DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Individual

Transformación Lineal y No Lineal para la Reducción de Dimensiones

4 de Julio — Métodos Numéricos

Integrante	LU	Correo electrónico
Mallol, Martín Federico Alejandro	208/20	martinmallolcc@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Resumen

Para este trabajo el enfoque fue puesto en dirimir las diferencias y similitudes entre varios metodos que transforman la dimensión de nuestros datos. Como así el efecto que producen cuando se las emplea a la par de un clasificador. En este caso, el famoso *k-Nearest Neighbors*. Para realizar tal estudio, se dispone del dataset *"Fashion MNIST"* de la base de datos **"MNIST"**. La estructura del informe es la siguiente:

- **Introducción**: Se estudia el comportamiento de **kNN** por sí solo como con los transformadores **PCA**, **t-SNE** (*T-distributed Stochastic Neighbor Embedding*) y **UMAP** (*Uniform Manifold Approximation and Projection*). Se observa el comportamiento de los nuevos métodos presentados y sus tiempos de ejecución para así compararlos con los métodos ya vistos el año pasado.
- **Desarrollo**: Se explica con qué herramientas fue realizado el Trabajo Práctico y de qué trata cada Notebook donde fue realizada la experimentación.
- **Experimentación, resultados y discusión**: Se revisita el modelo visto el año pasado en el cual se combinan **kNN** con **PCA**, realizando *Cross-validation* (con ayuda del algoritmo **K-Fold**). Se da un pantallazo inicial sobre como es el rendimiento de este clasificador cuando se lo utiliza por sí solo, y cuando es usado con una reducción de dimensión previa, mediante la ayuda del transformador **PCA** (*Principal Component Analysis*).

  Luego, se empieza a conocer un poco cómo se comportan los metodos **t-SNE** y **UMAP** y qué tan similares pueden llegar a ser con el ya familiar método **PCA**. A partir de esto, se comienzan a utilizar estos transformadores *no lineales* con kNN en pos de encontrar la combinación de parámetros ideal para el rendimiento del clasificador. Además, se pone el foco en los tiempos de computos que dichas transformaciones arrojan, los cuales habrá de todo tipo: acotados, estándar y muy grandes.
- **Conclusiones**: Se concluye que **t-SNE** y **UMAP** proveen resultados correctos, no así excepcionales. Se esperó un rendimiento mejor para la predicción, aunque no es decepcionante. La dimensión se reduce drásticamente obteniendo aceptables resultados, mas los tiempos de ejecución en algunos casos llegaron a ser desalentadores, en otros bastante bien estuvieron. Más todavia cuando se utiliza toda o casi toda la totalidad de elementos provistos por la muestra. Eso sí, el tiempo de ejecución, una vez hecha la reducción, sí mejora. La metrica accuracy ayudó a mostrar que los valores ideales para los métodos fueron los siguientes:
  kNN: **k = 7.**
  PCA: **Número de componentes = 114.**
  tSNE: **Número de componentes = 3 y perplejidad: 18.**
  UMAP: **Número de componentes = 15 y Distancia mínima = 0.99**

Palabras clave: *k-Nearest Neighbors*, *Principal Component Analysis*, *Cross-validation*, *T-distributed Stochastic Neighbor Embedding*, *Uniform Manifold Approximation and Projection.*

Índice

1. Introducción

Se buscará relacionar a varias transformaciones, tanto lineales (PCA) como no lineales (t-SNE y UMAP) en busca de ver sus similitudes y sus diferencias. La experimentación estará centrada en ver qué tan efectivos son estos métodos para la predicción de patrones en los dataset, con ayuda del algoritmo kNN. Por último se compararán los tiempos de cómputo de cada uno, y se intentarán encontrar los puntos fuertes y débiles de cada algoritmo, para poder saber en qué contexto utilizarlos.

Se presentará brevemente cada método, ya que lo importante y relevante en este trabajo es la experimentación con los mismos.

1.1. Presentación de los métodos

1.1.1. k-Nearest Neighbors (kNN)

La idea del método es clasificar a un vector (imagen) $y \in \mathbb{R}^{784}$ con la moda (la más repetida) de las clases de los k vectores $x_{i_1}, ..., x_{i_k} \in \mathcal{D}$ que minimizan la distancia euclídea respecto a dicho y. La dimensión del espacio es alta, por lo tanto el método, sin ninguna transformación previa, puede ser costoso.

1.1.2. Principal Components Analysis (PCA)

PCA permite "resumir" en unos pocos componentes la información almacenada en todos los valores de los $x_i \in \mathcal{D}$. Con el objetivo de mejorar no sólo el tiempo de cómputo del clasificador, si no también de acrecentar la efectividad del mismo. La transformación que realiza es lineal, en la cual en su algoritmo se da el uso de una matriz de covarianza y la busqueda de alfa autovectores, para armar una matriz de cambio de base $W \in \mathbb{R}^{\alpha \times m}$, con el fin de amplificar la varianza de los datos y así eliminar información "redundante" (existe pérdida de exactitud de la data).

1.1.3. T-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE busca preservar la similitud de vecinos de los datos originales cuando se reduce el espacio. La reducción que realiza es drástica, ya que se hace a una dimensió interpretable, que se puede graficar. O sea, a dimensiones muy bajas, se habla de 1,2 o 3 dimensiones. Su algoritmo calcula las similitudes de los puntos en la dimension original. Para hacer esto, se centra en cada

punto la distribucion gaussiana alrededor del mismo. Después se calcula la densisdad para cada punto dentro de dicha distribución.

Se crea un espacio más pequeño en dimensión distibuyendo los puntos de forma aleatoria y luego calculando sus similitudes en el mismo, pero esta vez usando una distribución t-Student.

