidas hasta el momento.

II. Método

La base de datos tiene la información de 398 autos. Las características que se presentan en la base de datos son: millas por galón, cantidad de cilindros, desplazamiento, caballos fuerza, peso, aceleración, año del modelo y origen.

Para la predicción y codificación de los datos se tienen las siguientes herramientas, que luego serán con mayor detalle:

- Los métodos de reducción de dimensionalidad. En este caso el PCA y LDA, métodos que se caracterizan por representar un conjunto de datos como otro conjunto de menor dimensión y con más información posible.
- Las redes neuronales. Un método de aprendizaje inspirado en las conexiones neuronales biológicas, que se entrenan mediante el procesamiento de ejemplos. La estructura de estas redes se puede apreciar en la Ilustración 1.

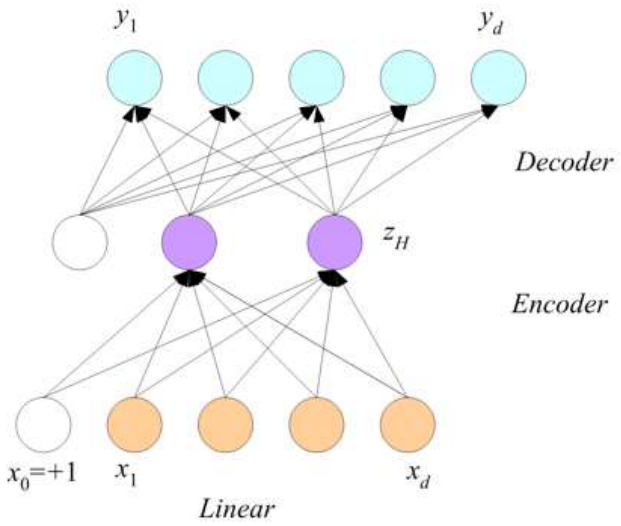

Ilustración 1. Estructura de una red neuronal (extraído de [1]*).*

Para la codificación de los datos se trabajará con el análisis de componente principal (PCA por sus siglas en inglés), un método estadístico que permite simplificar la complejidad la complejidad de un conjunto de datos en una cantidad n de dimensiones[2], [3].

Los autoencoders, de una forma parecida a los PCA es un método no supervisado del machine learning que sirve para identificar anomalías cuando os datos no están etiquetados.

Para la codificación por PCA se buscará una proyección de los datos hacia un nuevo espacio vectorial en el que se cumpla

$Z = W^T X$. Donde X y Z son la matriz con las características de los datos y la representación de estos en otro espacio vectorial, que se desea de menor dimensión.

Para representar los datos a una menor dimensión, se buscan los vectores w que correspondan a los mayores eigenvalores de la matriz de covarianza de X. En síntesis, se seleccionan:

$$W = \{w_1 \quad w_2 \quad \dots \quad w_j \quad \dots \quad w_d\}$$
$$tal\ que\ \Sigma_x w = \alpha w \wedge |w| = 1$$

Para la decodificación de los datos se aplica el siguiente concepto, conociendo que, cuando los vectores de una matriz son ortogonales entre sí, su producto devuelve la matriz unitaria. Los datos originales se recuperarían de manera tal que:

$$Z = W^T X$$

Ecuación 1. Reducción de dimensionalidad a partir de matriz de pesos.

$$W^{T^{-1}} Z = W^{T^{-1}} W^T X$$
$$W^{T^T} Z = IX$$
$$\boldsymbol{WZ = X}$$

Ecuación 2. Recuperación de dimensión original.

La red neuronal utilizada para el autoencoder poseerá una capa oculta que poseerá una función de transferencia lineal saturada entre 0 y 1 para la partida a la salida. En la capa de salida de la red, dígase los datos recuperados, se trabajará con una función de transferencia totalmente lineal de manera tal que $f(Z) = Z$. Los datos serán codificados y recuperados a través de los pesos de la misma red neuronal, por lo que la salida es presentada en el mismo proceso.

El segundo método de reducción de dimensionalidad utilizado, el análisis de discriminante lineal (LDA por sus siglas en inglés), consiste en la búsqueda de la mejor representación de un set de datos en menor cantidad de características a las presentadas[4]. El fin de esta sección es el de poder reducir la cantidad de datos representativos para los datos.

