Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Computación

Reducción de Dimensiones: Revisión y Aplicaciones en Clasificación Automática

Alumno: Andrés Bravo Núñez
Tutor: Valentín Cardeñoso Payo

Dedicado a
mi familia

Agradecimientos

En primer lugar, a mis profesores por la formación que he recibido durante estos años y de la que hago uso en este trabajo.

Gracias a mi tutor Valentín Cardeñoso por apoyarme en la realización de este trabajo de fin de grado y por preocuparse de que lo finalizara.

Por último y especialmente, a mi familia, por brindarme su apoyo en todo momento haciendo cualquier problema más llevadero.

Resumen

En la actualidad es habitual encontrar conjuntos de datos de alta dimensión. Este tipo de datos son difíciles de interpretar y presentan problemas como entrada a algoritmos de clasificación debido a la maldición de la dimensionalidad. Una manera de mejorar estos aspectos es la aplicación de métodos de reducción de la dimensión.

Este trabajo tiene como objetivo analizar y comparar un subconjunto de los métodos de reducción de dimensión existentes. Además, se hace una evaluación de los mismos tanto de una manera aislada mediante diferentes medidas de calidad como por su impacto en diferentes métodos de clasificación. Ambas evaluaciones se realizan sobre un conjunto de datos de referencia como es MNIST.

De una manera más concreta, se han evaluado los métodos de reducción usando medidas cuantitativas para medir la conservación de las propiedades locales y globables de los datos. Posteriormente se han comparado métodos de clasificación entrenados en un conjunto de datos reducido con métodos entrenados con los datos originales. Los resultados obtenidos muestran que cual es el mejor método de reducción de dimensión depende de nuestro objetivo (visualización o preprocesamiento). Además se comprueba que la reducción de dimensión es una estrategia viable para mejorar el comportamiento de un clasificador o para reducir el coste de entrenarlo, almacenarlo o usarlo.

Palabras claves: reducción de dimensionalidad, MDS, PCA, t-SNE, UMAP, clasificación automática

Andrés Bravo Núñez

Abstract

Nowadays it is common to find high dimensional data sets. This type of data is difficult to interpret and present problems as input to classification algorithms due to the curse of dimensionality. One way to improve these aspects is the application of dimension reduction methods This work aims to analyze and compare a subset of existing dimensional reduction methods. In addition, they are evaluated both in an isolated way by different quality measures and by their impact on different classification methods. Both evaluations are carried out on a reference data set such as MNIST.

More specifically, reduction methods have been evaluated using quantitative measures to measure the conservation of local and global properties of the data. Subsequently, classification methods trained on a reduced data set have been compared with methods trained on the original data.

The results obtained show that which is the best dimension reduction method depends on our objective (visualization or pre-processing). In addition, it is proven that dimensional reduction is a viable strategy to improve the behavior of a classifier or to reduce the cost of training, storing or using it.

Keywords: dimensionality reduction, MDS, PCA, t-SNE, UMAP, automatic classification

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En la actualidad existen muchos campos en los que se trabaja con conjuntos de datos de gran dimensión. Como ejemplo, dentro de la bioinformática podemos nombrar los estudios de transcriptómica de célula única en la que los datos son niveles de expresión de cientos de miles genes por cada célula individual estudiada [1]. Por otro lado, en el ámbito de la inteligencia artificial se trabaja con datos de alta dimensión como imágenes, textos o audios. Para manejar este tipo de datos es habitual reducir el número de dimensiones de los datos ya sea seleccionando las variables más relevantes o generando nuevas variables a partir de las existentes. Esta generación de nuevas variables a partir de todas las variables originales es lo que se conoce como reducción de dimensión.

La reducción de dimensión presenta varias ventajas, facilita la visualización y la interpretación [2], reduce el espacio necesario para el almacenamiento [3] y reduce la cantidad de ruido presente [4]. Además es habitual que los datos multidimensionales habiten un subespacio de menor dimensión y por tanto es posible reducir la dimensionalidad sin perder casi información[5].

Desde el punto de vista de la clasificación, los datos de alta dimensión son problemáticos. La precisión de un clasificador empeora en alta dimensión debido al fenómeno de la maldición de la dimensión [6]. Esto se traduce en que la dimensión que es capaz de admitir un clasificador escala con el número de muestras disponibles [7][8]. Otro problema posible si se tienen muchas dimensiones para relativamente pocas muestras es el del sobreajuste. Estás problemáticas pueden justificar la aplicación de un método de reducción como paso previo al entrenamiento de un clasificador.

1.1. Objetivos

En este contexto, el presente TFG pretende evaluar un conjunto de métodos de reducción de dimensión de una manera aislada y posteriormente aplicar los resultados obtenidos para entender el impacto de la reducción de dimensión sobre diferentes métodos de clasificación.

Para ello se plantean dos experimentos:

- La reducción de dimensión de un conjunto de datos seguida del cálculo de diferentes índices que miden la conservación de las propiedades de los datos.
- La comparación de diferentes clasificadores sobre el conjunto de datos original y sobre versiones reducidas del mismo bajo diferentes métodos.

Estos dos experimentos, se han realizado sobre el conjunto de datos de imágenes de dígitos escritos a mano MNIST [9]. Se ha escogido MNIST por ser un conjunto de datos en abierto, de alta dimensión y de referencia para la evaluación de técnicas de machine learning.

1.2. Estructura

Con estos objetivos planteados, la estructura que sigue este TFG es la siguiente:

- **Capítulo 2. Reducción de la dimensión.** En este apartado, se realiza una revisión teórica de diferentes métodos de reducción de dimensión, con especial énfasis en aquellos menos conocidos.
- **Capítulo 3. Clasificación.** En él se exponen los diferentes métodos de clasificación que serán evaluados.
- **Capítulo 4. Experimentos.** Se explica el planteamiento de los dos experimentos planteados y se presentan los resultados.
- **Capítulo 5. Conclusiones** Se cierra el trabajo presentando las conclusiones obtenidas.

Capítulo 2

Reducción de la dimensión

En este capítulo se presenta una visión tanto matemática como algorítmica de un conjunto de métodos de reducción de la dimensionalidad. Estudiándolos se pretende ayudar a seleccionar razonadamente uno u otro para formar parte de un *pipeline* de clasificación.

2.1. Consideraciones generales

La reducción de dimensionalidad tiene varios puntos positivos como paso previo al entrenamiento de un clasificador:

1. Elimina o reduce el número de atributos irrelevantes/redundantes para la tarea de clasificación. Esto es ventajoso pues existen métodos de clasificación que no funcionan bien bajo este tipo de ruido (Por ejemplo *support vector machines* [10]).
2. Ayudan a combatir el sobreajuste [11].
3. Reducen la cantidad de recursos necesarios para entrenar, almacenar o utilizar el clasificador.

Para esta revisión se han escogido siete métodos de reducción de la dimensionalidad:

- PCA
- MDS
- ISOMAP
- LLE
- Laplacian Eigenmap
- t-SNE
- UMAP

2.2. PCA

El análisis de componentes principales (PCA o *Principal component analysis*) [12] busca llevar los datos a un nuevo sistema de coordenadas de forma que los nuevos ejes apunten en las direcciones de máxima varianza. Las coordenadas de los datos según estos ejes se denominan componentes principales y se ordenan según la varianza que recogen: El primer componente principal representaría la dirección de máxima varianza, el segundo componente la dirección de máxima varianza ortogonal al anterior componente y sucesivamente hasta tener todos los componentes necesarios.

PCA se puede usar para la reducción de dimensión si no se utilizan todos los componentes principales para representar los datos. De esta forma se están proyectando los puntos sobre el hiperplano representado por los componentes principales y la proyección sobre este hiperplano recoge la máxima varianza posible (Entre todos los hiperplanos de una dimensión dada).

Para encontrar los componentes principales se puede centrar la matriz de datos y realizar su descomposición en valores singulares:

$$X = UDV^T \tag{2.1}$$

donde X es la matriz de datos centrados, U contiene los vectores singulares izquierdos, V los vectores singulares derechos y D es una matriz diagonal con los valores singulares de X en la diagonal. Los componentes principales se encuentran en las columnas de XV y la varianza que recogen se puede calcular como la suma de los valores singulares asociado a los componentes entre la suma de todos.

2.3. MDS

Multidimensional scaling [13] es un método para representar una matriz de disimilaridades en un cierto número de dimensiones. Se puede usar como una técnica de reducción de la dimensionalidad si se calculan las disimilaridades con respecto a la dimensión original y luego se representan en un espacio de dimensión menor.

La idea de MDS es que las distancias euclídeas en el espacio de llegada sean lo más cercanas posibles a las disimilaridades en el espacio original. Si tenemos observaciones $x_1, x_2..., x_N$ y una función de disimilaridad d, la función de coste a optimizar en MDS podría ser:

$$C = \sum_{i \neq j} (d(x_i, x_j) - \|z_i - z_j\|)^2 \tag{2.2}$$

donde z_i y z_j son los representantes en el espacio de llegada de x_i y x_j respectivamente. Esta función puede optimizarse mediante descenso del gradiente[1].

[1]El descenso del gradiente es un algoritmo que encuentra un mínimo local en una función diferenciable. Para encontrar un mínimo se mueve un punto candidato en el sentido contrario al gradiente (la derivada) de manera proporcional a la magnitud del gradiente. Se itera este proceso hasta convergencia o hasta alcanzar un número de iteraciones prefijado

Sin embargo, si nuestras disimilaridades son distancias euclídeas existe una solución analítica. Si nuestras observaciones originales se representan por la matriz X sabemos que la matriz de Gram se define como $B = XX^T$. Podemos obtener la matriz de Gram mediante nuestra matriz de distancias al cuadrado D.

$$B = \frac{-1}{2} C_n * D * C_n \tag{2.3}$$

Donde $C_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ sirve para centrar la matriz de distancias. Por tanto si descomponemos la matriz de Gram podremos recuperar la matriz X y con ella las coordenadas en el espacio de llegada. En este caso (usando distancias euclídeas) el resultado será equivalente al obtenido mediante PCA.

2.4. ISOMAP

Los métodos que hemos visto hasta ahora solo son capaces de realizar proyecciones lineales de los datos. Es posible que nuestros datos se encuentren sobre una variedad de Riemman (*Riemman manifold*) de pocas dimensiones embebida en un espacio de dimensión mayor. En este contexto las distancias euclídeas carecen de sentido. Sin embargo, si pudiéramos obtener una distancia con sentido podríamos aplicar multidimensional scaling provechosamente. Esta es la idea que subyace tras ISOMAP (*complete isometric feature mapping*)[14].

ISOMAP pretende aplicar MDS sobre la distancia geodésica (esto es siguiendo la superficie del espacio embebido) entre nuestros datos (ver la figura 2.1). Como esta distancia geodésica no se conoce es necesario estimarla. Para estimar la distancia geodésica se siguen una serie de pasos:

1. Se construye un grafo pesado conectando las observaciones vecinas con pesos sus distancias euclídeas. Para determinar la vecindad puede escogerse una cercanía mínima o un número k de vecinos más próximos. El grafo resultante debe ser conexo.

2. Se estiman las distancias geodésicas entre puntos como las sumas de los pesos del camino mínimo entre ellos.

Se puede demostrar que esta estimación es buena si los puntos conforman una muestra uniforme del espacio embebido. Si esta hipótesis no se cumple o existe ruido en los datos ISOMAP puede no dar buenos resultados.

2.5. LLE

Locally linear embedding [15] es un método de reducción de dimensionalidad que intenta preservar las relaciones locales entre puntos. Se basa en la intuición de que si los datos son una muestra suficientemente grande de una variedad, un punto y sus vecinos

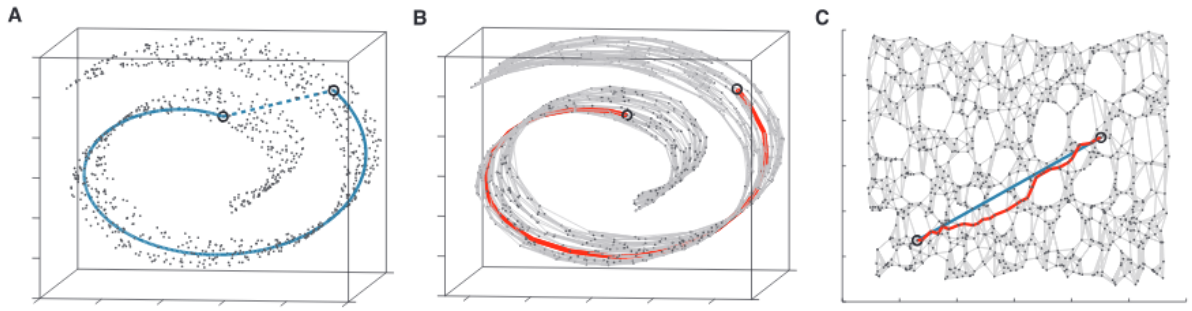

Figura 2.1: Aplicación de ISOMAP sobre el *swiss roll dataset*.En **A** se puede ver la discrepancia entre distancia geodésica y euclídea, en **B** en rojo se encuentra el camino mínimo que estima la distancia geodésica y en **C** una comparación entre la distancia geodésica real y la estimada. Imagen extraída de Tenenbaum et al. [14].

se situarán cerca de un área (*patch*) localmente lineal. Es decir, un punto puede ser aproximadamente reconstruido por una combinación lineal de sus vecinos. Los pesos de estas combinaciones lineales pueden obtenerse minimizando:

$$\mathcal{E}(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2 \tag{2.4}$$

Que es la suma de las distancias al cuadrado entre los puntos y sus reconstrucciones. La ecuación 2.4 esta restringida por $W_{ij} = 0$ si X_j no es vecino de X_i y por $\sum_j W_{ij} = 1$.

Una consideración importante es que para un punto determinado los pesos son invariantes a reescalados, rotaciones y translaciones del punto y sus vecinos. Es decir los pesos codifican propiedades locales de los vecindarios y no dependen del punto de referencia.

Una consecuencia de esto es que los pesos sirven para reconstruir las coordenadas de los puntos sobre la variedad. Si el punto X_i se mapea al punto sobre la variedad Y_i podemos obtener los puntos Y_i minimizando:

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2 \tag{2.5}$$

donde los W_{ij} se han obtenido con 2.4. Si hacemos que los puntos Y_i estén centrados y tengan covarianza 1 podemos resolver 2.5 como un problema de vectores propios (eigenvectors).

Podemos reescribir 2.5 como:

$$\Phi(Y) = Y^T (I - w)^T (I - w) Y = Y^T W Y \tag{2.6}$$

Usando un multiplicador de lagrange para la condición $\frac{1}{n}Y^TY = 1$:

$$\begin{aligned} L(Y,\mu) &= Y^TMT - \mu(\frac{1}{n}Y^TY - 1) \\ \frac{dL}{dY} &= 2MY - 2\mu\frac{1}{n}Y = 0 \\ MY &= \frac{\mu}{n}Y \end{aligned} \tag{2.7}$$

Por tanto Y es un valor propio de M y para minimizar la ecuación 2.5 necesitamos los vectores propios de M como coordenadas (descartando el menor que tendrá valor propio 1 por nuestras restricciones).

2.6. Laplacian eigenmaps

Laplacian eigenmaps [16] es una metodología que se parece a LLE en el sentido de intentar preservar la localidad de los datos. En este caso la localidad se basa en la distancia entre puntos vecinos. El objetivo es crear una representación de menor dimensión minimizando las distancias entre los puntos y sus vecinos de una manera ponderada (dando más importancia a los vecinos más cercanos). Las relaciones de vecindad se plasman en un grafo y usando teoría espectral de grafos resolvemos el problema de minimización.

