Informe Proyecto Final Reconocimiento de Patrones

Proyecto estándar: Clasificación de fashion-MNIST

Federico Aicardi - CI: 3.817.031-2
Fernando Viera - CI: 4.598.458-8

Indice

1. Introducción

En el área de reconocimiento de patrones, la base de dígitos manuscritos MNIST ha sido un estándar a la hora de comparar la performance de diferentes algoritmos. Esta base de datos consiste en imágenes de 28x28 píxeles de dígitos escritos a mano contando por lo tanto con 10 clases (los dígitos del 0 al 9).
Puesto que con técnicas relativamente sofisticadas de clasificación, utilizar dicha base como benchmark para comparar algoritmos ha quedado en relativo desuso debido a su baja dificultad, se ha propuesto la base alternativa FMNIST.
Al igual que la base MNIST, la nueva base propuesta consta de imágenes de 28x28 pixeles repartida en 10 clases, siendo los features disponibles los valores entre 0 y 255 correspondientes al nivel de gris de cada pixel (784 features).
La base de datos está en formato .csv, correspondiendo cada línea a un patrón. El primer valor es la clase (valor entre 0 y 9) y el resto de las columnas contiene valores entre 0 y 255 correspondientes a los valores del nivel de gris del pixel. El valor de la imagen en las coordenadas (i,j) está guardado en la característica correspondiente a la posición x = 1 + i * 28 + j, con i y j tomando valores entre 0 y 27. La primera línea del archivo “csv” es un encabezado indicando la clase y el número de característica.
Las clases representadas en la base son:

0. Camiseta
1. Pantalon
2. Pulover
3. Vestido
4. Saco
5. Sandalia
6. Camisa
7. Zapatilla
8. Bolso
9. Bota

1.1 Objetivos

El objeto del proyecto será trabajar con dicha base de datos desde la etapa de extracción y selección de características, evaluando el desempeño de los diferentes clasificadores con el objeto de lograr tasas mayores o iguales al 90% de efectividad. Se utilizarán las matrices de confusión para estudiar las parejas de clases que son altamente confundidas por el clasificador entrenado y de esta manera detectar y proponer soluciones adicionales que mejoren su desempeño.

2. Problema fashion-MNIST

Dado que el problema fashion-MNIST tiene características similares al MNIST se busca aplicar técnicas ya probadas sobre el mismo. En ambos casos se trata de imágenes de 28x28 pixels,

donde las características disponibles son los valores en la escala de grises en cada unos de los 784 pixels. Por este motivo se buscaron técnicas similares a las ya utilizadas sobre MNIST.

2.1. Benchmarks existentes

En la propia página de quienes proponen esta base como sustituta de la MNIST, Zalando Research, se pueden encontrar distintos resultados ya obtenidos, siendo éstos utilizados para obtener una primera aproximación al problema. La tabla de la figura 2.1 muestra algunos de los resultados registrados en dicha web.

Name	Parameter	Accuracy (mean)	Accuracy (std)
SVC	{"C":10,"kernel":"poly"}	0.897	0.000
SVC	{"C":100,"kernel":"poly"}	0.896	0.000
SVC	{"C":10,"kernel":"rbf"}	0.896	0.000
SVC	{"C":100,"kernel":"rbf"}	0.893	0.000
GradientBoostingClassifier	{"loss":"deviance","max_depth":10,"n_estimators":100}	0.888	0.001
SVC	{"C":1,"kernel":"rbf"}	0.884	0.000
GradientBoostingClassifier	{"loss":"deviance","max_depth":10,"n_estimators":50}	0.880	0.001
RandomForestClassifier	{"criterion":"entropy","max_depth":50,"n_estimators":100}	0.879	0.000
RandomForestClassifier	{"criterion":"entropy","max_depth":100,"n_estimators":100}	0.877	0.001

Figura 2.1

2.2. Primera aproximación

En primera instancia, se utilizó la herramienta Weka para atacar el problema. Para las primeras pruebas que se realizaron se utilizaron distintos clasificadores con los parámetros por defecto, y sin realizar extracción previa. A continuación se detallan las primeras pruebas realizadas con el fin de obtener una primera aproximación al problema.

2.2.1. SVM sin extraccion

Utilizando un clasificador SVM los resultados obtenidos fueron los siguientes,

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	51382	**85.6367 %**
Incorrectly Classified Instances	8618	14.3633 %
Kappa statistic	0.8404	
Mean absolute error	0.161	
Root mean squared error	0.2738	
Relative absolute error	89.4509 %	
Root relative squared error	91.2645 %	
Total Number of Instances	60000	

=== Confusion Matrix ===

a=0	b=1	c=2	d=3	e=4	f=5	g=6	h=7	i=8	j=9	<-- classified as
5007	28	93	239	16	3	568	0	46	0	**a=0**
39	5814	16	111	6	0	10	0	4	0	**b=1**
145	19	4602	64	661	2	482	0	25	0	**c=2**
287	94	67	5191	192	0	158	0	11	0	**d=3**
19	12	605	209	4693	0	446	0	16	0	**e=4**
4	2	1	3	0	5665	1	221	17	86	**f=5**
971	19	668	182	512	0	3558	1	88	1	**g=6**
0	0	0	0	0	210	0	5595	3	192	**h=7**
67	10	67	55	38	40	121	16	5584	2	**i=8**
0	1	1	2	0	105	0	215	3	5673	**j=9**

2.2.2. Random Forest sin extraccion

Luego, se atacó el problema nuevamente sin extracción, pero utilizando un clasificador Random Forest con los parámetros por defecto. Los resultados obtenidos en este caso se muestran a continuación,

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	52718	**87.8633 %**
Incorrectly Classified Instances	7282	12.1367 %
Kappa statistic	0.8651	

Mean absolute error	0.0495
Root mean squared error	0.1387
Relative absolute error	27.4792 %
Root relative squared error	46.2387 %
Total Number of Instances	60000

=== Confusion Matrix ===

a=0	b=1	c=2	d=3	e=4	f=5	g=6	h=7	i=8	j=9	<-- classified as
5226	5	81	196	29	3	405	0	55	0	**a=0**
10	5778	26	145	8	0	28	0	5	0	**b=1**
43	1	4901	61	687	1	271	0	35	0	**c=2**
120	13	42	5515	178	0	117	0	14	1	**d=3**
11	8	466	242	4972	1	277	0	23	0	**e=4**
0	1	0	3	0	5724	0	178	16	78	**f=5**
1019	12	725	153	564	1	3420	0	106	0	**g=6**
0	0	0	0	0	87	0	5637	10	266	**h=7**
8	4	27	21	27	18	43	11	5832	9	**i=8**
0	0	0	1	0	66	2	212	6	5713	**j=9**

2.2.3. PCA + Random Forest

Como última aproximación con la herramienta Weka, se realizó una extracción utilizando PCA como técnica para reducir la dimensión del problema. Al igual que en el caso anterior, la extracción y la clasificación se hicieron con los parámetros por defecto.