Este método utiliza el concepto de separabilidad entre las clases mediante el uso de las siguientes métricas:

1. Una métrica de dispersión total a lo largo del interno de las clases (S_w).
2. Una métrica de dispersión entre las clases (S_B).
3. Una función de costo $\left(J(w) = \frac{|wS_b w|}{|wS_w w|}\right)$.

El cálculo de S_w se puede computar como la suma de las matrices de dispersión para cada clase:

$$S_w = \sum_{i=1}^{k} S_i$$
$$S_i = \sum_{t} r_i^t (x_t - m_i)(x_t - m_i)^T$$
$$m_i = \frac{1}{N_i} \sum_{x \in C_i} x$$

Ecuación 3. Cálculo de matriz de dispersión total a lo interno de las clases (Sw).

El cálculo de S_B se puede computar estimando las distancias que tienen los centros de las clases al punto central de todas:

$$S_B = \sum_{i=1}^{k} N_i (m_t - m)(m_i - m)^T$$
$$S_B = \frac{1}{k} \sum_{i=1}^{k} m_i$$

Ecuación 4. Cálculo de matriz de dispersión entre las clases (SB).

Conociendo lo anterior, es demostrable que la solución óptima se consigue usando los vectores propios (eigenvectors) de los mayores eigenvalores de $S_W^{-1} S_B$.

$$W^* = eigVect\{S_W^{-1} S_B\}, \qquad tal\ que\ eigvl \neq 0$$

Ecuación 5. Cálculo de pesos para análisis de discriminante lineal.

Tan pronto de tenga un nuevo set de características de manera tal que $Z = W^T X$, los features serán clasificados a partir de una regresión lineal. Los pesos de una regresión lineal se estiman mediante las siguientes expresiones:

$$\boldsymbol{w} = (D^T D)^{-1} (D^T \boldsymbol{r})$$

Ecuación 6. Cálculo de peso para regresión lineal.

$$D = \begin{bmatrix} 1 & z_1^1 & \cdots & z_k^1 \\ 1 & z_1^2 & \cdots & z_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_1^N & \cdots & z_k^N \end{bmatrix}$$

Donde:

- z son los datos de entrenamiento obtenidos mediante la reducción por LDA.
- r son las etiquetas correspondientes a los datos de entrenamiento.

En caso de ser necesario, se aplicará el método de k vecinos más cercanos (kNN por sus siglas en inglés) a la reducción por LDA. Este método consiste en asignar una clase a un dato en base a votación según las clases que tengan los datos más cercanos a este.

La red neuronal utilizada para la predicción mediante reducción de dimensionalidad posee una capa neuronal oculta con tantas neuronas como dimensiones indica el LDA, [5]. Esta selección de cantidad de neuronas es para la comparación entre ambos métodos bajo el mismo nivel de reducción y así hacer énfasis en los efectos de los métodos en sí.

La clasificación a partir de la red neuronal se hará mediante la prueba de esta, introduciendo el set de datos de entrenamiento en la entrada y observando la salida que este retorna.

III. Resultados

A continuación, se presentarán los resultados obtenidos mediante la implementación de los códigos. Se empezará mostrando el autocodificador para demostrar la eficiencia de la reducción de dimensionalidad para almacenar datos de una manera comprimida.

A. Autoencoder: PCA versus red neuronal

Según lo descrito en la metodología, se empezará computando y preparando la base de datos en una matriz, que

llamaremos *X_tout* y posee todos los datos presentes en la base de datos.

Aplicando la matriz de covarianza y obteniendo los eigenvalores de esta, se presentan los siguientes valores:

```
[V D] = eig(cov(X_tout));
```

```
D =

  1.0e+05 *

    0.0000         0         0         0         0         0         0         0
         0    0.0000         0         0         0         0         0         0
         0         0    0.0000         0         0         0         0         0
         0         0         0    0.0001         0         0         0         0
         0         0         0         0    0.0002         0         0         0
         0         0         0         0         0    0.0027         0         0
         0         0         0         0         0         0    0.0151         0
         0         0         0         0         0         0         0    7.2775
```

Ilustración 2. Matriz de eigenvalores mediante PCA.