Por último, se intenta preservar la estructura local de los datos al minimizar la divergencia de Kullback–Leibler entre las distribuciones Gaussianas y t-Student calculadas anteriormente con respecto a las ubicaciones de los puntos en el espacio original y el de menor dimensión.

1.1.4. Uniform Manifold Approximation and Projection (UMAP)

UMAP A diferencia de t-SNE que busca mantener la estructura local en los datos, intenta preservar tanto la estructura local como la mayor porcion global de los datos. Se agrupan los puntos en distintos grupos, pero sin perder su enfoque global con los demás.

El método es modelado mediante una estructura topológica difusa. La transformación se realiza en pos de optimizar la representacion de la muestra de baja dimensión. Eso se realiza buscando una proyección de menor dimensión de la data que tenga la representación topológica difusa más similar posible.

2. Desarrollo

Todos los experimentos fueron realizados con Jupyter Notebook en archivos **.ipynb**. Allí se utilizaron diferentes librerías para los distintos métodos que eran necesarios para poder efectuar los tests. Las implementaciones de kNN, PCA, tSNE y KFold se importaron de la librería **"Scikit Learn"**. La metrica Accuracy también fue extraida de esta. Los estudios de rasgos generales de cada método fueron hechos en el notebook **2D.ipynb**. Mientras que el estudio de kNN con las distintas transformaciones fue realizado en **EXPERIMENTACION.ipynb**. Mientras que el código de UMAP fue instalado con python y solo hizo falta importarlo en el Notebook para que funcione.

Para calcular los tiempos de ejecución, se instaló vía python el codigo "timeƨ luego se lo importó al notebook. Mientras que, para graficar los resultados de los experimentos, se dispuso de las librerías matplotlib.pyplot (plots y scatterplots) y flameplot (heatmaps).

Por último, las muestras usadas a lo largo de la experimentación fueron tomadas de la base de datos **MNIST** sobre el dataset **Fashion MNIST** que se encuentra en *kaggle*[1].

3. Experimentación, resultados y discusión

3.1. Métricas utilizadas

Para la experimentación de este trabajo, se utilizó la métrica Accuracy. Esta es la tasa de efectividad lograda. En todos los experimentos que se hicieron con el fin de averiguar el rendimiento del clasificador kNN con cada transformador. Ninguna otra métrica formó parte de la experimentación. Accuracy provee la cantidad de aciertos totales por sobre la cantidad total de resultados obtenidos. O sea que calcula la media de todos los resultados obtenidos. Lo justo y necesario para este trabajo.

[1] `www.kaggle.com/zalando-research/fashionmnist`

3.2. Evaluación de los métodos a utilizar

3.2.1. K-fold cross validation

Para evaluar los distintos algoritmos de cada uno de los métodos (tanto viejos como nuevos) presentados, se dispuso del ya conocido "K-fold cross validation". Si no se lo utilizase sobre la muestra de prueba, es muy posible que los resultados arrojados por los métodos no sean representativos. Con el Cross-Validation se busca particionar los datos que en K grupos y se entrenan mientras que un integrante de dicho grupo se utiliza como validación. Se elijió el valor de K = 10. Que fue un número que en el Trabajo Práctico del año pasado mostró ser más que aceptable.

3.3. Estudio de rasgos generales de PCA, t-SNE y UMAP

3.3.1. Reducción a 2D

Primero se quiso ver como se comporta la reducción de la dimensión a 2D, a algo tangible, algo que el ojo humano pueda interpretar. Ya que si se piensa en dimensiones de 4 en adelante, es dificil imaginarse como podrían verse. Por lo tanto, tomando los datos óptimos de Perplexity en el caso de t-SNE, y de Minimum Distance en UMAP, se graficaron para los tres transformadores su reduccion a un espacio de dos dimensiones. El objetivo era ver cómo era dicha transformación y cómo, dependiendo el algoritmo (método) cambiaría la agrupación de elementos. Al ser t-SNE y UMAP métodos no lineales, se formó la hipótesis de que seguramente, sus figuras dibujadas en el plano de dos dimensiones serían más abstractas o, más bien, menos representables por una línea de regresión en contra parte con la figura de PCA. Como esta última transformación es una transformación lineal, se podría considerar que es más representable en el plano 2D.

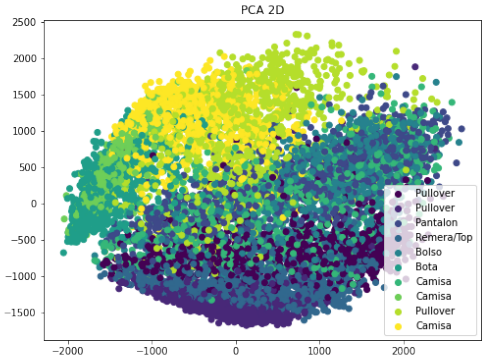

Figura 1: Gráfico de la transformación a 2d de $'PCA'$ con $'k = 7'$

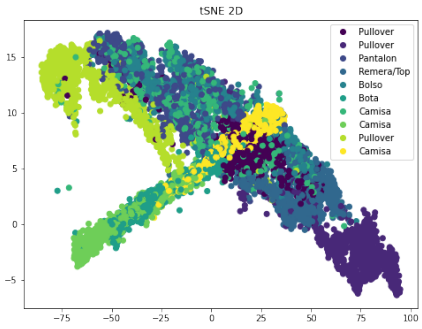

Figura 2: Gráfico de la transformación a 2d de $'t-SNE'$ con $'k = 7'$. Es muy curiosa la inesperada semejanza que este gráfico tiene con la imágen de un pájaro posando en una rama.