Lo primero que se necesita es definir la vecindad ya sea fijando el número de vecinos o por una distancia umbral. Después se da un peso a cada relación de vecindad usando un kernel gaussiano:

$$w_{ij} = e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}} \tag{2.8}$$

con lo que logramos una matriz de adyacencia pesada W. La función que se quiere minimizar es:

$$C = \sum \left\|y_i - y_j\right\|^2 w_{ij} \tag{2.9}$$

donde y_i es la representación del punto x_i en la dimensión reducida. Operando con este coste:

$$\begin{aligned} \sum \left\|y_i - y_j\right\|^2 w_{ij} = \sum y_i^2 + y_j^2 - 2y_i{y_j}^2 w_{ij} = \\ \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2\sum_{i,j} y_i y_j w_{ij} = 2Y^TLY \end{aligned} \tag{2.10}$$

donde D es la matriz de grado del grafo (la diagonal contienen los grados de los vértices) y L es la matriz laplaciana del grafo ($L = D - W$). Por tanto para minimizar la ecuación 2.9 basta con obtener los vectores propios de L (que contendrán las nuevas coordenadas de los puntos).

2.7. T-SNE

t-Distributed Stochastic Neighbor Embedding es una técnica desarrollada por Geoffrey Hinton y Laurens van der Maaten en el año 2008. En su artículo originario[17] se presenta

como una técnica especializada en la visualización de conjuntos de datos con muchas dimensiones.

Este método al igual que otros que hemos visto (ISOMAP y LLE) intenta mapear elementos desde un espacio de muchas dimensiones a uno de menos conservando las relaciones de vecindad entre elementos (una versión probabilística de estas relaciones). Para explicar t-SNE vamos a realizar un recorrido histórico del desarrollo del método.

2.7.1. SNE

Si queremos conservar las relaciones de vecindad entre puntos se pueden pensar diferentes opciones. SNE [18] opta por una versión probabilística basada en una medida de distancia. Por cada punto i se genera una distribución de probabilidad sobre todos los puntos de la forma:

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2 * \sigma_i^2)}{\sum_{i \neq k} exp(-||x_i - x_k||^2/2 * \sigma_i^2)} \tag{2.11}$$

Con $p_{ii} = 0$.

Habitualmente se describe esta ecuación como la probabilidad de que el punto i escoja al punto j como su vecino si los vecinos se escogieran de manera proporcional a su densidad en una $N(i, \sigma_i)$.

La idea detrás de la función es que todos los puntos son vecinos pero algunos vecinos son más importantes que otros. Las relaciones con los vecinos importantes son las que queremos conservar en el espacio de pocas dimensiones. Esta función permite dar mucho peso a los vecinos más cercanos (los más importantes) y menos a los lejanos. Cuantos vecinos son importantes queda determinado por el valor de σ_i. Los valores adecuados de σ_i se encuentran fijando una perplejidad y realizando una búsqueda binaria[2] para encontrar las σ que la produzcan. La perplejidad es una medida suavizada del número de vecinos que son importante para cada punto y queda definida como:

$$Perp(P_i) = 2^{H(P_i)} \tag{2.12}$$

Con $H(P_i)$ siendo la entropía de Shannon:

$$H(P_i) = -\sum_j p_{ij} \log_2 p_{ij} \tag{2.13}$$

En la figura 2.2 podemos apreciar el comportamiento de las distribuciones en función de la perplejidad.

Tenemos por tanto una matriz P en la que cada fila es una distribución de probabilidad que representa las relaciones de vecindad en el espacio de partida (de alta dimensión). Necesitamos una matriz análoga Q que cumpla el mismo rol en el espacio de llegada (de

[2]Recordemos que una búsqueda binaria se puede realizar sobre una función monotónicamente creciente

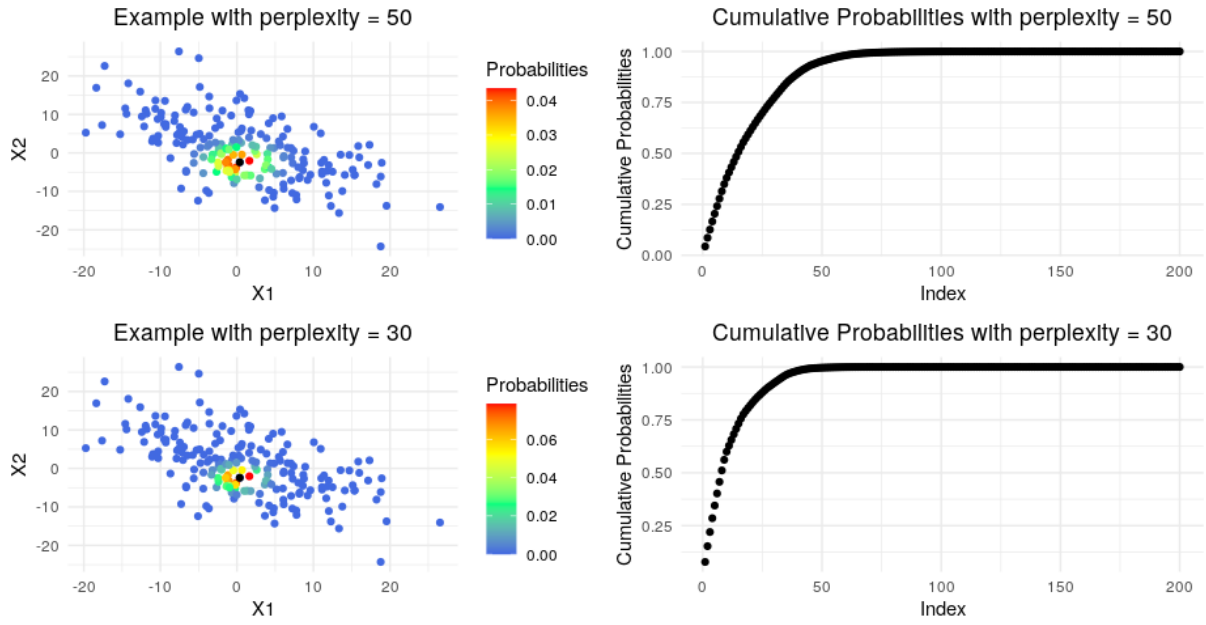

Figura 2.2: Relación entre perplejidad y vecindad. El número de puntos con una probabilidad relevante cambia con la perplejidad.

baja dimensión). Los elementos de Q quedan definidos como:

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{i \neq k} exp(-||y_i - y_k||^2)} \tag{2.14}$$

Con $q_{ii} = 0$. Los elementos q_{ij} siguen la misma formula que los p_{ij} si hacemos $\sigma_i = \frac{1}{\sqrt{2}}$.

Queremos que al reducir la dimensionalidad de los puntos, P quede conservada. Dicho de otra manera P y Q deben ser lo más similares posible. Una forma de hacer esto es usar la suma de las divergencias de Kullback-Leibler entre las distribuciones:

$$C = \sum_i KL(P_i||Q_j) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.15}$$

Esta medida de coste es asimétrica. Como consecuencia de esto cuesta más tener un p_{ij} grande representado por un q_{ij} pequeño que la situación inversa. Es decir lo importante es conservar los vecinos próximos (la estructura local). La minimización del coste se hace mediante descenso del gradiente. El gradiente de la ecuación 2.15 sigue la forma:

$$\frac{dC}{dy_i} = 2 \sum_j (p_{ij} - q_{ij} + p_{ji} - q_{ji})(y_i - y_j) \tag{2.16}$$

Los puntos en baja dimensión se inicializan como muestras de una Normal centrada en el origen con poca varianza y se actualizan según la formula:

$$Y(t) = Y(t-1) + \eta \frac{dC}{dy_i} + \alpha(Y(t-1) - Y(t-2)) \tag{2.17}$$

Donde η es la tasa de aprendizaje y α el momento.

Además en el artículo original se recomienda añadir ruido gaussiano a los puntos después de actualizarlos. La cantidad de ruido debe reducirse con el tiempo. Esto genera una especie de *simulated annealing* que ayuda a evitar mínimos locales.

SNE tiene muchos parámetros que interactúan entre ellos y además tiene un componente estocástico. Esto hace que encontrar un conjunto de parámetros con un buen comportamiento para un conjunto de datos requiera bastante tiempo.

2.7.2. Variantes de SNE

SNE simétrico[19]

En vez de minimizar la suma de las divergencias de Kullback-Lieber se pueden definir las matrices P y Q como una única distribución y minimizar únicamente la divergencia entre ellas. Los elementos q_{ij} quedan definidos como:

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_k - y_l||^2)} \tag{2.18}$$

y los elementos p_{ij} se simetrizan:

$$p_{ij} = \frac{p_{ij} + p_{ji}}{2n} \tag{2.19}$$

donde los elementos p_{ij} en la fracción se derivan con la fórmula 2.11. La ventaja es que ahora tenemos un gradiente más sencillo de optimizar:

$$\frac{dC}{dy_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \tag{2.20}$$

SNE uniforme[19]

SNE sufre del denominado *Crowding Problem*. Esto es que en un espacio de mayor dimensión se pueden colocar más puntos equidistantes (para entender esto piensa en como escala la superficie de una esfera frente a la de un círculo). Este hecho obliga a SNE a representar distancias moderadas con distancias grandes. Esto provoca que para una gran cantidad de puntos $p_{ij} > q_{ij}$. Por tanto, para minimizar el coste 2.16 los puntos son comprimidos en el centro de la representación.

Una primera idea para arreglar es mezclar los q_{ij} con una uniforme:

$$q_{ij} = \frac{(1 - \lambda) exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_k - y_l||^2)} + \frac{2\lambda}{N(N-1)} \tag{2.21}$$

De esta manera los q_{ij} no pueden bajar de un cierto valor y se evita el *Crowding Problem*. El problema de SNE uniforme es que no se puede aplicar directamente puesto que en las fases iniciales impide que los clusters puedan entrecruzarse. Por tanto, primero se realizaría SNE simétrico y posteriormente SNE uniforme.

2.7.3. tSNE

Por el contexto probabilístico del método existe una manera inteligente de solventar el *Crowding Problem*. Si utilizamos una distribución con colas más pesadas en el espacio de llegada, podemos representar distancias moderadas con distancias grandes y que $p_{ij} \approx q_{ij}$. La distribución de colas pesadas escogida es la t de student y las q_{ij} quedan definidas como:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{2.22}$$

y el gradiente como:

$$\frac{dC}{dy_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \tag{2.23}$$

Como conclusión podemos decir que t-SNE es un buen método de reducción de la dimensionalidad si queremos hacer una representación. Si queremos reducir la dimensión para un proceso posterior (un clustering por ejemplo) puede no ser adecuado por dos motivos:

1. No nos da una función con la que mapear futuras observaciones. Esto es un problema salvable puesto que se puede optimizar el gradiente 2.23 permitiendo únicamente a la nueva observación moverse [20].

2. Al aumentar el número de dimensiones a las que se quiere reducir el conjunto de datos se entorpece la preservación de las cualidades locales de los datos [17]. Es decir, funciona bien si queremos 2 o 3 dimensiones para visualizar pero si nuestro objetivo es otro puede no resultar adecuado.

3. La complejidad del algoritmo tanto temporal como espacial es $\mathcal{O}(n^2)$ [17] lo que puede ser problemático para conjuntos de datos con muchas muestras.

2.8. UMAP

UMAP (*Uniform Manifold Approximation and Projection*)[21] es un algoritmo de reducción de dimensión que intenta preservar la estructura local de los datos. Cuenta con una base teórica basada el topología algebraica y geometría Riemanniana. Da resultados competitivos comparada con t-SNE escalando computacionalmente mejor y preservando más de la estructura global de los datos.

2.8.1. Base matemática

Las matemáticas detrás de UMAP son muy abstractas. A continuación se presenta una descripción de las matemáticas que soportan el método [22].

El problema que se quiere resolver es el de encontrar una representación de un *dataset* en R^N en un espacio de menor dimensión. Para ello suponemos que los datos son muestras de una variedad de Riemman M que luego se mapean al espacio de dimensión N. Para facilitar las cosas realizamos una serie de asunciones:

1. Los datos son una muestra uniforme de M.

2. M está localmente conectado.

3. Hay suficientes datos como para que no haya puntos aislados. Un punto está aislado si es el único punto en un componente de M (M puede tener múltiples componentes no conectados entre si).

4. Tenemos una distancia[3] en M que es localmente constante.

Puesto que se quiere conservar la estructura de los datos queremos aproximar de alguna manera la distancia geodésica entre nuestros puntos. Tenemos una distancia en el espacio de ambiente ($\mathbb{R}^N$) y gracias a la asunción 4 también una en la variedad de Riemman. Con estas dos distancias podemos aproximar la distancia geodésica:

Lema 2.8.1 *Sea (M,d_M[4]) una variedad riemanniana en un ambiente $\mathbb{R}^N$ y $p \in M$ un punto. Suponemos que d_M es localmente constante en un vecindario abierto U tal que $p \in U$ de forma que d_M es una matriz diagonal constante con coordenadas ambiente. Sea B una bola en U, conteniendo p, con volumen $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ con respecto a d_M. Entonces la distancia geodésica en M desde p a un punto $q \in B$ es $\frac{1}{r} d_{\mathbb{R}^N}(p,q)$, donde r es el radio de B en $\mathbb{R}^N$ y $d_{\mathbb{R}^N}(p,q)$ la distancia de p a q en $\mathbb{R}^N$.* [5]

Puesto que asumimos que nuestros datos están uniformemente distribuidos en M, una bola B de un tamaño fijo contendrá aproximadamente el mismo número de puntos independientemente de donde se sitúe dentro de M. Por tanto podemos elegir un número de vecinos k y aproximar las distancias geodésicas (usando el lema 2.8.1) dando a r el valor de la distancia al k-vecino más cercano. De esta manera tenemos una forma de aproximar la distancia topológica desde un punto de M a cualquier otro punto de M y además bajo esta distancia garantizamos la uniformidad de los datos sobre M. El problema es que cada punto tiene su propia distancia (puesto que la distancia al k-vecino más cercano varia) lo cual implica que $d(a,b) \neq d(b,a)$. Es decir tenemos un conjunto de nociones de distancia incompatibles entre sí que debemos compatibilizar de algún modo. Para ello necesitamos conocer algunos fundamentos de teoría de las categorías y sobre conjuntos difusos de símplices.

[3]Una distancia en X es una función $d : X * X \to R$ tal que $d(x,y) \geq 0$,$d(x,y) = d(y,x)$,$d(x,y) = 0 if f x = y$, y $d(x,z) \leq d(x,y) + d(y,z)$

[4]La distancia asumida en la asunción 4

[5]La demostración de este lema puede encontrarse en el artículo original de UMAP.

Teoría de las categorías para UMAP

La teoría de las categorías es una rama de las matemáticas que intenta generalizar y unificar los conceptos del resto de las matemáticas. Comúnmente se la llama las matemáticas de las matemática. En el contexto de UMAP la teoría de las categorías nos da una adjunción entre conjuntos difusos de símplices y espacios pseudométricos extendidos. Para entenderlo empezaremos definiendo algunos conceptos:

Una categoría C es una colección de objetos y una colección de morfismos (los morfismos son arcos dirigidos que comienzan y terminan en un objeto). Además se cumplen una serie de propiedades:

- Los morfismos se pueden componer. Es decir si tenemos los objetos X,Y,Z y los morfismos $f : X \to Y$ e $g : Y \to Z$ entonces tendremos el morfismo composición $gf : X \to Z$.
- Todos los objetos tienen un morfismo identidad $Id : X \to X$

Dada una categoría C podemos tener su categoría opuesta C^{op} que contiene los mismos objetos que C pero en la que los morfismos se han invertido (se ha invertido la dirección de los arcos entre objetos).