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	51669	**86.115** %
Incorrectly Classified Instances	8331	13.885 %
Kappa statistic	0.8457	
Mean absolute error	0.081	
Root mean squared error	0.1688	
Relative absolute error	45.0273 %	
Root relative squared error	56.2677 %	
Total Number of Instances	60000	

=== Confusion Matrix ===

a=0	b=1	c=2	d=3	e=4	f=5	g=6	h=7	i=8	j=9	<-- classified as
5192	2	68	229	23	7	353	0	123	3	**a=0**
26	5742	16	164	11	0	27	0	12	2	**b=1**
86	1	4742	62	668	10	317	0	113	1	**c=2**
192	7	48	5408	185	6	121	0	29	4	**d=3**
16	4	459	242	4899	3	314	0	63	0	**e=4**
3	0	1	5	0	5490	0	245	27	229	**f=5**
1076	3	682	162	581	14	3272	0	209	1	**g=6**
1	0	0	0	0	182	0	5485	10	322	**h=7**
15	0	17	52	33	62	49	23	5737	12	**i=8**
0	0	1	0	0	82	0	210	5	5702	**j=9**

2.3. Principales problemas

A partir de las matrices de confusión obtenidas anteriormente podemos identificar las clases más problemáticas y será sobre estas que se deberá trabajar con el fin de lograr mejorar los accuracy obtenidos al momento.
Se puede ver que las clases que son mayormente confundidas son las 0, 2, 4 y 6. Esto se puede explicar por el parecido que tienen las imágenes pertenecientes a dichas clases. En la figura 2.2 se pueden ver los promedios de dichas clases observándose que efectivamente resultan difíciles de distinguir.

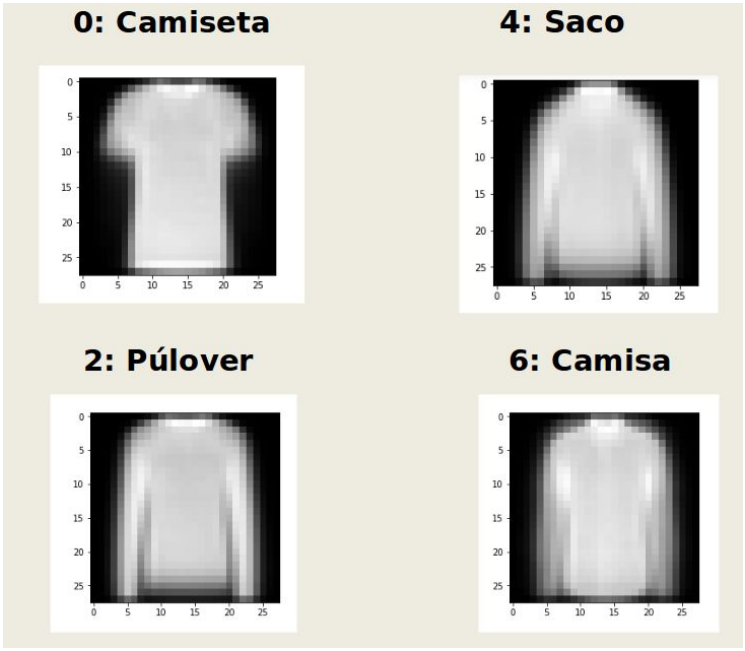

Figura 2.2

Por último, vale la pena mencionar que a partir del análisis de las diferentes matrices de confusión obtenidas, la mayor cantidad de errores se da en muestras cuya etiqueta real es clase 6 clasificadas como clase 0. Mejorar la clasificación entre estas dos clases, será tenido en cuenta para lograr obtener un mejor accuracy general.

3. Abordaje al problema

A continuación se plantean los diferentes caminos y métodos disponibles con los que será posible abordar el problema. Se considerarán los principales algoritmos y métodos existentes en la resolución de problemas de reconocimiento de patrones y se elaborará un plan estratégico que permitirá avanzar cuidadosamente, identificando y trabajando con los mejores resultados obtenidos en cada etapa hasta llegar al modelo final.

3.1. Planificación

El estudio del problema requerirá de emplear y probar los métodos más utilizados en este tipo de problemas, en particular se establecerán las diferentes etapas de trabajo a través de las cuales se definirán y aplicarán las técnicas de extracción, selección, y clasificación que mejor funcionen para resolver este problema en particular.
La planificación de trabajo está ordenada cronológicamente y se compone por las siguientes etapas:

a) Definir él o los clasificadores principales con los que se evaluarán resultados finales.
En esta primera instancia se establecen los clasificadores con el que se desarrollarán las pruebas y estudiará el problema. Para ello se observarán los resultados obtenidos con algunos clasificadores que se elegirán en base a Benchmarks encontrados y conocimientos teóricos adquiridos en la materia. Se realizarán ensayos con los clasificadores candidatos utilizando una cantidad reducida de muestras y se definirán 3 clasificadores a utilizar en las etapas siguientes.

b) Selección y/o extracción de características.
En esta etapa se implementarán las estrategias de búsqueda del sub-conjunto de características que mejor representen al espacio de muestras. En particular se buscará bajar la dimensionalidad del problema descartando las características que no contribuyan a la separabilidad. Se probarán un total de tres técnicas para la selección y extracción de características, selección y extracción según ganancia de información, según el análisis de componentes principales (PCA y KPCA), y por último se implementará el método de selección y extracción implementado en el artículo *"A Novel Feature Selection and Extraction Technique for Classification" de Goel, Vohra y Baksh*i.
Para evaluar cada método se utilizará un espacio de muestras reducido y se clasificará utilizando los 3 clasificadores seleccionados en la etapa anterior (con parámetros seteados según Benchmarks). Para realizar estas pruebas se ejecutarán algoritmos tipo "grid-search" con cada clasificador para encontrar el punto de mejor funcionamiento de los métodos de extracción y selección, además se identificará el clasificador que mejor funcione. Los algoritmos de extracción y selección que mejor aproximen al espacio de muestras serán probados con un número de muestras mayor evaluando su desempeño junto al mejor clasificador. Se ensayará también no solo el rendimiento de estos algoritmos por sí solos, sino que también se probará cómo responden en conjunto.