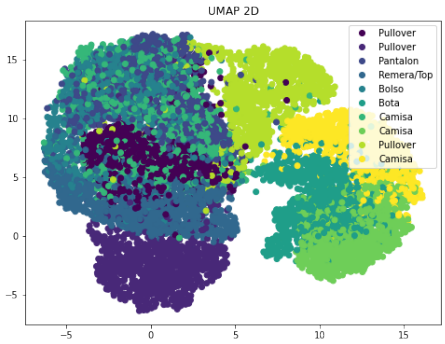

Figura 3: Gráfico de la transformación a 2d de $'UMAP'$ con $'k = 7'$

Se puede notar cómo se podría dibujar una línea recta en el gráfico de PCA con pendiente positiva. Mientras que en los otros gráficos eso no sería posible. No existe tal línea que pueda asemejar el comportamiento y patrón de los puntos ubicados en tales coordenadas. Tal vez en el de t-SNE, la línea de regresión puede mostrar muy bien el agrupamiento de los elementos de color oscuro, pero no el de los tonos verdes claro (camisas, pullover, botas), que tienen un patrón radicalmente distinto.

A priori ya estos graficos nos muestran las nuevas virtudes de t-SNE y UMAP en cuanto a reagrupamiento de distintas clases. Se ve como en estos dos métodos se logra realizar una separación mucho mas marcada de los diferentes tags del dataset, mientras que en PCA, a pesar de que existe una clara discriminación, no es tan tajante como en los otros dos metodos. Estas nuevas transformaciones son mas literales en la discriminación por tags que PCA.

3.3.2. Distribución de grupos en las tres transformaciones

Ya hecha la transformación en cada caso, se tendrían los datos listos para efectuar con el clasificador kNN el algoritmo de predicción. Por lo tanto, veamos que tanta coincidencia de vecinos hay entre cada método. Queremos ver qué tan similar es la distribución de elementos entre los tres métodos. Si los vecinos cercanos de cada uno coincide en un buen porcentaje, se prevee que el rendimiento de kNN con estos metodos sea parecido, si no, dependiendo el k que tome para kNN, el resultado del rendimiento podría variar considerablemente dependiendo qué transformación se escoja. Se implementó por lo tanto, el uso de heatmaps para poder cuantificar qué tan similares son cada reduccion a 2D entre PCA, t-SNE y UMAP.

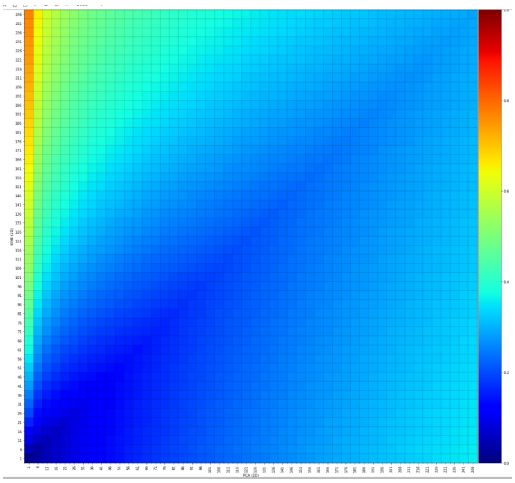

Figura 4: Heatmap de dimensiones 2D para $'PCA'$ y $'t-SNE'$. La X (PCA) e Y (tSNE) de este gráfico son los vecinos más cercanos.

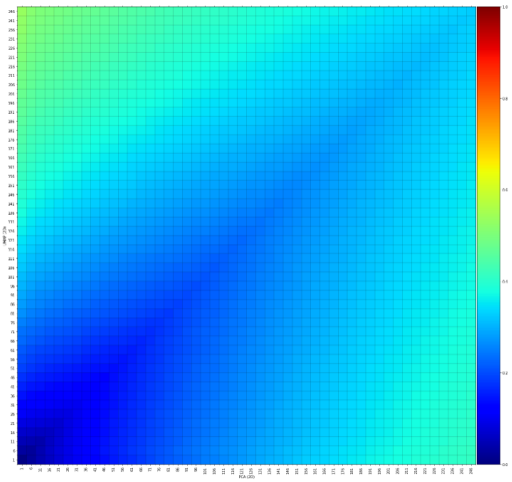

Figura 5: Heatmap de dimensiones 2D para $'PCA'$ y $'UMAP'$. La X (PCA) e Y (UMAP) de este gráfico son los vecinos más cercanos.

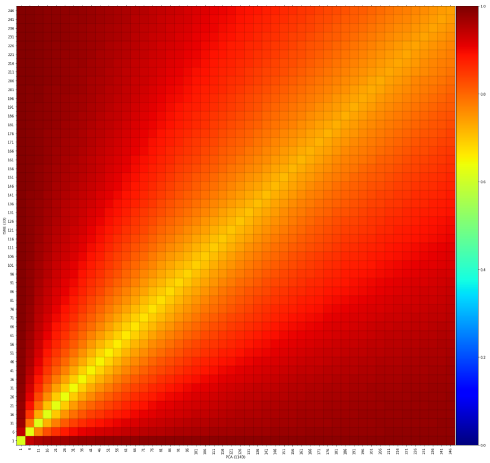

Figura 6: Heatmap de dimensiones óptimas para $'PCA(114componentes)'$ y $'UMAP(3componentes)'$. La X (PCA) e Y (tSNE) de este gráfico son los vecinos más cercanos.