En base a la teoría, se reconoce que la proporción de varianza, mientras mayor sea, más cercana a los datos originales es la compresión de datos [6]. Usualmente, la proporción es seleccionada como cualquier valor mayor al 90 % de la representación mediante la siguiente expresión:

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

Según lo visto en la base de datos actual, se tiene que k=1, pero, para mayor representación, se seleccionará k=2. A continuación se presenta la matriz de vectores propios con módulo unitario, presentada como V. Luego, se presenta la matriz de pesos generada a partir de los k eigenvalores seleccionados.

```
V =

   -0.0220   -0.0412    0.0602    0.5497    0.8306   -0.0372   -0.0173   -0.0076
   -0.9332    0.3586    0.0145   -0.0057   -0.0042   -0.0073    0.0135    0.0018
    0.0164    0.0010   -0.0116   -0.0041    0.0111   -0.3042    0.9454    0.1143
   -0.0071   -0.0068   -0.0827   -0.0060    0.0585    0.9481    0.2988    0.0386
    0.0001   -0.0002    0.0035    0.0040    0.0036   -0.0029   -0.1207    0.9927
   -0.0082    0.0203   -0.9881    0.1290   -0.0163   -0.0733   -0.0346   -0.0014
    0.0137    0.0337   -0.1114   -0.8237    0.5528   -0.0409   -0.0247   -0.0013
    0.3578    0.9317    0.0220    0.0535    0.0192    0.0115   -0.0032   -0.0005
```

Ilustración 3. Matriz "V" de vectores propios mediante PCA.

```
>> Wpca

Wpca =

   -0.0076   -0.0173
    0.0018    0.0135
    0.1143    0.9454
    0.0386    0.2988
    0.9927   -0.1207
   -0.0014   -0.0346
   -0.0013   -0.0247
   -0.0005   -0.0032
```

Ilustración 4. Matriz de pesos mediante PCA.

La primera transformación devuelve una representación de los datos a dos dimensiones generada mediante Ilustración 4 y la Ecuación 1.

```
>> size(Zpca)

ans =

   398     2
```

Ilustración 5. Tamaño de representación de los datos por PCA.

Aplicando la Ecuación 2 se busca recobrar los features o características originales, obteniéndose una matriz con el tamaño de la matriz original (ver Ilustración 6).

```
>> size(X_f)

ans =

   398     8
```

Ilustración 6. Tamaño de recuperación de datos por PCA.

Para obtener todos los resultados mencionados anteriormente se ha implementado el código de la Ilustración 7.

```
[V D] = eig(cov(X_tout));

Des = sum(D, 'all');
cant = 0;
posi = sum(D);
Eigenes = 0;
while Eigenes<=0.9 %% Ch6, Diapo 11
    cant = cant + 1;
    Eigenes = sum(posi(end-cant:end))/Des;
end
cant;

[M I] = maxk(sum(D), cant+1);
Wpca = V(:,I);

Zpca = X_tout*Wpca;
X_f = Zpca*Wpca';

MS_PCA = mean((X_tout-X_f).^2 )
```

Ilustración 7. Implementación de PCA en Matlab.

A parte, se entrena una red neuronal mediante el uso de las librerías brindadas por Matlab, configurando la función del codificador a una función lineal saturada y la salida del decodificador como puramente lineal. El código implementado se presenta en la Ilustración 8.

```
netEncoded = trainAutoencoder(X_tout', cant+1, 'EncoderTransferFunction', ...
    'satlin', 'DecoderTransferFunction', 'purelin');
generateFunction(netEncoded, 'NNEnc');

X_fNN = NNEnc(X_tout')';

MS_NN = mean((X_tout-X_fNN).^2)
```

Ilustración 8. Red neuronal para codificación implementada mediante librerías de Matlab.