c) Encontrar los parámetros de funcionamiento del clasificador principal.
Una vez identificado el clasificador principal que formará parte del modelo de clasificación final y el método de extracción y selección a utilizar, se procederá a realizar una corrida tipo "grid-search" con un número reducido de muestras para obtener el valor de los parámetros que mejor funcionan para el clasificador seleccionado.

d) Observación y evaluación de resultados obtenidos
Se analizarán los resultados obtenidos hasta el momento, al aplicar los algoritmos de selección y extracción de características y clasificar con los parámetros encontrados. A partir de aquí se trabaja con el modelo y la metodología que haya demostrado obtener los mejores resultados para clasificar la base de datos fashion-MNIST.

e) Propuesta de combinación de clasificadores por selección
Además de verificar los resultados con un sólo clasificador para todas las muestras, se propone también verificar el rendimiento de un modelo compuesto por la combinación de dos clasificadores, uno de primera etapa para clasificar y separar todas las muestras clasificando como una misma clase a las idénticas o más problemáticas. Luego de procesar los datos con

esta primera etapa, se clasificaran con un segundo clasificador dedicado especialmente a separar las clases críticas para el problema.

f) Ensayo con muestra de test final
Se ensayará el modelo encontrado con un set de datos para prueba de test final, a modo de evaluar y verificar el funcionamiento del modelo y clasificadores elegidos.

3.2. Técnicas para reducción de dimensionalidad

Se realizarán diferentes ensayos utilizando algunas de las técnicas de selección y extracción aprendidas en el curso, particularmente se considerarán:

- **PCA:** análisis de componentes principales (reducción de dimensionalidad a 2 o más componentes)
- **Ganancia de información:** selección de las características que aportan mayor información al definir la etiqueta de la muestra (reducción de dimensionalidad a 2 o más componentes)
- **Técnica aplicada en Paper de referencia:** se estiman las características que sean más relevantes para cada clase por separado (basándose en las relaciones entre pares de clases) y devuelve un escalar que representa la distancia (según Keivack-Leiver) de una muestra cualquiera con respecto al promedio de todas las muestras dentro de una clase en particular (se detalla más adelante en capítulo 4.5).

Se comprobará el funcionamiento de estas tres técnicas sobre la base de datos fashion-MNIST utilizando los clasificadores Random-Forest, SVM y Multilayer Perceptron.

3.3. Prueba y evaluación de algoritmos de clasificación

Considerando los resultados encontrados de los benchmarks disponibles en la web, se identificaron los clasificadores que mejor funcionan para clasificar la base de datos fashion-MNIST. Se trabajará con los 3 siguientes clasificadores:

- **Algoritmo SVM:** algoritmo de aprendizaje supervisado, proyecta las muestras según el hiperplano separador de máximo margen.
    - Ventajas: contempla los casos no linealmente separables, buen funcionamiento en espacios de muchas características.
    - Desventajas: precisa conjuntos de entrenamiento suficientemente grandes, alta complejidad temporal.

- **Clasificador Random Forest:** basado en el promedio de resultados de una combinación de de árboles.

- Ventajas: distinción de datos no linealmente separables, eficiente para base de datos grandes, generalmente arroja buenos resultados.
- Desventajas: puede sobre ajustarse al ser entrenado si la muestra es ruidosa.

- **Multilayer Perceptron (MLP):** red neuronal multicapa con funciones de activación lineales y no lineales en sus neuronas con método de convergencia según descenso por gradiente.
  - Ventajas: distinción de datos no linealmente separables.
  - Desventajas: puede sobre ajustar los datos en el entrenamiento, ineficiente si se cuenta con pocas muestras de entrenamiento.

Estos son los clasificadores que se utilizarán para las pruebas de selección y extracción y en base a los resultados obtenidos se definirá el clasificador a utilizar en el modelo final.

4. Selección y extracción de características con fashion-MNIST

Teniendo en cuenta la técnicas de selección y extracción vistas durante el curso, se aplicaron sobre la base algunas de ellas con el propósito de reducir la dimensionalidad del problema y obtener mejores resultados. Las técnicas seleccionadas para aplicar sobre la base fueron PCA, Kernel PCA y selección individual de características por ganancia de información, por test de hipótesis chi cuadrado y por F-test.
Para evaluar qué técnica de extracción nos permite mejorar la performance de nuestro clasificador tanto desde el punto de vista del accuracy como del costo computacional, se utilizan tres clasificadores con los parámetros obtenidos de la página de Zalando Research. Se realizó la extracción de features, para luego entrar en los tres clasificadores mencionados, SVM con kernel poly y C=10, Random Forest y un Multi-layer Perceptron con función de activación relu y 100 unidades en capa oculta. Al mismo tiempo, se realizó la clasificación con los 784 features para poder evaluar el efecto de la extracción realizada. La evaluación se realizó con el 10% de las muestras, y realizando validación cruzada de 5 particiones. A su vez se clasificó utilizando las 784 features a modo de poder comparar los accuracy obtenidos.

Para el caso de los 784 features, antes de pasar las muestras por la etapa de clasificación, estas serán normalizadas de manera que todas las características queden entre los valores [-1,1]. Las operaciones que se aplicarán a cada característica son las siguientes:

$$x_{ik} = 2 * (\frac{x_{ik}}{max\{x_{ik}\}} - 0.5) \qquad \forall\ i = 1, ..., c\ \ y\ \ k = 1, ..., p\ ,$$

Siendo c el número total de clases y p el número total de características.

De modo de ejemplo se grafican a continuación cómo quedan distribuido los pixeles de una de las muestras al ser normalizados.

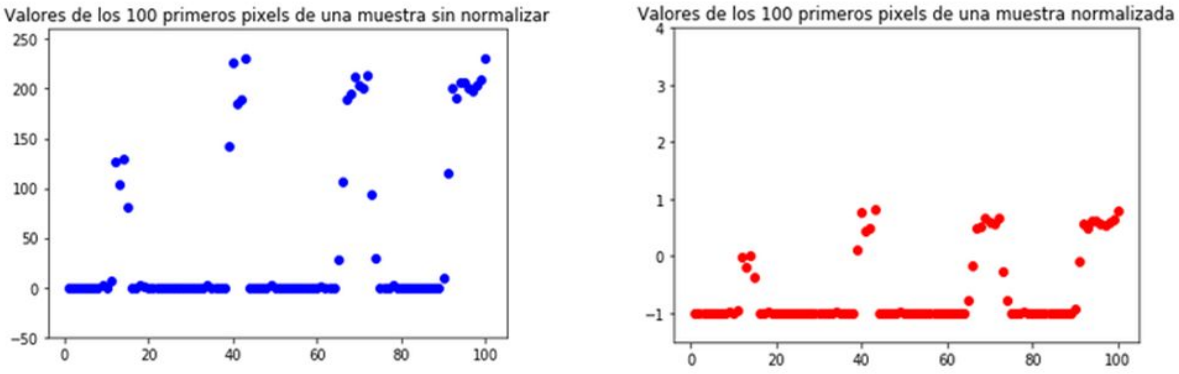

Figura 4.1

4.1. PCA

En primera instancia se decidió tratar las features del problema con PCA (Principal Components Analysis). Esta técnica consiste en encontrar las direcciones sobre las cuales proyectar los datos de forma que se maximice la varianza de los puntos proyectados. Esta es una técnica lineal no supervisada basada en vectores propios. La idea de PCA es usar un nuevo sistema de coordenadas dependiente de los puntos posicionando el primer eje en la dirección de mayor varianza, y así sucesivamente.