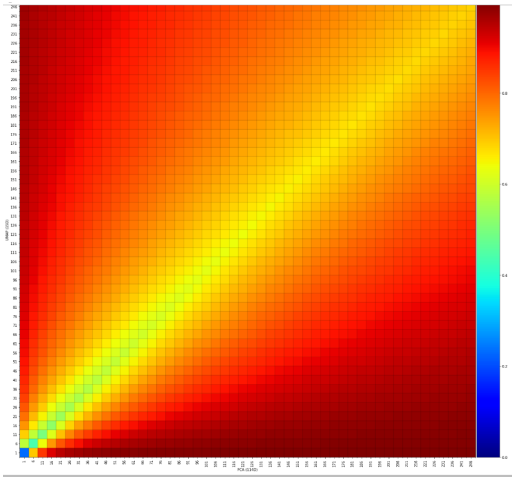

Figura 7: Heatmap de dimensiones óptimas para $'PCA(114componentes)'$ y $'UMAP(15componentes)'$. La X (PCA) e Y (UMAP) de este gráfico son los vecinos más cercanos.

Teniendo ya a mano los gráficos queda claro que donde mas difieren las transformaciones es cuando se toman pocos vecinos cercanos en consideración. Tambien la baja dimension tiene que ver. Se pierde mucha información en el caso de PCA y en UMAP un poco menos, no se pierde en tSNE (veremos más adelante como se comportó el accuracy con estos valores de dimensión). Esto sucede porque aquí es donde más entra en juego el criterio y la política de cada método. En el ambito 2D, notamos las mayores diferencias, mientras que, si tomamos la mejor reduccion de dimensión (lo veremos más adelante) para cada método, la semejanza es notable. Hay mucho rojo en los gráficos cuando se hace eso, por lo tanto queda visto que son bastante similares entre sí siempre y cuando los n componentes elegidos sean los ideales.

3.3.3. Dibujo de la muestra

Como último paso en este estudio de los rasgos generales de cada método, se dibujó en una matriz de 28x28 la imagen que mostraba el conjunto de datos ya transformado. En este caso, la transformación se hizo para dimensiones que dieron mejor rendimiento en la predicción para PCA y UMAP. Se utilizó la totalidad de la muestra de datos (60000) y no se muestra el resultado de tSNE porque eran 4 cuadrados ya que tSNE solo permite reducción de, como mucho, 3 dimensiones. Por lo que su imagen iba a ser predecible. Se intuyó que el dibujo que reflejaran tanto PCA como UMAP iba a, al menos, parecerse un poco a la imagen original. En años anteriores hubo experimentos sobre el dibujo de dígitos luego de realizar una reducción de dimensión con PCA, y los trazos de esta nueva muestra podían llegar a asemejarse bastante con la imagen del dígito original. No fue asi en este caso, donde la imagen que arrojan PCA y UMAP nada tienen que ver con la original. De todas maneras, valió la pena hacer el intento aunque no se llegara a lo esperado.

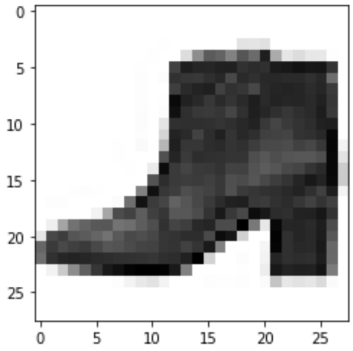

Figura 8: Dibujo original de la muestra de 60000 datos

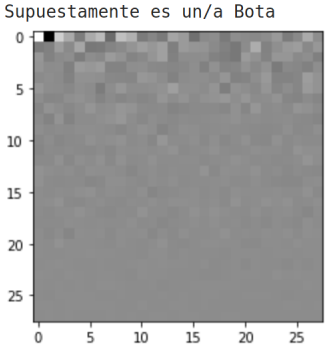

Figura 9: Dibujo de la muestra transformada a una dimensión de 114 mediante PCA.

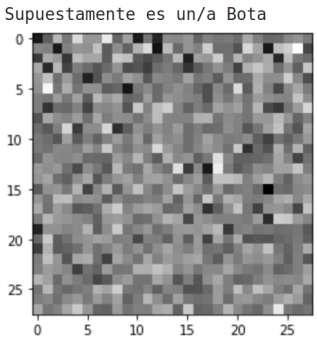

Figura 10: Dibujo de la muestra transformada a una dimensión de 15 mediante UMAP.

3.4. Estudio de kNN y las transformaciones lineales y no lineales

3.4.1. Estudio de k

El año pasado se estudió la diferencia entre el método kNN con y sin PCA. Se vuelve a realizar este experimento, no tan detallado como la anterior vez, para poder mostrar en qué se diferencia el rendimiento del clasificador con y sin transformación previa. Se busca encontrar el número de vecinos óptimo como así también la mejor reduccion posible en relación a la efectividad que pueda arrojar la métrica accuracy.

La muestra tomada será de unos 10000 ejemplares de entrenamiento. Para facilitar los tiempos de ejecución, ya que si se tomara toda la muestra para hacer todas las experimentaciones, llevaría demasiado tiempo. Esta cantidad de datos se guarda para la etapa final de la experimentación, donde se trata sobre la variación en las muestras elegidas.

Primero se ve como cae el rendimiento a mayor cantidad de vecinos cercanos. Es notable la pendiente negativa. Este gráfico arroja como resultado que el k ideal para la realización de kNN es 7.

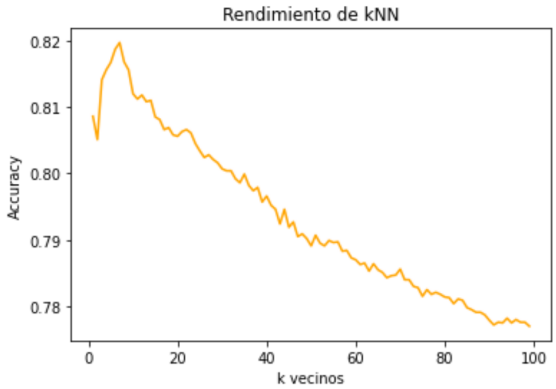

Figura 11: Variación de la efectividad del clasificador kNN sin transformación previa.

