DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Informe

4 de julio de 2022

Métodos Numéricos

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
https://exactas.uba.ar

Índice

1. Introducción

El objetivo de este trabajo es el desarrollo de un clasificador basado en k Nearest Neighbors (kNN) y Principal Component Analysis (PCA). El set de datos sobre el que se trabajará es "Fashion mnist"[1] cuyas instancias representan distintos tipos de vestimenta. De manera similar a MNIST consiste en imágenes de 28x28 en escala de grises representadas mediante vectores de tamaño 784. Para hacer un análisis de la efectividad de los métodos a probar, se utilizará la técnica de K-Fold cross validation.

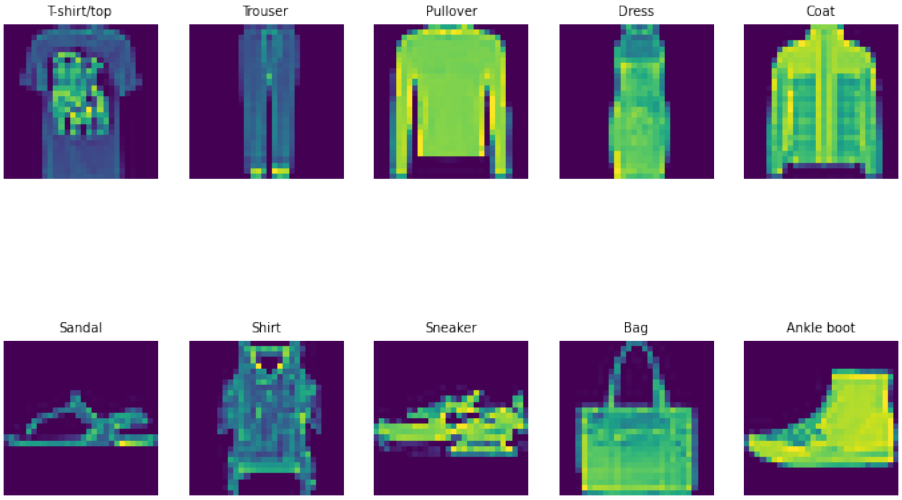


Figura 1: Ejemplos de instancias del dataset Fashion mnist

2. Experimentación

Si bien "Fashion mnist"[1] provee un set de datos etiquetados de 60.000 instancias y uno de 10.000, para realizar la experimentación que se presenta a continuación te utilizó una muestra de tamaño 3000 tomada considerando el balance de las clases. A menos que se indique lo contrario, el conjunto de entrenamiento y validación que se menciona en los experimentos se obtiene de la muestra mencionada.

2.1. K-Fold cross validation

KFold cross validation consiste en dividir el conjunto de prueba en **K** particiones. Luego, tomar cada una de las particiones como conjunto de validación y el resto como conjunto de entrenamiento. Luego uno puede promediar las métricas obtenidas para las distintas particiones. De esta manera se busca independizar el análisis del modelo del conjunto de datos utilizado.

Figura 2: Diagrama K-Fold cross validation[2]

Para poder utilizar este método de validación, es necesario elegir un K apropiado. Dado que hay que ajustar un modelo para cada una de las particiones hechas, buscamos encontrar un valor de **K** que permita hacer un buen análisis

del modelo sin aumentar demasiado los tiempos de ejecución. Si bien un **K** igual al tamaño del conjunto (Leave-One-Out-Cross-Validation LOOCV) provee la mayor cantidad de conjuntos de entrenamiento distintos, haría muy costoso realizar experimentación sobre otros parámetros. Aún así es posible utilizarlo como referencia. En la figura 3, se puede observar la precisión promedio para cada partición del conjunto con una indicación del máximo y mínimo. Si se asume LOOCV como una estimación óptima, hay que buscar un valor de **K** que estime de manera similar con una varianza relativamente baja. En base a los resultados de la figura 3 se decidió tomar **K**=10 para realizar la toda la experimentación presentada en este informe.