	PCA			784 features
Clasificador	100	200	300	
SVM	0.8501	0.8513	0.8533	0.8611
Random Forest	0.7846	0.7723	0.7743	0.8516
Multi-layer Perceptron	0.8345	0.8318	0.8266	0.798

Como se puede ver por los resultados obtenidos, para el caso de SVM, realizar PCA permite bajar la dimensionalidad del problema sin perder accuracy. El caso del Multi-layer Perceptron es aún más drástico, ya que el accuracy obtenido mejora considerablemente luego de aplicar PCA. En el caso de Random Forest no se logran buenos resultados perdiéndose puntos de score a costo de reducir la dimensión del problema. Se puede ver, a partir del código ejecutado, que las primeras 80 features resultantes de aplicar PCA, representan poco más del 90% de varianza total. Por este motivo, parece razonable quedarse con las primeras 200 features conteniendo más del 95% de varianza.

4.2. Kernel PCA

En segunda instancia se aplicó Kernel PCA sobre las características del problema. Esta técnica consiste en buscar una dimensión mayor donde los datos se logren separar de mejor manera, para luego aplicar PCA en ellos. Al igual que para PCA, se utilizaron el 10% de las muestras.

	Kernel PCA			784 features
	100	200	300	
SVM	0.8507	0.8525	0.8523	0.8611
Random Forest	0.7905	0.7847	0.7667	0.8516
Multi-layer Perceptron	0.8362	0.828	0.8285	0.798

4.3. Selección individual de características

Por último se realizó selección de features mediante selección individual de características. Primero se realizó la evaluación por test de hipótesis chi cuadrado, de manera de seleccionar las features más significativas para el problema. En este caso, los resultados obtenidos para los tres clasificadores seleccionados fueron los siguientes,

	SelectKBest (Chi cuadrado)			784 features
	100	200	300	
SVM	0.7335	0.7863	0.8033	0.8611
Random Forest	0.7938	0.8173	0.8308	0.8516
Multi-layer Perceptron	0.6998	0.738	0.7447	0.798

En segundo lugar se utilizó el criterio de ganancia de información los accuracy obtenidos fueron los siguientes.

	SelectKBest (Ganancia de información)			784 features
	100	200	300	
SVM	0.7052	0.7747	0.8008	0.8611
Random Forest	0.7768	0.8197	0.8278	0.8516

Multi-layer Perceptron	0.6455	0.7257	0.7427	0.798

Por último, se seleccionaron las K mejores features por f_classif (valor por defecto del SelectKBest) el cual utiliza F-test para ordenar las features. Los resultados obtenidos se muestra en el siguiente cuadro.

	SelectKBest ()			784 features
	100	200	300	
SVM	0.6873	0.7778	0.805	0.8611
Random Forest	0.7768	0.8197	0.8278	0.8516
Multi-layer Perceptron	0.6455	0.7257	0.7427	0.798

4.4. Combinación de métodos de extracción

Finalmente, dentro de las técnicas estudiadas en el curso, se probó realizar una combinación de extracciones para estudiar si se logra una mejoría en el accuracy obtenido. A partir de las matrices de confusión obtenidas en test del procedimiento de validación cruzada de 5 folds con el 10% de las muestras mostradas en las figuras 4.2 y 4.3, se puede ver que para PCA con 200 features y KBest con 100 features los errores de clasificar con etiqueta 0 muestras cuya etiqueta real es 6 es menor en el caso de KBest. Sin embargo el accuracy general es mayor para PCA. Por este motivo se decidió probar una extracción que combine al menos las primeras 150 features obtenidas con PCA (las cuales acumulan el 95% de la varianza) y agregarle al menos las primeras 25 de KBest seleccionadas con la función de score f_classif.