Para reconocer la eficiencia del método de codificación utilizado la métrica de error cuadrado promedio, tanto para PCA como para el autoencoder aplicado por una red neuronal.

```
>> MS_PCA'

ans =

   1.0e+03 *

    1.9240
    0.0048
    0.0781
    1.3670
    0.0000
    0.2150
    5.8404
    0.0079
```

```
>> MS_NN'

ans =

   1.0e+03

    0.0179
    0.0027
    1.3584
    0.3499
    0.0231
    0.0068
    0.0131
    0.0008
```

Ilustración 9. Error medio cuadrado para PCA y red neuronal (NN).

B. Extracción de características representativas: LDA versus red neuronal.

Para esta sección, debido a que se preparará un clasificador, la base de datos ha sido seccionada entre set de entrenamiento (con un 75 % de los datos) y set de prueba (con un 25 % de los datos). La proporción de los datos ha sido distribuida de manera aleatoria para evitar posible efecto debido al orden.

En la base de datos se tiene como clase la cantidad de cilindros que posee el automóvil en cuestión y como features los demás datos.

En base a la Ecuación 3, se calcula la dispersión total a lo interno de las clases tal y como se presenta en la Ilustración 10. El resultado obtenido se presenta en la Ilustración 11.

```
[fcyl Ncyl] = hist(traindata.cylinders, min(traindata.cylinders):1:max(traindata.cylinders));
% Si = 0;
Sw = 0;
Sb = 0;
[a, b] = size(X);
mcent = nan(length(Ncyl(fcyl~=0)), b);

for i = Ncyl(fcyl~=0)
    ind = find(Ncyl==i);
    Xi = X(traindata.cylinders(:)==i,:);
    mi = mean(Xi);
    Si = i*(X-mi)'*(X-mi);
    Sw = Sw + Si;
    i;

    mcent(ind, :) = mi;
end

mt = mean(mcent, 'omitnan');
```

Ilustración 10. Cálculo de Sw implementado en Matlab.

```
>> Sw

Sw =

   1.0e+09 *

    0.0007   -0.0086   -0.0030   -0.0678    0.0002    0.0002    0.0000
   -0.0086    0.1636    0.0499    1.1619   -0.0022   -0.0019   -0.0008
   -0.0030    0.0499    0.0184    0.3645   -0.0010   -0.0008   -0.0002
   -0.0678    1.1619    0.3645    9.5066   -0.0130   -0.0117   -0.0057
    0.0002   -0.0022   -0.0010   -0.0130    0.0001    0.0001    0.0000
    0.0002   -0.0019   -0.0008   -0.0117    0.0001    0.0001    0.0000
    0.0000   -0.0008   -0.0002   -0.0057    0.0000    0.0000    0.0000
```

Ilustración 11. Matriz de dispersión total a lo interno de las clases.

Luego, con uso de la Ecuación 4 se calcula la dispersión entre clases tal y como se presenta en la Ilustración 12. El resultado obtenido se aprecia en la Ilustración 13.

```
mt = mean(mcent, 'omitnan');

for i = Ncyl(fcyl~=0)
    ind = find(Ncyl==i);

    Xi = X(traindata.cylinders(:)==i,:);
    mi = mean(Xi);

    Sj = fcyl(ind)*(mi-mt)'*(mi-mt);
    Sb = Sb + Sj;
end
```

Ilustración 12. Cálculo de SB implementado en Matlab.

```
>> Sb

Sb =

   1.0e+08 *

    0.0001   -0.0017   -0.0005   -0.0137    0.0000    0.0000    0.0000
   -0.0017    0.0291    0.0090    0.2146   -0.0004   -0.0004   -0.0001
   -0.0005    0.0090    0.0030    0.0688   -0.0001   -0.0001   -0.0000
   -0.0137    0.2146    0.0688    1.6815   -0.0031   -0.0028   -0.0005
    0.0000   -0.0004   -0.0001   -0.0031    0.0000    0.0000    0.0000
    0.0000   -0.0004   -0.0001   -0.0028    0.0000    0.0000    0.0000
    0.0000   -0.0001   -0.0000   -0.0005    0.0000    0.0000    0.0000
```

Ilustración 13. Matriz de dispersión entre clases (SB).