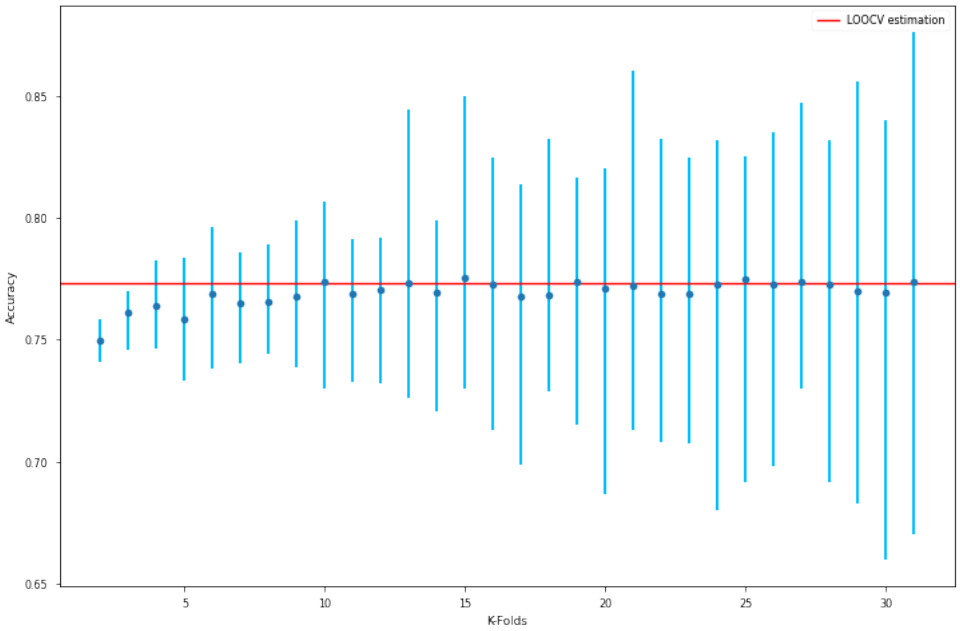


Figura 3: Accuracy promedio de kNN para distintos **K**

2.2. k-Nearest Neighbors (kNN)

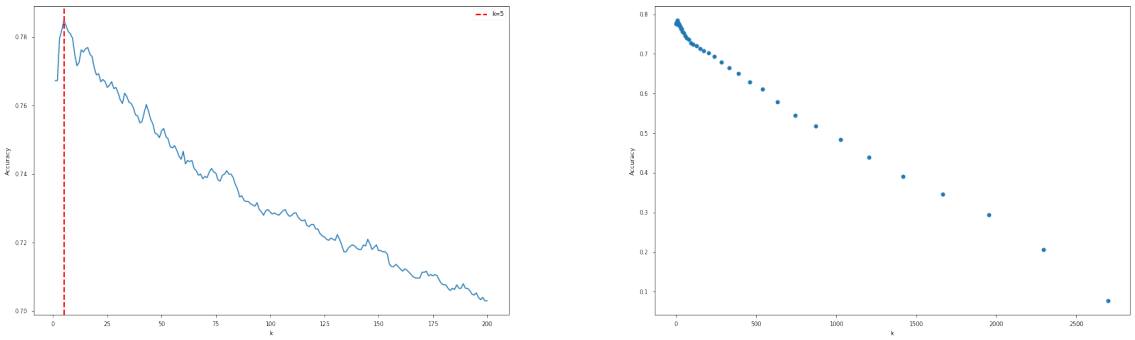
2.2.1. Precisión

En esta sección se desarrolla la experimentación realizada sobre el parámetro **k** de kNN. Con el objetivo de observar si hay alguna relación entre el valor de **k** y el tamaño del conjunto de entrenamiento, se realizó una serie de test cuyos resultados se pueden observar en la figura 4. Se ajustó el modelo con conjuntos de entrenamiento de tamaño creciente, utilizando distintas estrategias para elegir la cantidad de vecinos. En base a los resultados parece no haber tal relación, dado que tanto para valores constantes como para valores proporcionales al tamaño del conjunto de entrenamiento, la precisión varía muy poco (a excepción de los tamaños más bajos).

Adicionalmente podemos observar otros comportamientos, por un lado la precisión aumenta a medida que crece el conjunto de entrenamiento. Esto es esperable dado se posee más información a la hora de clasificar. Por otro lado parece ser que se obtiene una precisión mayor para valores de **k** mas bajos.