Luego se estudia cuál es la mejor reducción posible de dimensión para utilizar con PCA. Se podría haber seguido buscando con valores mayores a 150, pero el gráfico muestra que los valores no mejoran, como si se empezara a amesetar el dibujo. Por lo tanto, teniendo en cuenta la cantidad de tiempo de computo y energía computacional que requiere hacer más ejecuciones para llegar a valores de alfa, por ejemplo, mayores a 200, se decide hacer el corte en 150 y elegir el más alto. En este caso, el alfa más eficaz (en cuanto a la métrica accuracy) resulta ser el 114.

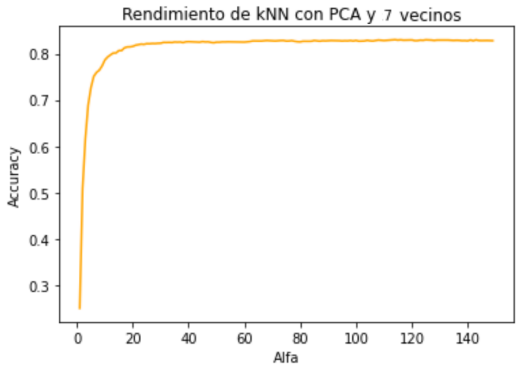

Figura 12: Variación de la efectividad del clasificador kNN según la reducción de dimensión mediante PCA.

Poniendo en comparación kNN con y sin PCA, se muestra la diferencia de rendimiento según los vecinos elegidos. Aunque haya una drástica reducción en la dimensión, PCA ayuda a obtener mayor performance en kNN. Aunque aumente la eficacia, el comportamiento de kNN sigue siendo igual, en el sentido de que la pendiente de ambos dibujos son casi idénticas.

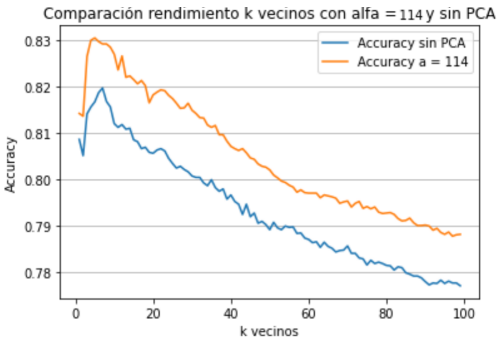

Figura 13: Variación de la efectividad del clasificador kNN para cada k con y sin PCA.

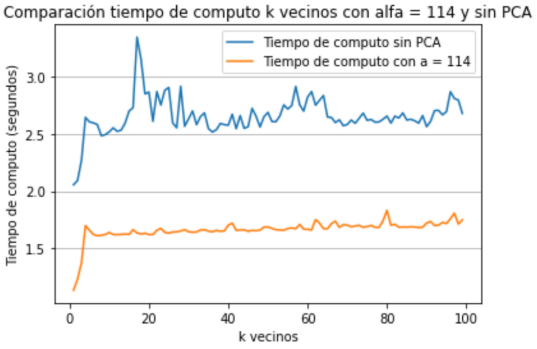

Figura 14: Tiempo de cómputo en segundos del algoritmo kNN con y sin transformación PCA.

El análisis de los temas ya vistos termina con la comparación del tiempo de cómputo de kNN con y sin PCA. Esperanzador resulta ser el resultado cuando vemos el gráfico. Llega a haber una diferencia del doble de tiempo, por ejemplo, cuando tomamos k igual a 20, donde el kNN con X reducido mediante PCA performa en 1,7 segundos mientras que el clasificador kNN sin reducción de dimensión previa lo hace en unos 3,4 segundos. Notable mejora. Aún así, esta comparación se realiza sin tener en cuenta el tiempo que le requiere a PCA realizar la transformación. De esto se hablará mas adelante.

Con PCA y kNN y su relación ya estudiada. Veamos cómo se comporta el clasificador de vecinos con transformaciones no lineales, tales como t-SNE y UMAP.

3.4.2. Estudio de t-SNE y kNN

Se testea qué tan bien se llevan t-SNE y kNN. Y como bonus, tomamos uno de los parámetros que provee t-SNE y se juega con él para ver en qué afecta esto al rendimiento del método.

Lastimosamente al poder hacer solo reducciones hasta 3D, tSNE no va a proveer muchas instancias. Aún así, con 3 alcanza para ver la clara mejora que hay si tomamos la tercera dimensión como ejemplo. Provee una accuracy mejor que PCA con alfa igual a 3. Por lo tanto, si sí o sí se necesitan 3 dimensiones, t-SNE empieza a ganar buena fama, mas no se sabe aún como puede compararse t-SNE con PCA y UMAP cuando estos tienen su mejor número de componentes en términos de rendimiento.

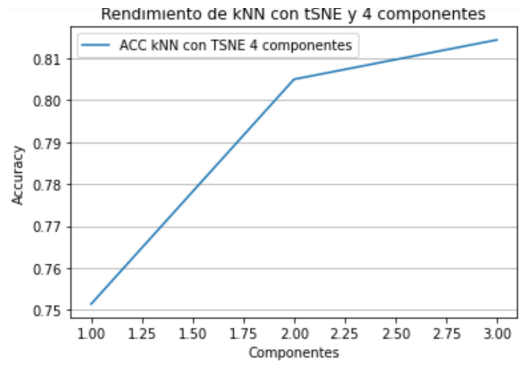

Figura 15: Variación de la efectividad del clasificador kNN con reduccion tSNE segun la dimensión.