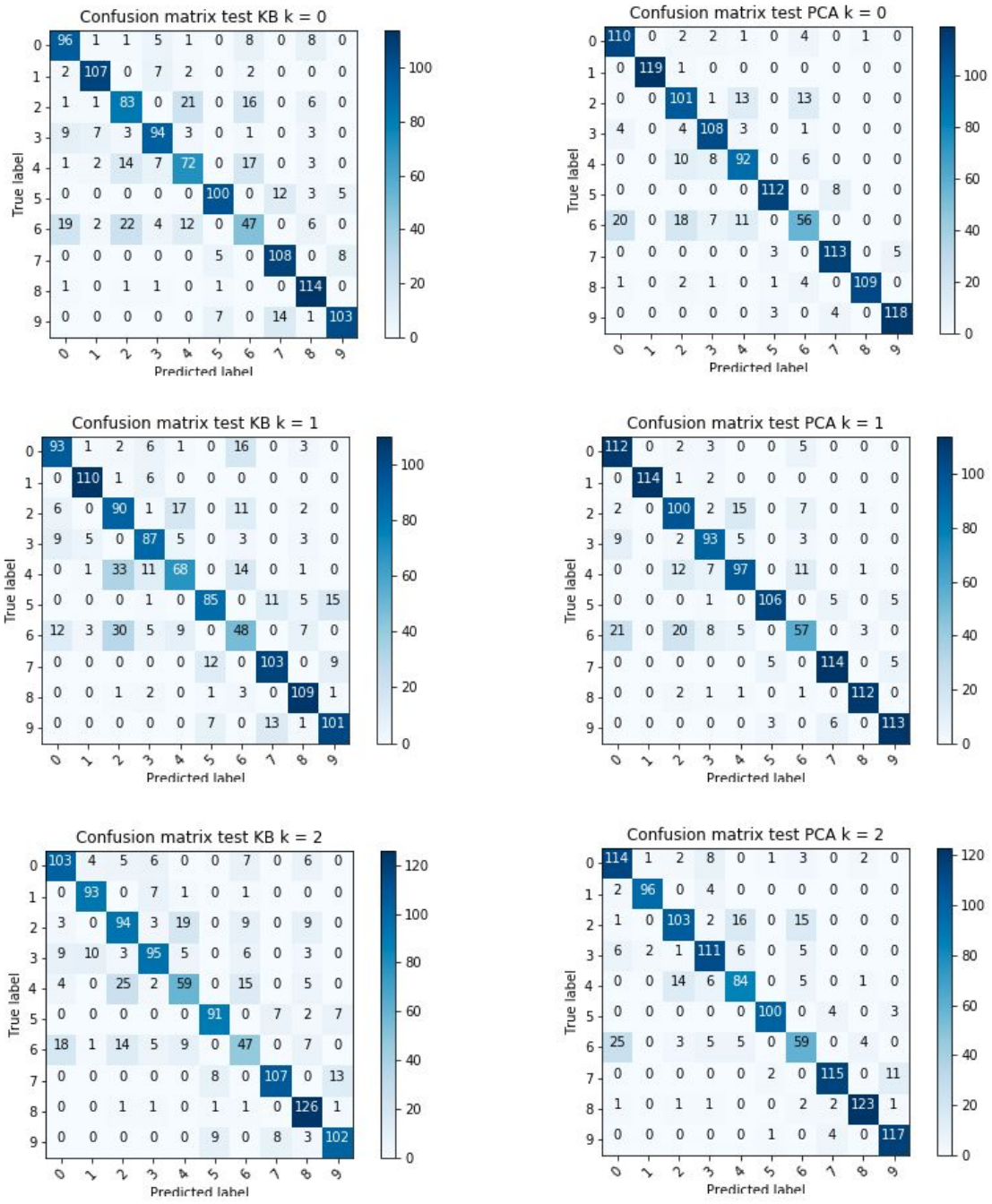
Confusion matrix test KB k = 0
Confusion matrix test PCA k = 0
Confusion matrix test KB k = 1
Confusion matrix test PCA k = 1
Confusion matrix test KB k = 2
Confusion matrix test PCA k = 2
True label
Predicted label

Figura 4.2

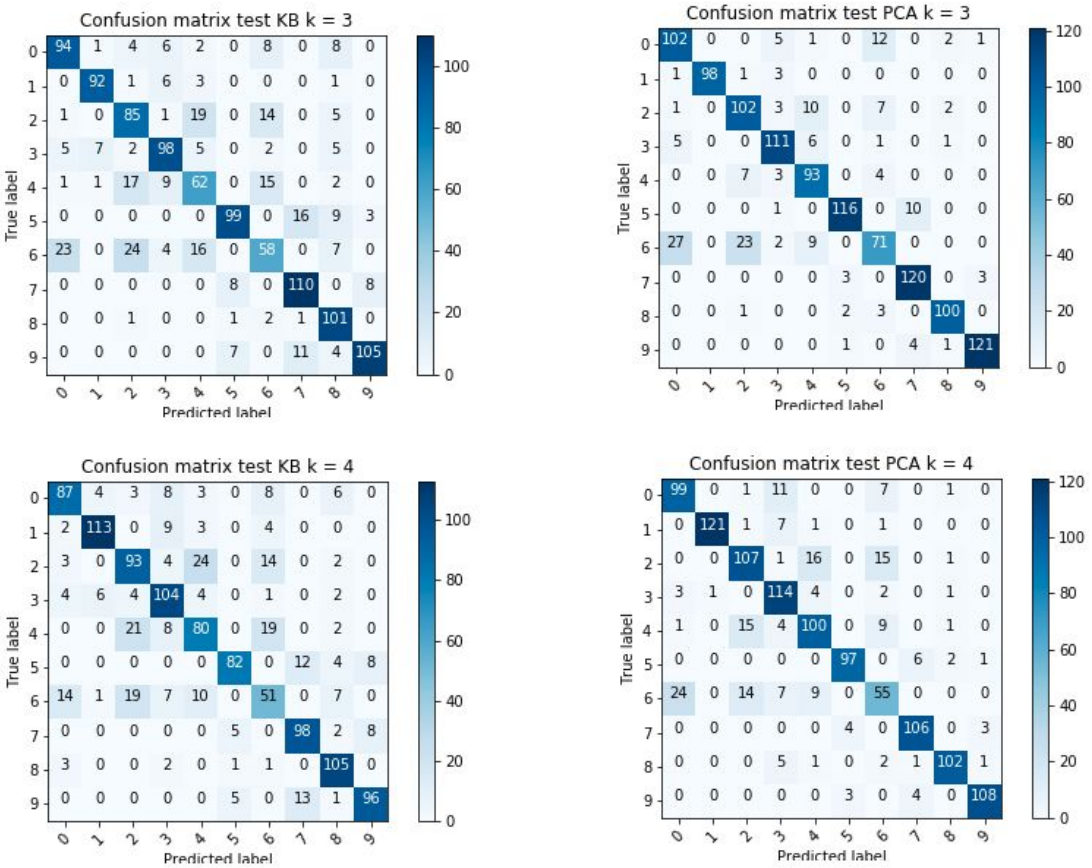

Figura 4.3

Para evaluar la union de métodos de extracción se clasificó con un SVM con kernel poly y C=10 variando la cantidad de features seleccionadas de cada extracción, utilizando 10% de las muestras y realizando validación cruzada de 5 folds. A continuación se muestran los resultados obtenidos variando la cantidad de features de cada extracción.

Features PCA	Features KBest	Accuracy
150	50	0.8555
160	40	0.8547
170	30	0.8562
175	**25**	**0.8565**
180	20	0.8542
180	30	0.8555

4.5. Paper estudiado

Finalmente se estudio el paper “A Novel Feature Selection and Extraction Technique for Classification” de Kratarth Goel, Raunaq Vohra y Ainesh Bakshi. Este paper plantea una técnica para seleccionar y extraer features dependientes de las clases (Class Dependent Features - CDFs). Lo que se busca en el paper es encontrar las features innatas a cada clase para extraerlas. Se focaliza en mejorar el accuracy de clasificación y al mismo tiempo controlar el costo computacional bajando la dimensión del problema.

4.5.1. Selección

La propuesta del paper para encontrar los features importantes de cada clase, es, primero obtener una medida T para cada clase. Para el caso de la base fahion-MNIST lo que se hace es promediar la intensidad de cada uno de los pixeles para cada clase. Por lo tanto cada medida T de las 10 clases tiene la misma dimensión que cada una de las imágenes de la base. De esta manera, para cada clase P_C se tiene que la función f, que representa la sumatoria en todas las instancias de la clase, viene dada por

$f(P_C) = \{a_{c1},\ a_{c2},\ \dots,\ a_{c784}\}$ donde, $a_{ci} = \sum_{k=1}^{M} p_k(i)$ siendo M la cardinalidad de la clase C, y $p_k(i)$ la intensidad del píxel i en la imagen k perteneciente a la clase C.