Figura 4: Accuracy de kNN para distintas maneras de tomar valores para **k**

Con los resultados anteriores en mente, se realizaron los experimentos de la figura 5, para los cuales se ajustó el modelo con un conjunto de entrenamiento y se analizó la precisión alcanzada para distintos valores de **k**. En 5(a) se observa que los valores óptimos se encuentran en el rango de 1 a 25 aproximadamente y que la precisión decrece a medida que aumenta la cantidad de vecinos. En este caso particular la mayor precisión se encuentra para **k** igual a 5. A modo de confirmar la tendencia observada en 5(a), se realizó el experimento 5(b) donde se observa claramente el decrecimiento de precisión a lo largo del rango de valores posibles.



(a) **k** tomados de manera lineal

(b) **k** tomados de manera logarítmica

Figura 5: Accuracy de kNN para distintos valores de **k**

2.2.2. Performance

Con respecto al tiempo de ejecución de kNN, como es de esperarse el valor de aumenta dicho tiempo. Dado que el peor caso es igual (sigue dependiendo del tamaño del conjunto de entrenamiento y no de k), el valor afecta las constantes pero no la complejidad temporal. De todas maneras, se encontró que no hay relación entre el valor de **k** y el tamaño del conjunto, más aún parece ser que el modelo se desempeña mejor para valores más bajos.

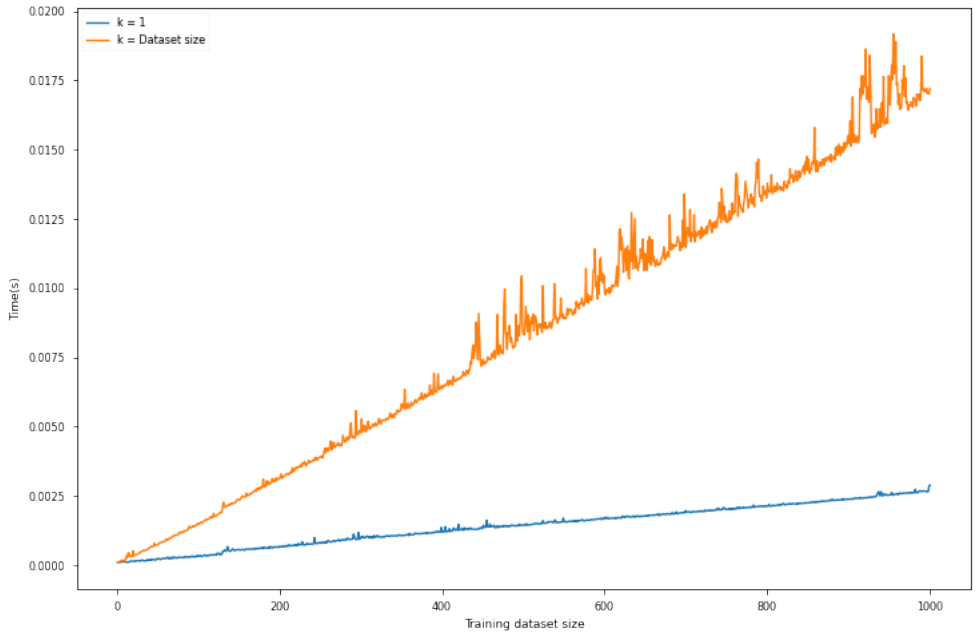


Figura 6: Tiempo de ejecución kNN para distintos tamañós de conjunto de entrenamiento

2.3. Principal component analysis (PCA)

Una manera de mejorar tanto la precisión como el tiempo de ejecución es implementar un método de reducción de dimensión. En este caso se implementó PCA. Se espera que la reducción de la dimensión produzca una mejora tanto en el tiempo de ejecución como en la precisión del modelo. Se espera que al utilizar únicamente los α componentes más significativos se descarte información "redundante" que pueda estar generando ruido a la hora de clasificar. La mejora en tiempo de ejecución es más evidente dado que se reduce el tamaño de las instancias que toma el algoritmo de kNN como entrada. El valor de α determina la dimensión de la transformación resultante.