Un functor es un mapeo entre categorías (si C y D son categorías $F : C \to D$ es un functor entre ellas). Los functores deben preservar los morfismos identidad y preservar la composición de los morfismos.

También existen mapeos entre functores que se conocen como transformaciones naturales. Sean C y D categorías con $F : C \to D$ y $G : C \to D$ functores entre ellas. Una transformación natural $\alpha : F \to G$ asocia a cada objeto X en C un morfismo $\alpha_x : F(X) \to G(X)$ de tal forma que para cada morfismo $f : X \to Y$ en C se cumple $G(f) \circ \alpha_x = \alpha_y \circ F(f)$ en D.

Por último definimos el concepto de adjunción. Una adjunción es un par de functores entre dos categorías que son *casi* inversos el uno del otro. Si C y D son categorías y $L : C \to D$ y $R : D \to C$ son functores entre ellas se dice que L y R son una adjunción si para cualquier $X \in C$ y $Y \in D$ existe un isomorfismo:

$$\alpha_{X,Y} : Homset(X, R(Y)) \to Homset(L(X), Y) \tag{2.24}$$

donde $Homset$ es el conjunto de morfismos entre ambos objetos.

Complejos difusos de símplices para UMAP

Un símplice es la versión generalizada de un triángulo. Un 0-símplice es un punto, un 1-símplice es un segmento, un 2-símplice es un triángulo, un 3-símplice es un tetraedro y sucesivamente. Los Complejos de símplices son colecciones de símplices tal que si un símplice está en esta colección también lo han de estar los símplices que conforman sus caras(Por ejemplo si tenemos un 2-símplice (un triángulo) también debemos tener los 3 1-símplices que conforman sus caras (los segmentos que conforman el triángulo). Podemos

describir un complejo de símplices X mediante una serie de conjuntos X^n que contienen los n-símplices. Por ejemplo dos triángulos unidos por una de sus caras podría describirse como:

$$X_0 = \{\{a\}, \{b\}, \{c\}, \{d\}\}$$
$$X_1 = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{c, d\}\}$$
$$X_2 = \{\{a, b, c\}, \{a, c, d\}\}$$

Es decir estamos representando un complejo de símplices mediante un conjunto de símplices. Podemos describir esta idea de conjunto de símplices en el lenguaje de la teoría de las categorías. Sea Δ la categoría de conjuntos ordenados $[n]$ con morfismos que preservan el orden $[m] \to [n]$. Un conjunto de símplices es un functor $X : \Delta^{op} \to SET$, con SET siendo la categoría de los conjuntos con morfismos funciones entre conjuntos. Esta definición es difícil de entender pero si nos fijamos, un símplice puede verse con un conjunto ordenado (los vértices siguen un cierto orden) y los morfismos $[n] \to [m]$ de Δ^{op} puede verse como la extracción de símplices más pequeños desde uno más grande. El functor X mapea $[n]$ a $[X_n]$ y el mapeo de los morfismos fuerza a que se cumpla que para que un conjunto de símplices contenga un n-símplice debe contener todos sus componentes.

Para el caso de los complejos difusos de símplices sustituimos los conjuntos por conjuntos difusos. Un conjunto difuso es un conjunto en el que hay una función de membresia para los elementos $\mu : elemento \to [0, 1]$ que puede pensarse como la probabilidad de pertenencia al conjunto del elemento. Con esto en mente se puede definir un conjunto difuso de símplices como un functor $X : \Delta^{op} \to Fuzz$, con $Fuzz$ siendo la categoría de conjuntos difusos.

Convirtiendo espacios métricos a conjuntos difusos de símplices

Recordemos lo que queríamos hacer. Tenemos una manera de estimar las distancias geodésicas entre los puntos mediante el lema 2.8.1. Pero esto nos da nociones diferentes de distancia según el punto de referencia que usemos. Lo que queremos es poder unificar toda las nociones de distancias en una sola. Posteriormente esta noción de distancia unificada la podemos intentar conservar en nuestro espacio de dimensión reducida.

El primer problema que tenemos es que nuestras nociones de distancia no nos aseguran la asunción 2 (la variedad M está localmente conectada) y además no están completamente definidas pues solo nos da la distancia del punto de referencia a los demás. Para arreglarlo forzamos a que la distancia con el vecino más cercano sea 0 y las distancias desconocidas las suponemos infinitas:

$$d_{x_i}(x_j, x_k) = \begin{cases} \frac{1}{r_i}(d_{\mathbb{R}^N}(x_j, x_k) - \rho_i) & \text{si } j = i \text{ o } k = i \\ \infty & \text{en otro caso} \end{cases} \tag{2.25}$$

De esta manera conseguimos un espacio pseudo-métrico extendido[6].

[6]Un espacio pseudo-métrico se compone de un conjunto X y una función $d : X * X \to \mathbb{R} \cup \{\infty\}$ de forma que $d(x, y) \geq 0$, $d(x, y) = 0$, $d(x, y) = d(y, x)$ y $d(x, z) = \infty$ o $d(x, z) \leq d(x, y) + d(y, z)$

Sea **EPMet** la categoría de espacios pseudo-métricos extendidos en la que los morfismos son mapeos no expansivos[7]. Sea **FinEPMet** la sub-categoría de **EPMet** que solo contiene espacios métricos finitos (con un número finito de puntos). Nuestros espacios (los extraídos de los datos) pertenecen a **FinEPMet**. Sea **sFuzz** la categoría de conjuntos difusos de símplices con morfismos transformaciones naturales entre ellos. Sea **Fin-sFuzz** la subcategoría de **sFuzz** que consiste de conjuntos difusos de símplices con un número finito de símplices no degenerados [8]

Teorema 2.8.2 *Existe una adjunción entre* ***FinEPMet*** *y* ***Fin-sFuzz*** *dada por* $FinSing$: ***FinEPMet*** $\rightarrow$ ***Fin-sFuzz*** *y por* $FinReal$: ***Fin-sFuzz*** $\rightarrow$ ***FinEPMet****, con* $FinReal \dashv FinSing$

El functor $FinReal$: **Fin-sFuzz** $\rightarrow$ **FinEPMet** toma un conjunto finito se símplices X y lo transforma en un espacio métrico T cuyos elementos son los 0-símplices de X. La métrica de T se define como:

$$d_T(x, y) = min_{U \in A} - log(\mu(U)) \tag{2.26}$$

Donde μ es la función de membresía de X y A es el conjunto de subconjuntos de T que contienen x e y. Es decir que los miembros de A también pueden ser entendidos como posibles símplices y por tanto se les puede aplicar μ.

Por su parte $FinSing$: **FinEPMet** $\rightarrow$ **Fin-sFuzz** convierte un espacio pseudo-métrico extendido y finito Y a un conjunto difuso de símplices. Para cada subconjunto de puntos (que definen un símplice en el espacio de llegada) la función de membresía es:

$$\mu(\{x_0, ..., x_n\}) = min_{i,j} e^{-d(x_i, x_j)} \tag{2.27}$$

Con las herramientas que tenemos hasta ahora podemos transformar un conjunto de datos a un conjunto de espacios métricos y estos en un conjunto de conjuntos difusos de símplices. Sin embargo, queremos un único conjunto de símplices que resuma la totalidad de los datos por lo que tomamos la unión de los conjuntos de símplices:

$$\bigcup_{i=1}^{n} FinSing(d_{x_i}) \tag{2.28}$$

donde d_{x_i} es el espacio métrico definido por x_i. Existen diferentes posibilidades para definir la unión de conjuntos difusos. Puesto que todos nuestros conjuntos tienen los mismos objetos una definición razonable para la función de membresía de la unión de dos conjuntos difusos es $(\mu \cup \upsilon)(a) = \mu(a) \perp \upsilon(a)$ con $\perp$ alguna t-conorma. La t-conorma usada en UMAP es $x \perp y = x + y - xy$ que es equivalente a la probabilidad de la unión de sucesos independientes.

[7] Para un mapeo f la distancia entre $f(a)$ y $f(b)$ no es mayor a la que hay entre a y b

[8] Un símplice es degenerado cuando tiene puntos coincidentes. Por ejemplo un 2-símplice(triángulo) en el que dos de sus puntos son los mismos es más un 1-símplice(segmento) que un 2-símplice(triángulo).

Reduciendo la dimensión

Tenemos una representación de nuestros datos mediante un conjunto difuso de símplices. El problema se reduce a evaluar la similitud entre este conjunto difuso de símplices y un conjunto candidato en baja dimensión. Definimos la entropía cruzada C de dos conjuntos difusos con los mismos elementos, (A, μ) y (A, υ) como:

$$C((A,\mu),(A,\upsilon)) = \sum_{a\in A}\left(\mu(a)log\left(\frac{\mu(a)}{\upsilon(a)}\right) + (1-\mu(a))log\left(\frac{1-\mu(a)}{1-\upsilon(a)}\right)\right) \tag{2.29}$$

Esta fórmula la podemos interpretar como dos partes:

- $\mu(a)log\left(\frac{\mu(a)}{\upsilon(a)}\right)$ aporta una fuerza atractiva y se optimiza al representar elementos cercanos en el espacio de origen como elementos cercanos en el de llegada.
- $(1-\mu(a))log\left(\frac{1-\mu(a)}{1-\upsilon(a)}\right)$ aporta una fuerza repulsiva y se optimiza al representar elementos lejanos en el espacio de origen como elementos lejanos en el de llegada. La ecuación 2.29 sirve para comparar dos conjuntos difusos sin embargo nuestro conjunto de símplices se compone de varios conjuntos difusos (uno para los 0-símplices, otro para los 1-símplices etc.). Si se quisiera podríamos calcular la entropía cruzada de todos los pares de conjuntos y hacer una suma de los mismos. Desde un punto de vista práctico se comparan únicamente los 1-símplices por razones computacionales y además porque el teorema del nervio nos asegura que los 1-símplices ya recogen la topología del espacio.

2.8.2. Algoritmo

Una vez descritas las matemáticas tras UMAP es fácil imaginarse el algoritmo para implementarlo(ver algoritmo 1).

Algorithm 1 UMAP

```
function UMAP(X, n, d,min-dist,n-épocas)
    for all x ∈ X do
        fs-set[x] = LocalFuzzySimplicialSet(X,x, n)
    top-rep = ⋃_{x∈X} fs-set[x]
    Y = InicializarEmbedding(top-rep,d)
    Y = OptimizarEmbedding(top-rep,Y, min-dist, n-épocas)
    return Y
```

El cálculo de los conjuntos difusos de símplices necesita del cálculo de las estimaciones de las distancias geodésicas. Como vimos en el lema 2.8.1 esto se hace corrigiendo la distancia euclídea con la distancia al k-vecino más cercano pero en la práctica se usa una versión suavizada de esta (ver la fuente original[21] para obtener más detalles del procedimiento). Por lo demás, para inicializar los datos en dimensión reducida se usa

Laplacian eigenmaps y para la optimización de la reducción de dimensión se usa descenso del gradiente estocástico sobre la función de coste (ver ecuación 2.29).

UMAP ha desplazado a T-SNE como método considerado estado del arte en reducción de dimensionalidad. Esto se debe a que el algoritmo de UMAP tiene menor complejidad ($\mathcal{O}O(n^1$,14) frente al $\mathcal{O}(n^2)$ de t-SNE) y no tiene problemas para reducir conjuntos de datos a más de 3 dimensiones.

2.9. Evaluación de la reducción de dimensión

Habitualmente es difícil escoger la técnica de reducción de dimensionalidad adecuada para un problema específico. Si además de escoger la técnica de reducción de dimensionalidad nos planteamos la optimización de los parámetros el problema se agrava. Y si además nos planteamos un problema de clasificación en el que tenemos que optimizar la combinación de reducción de dimensión y clasificador (con todos los parámetros) el problema puede ser inabordable. La metodología habitual escoge el método de reducción de dimensionalidad por un criterio cualitativo y posteriormente se intenta optimizar en conjunto al clasificador escogido. En este TFG seguiremos el trabajo de Espadoto et al. [23] en el que se plantea un índice de evaluación de las técnicas de reducción de dimensionalidad basado en 5 medidas cuantitativas:

1. Fiabilidad(*Trustworthines*)(M_t): Mide la proporción de puntos cercanos en los datos originales que también son cercanos en la proyección. Toma un valor entre 0 y 1, siendo 1 lo mejor y se define como:

$$1 - \frac{2}{NK(2n - 3K - 1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(K)}} (r(i,j) - K) \tag{2.30}$$

   Siendo $U_i^{(K)}$ los puntos que son K vecinos de i en la proyección pero no en la dimensión original, y $r(i,j)$ el ranking del punto j dentro de los vecinos de i en la proyección. Se ha escogido una $K = 7$ para este trabajo.

2. Continuidad(M_c): Mide la proporción de puntos cercanos en la proyección que también lo están en la dimensión original. Toma un valor entre 0 y 1, siendo 1 lo mejor y se define como:

$$1 - \frac{2}{NK(2n - 3K - 1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(K)}} (\hat{r}(i,j) - K) \tag{2.31}$$

   Siendo $V_i^{(K)}$ los puntos que son K vecinos de i en la dimensión original pero no en la proyección, y $\hat{r}(i,j)$ el ranking del punto j dentro de los vecinos de i en la dimensión original. Siguiendo el criterio del artículo original [23] se ha escogido una $K = 7$ para este trabajo [23].

3. Stress normalizado(M_σ): Es una medida de la conservación de la distancia entre puntos. Pueden utilizarse diferentes distancias pero habitualmente se usa la distancia euclídea. Se define como:

$$\frac{\sum_{ij}(d(x_i,x_j)-d(P(x_i),P(x_j)))^2}{\sum_{ij}(d(x_i,x_j))^2} \tag{2.32}$$

   Con d siendo la distancia y $P(X_i)$ la proyección del punto i. Menores valores implica mayor conservación de distancia (siendo 0 lo mejor). En Espadoto et al. [23] aseguran que esta medida solo puede tomar valores entre 0 y 1 aunque observando la fórmula concluimos que esto es erróneo (Si las distancias en la proyección son lo suficientemente diferentes a las originales se obtienen valores mayores a 1).

4. *Neighborhood hit(M_{NH})*: Proporción de K-vecinos de un punto i en la proyección que tienen la misma etiqueta que el punto i promediado para todos los puntos. Siguiendo el criterio del artículo original [23] se ha escogido una $K = 7$ para este trabajo [23].