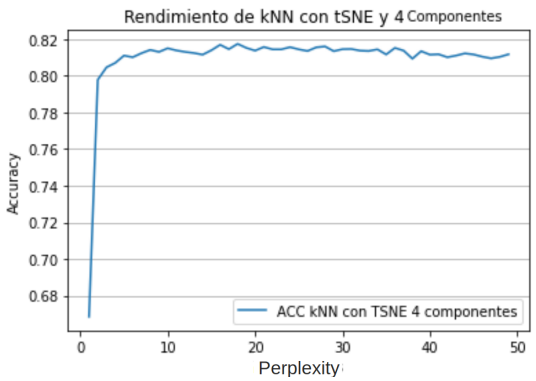

Figura 16: Variación de la efectividad del clasificador kNN con reduccion tSNE segun el Perplexity.

Se experimentó con el valor de Perplexity entre 1 y 50, ya que en el paper oficial del método se provee la siguiente información: *"The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50."*. Según el paper, el Perplexity se puede interpretar como una medida que suaviza el número efectivo de vecinos que se terminan teniendo en cuenta en el algoritmo.

Al ver el gráfico, se entiende por qué el valor tomado generalmente varía entre esos valores mencionados. Hay una clara meseta donde el valor no varía mucho, de hecho, la pendiente empieza a bajar, por lo tanto, con elegir un perplexity entre 10 y 20 suena como una buena decisión. El gráfico arroja como el mejor valor para Perplexity el 18.

Con los valores óptimos ya conocidos, ya tenemos listas dos instancias para el estudio de muestras. Los valores para PCA y t-SNE ya están. Falta averiguar el escenario óptimo para UMAP.

3.4.3. Estudio de UMAP y kNN

Ahora hay que encontrar la relación entre kNN y UMAP. Como sucede con PCA y t-SNE, se empieza con el estudio de los componentes a tener en cuenta para la reducción de la dimensión del espacio. Luego, se toma un parámetro del método UMAP, en este caso, la Minimum Distance. Segun la web del método, este parámetro trata sobre proveer la mínima distancia que debe haber entre todos los puntos en la reducción de la dimensión.

El gráfico dice que su dibujo es muy similar al de PCA y sus componentes. Tambien se empieza a mesetar aunque los valores sufren una mayor variación que en el caso de PCA, donde practicamente parece un dibujo de una funcion logarítmica perfecta.

El valor que se impone sobre el resto en este caso es 15. Buena noticia que haya un bajo número, esto reducirá considerablemente los tiempos de ejecución de kNN.

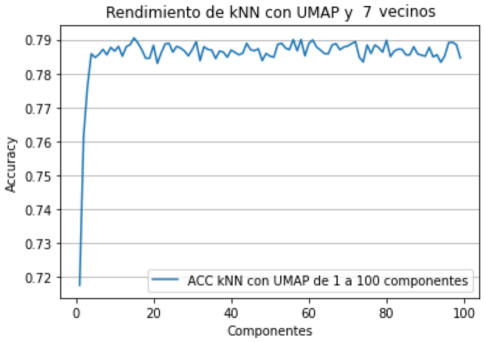

Figura 17: Variación de la efectividad del clasificador kNN con reduccion UMAP segun la dimensión.

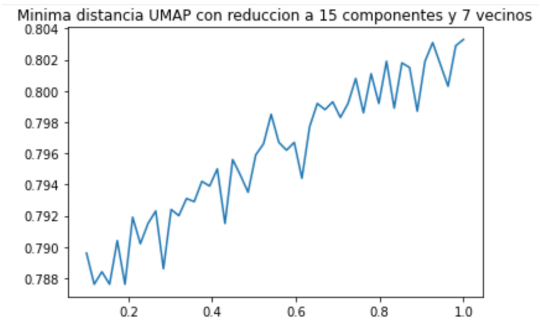

Figura 18: Variación de la efectividad del clasificador kNN con reduccion UMAP segun el Minimum Distance.

El valor por defaul de Minimum Distance es 0.1, y puede variar entre 0 y 0.99. Por lo tanto, se toma una muestra entre este rango de valores para ver si hay un cambio significativo en la performance de kNN con la transformación. Se llega a un gráfico lineal con pendiente positiva, dando a relucir el buen rendimiento que arroja una mayor Minimum Distance, en este caso, el valor ganador es 0.99.

Ya se descubrieron los valores necesarios para realizar el experimento sobre las distintas muestras. Primero se van a ver un par de diferencias más entre cada método.

3.4.4. kNN entre PCA, t-SNE y UMAP

Primero se estudia la diferencia entre las dimensiones de PCA y UMAP. A priori se nota que UMAP termina siendo mucho más estable. La variación de la efectividad va de 0.72 a 0.79, un spread del 0.07. Mientras que PCA tiene una variación de 0.3 a 0.83, spread del 0.53, mucho mayor. Con este primer dibujo se nota la superioridad de PCA en dimensiones altas por sobre UMAP, pero también UMAP saca a relucir su poderío en las dimensiones más bajas. Al ser no lineal como t-SNE, se supone que este último sera también más efectivo que PCA en bajas dimensiones. Por lo tanto, se plotea un grafico sobre la efectividad en bajas dimensiones para los tres transformadores más abajo.

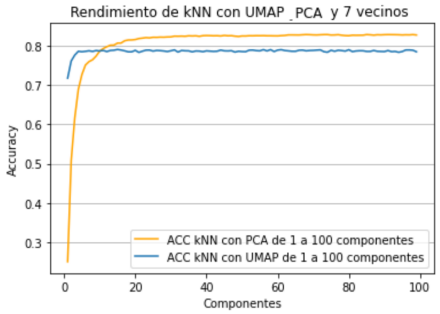

Figura 19: Variación de la efectividad de kNN con reduccion hasta 100D para PCA y UMAP.