2.3.1. Visualización de categorías

Si se quiere visualizar los datos del conjunto de entrenamiento, es útil buscar una representación en $\mathbb{R}^2$ que permita comprender un poco mejor que esta sucediendo. Una opción es realizar el cambio de base de PCA utilizando únicamente los dos primeros componentes. Para los experimentos siguientes, se utilizó el conjunto de entrenamiento original de tamaño 60.000. En la figura 7 se puede ver lo mencionado anteriormente. Si bien se puede distinguir cierta separación entre las distintas clases, claramente dos componentes no serán suficientes ni para visualizar ni para clasificar.

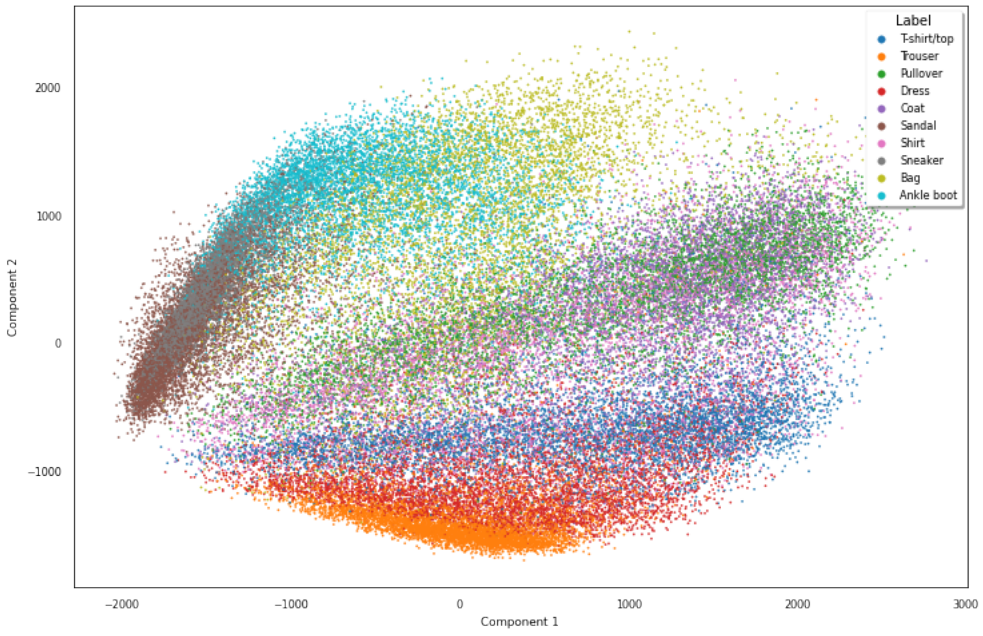


Figura 7: Proyección del dataset de entrenamiento al espacio generado por los primeros dos componentes principales

Para visualizar mejor se pueden utilizar otros métodos como T-distributed stochastic neighbor embedding (TSNE) que permiten visualizar en $\mathbb{R}^2$ datos de dimensión mucho mayor.

En la figura 9(a) se ve la aplicación de TSNE al conjunto de entrenamiento. Se puede ver más claro que categorías serán más fáciles de clasificar. Las imágenes de *Trouser*, *Bag* y *Dress* se encuentran bastante distinguidas del resto dado que tienen poca similitud con el resto. Las clases más problemáticas serán las que presentan diferencias más sutiles entre sí, estas son *Sandal*, *Sneaker* y *Ankle boot*; *Pullover*, *Coat*, *Shirt* y *Tshirt/Top*. El segundo conjunto de clases será más problematico que el primero.