5. *Shepard goodness(M_S)*: El diagrama de Shepard es un *scatterplot* de las distancias originales vs las distancias en la proyección. El shepard goodnes es una medida resumen del diagrama de Shepard que consiste en computar la correlación de spearman entre las distancias. Puede tomar valores entre -1 y 1

La medida resumen planteada por Espadoto et al. [23] es:

$$\mu = \frac{1}{5}(M_{NH} + M_t + M_c + (1 - M_\sigma) + M_S) \tag{2.33}$$

Esto lo justifican alegando que todas las medidas toman valores en $[0, 1]$ y que las consideran igual de importantes. Como sabemos que M_σ puede ser mayor que 1 y que M_S menor que 0 redefinimos la medida como:

$$\mu = \frac{1}{5}(M_{NH} + M_t + M_c + (1 - M_\sigma^{'}) + M_S^{'}) \tag{2.34}$$

con:

$$M_\sigma^{'} = \begin{cases} M_\sigma & \text{si } M_\sigma < 1 \\ 1 & \text{en otro caso} \end{cases} \tag{2.35}$$

y:

$$M_S^{'} = \begin{cases} M_S & \text{si } M_S > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.36}$$

Pensamos que si un método de reducción de dimensionalidad tiene $M_\sigma \geq 1$ o $M_S \leq 0$ (para una dimensión concreta), este método no pretende optimizar ninguna de estas medidas y no requiere de mayor penalización. Además, de esta manera conseguimos que nuestra medida resumen tome valores en $[0, 1]$ facilitando su interpretación. De las medidas escogidas la fiabilidad, la continuidad y el *neighborhood hit* puntúan la conservación de la estructura local de los datos y las otras dos: el stress y el *Sheppard goodnes* puntúan la conservación de la estructura global.

Capítulo 3

Clasificación

En este capítulo se exponen de manera sucinta los métodos de clasificación que se usarán en la parte experimental del trabajo.

3.1. SVM

Las máquinas de vectores soporte o SVM (*Support vector machines*) son un método de clasificación que encuentra el hiperplano con mayor margen entre dos clases. Tenemos L puntos de entrenamiento y cada punto x_i tiene una clase $y_i \in \{-1, 1\}$. Supongamos que las dos clases son linealmente separables por un hiperplano de la forma:

$$wx + b = 0$$

donde w es normal al hiperplano y $\frac{b}{\|w\|}$ es la distancia del hiperplano al origen. Es decir queremos seleccionar w y b tales que:

$$x_i w + b \geq 1 \text{ si } y_i = 1$$
$$x_i w + b \leq -1 \text{ si } y_i = -1$$

lo que podemos combinar en:

$$y_i(x_i w + b) - 1 \geq 0$$

El margen del hiperplano es $\frac{1}{\|w\|}$ y por tanto podemos plantear el problema que optimiza SVM como:

$$min \, \|w\| \text{ sujeto a } y_i(x_i w + b) - 1 \geq 0 \; \forall i \tag{3.1}$$

Este problema es extensible al caso en el que las clases no sean linealmente separables por unos pocos puntos. Esto se consigue añadiendo una penalización por cada fallo:

$$min \, \|w\| + C \sum_i \xi_i \text{ sujeto a } y_i(x_i w + b) - 1 + \xi_i \geq 0 \; \forall i \tag{3.2}$$

Donde ξ_i es 0 si el punto i esta bien clasificado o es la distancia al hiperplano que contiene los vectores soporte (estos son los puntos más cercanos al hiperplano de separación) de su

clase en otro caso. C es un parámetro que permite controlar la penalización por clasificar erróneamente.

Además, SVM puede separar datos que no son linealmente separable mediante funciones kernel. Este método se basa en que en muchos problemas de clasificación, los datos no son separables en la dimensión original, pero si se mapean de manera adecuada a una dimensión mayor si lo son. Es decir los puntos x_i no son linealmente separables pero los puntos $\phi(x_i)$ si lo son para un $\phi()$ adecuado. En el caso de SVM no hace falta calcular el mapeo explícitamente puesto que solo necesitamos el producto escalar entre los puntos. Por tanto nos basta una función:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{3.3}$$

Algunas funciones kernel de ejemplo son:

- Kernel de base radial:

$$k(x_i, x_j) = e^{-\frac{\left\| x_i - x_j \right\|^2}{2\sigma^2}} \tag{3.4}$$

- Kernel polinómico:

$$k(x_i, x_j) = (x_i * x_j + a)^b \tag{3.5}$$

3.2. Regresión logística

La regresión logística es un modelo de regresión utilizado para estudiar la relación entre una respuesta binaria y un grupo de variables independientes. La idea básica es ajustar un modelo de la forma:

$$\ln(\frac{\pi}{1-\pi}) = \beta_0 + \beta_1 X_1 + ... + \beta_k X_k \tag{3.6}$$

Donde π es la probabilidad de éxito (es decir que la respuesta sea de una clase que hemos fijado como éxito), las β son los coeficientes y las X las variables independientes. Moviendo los términos de la ecuación llegamos a:

$$P(exito|X = x; \beta) = \frac{e^{\beta_0 + \beta_1 X_1 + ... + \beta_k X_k}}{1 + e^{\beta_0 + \beta_1 X_1 + ... + \beta_k X_k}} \tag{3.7}$$

donde hemos sustituido π por $P(exito|X = x; \beta)$ para hacer explicita la dependencia de la probabilidad con respecto a los coeficientes y los valores. Para ajustar el modelo basta con encontrar los coeficientes que maximicen la verosimilitud. La función de logverosimilitud es:

$$l(\beta) = \sum_{i=1}^{N} \log(P(exito|X = x_i; \beta)) \tag{3.8}$$

Para que la regresión logística sea un método de clasificación basta con dar un umbral a la probabilidad para asignar un punto a una clase o a la otra. Para extender el método al caso multiclase existen dos métodos:

- Ajustar varios modelos para dos clases y clasificar una instancia con la clase más votada

- Elegir una clase y ajustar un modelo por cada clase restante teniendo en cuenta la restricción de que la probabilidad total debe ser 1. De esta manera la probabilidad de una clase i sería:

$$P(i|X = x; \beta) = \frac{e^{\beta_i X}}{1 + \sum_{j \neq w} e^{\beta_j X}} \tag{3.9}$$

  donde w es la clase base. La probabilidad de la clase base sería:

$$P(w|X = x; \beta) = \frac{1}{1 + \sum_{j \neq w} e^{\beta_j X}} \tag{3.10}$$

3.3. Perceptrón multicapa

El perceptrón multicapa es una de las redes neuronales más simples. El elemento básico de las redes neuronales artificiales es la neurona artificial. La neurona artificial imita de una manera simplificada la neurona biológica. El modelo de neurona que se utiliza es el propuesto por McCulloch-Pitts [24]. Según ese modelo la neurona se comporta como una función matemática:

$$y = \sigma\left(\sum_{j=0}^{m} w_j x_j\right) \tag{3.11}$$

donde x_j son los valores de entrada de la neurona, w_j los pesos y σ una función de activación (funciones de activación habituales son la sigmoide o la softmax). Estas neuronas se pueden acoplar de diferentes maneras para generar modelos más complejos. Una manera sencilla de hacerlo es organizar las neuronas en capas y conectar las salidas de las neuronas de una capa a las entradas de las neuronas de la capa siguiente. Si la red neuronal solo tiene dos capas (una de entrada y otra de salida) se la denomina perceptrón y si tiene más se la denomina perceptrón multicapa (como se muestra en la figura 3.1).

Los pesos de las neuronas en un perceptrón multicapa se optimizan utilizando un conjunto de datos de entrenamiento (donde conocemos la salida que debe dar el perceptrón para un conjunto de entradas) y el algoritmo de retropropagación (*backpropagation*). La retropropagación no es más que una manera eficiente de calcular el gradiente de una función de coste con respecto a los pesos. La idea es calcular el coste para un valor de entrada y posteriormente calcular el gradiente para los pesos desde la capa de salida hacia la de entrada utilizando la regla de la cadena de manera adecuada. Una vez obtenidos los gradientes usando retropropagación los pesos pueden ser actualizados utilizando cualquier variante de descenso del gradiente.

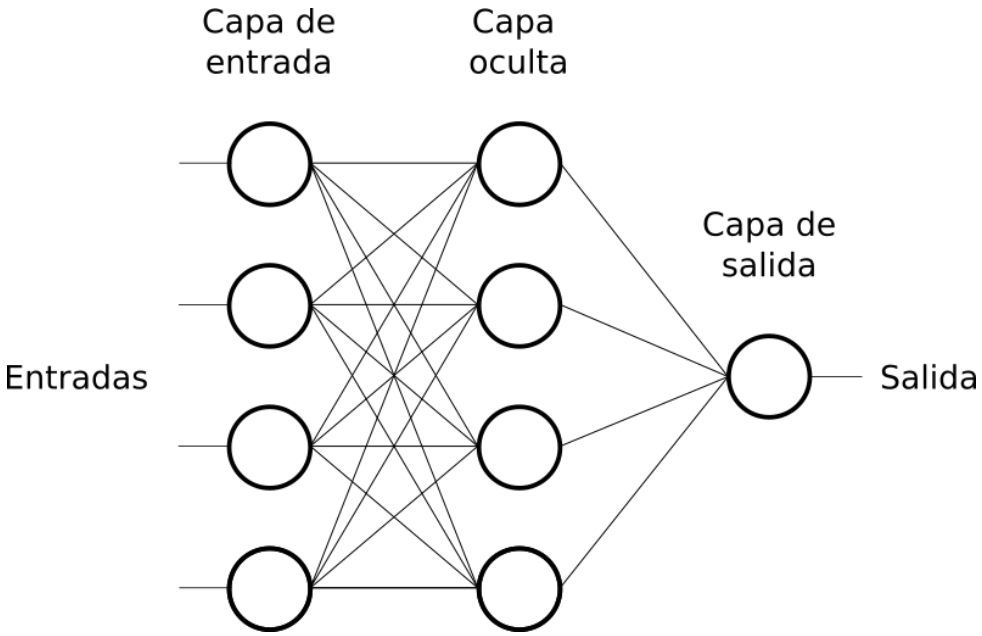

Figura 3.1: Organización de neuronas en forma de perceptrón multicapa

El perceptrón multicapa puede ser usado para clasificación si, en la última capa, se pone una neurona por clase y como función de coste se utiliza la entropía cruzada:

$$C = -\sum_{i=1}^{clases} y_i \log \hat{y}_i \tag{3.12}$$

donde y son los valores de salida de la red neuronal e $\hat{y}$ los valores de salida esperados para la entrada usada.

Capítulo 4

Experimentos

En este capítulo se describen los experimentes realizados y se estudian los resultados obtenidos. Todos los experimentos se han realizado usando herramientas estándar de aprendizaje automático basadas en Python (ver Apéndice C). Todo el código está disponible en un repositorio de acceso abierto[1].

4.1. Evaluando los métodos de reducción de dimensión

Se ha realizado una evaluación de seis métodos de reducción de dimensionalidad: Análisis de componentes principales, Isomap, *Locally linear embedding*, Laplacian eigenmaps, t-SNE y UMAP (Todos descritos en el Capítulo 2 de esta memoria) sobre el conjunto de datos MNIST. Para cada método se ha realizado una búsqueda en rejilla (ver la tabla 4.1) intentando maximizar el valor del índice expuesto en la Sección 2.9 para una dimensión concreta. Para evitar tiempos de ejecución prolongados se ha realizado la evaluación sobre una muestra aleatoria de 10.000 imágenes sobre el total de 60.000 que componen el conjunto de entrenamiento de MNIST. Sobre los valores de la rejilla:

- Se ha escogido reducir hasta 200 dimensiones puesto que al realizar una PCA se comprobó que con 200 componentes principales ya se recoge mucha más del 95 % de varianza. Por tanto conservar más dimensiones parece innecesario.
- Con el método tSNE se ha reducido solo hasta 9 dimensiones por ser un método especialmente lento.
- El número de vecinos usado para UMAP es muy diferente al usado en el resto de métodos. Se han escogido estos valores (ver la tabla 4.1) porque en UMAP el número de vecinos sirve para centrarse más en la estructura local(menos vecinos) o en la global(más vecinos). Nos interesa este efecto y para verlo se necesita modificar el número de vecinos sustancialmente.

[1] `https://github.com/andbrav/TFG-INF`

Método	Parámetros		
PCA	Dimensiones 1, 2, ... 10, 50, 100, 200		
ISOMAP	Dimensiones 1, 2, ... 10, 50, 100, 200	Vecinos 1, 2, 3, 4, 5, 6, 7, 8, 9	
LE	Dimensiones 1, 2, ... 10, 50, 100, 200	Vecinos 1, 2, 3, 4, 5, 6, 7, 8, 9	
LLE	Dimensiones 1, 2, ... 10, 50, 100, 200	Vecinos 1, 2, 3, 4, 5, 6, 7, 8, 9	
TSNE	Dimensiones 1, 2, ... 9	Perplejidad 30, 100, 250, 500	Exageración None, 2, 4
UMAP	Dimensiones 1, 2, ... 10, 50, 100, 200	Vecinos 5, 50, 500, 1000	Distancia mínima 0, 0.1, 0.25, 0.5, 0.75, 0.99

Cuadro 4.1: Valores de los parámetros que se han probado para la evaluación de los métodos de reducción de dimensión.

Es interesante observar como depende el índice del método usado y dentro de cada método de sus parámetros (el número de dimensiones al que se reduce se considera un parámetro). Para observar esto se ha realizado un estimador kernel de densidad del valor del índice de cada método (Figura 4.1). Como se puede pbservar, PCA es el método que arroja mejores resultados aunque la mejora del índice se debe únicamente al aumento de las dimensiones conservadas. Los kernel de densidad de ISOMAP y Laplacian eigenmaps muestran distribuciones sesgadas hacia la derecha y con un rango amplio, esto nos indica que son métodos que dan buenos resultados si se ajustan sus parámetros adecuadamente. Por último, LLE, t-SNE y UMAP son métodos cuyos kernels de densidad son apuntados, es decir leptocúrticos, y con un rango pequeño por lo que al aplicarlos se obtienen resultados parecidos independientemente de los parámetros usados.