Luego para obtener el promedio T de cada clase se realiza el promedio sobre cada a_{ci},

$$T(P_C) = \{q_{c1},\ q_{c2},\ \dots,\ q_{c784}\} \text{ con } q_{ci} = a_{ci}/M \quad \forall i \in [1, 784]$$

La medida T de cada una de las clases es mostrada en la figura 4.4 debajo.

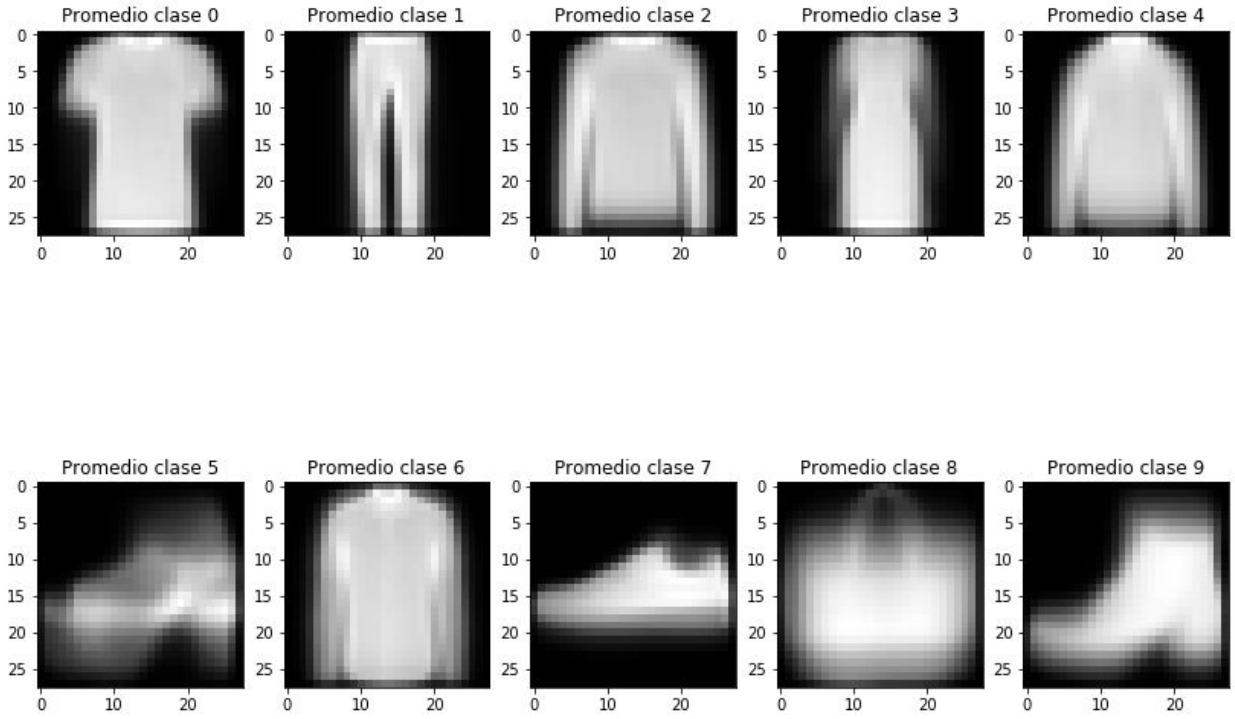

Figura 4.4

Una vez que se tiene una medida T de cada clase de la base, se establece una relación R_{xy} entre el par de clases P_x y P_y, que indique el grado de similitud o disparidad entre las mismas.

$$R_{xy} = \{q_{xi}/q_{yi} \mid \forall\ q_{xi} \in T(P_x)\ y\ \forall\ q_{yi} \in T(P_y)\}$$

Se toma la media de todos los valores $q_{xi}/q_{yi} \in R_{xy}$

$$\mu_{xy} = \frac{\sum_{i=1}^{784} q_{xi}/q_{yi}}{784}$$

Se generan dos thresholds $\tau\ y\ \tau'$ dados por

$$\tau = b\mu_{xy} \text{ y } \tau' = b'\mu_{xy} \text{ con } b,\ b' \in \mathbb{R}$$

Para la selección, el criterio propuesto es el siguiente, valores de $q_{ci} \in T(P_C)$ mayores que τ o τ' serán elegidos como features para la clase c. Es decir que sólo aquellos pixels pertenecientes a $T(P_C)$ cuyo valor de intensidad sea mayor que τ o τ' serán seleccionados como Class Dependent Features (CDFs). De esta forma, los valores b y b' pueden ser pensados como parámetros que controlan la dimensionalidad del problema. Así, se obtiene la muestra transformada

$$p'_k(i) = p_k(i)\ si\ q_{ci} > \tau\ o\ q_{ci} > \tau';\ de\ lo\ contrario\ nulo$$

La figura 4.5 muestra en oscuro los pixels elegidos en cada caso. 4.5.a los seleccionados para las clases 0 y 1 respectivamente; y 4.5.b los seleccionados para las clases 0 y 6 respectivamente.

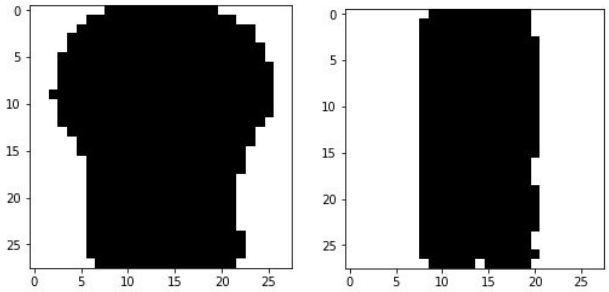

Figura 4.5.a

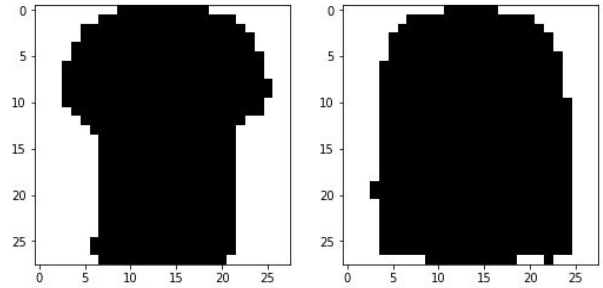

Figura 4.5.b

4.5.2. Extracción

Para realizar la extracción de los CDFs para el par de clases x e y, se utiliza el concepto de divergencia de Kullback-Leibler (KL). La divergencia Kullback-Leibler $D_{KL}(P||Q)$,es una medida no simétrica que mide la pérdida de información cuando se utiliza Q para aproximar a P. La propuesta del paper es usar el siguiente vector de features F con el propósito de clasificar las clases x e y,

$$F_{xy}(k) = D_{KL}(p_k^{'}||T(P_x))$$

4.5.3. Pruebas sobre fashion MNIST

Para evaluar la utilidad de la extracción propuesta en el paper se tomaron las muestras de clase 0 y 6, es decir, las muestras más complejas y las más fácilmente confundibles. Se les realizó la extracción propuesta por el paper, la extracción propuesta anteriormente (175 PCA + 25 KBest) y la clasificación con los 784 features.