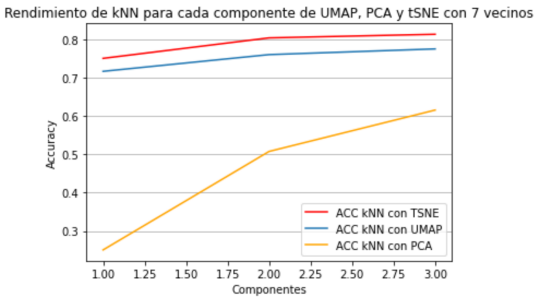

Figura 20: Variación de la efectividad de kNN con reduccion hasta 3D para las 3 transformaciones.

Aquí se ratifica la hipótesis previamente formulada donde se ve claramente la superioridad de los metodos no lineales por sobre el lineal en bajas dimensiones. Son más exactos y fidedignos en la reducción drástica de dimensiones. Un problema que surge aquí, es que el tiempo que tomaron los componentes no lineales fue muchísimo mayor que PCA. Se estudiará que tan importante fue esto.

3.5. Estudio de muestras

En ultima instancia, se decidió ver qué pasa cuando empleamos el escenario óptimo para cada transformador, junto con kNN. Se pensó que el algoritmo de kNN quedaria con peor rendimiento, a lo sumo, rendiría mejor que tSNE, ya que este método reduce demasiado la dimension. Pensar que si se encuentra con una muestra de 50000 elementos, reducir a 3 dimensiones tal dataset implica una perdida de exactitud en algun punto. Uno querría creer que esto es lo que sucede. Otra hipótesis que hubo es que, como UMAP tiene más componentes a tener en cuenta, significa que posee una mayor personalización, por lo que podría hacerlo más efectivo que PCA, donde solo se provee la cantidad de componentes a reducir la dimensión y listo. Este no fue el caso.

Ambas hipotesis fueron desmentidas, ya que el grafico arroja contundentes resultados donde se puede notar exactamente que PCA y kNN sin transformación son superiores a ambos métodos no lineales. En la única instancia donde tSNE tuvo resultados prometedores fue en la de una muestra de 2500 elementos. Pero esto es mas que despreciable ya que no es para nada representativo. Si se agarrase otra muestra de la misma cantidad, podría dar una efectividad completamente distinta. PCA termina como claro vencedor, mientras que kNN sin transformación le sigue, ganándole a tSNE y UMAP. El gran perdedor es UMAP, que es quien arrojó peores resultados. Hasta performa peor que tSNE, una transformacion de muchísimas menos dimensiones.

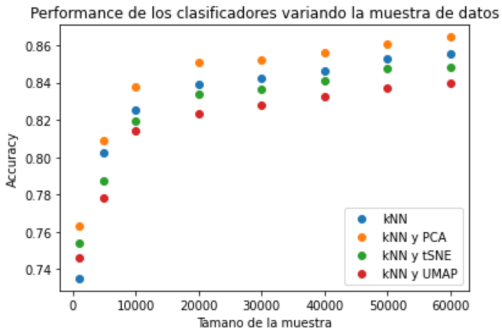

Figura 21: Efectividad de kNN sin reducción y con reduccion óptima para PCA, t-SNE y UMAP, según tamaño de la muestra inicial.

3.6. Estudio de tiempos de cómputo

El último testeo se enfocó en los tiempos de ejecución y cómputo para cada método. Fue curioso ver como, algunos tardaban muchos minutos en realizar la transformación. Pero, una vez ya hecha, el algoritmo de kNN con los datos transformados era rapidísimo. Un arma de doble filo. La elección de qué elemento usar queda ya en la preferencia de cada uno.

Las primeras dos figuras se centran en el tiempo de computo de kNN luego de haber sido transformada la muestra. No centra su atención en cuánto le toma a cada transformación realizarse. En este caso, vemos un buen punto positivo para UMAP (teniendo en cuenta su mal rendimiento comparado con las otras transformaciones en el estudio de muestras). Aquí suele ser constantemente el método que más facilita la ejecución rápida de kNN. Un punto en contra para PCA aquí es que suele dar la transformación que más hace tardar a kNN. Aún asi, su comportamiento es casi identico al tiempo de UMAP, basta con verlo en la Figura 22.

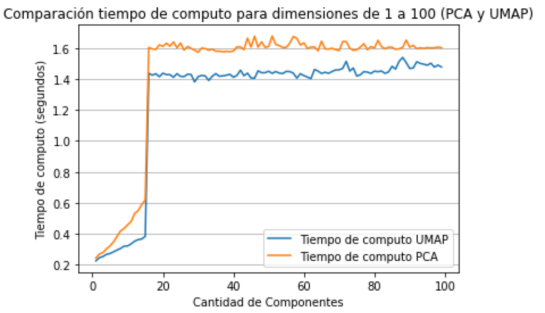

Figura 22: Tiempo de cómputo del algoritmo kNN con reduccion para PCA y UMAP, según componentes.

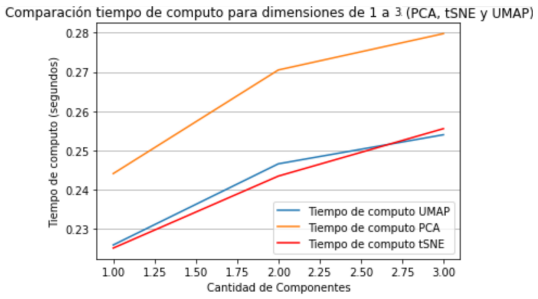

Figura 23: Tiempo de cómputo del algoritmo kNN con reduccion para PCA,t-SNE y UMAP, según componentes.