También es interesante estudiar si la calidad de los métodos depende de las dimensiones conservadas. Para ello se ha realizado una proyección PCA a dos dimensiones de las 5 medidas que conforman el índice de los mejores modelos para una dimensión dada (se pueden ver todos los gráficos en el anexo B). Podemos comprobar que al reducir a 2 dimensiones (figura 4.2) los métodos que salen como buenos son tSNE, UMAP y un poco por detrás PCA. Este hecho concuerda con lo que obtuvieron Espadoto et al [23](en el que solo redujeron a dos dimensiones). Al elevar el número de dimensiones podemos ver (por ejemplo en 10 dimensiones en la figura 4.3) que el mejor método es PCA y el resto de métodos son bastante peores. Esto se cumple desde 4 dimensiones como se puede comprobar en el anexo B. La pérdida de calidad por tSNE y UMAP al elevar las dimensiones se debe a que estos métodos solo se centran en conservar las relaciones

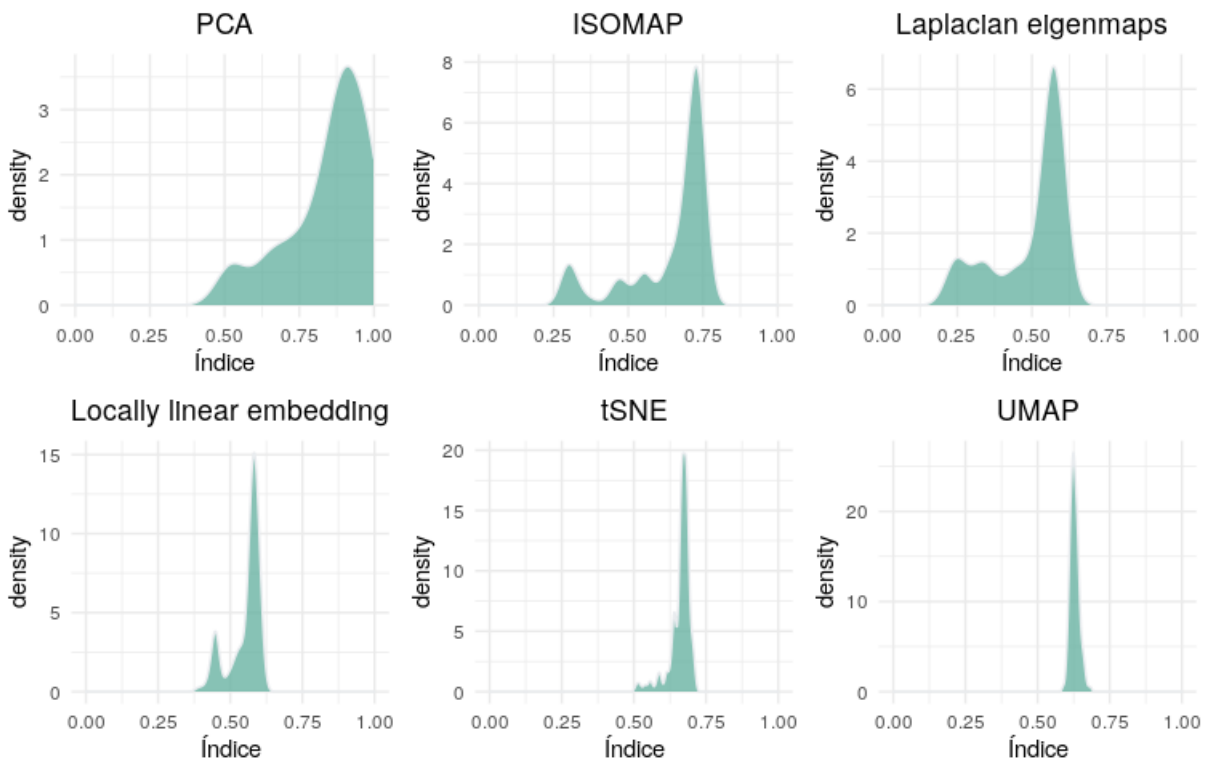

Figura 4.1: Estimadores kernel de densidad del índice para cada método de reducción de densidad.

locales de los datos y por tanto obtienen bajas puntuaciones en las medidas que puntúan la conservación de la estructura global (stress (cuanto más bajo mejor) y la correlación). En cambio PCA intenta conservar tanto aspectos locales como globales de los datos y lo hace mejor cuantas más dimensiones se conserven. Concluimos que si se quiere reducir la dimensión para visualización los mejores métodos son tSNE y UMAP. En cambio, si queremos reducir la dimensión como paso previo a otro proceso (posiblemente conservando más de dos dimensiones) y/o nos interesa la estructura global de los datos es difícil superar a PCA.

4.2. Clasificadores para MNIST

A partir de los resultados anteriores se ha evaluado el impacto de 5 métodos de reducción de dimensión PCA, ISOMAP, *Locally linear embedding*, tSNE y UMAP (no se ha utilizado laplacian eigenmaps debido a que no dispone de un método de mapeo de nuevos puntos) como paso previo al entrenamiento de tres clasificadores con estrategias diferentes: SVM, regresión logística y perceptrón multicapa. Para evitar tiempos excesivos, se ha escogido un subconjunto aleatorio de puntos (intentando mantener el equilibrio de las clases) sobre los que se ha calculado la reducción de dimensión y el resto de puntos han sido proyectados sobre esta reducción. Se ha usado una muestra de 30.000 puntos para

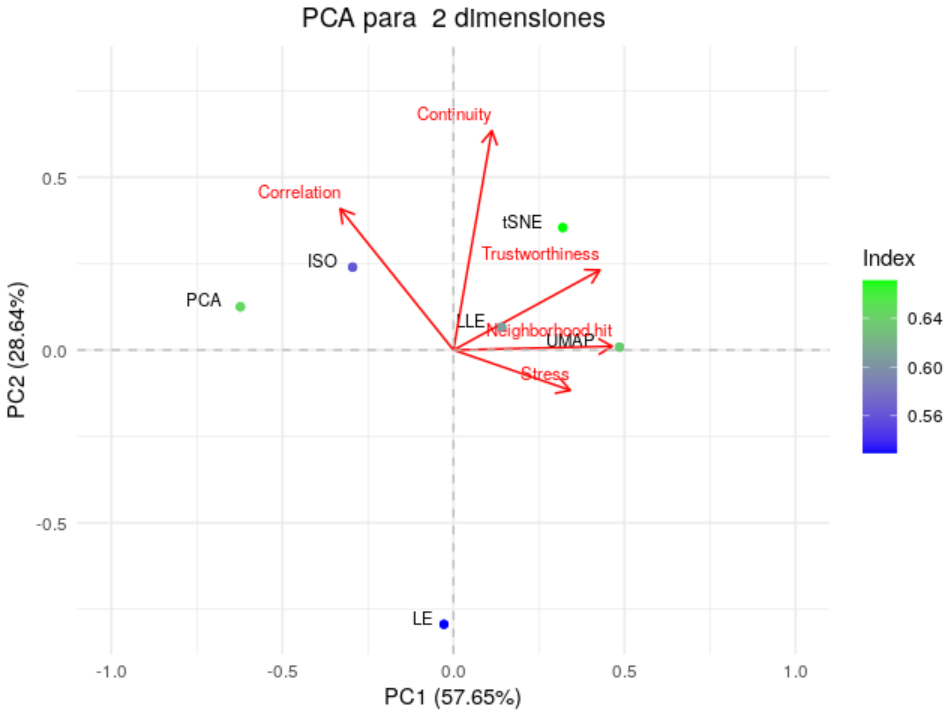

Figura 4.2: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 2 dimensiones.

los métodos más rápidos (PCA, ISOMAP y LLE) y 20.000 puntos para los más lentos (tSNE y UMAP). Esta diferencia de tamaños no supone problema alguno puesto se van a comparar los resultados de los clasificadores con los de uno de referencia y no entre ellos. Como parámetros para la reducción de dimensión se han utilizado los mejores según el experimento del apartado 4.1 (los que maximizan el índice) que se pueden ver en el anexo C.

4.2.1. SVM

En primer lugar, se ha ajustado un modelo SVM a los datos originales. Se ha utilizado una aproximación descrita por Scholkopf y Decoste [25]:

1. Se ha ajustado un modelo SVM a los datos y se han extraído los vectores soporte.
2. Se han generado muestras virtuales a partir de los vectores soporte. En nuestro caso, se han generado translaciones de los vectores soporte 1 pixel en las 8 direcciones posibles.
3. Se entrena un modelo SVM con los vectores soporte más las muestras virtuales.

Utilizando un kernel de base radial se ha conseguido una precisión del 98.96 %. Utilizando un kernel polinómico Scholkopf y Decoste consiguieron una precisión del 99.2 %. Con los

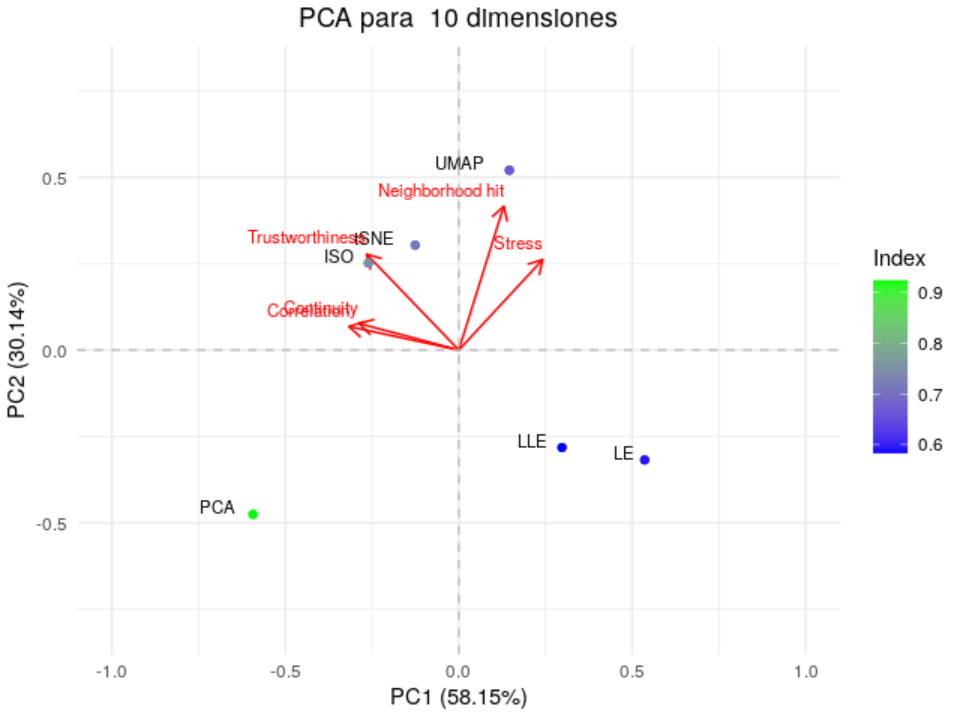

Figura 4.3: PCA de las 10 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 10 dimensiones.

conjuntos de datos reducidos se han entrenado clasificadores SVM con kernel de base radial y también se ha aplicado la estrategia de las muestras virtuales (generando las muestras con el dataset original y proyectando la nueva muestra a baja dimensión). Los resultados se muestran en la tabla 4.2.

Comprobamos que no hay ningún modelo que supere la precisión de nuestros modelos de referencia. Un aspecto a destacar es que la estrategia de generar muestras virtuales solo parece beneficiosa cuando se usa PCA para reducir a un número considerable de dimensiones. El mejor modelo conseguido llega a una precisión del 99.11 % usando PCA para reducir a 100 dimensiones y muestras virtuales. Utilizando pocas dimensiones es fácil conseguir modelos con entorno al 95 % de precisión. Un detalle interesante, es que usando los datos de 5 dimensiones generados con UMAP se obtiene un 85 % de precisión, lo que parece escaso observando como se comporta el método para el resto de dimensiones. Hipotetizamos que esto se debe a que el valor de los parámetros de UMAP para esta dimensión es bastante diferente al del resto (ver anexo C para ver los valores de los parámetros). Concluimos que SVM puede verse perjudicado por la reducción de dimensión aunque no de una manera exagerada. Parece posible encontrar un modelo igual de bueno que el original dedicando suficiente esfuerzo. Además parece que si se usa UMAP y SVM conjuntamente los parámetros de UMAP pueden tener mucha importancia como se ha visto anteriormente.

Dim	PCA		ISOMAP		LLE		tSNE		UMAP	
	Prec. SVM	Prec. VSVM	Prec. SVM	Prec. VSVM	Prec. SVM	Prec. VSVM	Prec. SVM	Prec. VSVM	Prec. SVM	Prec. VSVM
1	31.84	22.76	35.41	-*	55.64	60.58	83.19	78.78	63.69	60.88
2	47.87	39.36	51.62	-*	69.61	67.34	90.82	88.91	93.89	93.25
3	54.68	50.79	72.30	-*	87.93	87.20	93.87	94.06	94.17	93.51
4	66.02	61.07	78.89	-*	86.89	86.11	94.47	94.40	94.49	95.23
5	77.30	73.05	90.34	-*	91.77	89.70	94.48	94.48	84.88	85.77
6	84.93	81.92	94.18	94.28	95.15	95.33	97.70	94.67	94.88	94.79
7	88.87	86.86	94.73	95.08	95.17	95.42	94.76	94.76	95.16	95.11
8	91.21	90.11	95.08	95.08	95.50	95.84	94.84	94.86	95.31	95.31
9	92.34	91.88	95.32	95.69	95.95	95.86	94.71	94.81	95.19	95.16
10	93.65	93.83	95.96	95.69	95.84	96.26	-	-	95.28	95.19
50	98.32	99.04	96.82	97.24	96.51	96.22	-	-	94.72	93.89
100	98.39	99.11	96.94	97.36	96.74	96.87	-	-	95.03	95.04
200	98.33	99.11	97.01	97.55	96.84	96.83	-	-	95.19	95.27

Cuadro 4.2: Precisiones sobre el conjunto de test de MNIST conseguidas con SVM y SVM con muestras virtuales (VSVM) más reducción de dimensiones.* No se han podido evaluar estos modelos por problemas de memoria.

4.2.2. Regresión logística

Se han ajustado modelos de regresión logística al conjunto de datos original y a los conjuntos reducidos. Se han probado distintos valores para el parámetro de regularización C: {0.001, 0.01, 0.1, 1, 10, 100, 1000} donde valores más pequeños implican mayor fuerza regularizadora[2]. En el dataset original usando regresión logística se ha conseguido una precisión del 92.63 % (con C = 0.1). Los resultados conseguidos con los datos reducidos se presentan en la tabla 4.3.

Como se puede ver usando únicamente 6 dimensiones ya hay tres métodos (LLE, tSNE, UMAP) que permiten mejorar la clasificación frente a un modelo que usa los datos completos. Además el mejor modelo (LLE y C=1000)consigue una precisión del 96.73 % más de 4 puntos porcentuales por encima del modelo con los datos completos. Concluimos por tanto que la reducción de dimensiones es muy recomendable en el caso en el que se vaya a usar regresión logística para la clasificación.

[2]La regularización consiste en penalizar modelos complejos modificando la función de coste que se optimiza al ajustar el modelo. En el contexto de la regresión logística es una penalización a coeficientes más altos. Es una medida que ayuda a combatir el sobreajuste

Dim	PCA		ISOMAP		LLE		tSNE		UMAP	
	C	Precisión	C	Precisión	C	Precisión	C	Precisión	C	Precisión
1	10	31.34	10	33.82	1000	49.52	100	77.3	1000	60.83
2	10	44.88	1	47.79	1000	62.56	0.01	84.05	1000	89.7
3	10	48.55	100	65.01	1000	83.99	1	88.66	100	91.49
4	100	57.77	10	71.83	1000	85.55	0.1	93.24	100	91.06
5	10	68.85	10	84.94	1000	88.42	0.1	93.68	1	78.9
6	1	73.73	10	89.51	10	94.37	0.01	93.55	0.1	93.41
7	1	76.48	10	90.81	1000	94.37	0.01	94.26	100	94.76
8	1	79.12	10	91.85	1000	95.12	0.1	94.39	100	94.22
9	100	79.37	100	91.97	1000	95.11	0.1	94.07	100	94.61
10	1	81.02	100	92.98	1	95.10	-	-	100	94.81
50	1	91.26	1	95.96	1000	96.48	-	-	0.01	94.78
100	0.1	92.06	1	96.31	1000	96.73	-	-	0.1	95.09
200	1	92.55	0.1	96.42	1	96.72	-	-	0.1	95.37

Cuadro 4.3: Precisiones sobre el conjunto de test de MNIST conseguidas con regresión logística más reducción de dimensiones.

4.2.3. Perceptrón multicapa

En las tareas de clasificación de imágenes es habitual el uso de redes convolucionales. Las redes convolucionales tienen una precisión de entorno al 99 % para el conjunto MNIST [26]. Una vez se reduce la dimensión de las imágenes habitualmente[3] no es posible usar redes convolucionales. Por este motivo se ha optado por un perceptrón multicapa. La arquitectura escogida se conforma por una capa de entrada, 4 capas ocultas con 100 neuronas cada una y una de salida. Se ha escogido esta arquitectura realizando pruebas preliminares buscando resultados razonablemente buenos. Esta arquitectura sobre el conjunto de datos original consigue una precisión del 97 %. Se ha entrenado esta red neuronal con los diferentes conjuntos de datos reducidos y se presentan los resultados en la tabla 4.4.