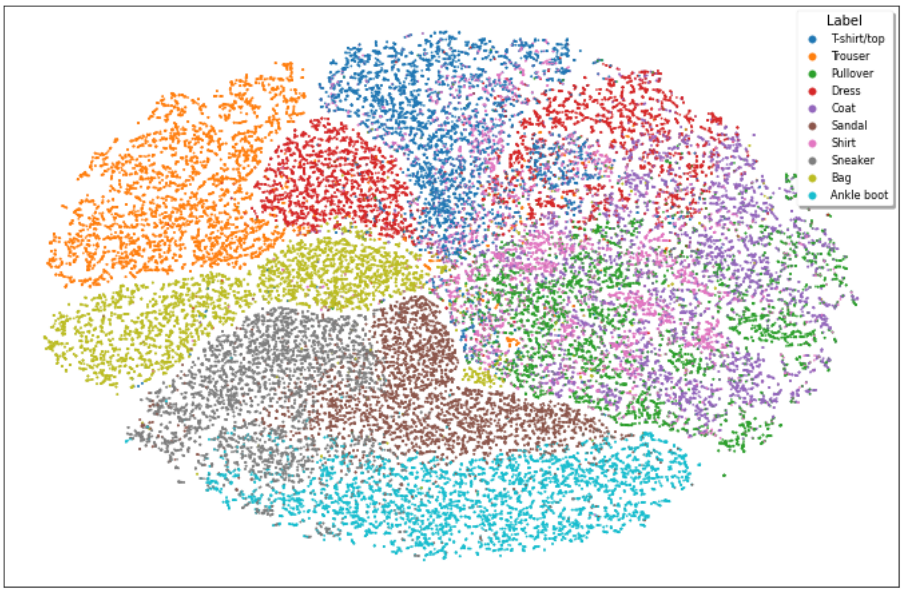
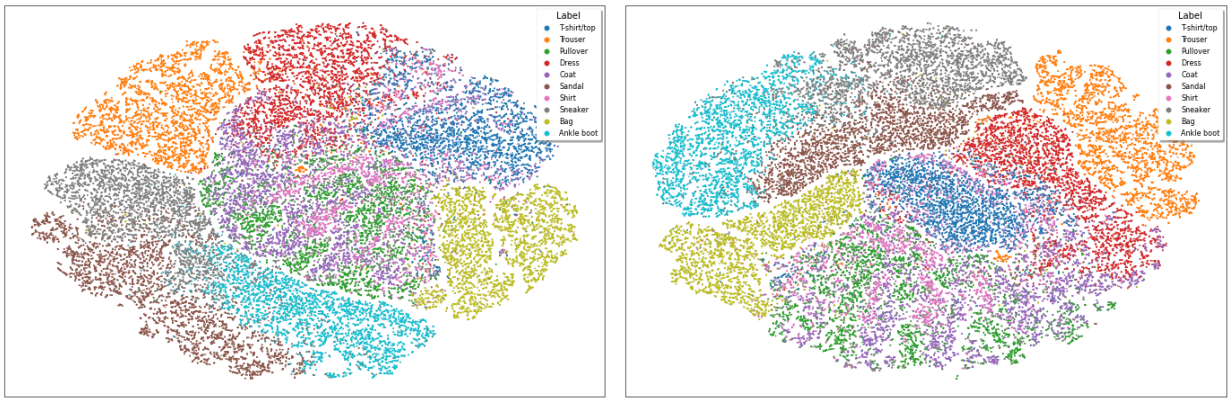


Figura 8: TSNE sobre conjunto de entrenamiento

Con el objetivo de observar el efecto de la reducción de dimensión, se aplicó TSNE al conjunto de entrenamiento transformado a través de PCA con distintos valores de α. En la figura 9 se pueden observar los resultados para $\alpha = 50$ y $\alpha = 100$. No parece haber una mejora significativa en la diferenciación de clases, al menos utilizando TSNE. Es posible que PCA solo produzca mejoras en tiempo de ejecución y no en precisión a la hora de clasificar.



(a) $\alpha = 50$ (b) $\alpha = 100$

Figura 9: TSNE sobre el conjunto de entrenamiento luego de aplicar PCA

2.3.2. Varianza explicada

Al realizar una reducción de dimensión hay una perdida de información, en el caso de PCA la reducción se hace de tal manera de quedarse con la información que mayor diferencie a las categorías. En la figura 10 se observa la varianza explicada por cada componente y la acumulada. A partir de $\alpha = 20$ se acumula una varianza explicada del 80 %. A medida que crece la cantidad de componentes, la varianza explicada que aporta cada uno disminuye significativamente, por lo tanto es posible que a partir de cierto valor se obtengan resultados similares a un costo de ejecución mayor.