Estos graficos anteriores parecieron un poco "mentirosos"(no lo son) en el sentido de que, tomaba muchísimo tiempo, por ejemplo, computar cualquier cosa que tuviera que utilizar tSNE para reducir la dimensión. Un tiempo realmente exagerado. Y en la Figura 23 este método arroja tiempos ínfimos (pero lo que da esos tiempos es la muestra ya transformada). Por lo tanto esto motivó a estudiar realmente cuánto tiempo tarda cada método en realizar la transformación del espacio. Los resultados fueron reveladores.

Claramente hay un categórico distanciamiento entre t-SNE y el resto. La cantidad de tiempo que toma llega hasta más de los 500 segundos (9 minutos aproximadamente) para reducir una

muestra. Completamente distinto a los demás. Los vencedores terminan siendo PCA y UMAP en especial. Performando muy por debajo de tSNE y kNN sin reducción. Esto habla de la calidad y rapidez del algoritmo de estos métodos para reducir tantas dimensiones a una porción muy acotada. Esto da a pensar que tal vez t-SNE se especialice más que nada en la visualización de datos (1D, 2D o 3D). Ya que para proveer una transformación rápida, no resulta ser el más beneficioso.

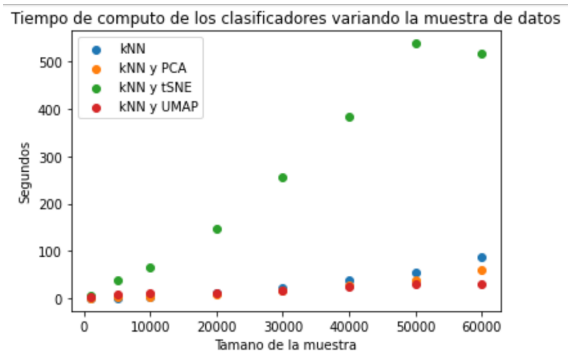

Figura 24: Tiempo de cómputo del algoritmo kNN con y sin reduccion para PCA, t-SNE y UMAP, según el tamaño inical de la muestra.

4. Conclusiones

Se concluye que los resultados arrojados por k-Nearest Neighbors a la par con los de las transformaciones PCA, t-SNE y UMAP son bastante eficientes para el reconocimiento de dígitos. Aún así, su performance no fue la esperada. Dan resultados mucho menores a los que arrojó la implementación de kNN de nuestro Trabajo Práctico el año pasado, donde se llegaron a alcanzar aciertos con una efectividad que rondaba el 0.95, mientras que en este trabajo la mayor efectividad alcanzada por cualquier método, con la mayor muestra posible, fue poco mayor al 0.86. Un spread considerable entre ambos resultados.

Las transformaciones presentadas no se parecen con PCA cuando las tres se encuentran en bajas dimensiones. La diagramación de los puntos en el plano no es similar. Esto cambia cuando se toma para cada método el número de componentes ideal. Allí también se emparejan los valores de sus predicciones y la curva de las mismas.

kNN demostró por sí solo ser bastante fiable. Performa mejor que con tSNE y UMAP. Aún así, sin la muestra transformada tarda más tiempo que con la transformación, a tener en cuenta.

PCA demostró ser la transformación más efectiva, superando a kNN por si sólo. Su procesode transformación no tarda demasiado en computar, aunque su dimensión óptima es mayor que la de los métodos no lineales por mucha diferencia, por lo que el tiempo de ejecucion de kNN con la muestra transformada, no es de lo más rapida, aunque tampoco es que sea lenta, para nada. No es un método para nada efectivo en bajas dimensiones, donde se suele querer visualizar los datos. Sus puntos estan muy desperdigados en el plano y no hay una discriminación por grupos clara. Aún asi, al ser lineal, si se quisiera medir un comportamiento del data frame con una línea de regresión, se puede hacer tranquilamente, mientras que en los no lineales no.

tSNE termina siendo útil para la visualización de datos y para la buena representación en la reducción de muchos datos a muy bajas dimensiones. Compite con dimensiones abismalmente más grandes hechas por otros métodos y está par a par en cuanto a efectividad de predicción se refiere. Lamentablemente, estas virtudes las acompaña con un gigante tiempo de procesamiento, tarda mucho en efectuar su algoritmo, llegando a estar mas de 9 minutos para realizar una reducción

dimensional. Grandes pros y una gran contra para este método.

UMAP empezó sin dar resultados brillantes, siempre daba menor efectividad que algún otro método. De hecho, es el método que, en situacion óptima, peor performa contra el resto, en todos los tamaños de muestra representativos. Su virtud radica en el poco tiempo de ejecución que presenta. La reducción de dimensión la hace muy rápido y tampoco es que su predicción sea mala por estar debajo que las otras.

Por lo tanto, podemos sacar el siguiente resúmen de las conclusiones para cada transformación:

kNN: Es un algoritmo muy efectivo, pero da peores resultados que su implementación hecha el año pasado.
PCA: Es la transformación que hace más efectiva la predicción, pero no sirve para dimensiones bajas.
t-SNE: Ideal para bajas dimensiones (interpretables para el ojo humano, 1D, 2D, 3D). Demasiado tiempo para realizar el algoritmo.
UMAP: Efectividad aceptable, ideal cuando se necesitan bajos tiempos de cómputo.

Referencias

[1] https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

[2] https://umap-learn.readthedocs.io/en/latest/parameters.html

[3] https://www.kaggle.com/datasets/zalando-research/fashionmnist

[4] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[6] https://github.com/erdogant/flameplot/