Algo que destaca a primera vista, son los buenos resultados que se consiguen con tSNE y UMAP con muy pocas dimensiones. Por ejemplo, el perceptrón multicapa con los datos de una dimensión generados con tSNE consigue una sorprendente precisión del 92.35 %. Esto lleva a pensar que para las redes neuronales las propiedades locales de los datos son especialmente importantes. Por tanto, si se va a reducir la dimensión como paso previo a usar redes neuronales interesa usar un método de reducción de dimensión que conserve

[3]Algunos métodos como PCA sí permiten reconstruir el formato de imagen

Dim	PCA	ISOMAP	LLE	tSNE	UMAP
1	31.17	35.54	56.40	92.52	64.93
2	47.58	50.29	68.44	93.03	93.82
3	53.30	70.45	82.36	94.23	93.87
4	64.50	76.25	85.66	94.67	94.16
5	74.96	88.46	91.00	94.93	83.82
6	81.85	92.63	94.54	94.95	93.77
7	86.87	93.59	91.60	95.01	94.90
8	89.22	94.09	95.27	95.02	94.91
9	90.64	94.64	95.53	94.80	95.12
10	91.94	94.99	95.54	-	94.98
50	97.44	96.09	95.82	-	94.70
100	96.97	95.94	96.54	-	93.82
200	96.79	96.21	96.75	-	94.55

Cuadro 4.4: Precisiones sobre el conjunto de test de MNIST conseguidas con un perceptrón multicapa más reducción de dimensiones.

las propiedades locales de los datos. Por lo demás las redes neuronales entrenadas con los conjuntos de datos reducidos no se acercan al 99 % de precisión conseguida por las redes convolucionales con los datos originales. Aún así la reducción de dimensión puede ser interés desde el punto de vista de la conservación de recursos.

Capítulo 5

Conclusiones

En este TFG se ha planteado la revisión y evaluación de un conjunto de métodos de reducción de dimensión. Se han explicado diferentes estrategias desde métodos más clásicos como PCA a métodos muy novedosos como UMAP. Se ha realizado una evaluación de los mismos sobre el conjunto MNIST utilizando medidas cuantitativas. Los resultados indican que la elección del método de reducción depende del uso que se quiera hacer de los datos reducidos. Además se ha estudiado el impacto de la reducción de dimensión sobre distintas estrategias frente a un problema de clasificación. La principal conclusión obtenida es que la reducción de dimensión es una estrategia viable para mejorar un clasificador o para reducir el uso de recursos por el mismo. Otra conclusión importante, es que parece que cual es el mejor método de reducción de dimensión depende de a cuantas dimensiones queramos reducir. En este proyecto, quedan abiertas lineas de trabajo futuro, ya sea ampliando el trabajo realizado con más métodos o utilizando otros conjuntos de datos. Además, se espera poder utilizar lo aprendido en futuros trabajo puesto que la reducción de dimensión se usa en diferentes sectores como la bioinformática, la medicina y en general en cualquier disciplina que trabaje con gran cantidad de variables.

Parte I

Apéndices

Apéndice A

Software utilizado

En este apéndice listaremos que software se ha usado para cada método de reducción de la dimensionalidad. Para PCA, ISOMAP, LLE (*Locally linear embedding*) y Laplacian eigenmaps se ha utilizado la implementación de sklearn. El código se puede encontrar en:

- **PCA**: `https://github.com/scikit-learn/scikit-learn/tree/master/sklearn/decomposition`
- **ISOMAP**: `https://github.com/scikit-learn/scikit-learn/tree/master/sklearn/manifold`
- **LLE**: `https://github.com/scikit-learn/scikit-learn/tree/master/sklearn/manifold`
- **Laplacian Eigenmaps**: `https://github.com/scikit-learn/scikit-learn/tree/master/sklearn/manifold`

Para tSNE se ha utilizado la implementación OpenTSNE[27] que está escrita en Python y destaca por su facilidad de uso y su excelente documentación. Esta implementación permite añadir observaciones al mapping realizado por TSNE. Su código puede encontrarse en: `https://github.com/pavlin-policar/openTSNE`. Existe una implementación más eficiente escrita en c++[28] que se ha descartado por no implementar el añadido de nuevas observaciones al mapping.

Por último, para UMAP se ha utilizado la implementación de los autores del método. Está escrita en python, tiene una buena documentación y puede encontrarse en: `https://github.com/lmcinnes/umap`

Apéndice B

Figuras PCA

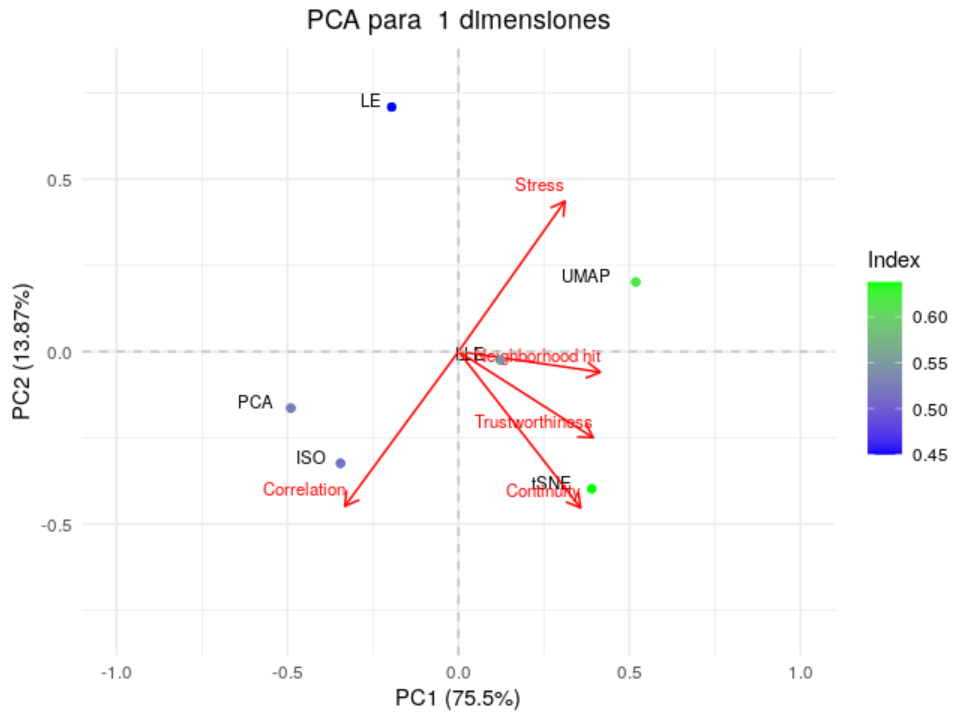

Figura B.1: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a una dimensión.

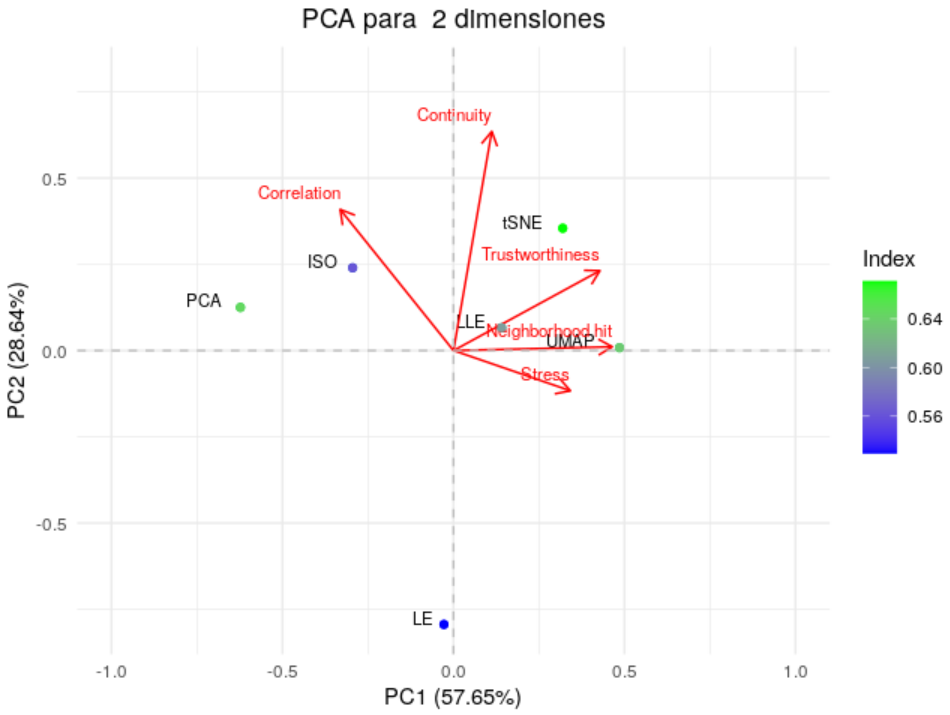

Figura B.2: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 2 dimensiones.

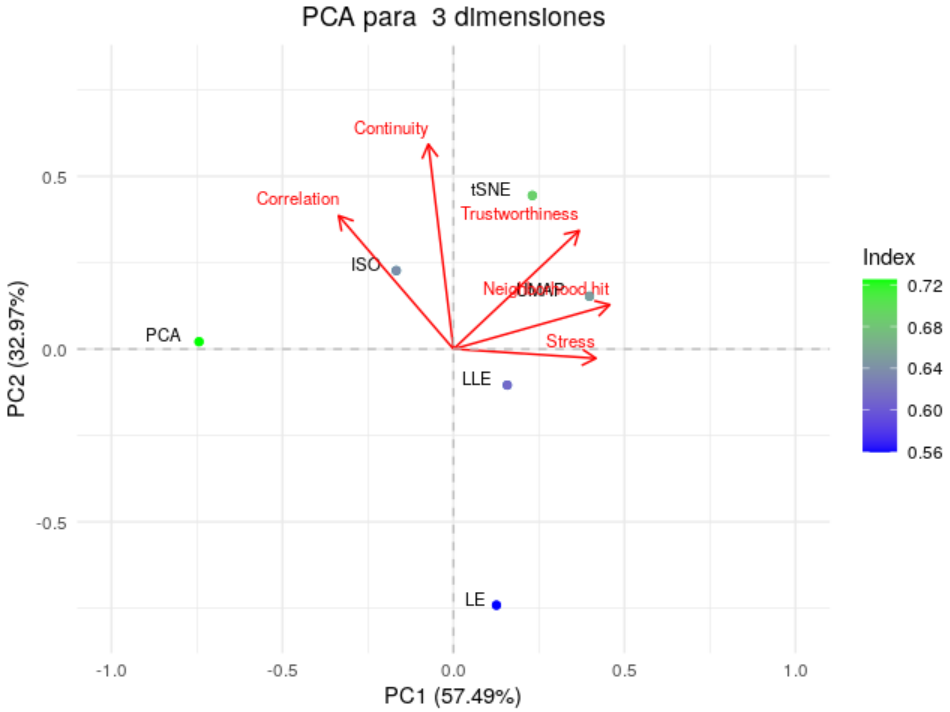

Figura B.3: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 3 dimensiones.

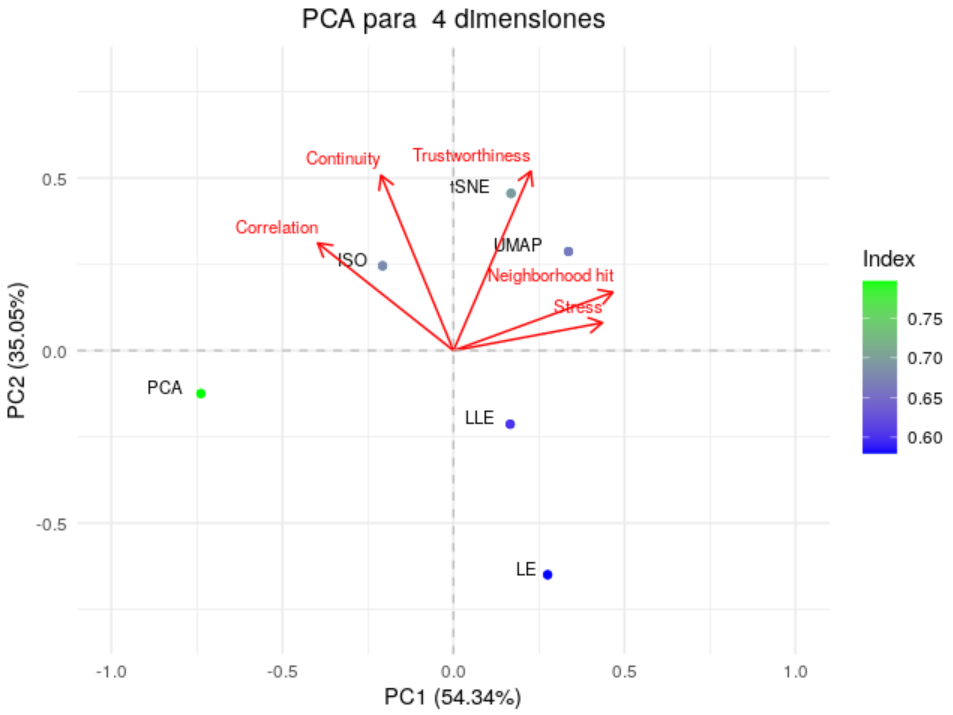

Figura B.4: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 4 dimensiones.

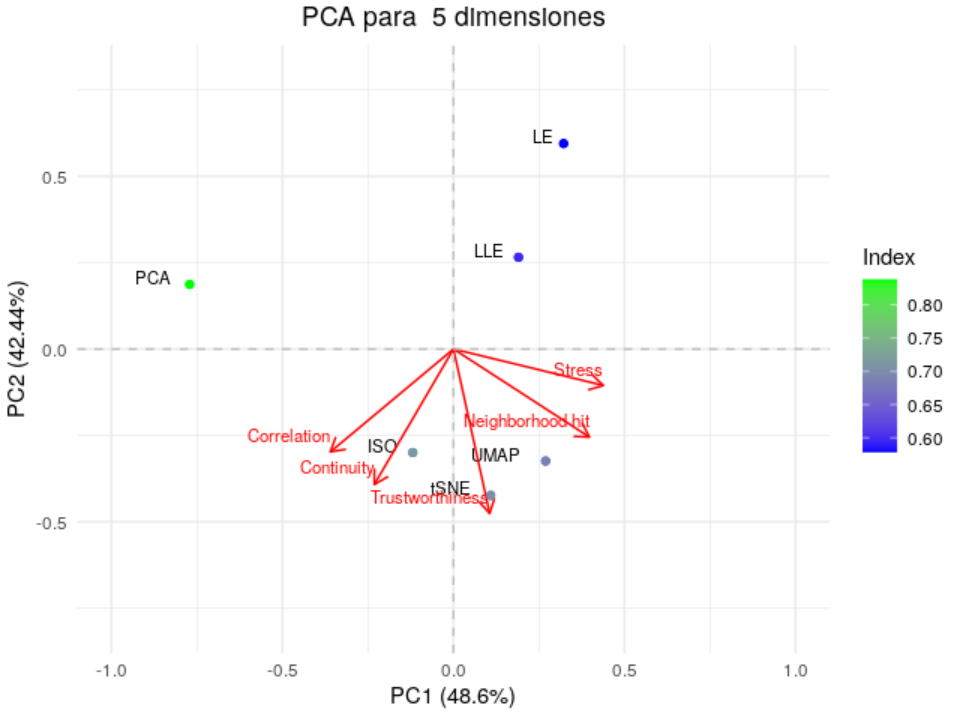

Figura B.5: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 5 dimensiones.