El resultado de cada una de las extracciones se utilizó para entrenar y testear un clasificador SVM con kernel poly y C=10 con validación cruzada de 10 folds. Los resultados obtenidos fueron los siguientes.

SVM kernel poly C=10	
Selección y extracción	**Accuracy**
Extraccion paper	0.5
784 features (Sin extraccion)	0.8752
175 PCA + 25 KBest	0.8579

También se probó variando los parámetros del clasificador a kernel rbf, C=7, y gamma=0.5. Con esta configuración se obtuvieron los resultados tabulados debajo.

SVM kernel rbf C=7, gamma=0.5	
Selección y extracción	**Accuracy**
Extraccion paper	0.5
784 features (Sin extraccion)	0.5069
175 PCA + 25 KBest	0.8829

Por último, se probó realizar la extracción propuesta por el paper para entrar en un clasificador Random Forest. El resultado obtenido es el siguiente,

Random Forest	
Selección y extracción	**Accuracy**
Extraccion paper	0.7478
784 features (Sin extraccion)	0.8447
175 PCA + 25 KBest	0.8242

4.6. Conclusiones de la extracción

Luego de analizar los resultados obtenidos aplicando la extracción propuesta en el paper, y dado que la misma no logra mejorar el accuracy para las clases más complejas, decidimos

continuar con la extracción con la cual se obtuvieron mejores resultados, es decir 175 PCA + 25 KBest.

5. Selección de parámetros del clasificador

Al identificar la técnica de extracción y el clasificador que mejor se desenvuelve al afrontar el problema de reconocimiento propuesto por la base de datos fashion-MNIST, resta seleccionar los parámetros de funcionamiento del clasificador C-SVM, que según se demostró es el que mejor funcionó para clasificar las muestras (luego de aplicar la extracción). Para ello se realizarán pruebas de rendimiento, en una primera etapa con un espacio reducido de muestras, utilizando como criterios de evaluación scores y complejidad temporal.
Las pruebas a continuación se realizarán aplicando la técnica de selección y extracción encontrada en la etapa anterior, la cual está compuesta por la unión de las técnicas PCA y SelectKBest (selección de las k mejores características basado en tests de varianza univariada). Aplicar estas técnicas permitirá bajar la dimensionalidad del espacio de características de 784 a 200 manteniendo cerca del 95% la información, tal como se demostró en el capítulo anterior.
La estrategia a emplear será ejecutar pruebas tipo ”grid-search” variando los parámetros de un clasificador C-SVM (SVC según librería sklearn) con tests de validación cruzada tomando el 10% del espacio de muestras original.
Estas pruebas definirán los parámetros de ajuste con los que se inicializará el clasificador principal a utilizar.
Todos los test que siguen a continuación serán realizados utilizando módulos de la librería “scikit-learn” disponible para Python.

Antes de analizar los resultados obtenidos, se detallan a continuación algunas generalidades de la técnica de clasificación a utilizar y descripción y funcionalidad de los parámetros que se variarán en el test de “grid-search”.

5.1. C-SVM - Generalidades

Los valores de los parámetros encontrados anteriormente ajustan el algoritmo empleado por el clasificador C-SVM a los parámetros del problema.

El modelo de clasificación SVM busca el hiperplano separador (en el espacio transformado) que maximiza el margen de decisión, encontrando los parámetros (w, b) correspondientes que definen la superficie de decisión $g(x) = w^T \Phi(x) + b,\ S = \{x : g(x) = 0\}$, donde cada nueva muestra se clasificará según signo($g(x)$). Las muestras que caen sobre los hiperplanos de máximo margen conforman los vectores de soporte (SV) y por lo tanto cumplen $y_i\, g(x_i) = 1.$

Para el caso no linealmente separable (C-SVM) se define una penalización para los errores aceptando algunos errores de clasificación en entrenamiento, de forma que cada muestra se penaliza con un valor " ξ " tal que;

$$\xi = \left\{ 0 \; si \; x_i \; está \; bien \; clasificado, \quad \left| y_i - g(x_i) \right| \; si \; no \right\} \qquad \text{con} \quad g(x_i) = w^T \Phi(x_i) + b$$

Considerando el valor de penalización la condición de clasificación es

$$y_i \left(w^T \Phi(x_i) + b \right) \geq 1 - \xi_i , \;\; \xi_i \geq 0, \; i = 1, ..., N$$

Entonces el algoritmo C-SVM da lugar al siguiente problema de optimización:

$$\begin{cases} \min_{w,b} \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \zeta_i \right\} , C > 0 \\ \text{sujeto a} \begin{cases} y_i (w^T \phi(x_i) + b) \geq 1 - \zeta_i , \\ \zeta_i \geq 0, \; i = 1, \ldots, N \end{cases} \end{cases}$$

Donde $J(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i$ es la función de costo original.

Los parámetros que se variarán para encontrar el punto de mejor funcionamiento son "Kernel", "C", y "gamma".
A continuación se detalla en líneas generales la funcionalidad de cada uno de ellos en el algoritmo C-SVM.

- Parámetro " Kernel": identifica la función que es utilizada para proyectar las muestras a un espacio de características de mayor dimensionalidad de modo que las mismas puedan ser separables por infinitos hiperplanos.
  Las funciones que se considerarán son:
  - "poly": función polinómica, ej.: $k(x_i , x_j) = (x_i \, x_j)^n$ con $x_j \in SV$
  - "rbf": función base radial gaussiana, ej.: $k(x_i , x_j) = exp(\frac{-(x_i - x_j)^2}{2 c})$ con , c cte
  - "sigmoid": función sigmoide, ej.: $k(x_i , x_j) = \frac{1}{1+exp(-(x_i - x_j))}$ con $x_j \in SV$

- Parámetro " C": modificar el valor de C implica mover los márgenes de decisión penalizando más o menos el error de clasificación, de esta manera se genera un compromiso entre mayor margen (mejor generalización) y mayor cantidad de errores de entrenamiento.