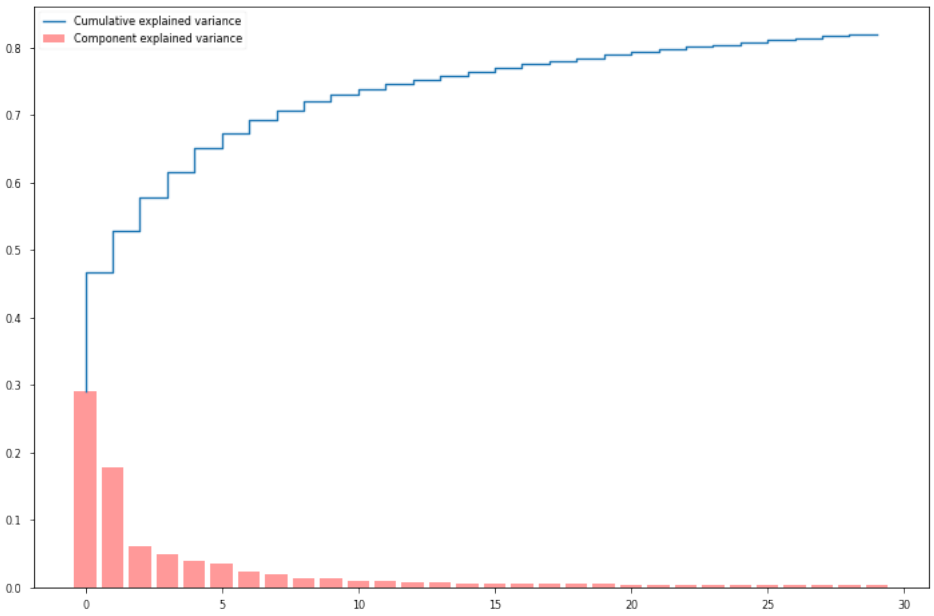


Figura 10: Varianza explicada individual y acumulada para cada componente de PCA

2.4. kNN + PCA

Luego de analizar tanto los parámetros de kNN como de PCA por separado, se experimentó sobre los métodos utilizados en conjunto. En esta sección se busca ver como se relacionan **k** y α y como son los resultados tanto de precisión como de tiempo de ejecución.