Calculando el producto entre ambas matrices de dispersión y obteniendo los eigenvalores de la manera presentada en la Ecuación 5, se presentan los siguientes valores:

```
D =

    0.0191         0         0         0         0         0         0
         0    0.0117         0         0         0         0         0
         0         0    0.0069         0         0         0         0
         0         0         0    0.0012         0         0         0
         0         0         0         0   -0.0000         0         0
         0         0         0         0         0    0.0000         0
         0         0         0         0         0         0    0.0000
```

Ilustración 14. Matriz de eigenvalores mediante LDA.

En base a la teoría, se reconoce que el tamaño de la nueva dimensión gracias a la reducción por LDA corresponde a cantidad de eigenvalores diferentes de cero.

Según lo visto en la base de datos actual, se tiene que k=4, pero. A continuación, se presenta la matriz de vectores propios con módulo unitario, presentada como V. Luego, se presenta la matriz de pesos generada a partir de los k eigenvalores diferentes de cero.

```
V =

    0.2400   -0.2074   -0.3501    0.7153    0.0562    0.0436   -0.1082
   -0.0147   -0.0257    0.0187   -0.0166    0.0108   -0.0179   -0.0000
    0.0339   -0.0242   -0.0849   -0.0823   -0.0995    0.0267   -0.0095
   -0.0021    0.0028   -0.0019    0.0123    0.0020    0.0009   -0.0006
    0.1610   -0.1679    0.0606    0.5280   -0.9139    0.0366   -0.5517
   -0.1136    0.1072    0.1752   -0.1596    0.1560   -0.0884    0.6973
   -0.9498    0.9571   -0.9141    0.4206    0.3567   -0.9939   -0.4445
```

Ilustración 15. Matriz "V" de vectores propios mediante LDA.

```
>> Wlda

Wlda =

    0.2400   -0.2074   -0.3501    0.7153
   -0.0147   -0.0257    0.0187   -0.0166
    0.0339   -0.0242   -0.0849   -0.0823
   -0.0021    0.0028   -0.0019    0.0123
    0.1610   -0.1679    0.0606    0.5280
   -0.1136    0.1072    0.1752   -0.1596
   -0.9498    0.9571   -0.9141    0.4206
```

Ilustración 16. Matriz de pesos mediante LDA.

La primera transformación devuelve una representación de los datos a dos dimensiones generada mediante Ilustración 16 y la Ecuación 1.

```
>> size(Zlda)

ans =

   299     4
```

Ilustración 17. Tamaño de representación de los datos de entrenamiento por LDA.

Tómese en cuenta que la cantidad de datos en esta trasformación corresponde a los datos de entrenamiento, aproximadamente un 75 % de los datos totales. El otro 25 % se aprecia transformado y guardado como Z2lda (véase la Ilustración 18).

```
>> size(Z2lda)

ans =

    99     4
```

Ilustración 18. Tamaño de representación de los datos de prueba por LDA.

Con la reducción obtenida para los datos del entrenamiento y los datos del objetivo del set se ha ideado calcular los pesos de la regresión lineal. La regresión lineal, tal y como se presenta en la Ecuación 6. La computación de los pesos de la regresión lineal se puede observar en la Ilustración 19.

```
Zlda = [ones(length(traindata.cylinders), 1) X*Wlda];
Z2lda = [ones(length(testdata.cylinders), 1) X2*Wlda];

w = inv(Zlda'*Zlda)*Zlda'*r;
cylinderML = round(Z2lda*w);
e = [cylinderML testdata.cylinders cylinderML==testdata.cylinders];
MSE_LDA_Reg = mean((cylinderML-testdata.cylinders).^2)
```

Ilustración 19. Implementación de LDA en Matlab.

A parte, se entrena una red neuronal mediante el uso de las librerías brindadas por Matlab, configurando la función del codificador a una función lineal saturada y la salida del decodificador como puramente lineal. El código implementado se presenta en la red neuronal ha sido extraído de las herramientas de Matlab, a partir de este se obtuvo la función para la predicción de los datos de prueba (ver Ilustración 20).

```
%% Comprobación para Neural Network como predictor

cylinderML3 = myNeuralNetworkFunction(X2')'

MSE_LDA = mean((cylinderML3-testdata.cylinders(:)).^2)
R2_LDA = corr2(cylinderML3, testdata.cylinders)
[cylinderML3 testdata.cylinders];
```

Ilustración 20. Comprobación para red neuronal como clasificador.