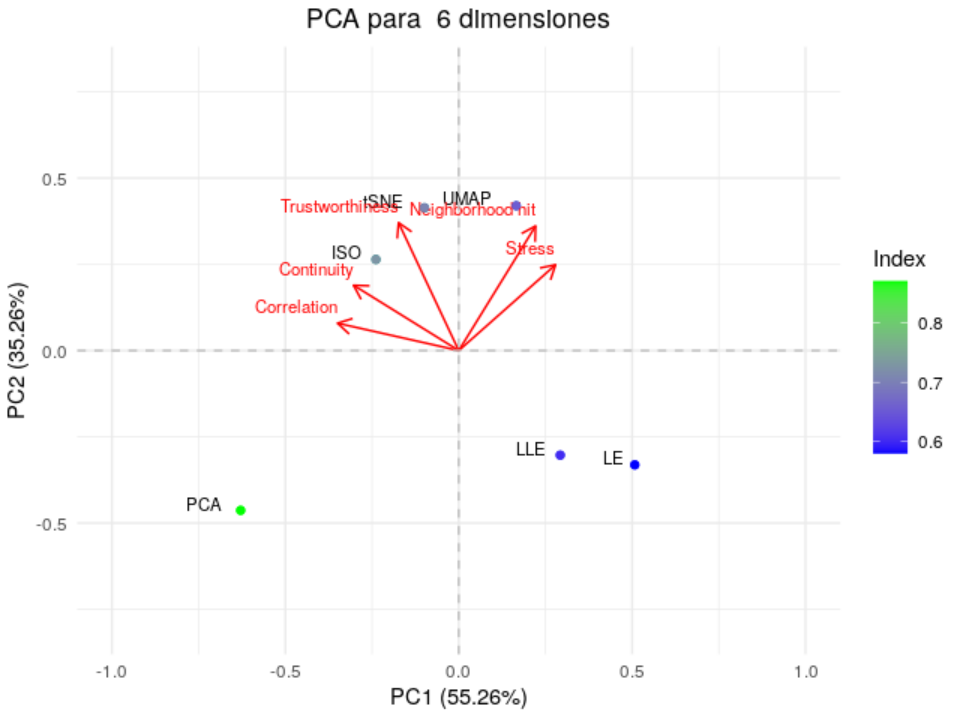

Figura B.6: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 6 dimensiones.

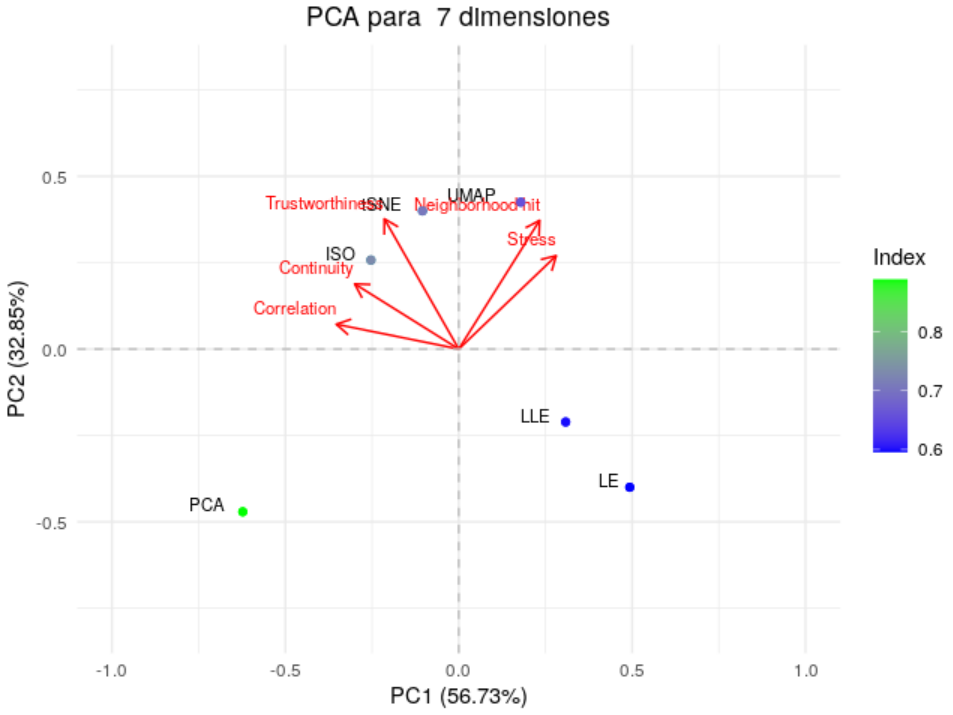

Figura B.7: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 7 dimensiones.

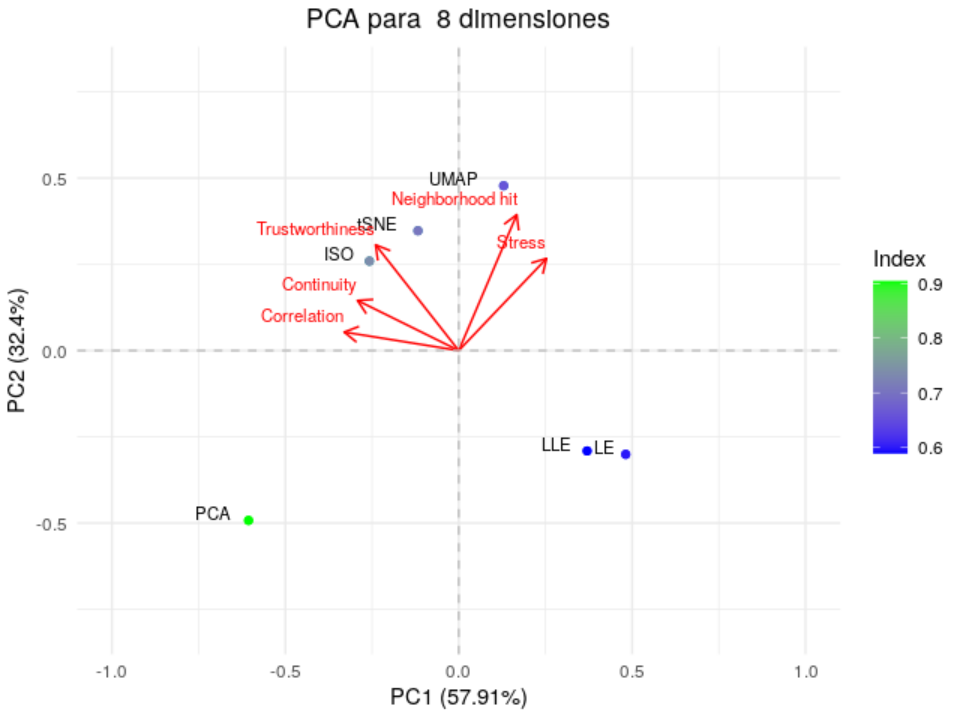

Figura B.8: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 8 dimensiones.

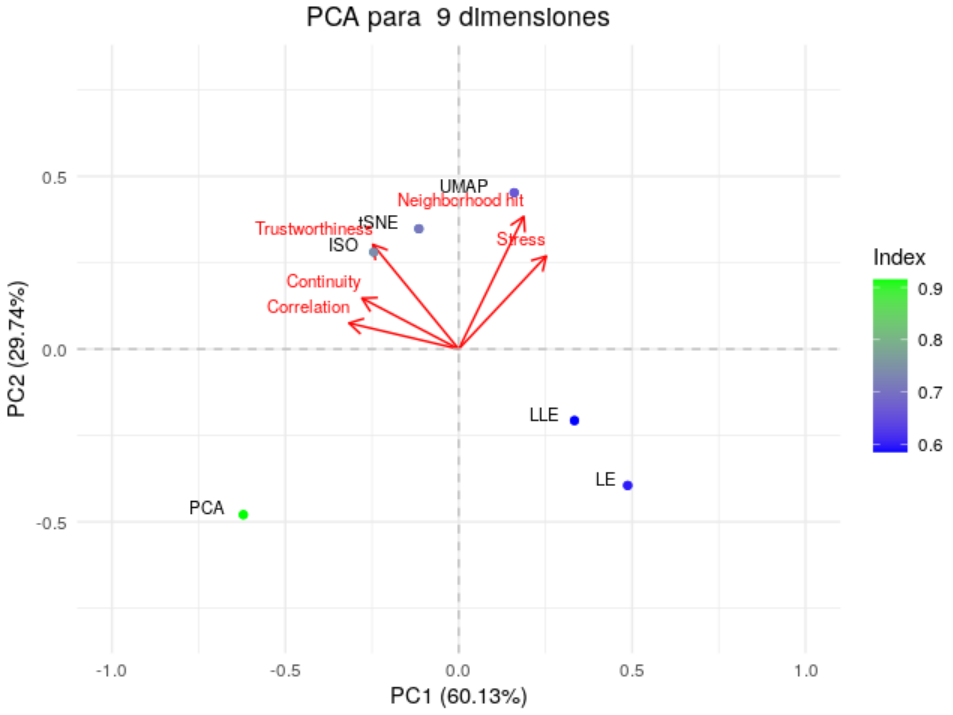

Figura B.9: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 9 dimensiones.

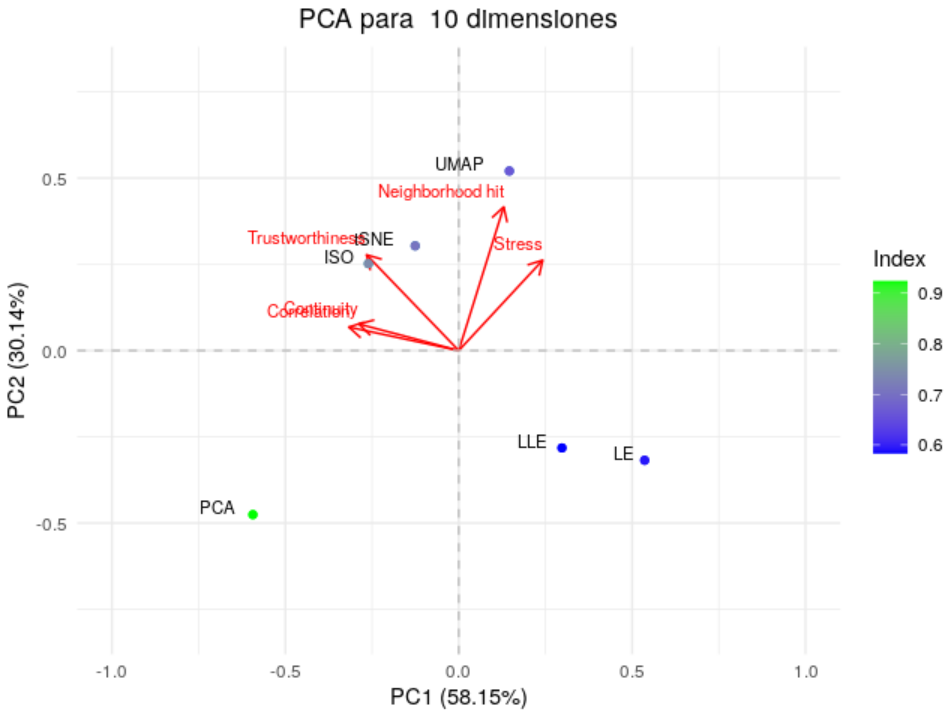

Figura B.10: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 10 dimensiones.

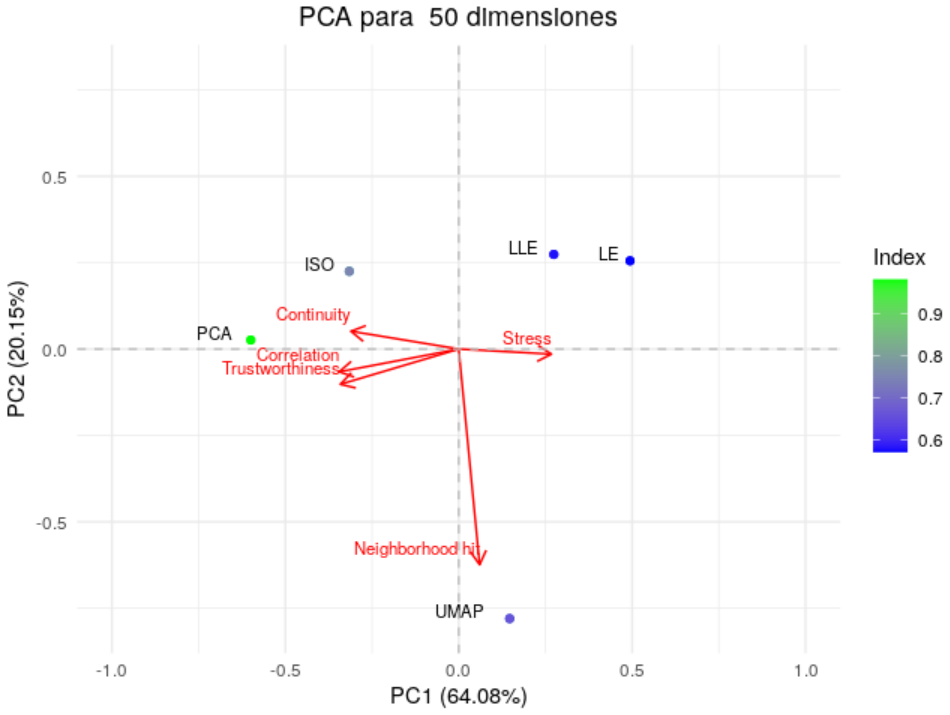

Figura B.11: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 50 dimensiones.

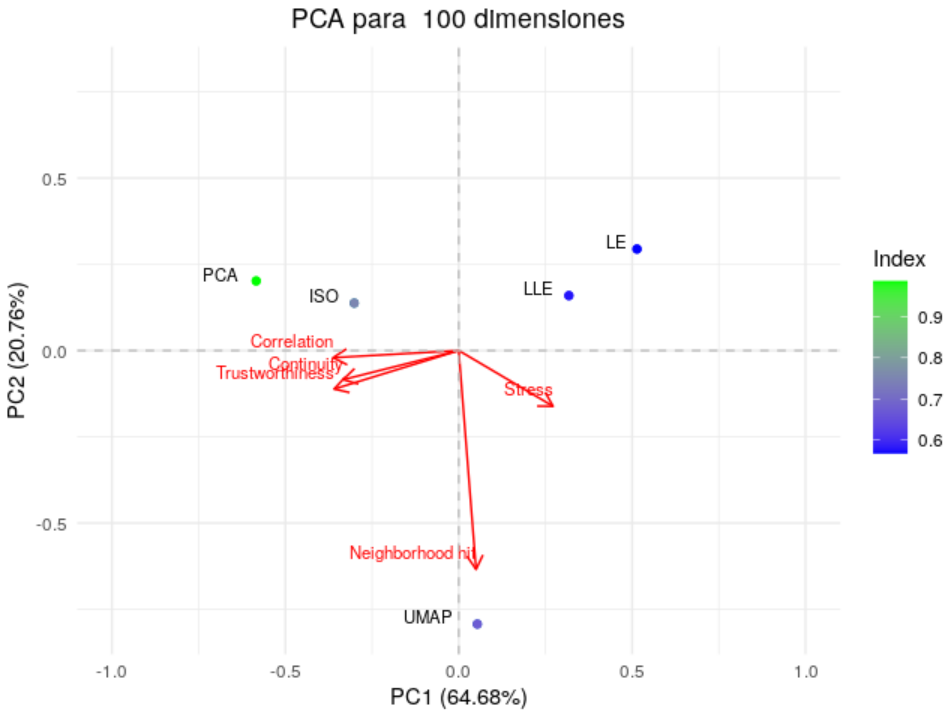

Figura B.12: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 100 dimensiones.