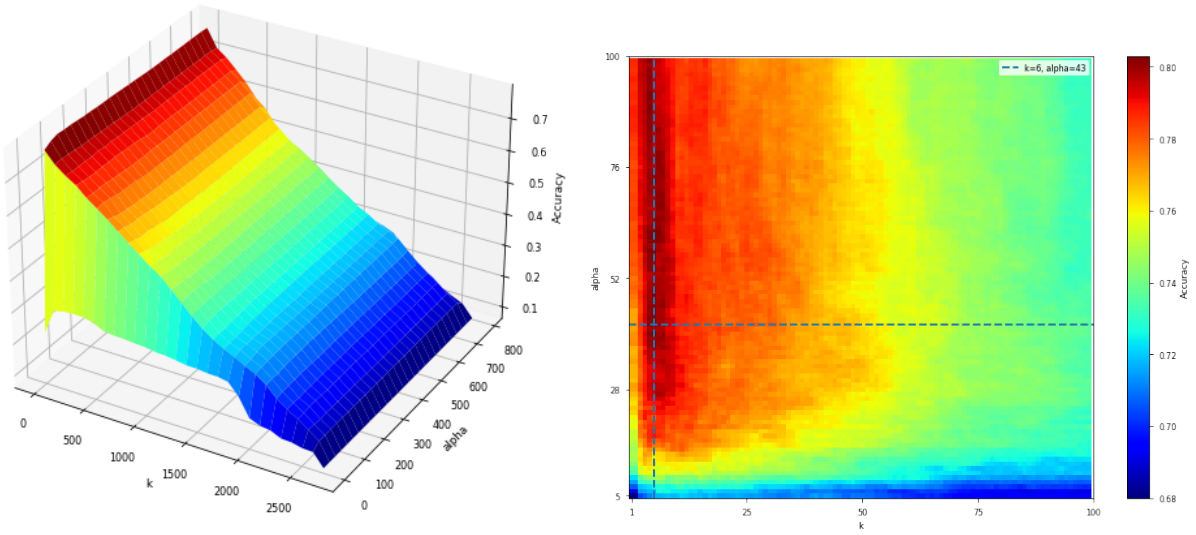


Figura 11: Precisión obtenida para distintos valores de **k** y α

En la figura 11 se observa claramente que los valores óptimos para ambos parámetros en este caso particular se encuentran para **k** entre 1 y 10 aproximadamente y para α de 28 en adelante. Para los valores de **k** se observa la tendencia de los experimentos anteriores, la mejor performance se obtiene para valores de más bajos. A excepción de valores muy bajos de α, la precisión obtenida no parece variar demasiado. Dado que se utiliza PCA para reducir la dimensión, son de interés los valores de α más bajos posibles. Particularmente para este conjunto de entrenamiento, el valor de precisión máxima se encontró para **k**= 6 y $\alpha = 43$.

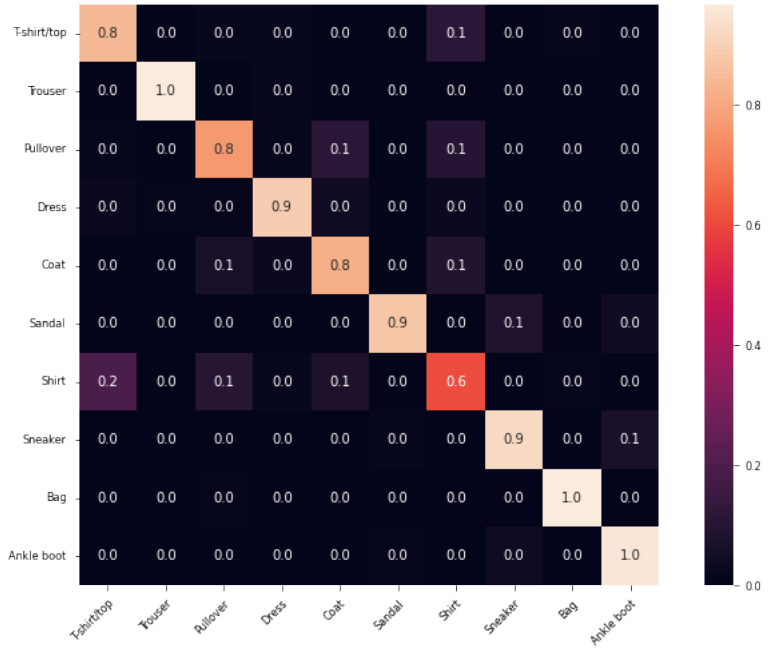


Figura 12: Matriz de confusión kNN + PCA conjuntos de entrenamiento y validación originales. True Label(y) vs. Predicted Label(x)

A raíz de los resultados anteriores, se realizó una última serie de experimentos. Por un lado se ajusto el modelo de kNN con el conjunto de entrenamiento original y **k**= 6 y se utilizó para predecir las etiquetas del conjunto de validación original. El tiempo de ajuste y predicción fue de 460,510s obteniendo una precisión de 0.860. Por otro lado se aplicó PCA con $\alpha = 43$ y se ajustó kNN con **k**= 6. El ajuste y transformación de PCA tomó 4,740s, el tiempo de ajuste y predicción de kNN tomó 21.232s, obteniendo así un total de 25,972s. Se puede ver claramente la diferencia entre kNN solo y kNN + PCA en tiempo de ejecución. No fue así para la precisión, para kNN + PCA se obtuvo una precisión de 0,859 que es muy similar a la obtenida para kNN.

En la figura 12 se puede ver la matriz de confusión de el experimento kNN + PCA. Como se había indicado anteriormente, las clases para las cuales la performance fue peor son *Pullover*, *Coat*, *Shirt* y *Tshirt/Top*, siendo *Shirt* particularmente mala.

3. Conclusión

En conclusión se logro encontrar rangos de valores óptimos tanto para **k** como para α. Si bien no se logró obtener una mejora de precisión utilizando PCA, si se pudo observar una mejora significativa en tiempo de ejecución sin perjudicar demasiado la precisión. Se pudieron comprobar los problemas observados mediante TSNE con respecto a la predicción de ciertas clases. Para un trabajo futuro se podría buscar soluciones a los problemas mencionados mediante otros métodos de clasificación distintos.

Referencias

[1] `https://github.com/zalandoresearch/fashion-mnist`

[2] By Gufosowa - Own work, CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=82298768`