Para reconocer la eficiencia del método de codificación utilizado la métrica de error cuadrado promedio para ambos métodos. Igualmente, se ha utilizado la tasa de aciertos debido a que estamos tratando un clasificador con objetivo discreto. En la Ilustración 21 se presentan los aciertos y error medio cuadrado mediante el uso de LDA. En la Ilustración 22 se puede observar la misma métrica para el uso de la red neuronal.

```
aciertosLDA =

    0.7475


MSE_LDA_Reg =

    0.2525
```

Ilustración 21. Tasa de aciertos y error medio cuadrado para LDA.

```
aciertosLDA_NN =

    0.9495


MSE_LDA_NN =

    0.2525
```

Ilustración 22. Tasa de aciertos y error medio cuadrado para red neuronal.

Viendo una gran diferencia de eficiencia en la predicción mediante regresión lineal por PCA frente a la red, se ha procurado confirmar la tasa de aciertos del primero utilizando otro método bien conocido: kNN.

El método utilizado ha sido preseleccionado debido a que el método de reducción de dimensionalidad tiende a aislar las clases, dando así mejor detalle de las clases según la ubicación de las características. En la Ilustración 23 se presenta la nueva tasa de aciertos mediante kNN aplicado en conjunto a LDA.

```
kNNlda =

    0.9697
```

Ilustración 23. Tasa de aciertos para kNN aplicada a reducción por LDA.

IV. Análisis

Si se presta atención al error medio cuadrado de cada una de las características recuperadas mediante PCA y la red neuronal. Se puede apreciar que la red neuronal es más precisa a la hora de codificar que el análisis de componentes principales, aunque se esté trabajando en una misma dimensión.

La razón por la que la red es más precisa es que esta se adapta a los datos en su máxima expresión, garantizando en cada paso de su estimación un mejor detalle de los pesos precisos. Es decir, el método por redes neuronales parece adaptarse mucho mejor a los detalles del dataset, mientras que el método de componentes principales tiende a perder detalles y se centra más en describir el conjunto de datos.

En cuanto a la predicción mediante la representación de los datos por LDA. Se puede observar que la tasa de aciertos con la red neuronal es mucho mayor que con la reducción por LDA; nos referimos a una diferencia de cerca del 20 % de acierto.

Paralelamente, ha de reconocerse que el método de redes neuronales implica mayor tiempo de entrenamiento, por lo que es importante hacer un análisis de si vale la pena dicho tiempo según la base de datos que se quiera implementar en el momento.

Finalmente, luego de observar la eficiencia del método de LDA y reconociendo que este "detalla" las clases en sus respectivas dimensiones. Se considera que, en general, el método de reducción por LDA aprecia un menor sobreajuste (overfit) a los datos de entrenamiento y que, por lo tanto, tiende a dar mejores resultados [7]. Además de que este método es mucho más rápido a la hora de entrenar a la máquina para la clasificación.

V. Referencias

[1] E. Alpaydin, «Introduction to machine learning: Multilayer perceptrons.», 2014.

[2] J. A. Rodrigo, «Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE». 2017, [En línea]. Disponible en: https://www.cienciadedatos.net/documentos/35_principal_component_analysis.

[3] J. A. Rodrigo, «Detección de anomalías: Autoencoders y PCA». 2020, [En línea]. Disponible en: https://www.cienciadedatos.net/documentos/52_deteccion_anomalias_autoencoder_pca.html.

[4] J. Brownlee, «Linear Discriminant Analysis for Machine Learning». 2016, [En línea]. Disponible en: https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/.

[5] Comunidad, «ML | Linear Discriminant Analysis». 2021, [En línea]. Disponible en: https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/.

[6] E. Alpaydin, «Introduction to machine learning: Dimensionality reduction», pp. 8-13, 2014.

[7] Na8, «Qué es overfitting y underfitting y cómo solucionarlo». 2017, [En línea]. Disponible en: https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/#:~:text=En Resumen&text=Overfitting indicará un aprendizaje "excesivo,comprender nuevos datos de entrada.