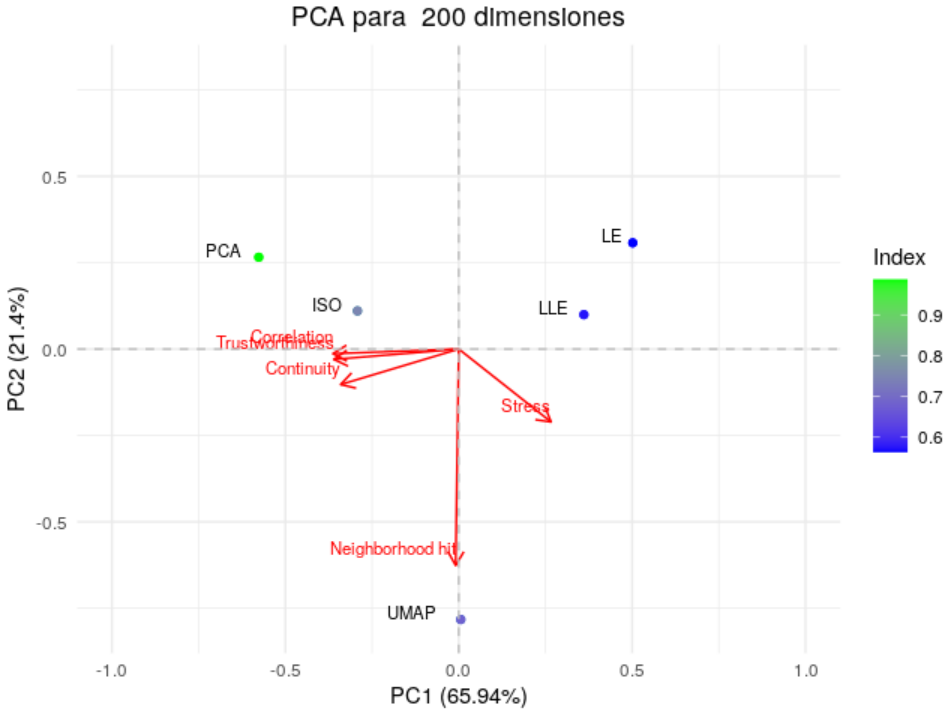

Figura B.13: PCA de las 5 medidas que conforman el índice para los métodos con los mejores parámetros reduciendo a 200 dimensiones.

Apéndice C

Mejores métodos de reducción de dimensión

Dimensión=1							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.62	0.84	0.59	0.35	0.39	0.52
ISOMAP	vecinos=9	0.64	0.90	0.76	0.37	0.42	0.51
LLE	vecinos=6	0.74	0.94	1	0.69	0.35	0.54
TSNE	perp=30* exag=0*	0.94	0.97	0.96	0.90	0.34	0.64
UMAP	vecinos=500 distm=0.5**	0.89	0.95	0.98	1	0.23	0.62
Dimensión=2							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.74	0.94	0.42	0.47	0.50	0.65
ISOMAP	vecinos=4	0.77	0.96	3.81	0.54	0.53	0.56
LLE	vecinos=5	0.87	0.96	1	0.81	0.40	0.61
TSNE	perp=100* exag=0*	0.98	0.98	0.97	0.93	0.44	0.67
UMAP	vecinos=5 distm=0.25**	0.96	0.97	0.99	0.97	0.28	0.64

Cuadro C.1: Mejores resultados conseguidos en el primer experimento para 1 y 2 dimensiones. * perp es perplejidad y exag es exageración. ** distm es distancia mínima.

Dimensión=3							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.84	0.97	0.32	0.51	0.72	0.72
ISOMAP	vecinos=3	0.89	0.98	8.71	0.71	0.61	0.64
LLE	vecinos=6	0.89	0.96	1	0.86	0.33	0.61
TSNE	perp=500* exag=0*	0.98	0.98	0.98	0.92	0.53	0.69
UMAP	vecinos=5 distm=0.75**	0.98	0.97	0.99	0.97	0.32	0.65
Dimensión=4							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.90	0.98	0.25	0.63	0.72	0.80
ISOMAP	vecinos=8	0.94	0.99	2.95	0.75	0.70	0.68
LLE	vecinos=5	0.91	0.97	1	0.87	0.25	0.60
TSNE	perp=500* exag=0*	0.99	0.98	0.98	0.93	0.55	0.70
UMAP	vecinos=5 distm=0.99**	0.99	0.97	0.99	0.97	0.37	0.66
Dimensión=5							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.94	0.99	0.21	0.71	0.76	0.84
ISOMAP	vecinos=8	0.97	0.99	3.56	0.86	0.73	0.71
LLE	vecinos=6	0.94	0.97	1	0.88	0.19	0.60
TSNE	perp=500* exag=0*	0.99	0.99	0.98	0.93	0.57	0.70
UMAP	vecinos=1000 distm=0.99**	0.98	0.98	0.99	1	0.45	0.69

Cuadro C.2: Mejores resultados conseguidos en el primer experimento para 3, 4 y 5 dimensiones. * perp es perplejidad y exag es exageración. ** distm es distancia mínima.

Dimensión=6							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.97	0.99	0.17	0.78	0.76	0.87
ISOMAP	vecinos=9	0.98	1	3.51	0.89	0.76	0.73
LLE	vecinos=4	0.92	0.97	1	0.89	0.24	0.60
TSNE	perp=500* exag=0*	1	0.99	0.98	0.93	0.59	0.71
UMAP	vecinos=5 distm=0.99**	0.99	0.97	0.99	0.97	0.33	0.65
Dimensión=7							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.98	0.99	0.15	0.82	0.80	0.89
ISOMAP	vecinos=9	0.99	1	3.91	0.90	0.77	0.73
LLE	vecinos=6	0.94	0.98	1	0.91	0.17	0.60
TSNE	perp=500* exag=0*	1	0.99	0.98	0.93	0.59	0.71
UMAP	vecinos=5 distm=0.5**	0.99	0.97	0.99	0.97	0.34	0.66
Dimensión=8							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.98	1	0.13	0.85	0.82	0.90
ISOMAP	vecinos=9	0.99	1	4.28	0.92	0.78	0.74
LLE	vecinos=4	0.93	0.97	1	0.91	0.13	0.59
TSNE	perp=500* exag=0*	1	0.99	0.98	0.94	0.59	0.71
UMAP	vecinos=5 distm=0.99**	0.99	0.98	0.99	0.97	0.35	0.66

Cuadro C.3: Mejores resultados conseguidos en el primer experimento para 6, 7 y 8 dimensiones. * perp es perplejidad y exag es exageración. ** distm es distancia mínima.

Dimensión=9							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.99	1	0.12	0.86	0.84	0.92
ISOMAP	vecinos=9	0.99	1	4.59	0.92	0.79	0.74
LLE	vecinos=6	0.95	0.98	1	0.92	0.08	0.58
TSNE	perp=500* exag=0*	1	0.99	0.98	0.94	0.60	0.71
UMAP	vecinos=5 distm=0.75**	0.99	0.97	0.99	0.97	0.37	0.66
Dimensión=10							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	0.99	1	0.10	0.88	0.85	0.92
ISOMAP	vecinos=9	0.99	1	4.83	0.92	0.80	0.74
LLE	vecinos=9	0.96	0.98	1	0.91	0.07	0.58
TSNE	-	-	-	-	-	-	-
UMAP	vecinos=5 distm=0.75**	0.99	0.97	0.99	0.97	0.38	0.66
Dimensión=50							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	1	1	0.01	0.94	0.98	0.98
ISOMAP	vecinos=9	1	1	8.84	0.93	0.83	0.75
LLE	vecinos=9	0.98	0.98	1	0.94	-0.02	0.58
TSNE	-	-	-	-	-	-	-
UMAP	vecinos=5 distm=0.1**	0.99	0.97	0.99	0.97	0.38	0.66

Cuadro C.4: Mejores resultados conseguidos en el primer experimento para 9, 10 y 50 dimensiones. * perp es perplejidad y exag es exageración. ** distm es distancia mínima.

Dimensión=100							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	1	1	0	0.93	1	0.99
ISOMAP	vecinos=9	1	1	11.01	0.93	0.83	0.75
LLE	vecinos=9	0.98	0.97	1	0.94	-0.05	0.58
TSNE	-	-	-	-	-	-	-
UMAP	vecinos=5 distm=0.75**	0.99	0.97	0.99	0.97	0.42	0.67
Dimensión=200							
Método	Parámetros	Fiabilidad	Continuidad	Stress	Neigb	Corr	Índice
PCA	-	1	1	0	0.93	1	0.99
ISOMAP	vecinos=9	1	1	13.67	0.93	0.83	0.75
LLE	vecinos=7	0.99	0.95	1	0.93	-0.07	0.57
TSNE	-	-	-	-	-	-	-
UMAP	vecinos=5 distm=0.75**	0.99	0.97	0.99	0.97	0.46	0.68

Cuadro C.5: Mejores resultados conseguidos en el primer experimento para 100 y 200 dimensiones. * perp es perplejidad y exag es exageración. ** distm es distancia mínima.

Bibliografía

[1] Itamar Kanter y Tomer Kalisky. «Single cell transcriptomics: methods and applications». En: *Frontiers in oncology* (mar. de 2015). DOI: 10.3389/fonc.2015.00053.

[2] Christopher Burges. «Geometric Methods for Feature Extraction and Dimensional Reduction - A Guided Tour». En: jul. de 2010, págs. 53-82. DOI: 10.1007/978-0-387-09823-4_4.

[3] Jorge Hurtado. «Dimension Reduction and Data Compression». En: ene. de 2004, págs. 81-105. ISBN: 978-3-642-53576-5. DOI: 10.1007/978-3-540-40987-8_3.

[4] Sunil Jha y RDS Yadava. «Denoising by Singular Value Decomposition and Its Application to Electronic Nose Data Processing». En: *Sensors Journal, IEEE* 11 (feb. de 2011), págs. 35-44. DOI: 10.1109/JSEN.2010.2049351.

[5] Luis Jimenez y David A. Langrebe. «Supervised Classification in High Dimensional Space: Geometrical, Statistical, and Asymptotical Properties of Multivariate Data». En: *IEEE TRANSACTIONS ON SYSTEM, MAN, AND CYBERNETICS, VOLUME 28, PART C* 28 (1998), págs. 39-54.

[6] Mario Koppen. «The Curse of Dimensionality». En: *5th Online World Conference on Soft Computing in Industrial Applications* (2000).

[7] Keinosuke Fukunaga y Raymond Hayes. «Effects of Sample Size in Classifier Design». En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 11 (sep. de 1989), págs. 873-885. DOI: 10.1109/34.31448.

[8] Anil Jain, Robert Duin y Jianchang Mao. «Statistical Pattern Recognition: A Review». En: *IEEE Trans. Pattern Anal. Mach. Intell.* 22 (ene. de 2000), págs. 4-37. DOI: 10.1109/34.824819.

[9] Yann Lecun y col. «Gradient-Based Learning Applied to Document Recognition». En: *Proceedings of the IEEE* 86 (dic. de 1998), págs. 2278-2324. DOI: 10.1109/5.726791.

[10] Jason Weston y col. «Feature selection for SVMs». En: vol. 13. Ene. de 2000, págs. 668-674.

[11] Frank Plastria, Steven De Bruyne y Emilio Carrizosa. «Dimensionality Reduction for Classification». En: oct. de 2008, págs. 411-418. DOI: 10.1007/978-3-540-88192-6_38.

[12] H Hotelling. «Analysis of a complex of statistical variables into principal components.» En: *Journal of Educational Psychology* (1933). DOI: 10.1037/h0071325.

[13] Warren S Torgerson. «Multidimensional scaling: I. Theory and method». En: *Psychometrika* (1952). DOI: 10.1007/BF02288916.

[14] Joshua Tenenbaum, Vin Silva y John Langford. «A Global Geometric Framework for Nonlinear Dimensionality Reduction». En: *Science* 290 (ene. de 2000), págs. 2319-2323. DOI: 10.1126/science.290.5500.2319.

[15] Sam Roweis y Lawrence Saul. «Nonlinear Dimensionality Reduction by Locally Linear Embedding». En: *Science (New York, N.Y.)* 290 (ene. de 2001), págs. 2323-6. DOI: 10.1126/science.290.5500.2323.

[16] Mikhail Belkin y Partha Niyogi. «Laplacian Eigenmaps for Dimensionality Reduction and Data Representation». En: *Neural Computation* 15 (jun. de 2003), págs. 1373-. DOI: 10.1162/089976603321780317.

[17] Laurens van der Maaten y Geoffrey Hinton. «Viualizing data using t-SNE». En: *Journal of Machine Learning Research* 9 (nov. de 2008), págs. 2579-2605.

[18] Geoffrey Hinton y Sam Roweis. «Stochastic Neighbor Embedding». En: 15 (jun. de 2003).

[19] James Cook y col. «Visualizing Similarity Data with a Mixture of Maps.» En: *Journal of Machine Learning Research - Proceedings Track* 2 (ene. de 2007), págs. 67-74.

[20] Dmitry Kobak y Philipp Berens. «The art of using t-SNE for single-cell transcriptomics». En: *Nature Communications* 10 (dic. de 2019). DOI: 10.1038/s41467-019-13056-x.

[21] Leland McInnes y John Healy. «UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction». En: (feb. de 2018).

[22] Jackson Adelse. *The mathematics of UMAP*. 2019. URL: https://www.biorxiv.org/content/early/2019/08/13/731877.

[23] Mateus Espadoto y col. «Towards a Quantitative Survey of Dimension Reduction Techniques». En: *IEEE Transactions on Visualization and Computer Graphics* PP (sep. de 2019), págs. 1-1. DOI: 10.1109/TVCG.2019.2944182.

[24] Frederic B. Fitch. «Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133.» En: *Journal of Symbolic Logic* 9.2 (1944), págs. 49-50. DOI: 10.2307/2268029.

[25] Dennis Decoste y Nello Cristianini. «Training Invariant Support Vector Machines». En: *Machine Learning* 46 (ago. de 2003). DOI: 10.1023/A:1012454411458.

[26] Yann Lecun y col. «Gradient-Based Learning Applied to Document Recognition». En: *Proceedings of the IEEE* 86 (dic. de 1998), págs. 2278-2324. DOI: 10.1109/5.726791.

[27] Pavlin G. Poličar, Martin Stražar y Blaž Zupan. «openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding». En: *bioRxiv* (2019). DOI: 10.1101/731877. eprint: https://www.biorxiv.org/content/early/2019/08/13/731877.full.pdf. URL: https://www.biorxiv.org/content/early/2019/08/13/731877.

[28] George Linderman y col. «Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data». En: *Nature Methods* 16 (mar. de 2019), pág. 1. DOI: 10.1038/s41592-018-0308-4.