- Parámetro " gamma": El factor gamma define hasta dónde llega la influencia de los vectores de soporte en la clasificación, se puede ver como el inverso del radio de influencia de las muestras seleccionadas (por el modelo) como vectores de soporte.
  En el caso en que gamma sea muy grande, el radio del área de influencia de los vectores de soporte sólo incluirá al mismo vector de soporte lo que provocará un "overfitting" en el modelo.

5.2. Grid-search con SVM

La prueba a realizar en esta etapa se hará con un grupo reducido de muestras de manera de acelerar el proceso. Para reducir el espacio de muestras y trabajar con el 10% de las mismas se utiliza la función "train_test_split()" de scikit-learn la cual permite particionar el espacio de datos. Luego se seleccionan y extraen las características más relevantes según las técnicas descritas en la etapa anterior, luego de la extracción se normalizan las muestras (ver cap.4), y posteriormente se clasifican utilizando como estrategia validación cruzada dividiendo el espacio de muestras en 5 grupos.
Para emplear la estrategia de validación cruzada se utilizaron algunas funciones especiales como "cross_val_score()" y "GridSearchCV()" de la librería "sklearn.model_selection" (scikit-learn).

Se hacen dos búsquedas del tipo "grid-search", la primera para encontrar los parámetros de funcionamiento óptimo del modelo constituido por un solo clasificador C-SVM dedicado a todas las clases, la segunda búsqueda para encontrar el punto de funcionamiento óptimo de un modelo propuesto por la combinación de dos clasificadores C-SVM (se detalla más adelante).

5.2.1. Ensayos con un sólo clasificador C-SVM

El resultado de la prueba realizada tomando el 10% de las muestras y variando los parámetros del clasificador C-SVM es el siguiente:

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma) Estrategia: validación cruzada con el 10% de muestras y kfolds=5		
kernel= 'rbf'	C= 5	gamma= 0.5
crossval-accuracy: **87,2 % - máximo valor obtenido en test grid-search (*)**		

*****La prueba "grid-search" se realizó para los parámetros kernel, C y gamma variando entre:

- Kernel= ['poly','rbf','sigmoid']
- C= [1,2,3,4,5,7,10,15,30,50,100,150,200,250,500,1000,1500]
- gamma= ['auto',0.0001, 0.005, 0.01, 0.1, 0.5, 0.8, 1.0]

A partir del resultado anterior se obtienen los parámetros del clasificador con los que se logra la mejor exactitud (kernel ='rbf', C =5 y gamma=0.5). Más adelante se realizará una segunda prueba, con mayor cantidad de muestras, para encontrar el punto óptimo de funcionamiento dentro del entorno del punto encontrado en esta oportunidad.

5.2.2. Ensayos con combinación de clasificadores C-SVM

Además del modelo básico formado por un sólo clasificador para todas las clases, se propone un modelo de clasificación compuesto por la combinación de dos clasificadores idénticos al anterior, con la diferencia de tener uno de ellos dedicado a separar las muestras de mayor complejidad de las de menor complejidad por separado. Es decir la primera etapa de clasificación es para separar las clases 0,2,4,y 6 de las restantes, y la segunda etapa estará dedicada a separar únicamente estas 4 clases más complejas entre sí.
La siguiente imagen (figura 5.1) detalla el esquema modular del clasificador en cuestión:

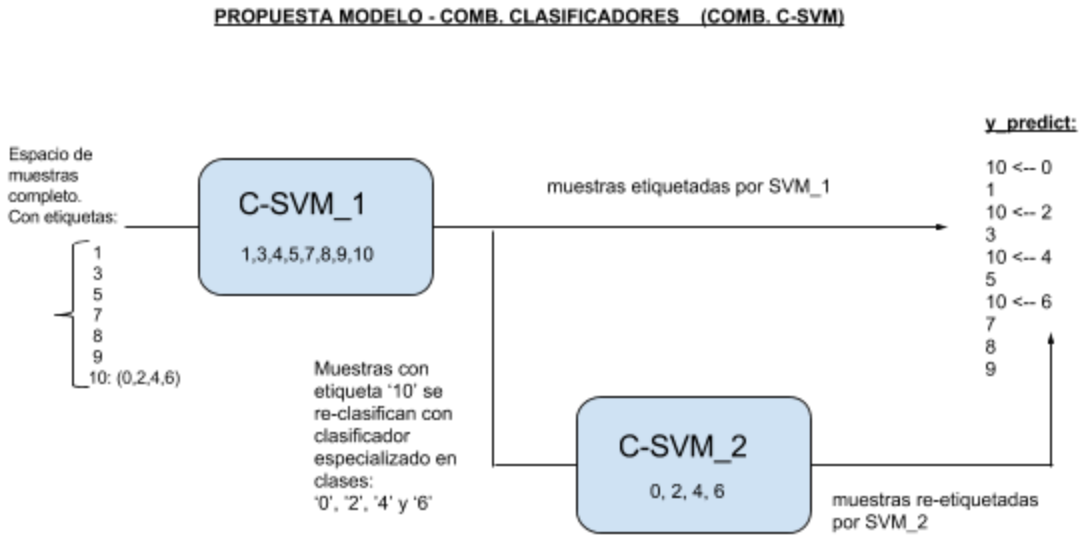

Figura 5.1

Al igual que en el caso anterior la prueba se realiza tomando el 10% de las muestras y variando los parámetros de ambos clasificadores C-SVM por igual, el punto de funcionamiento óptimo encontrado es el siguiente:

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: 2xC-SVM (kernel, C, gamma) Estrategia: validación cruzada con el 10% de muestras y kfolds=5		
kernel= 'rbf'	C= 1500	gamma= 1.0
crossval-accuracy: **83.1 % - máximo valor obtenido en test grid-search (*)**		

*****La prueba “grid-search” se realizó para los parámetros kernel, C y gamma variando entre:

- Kernel= [‘poly’,’rbf’,’sigmoid’]
- C= [1,2,3,4,5,7,10,15,30,50,100,150,200,250,500,1000,1500]
- gamma= [‘auto’, 0.005, 0.01, 0.1, 0.5, 0.8, 1.0]

A partir del resultado anterior se concluye que el clasificador constituido por la combinación de clasificadores C-SVM no mejora el rendimiento en comparación al modelo constituido por un único C-SVM. A continuación se verificará dicha afirmación al realizar la prueba de clasificación para el 100% de las muestras para ambos modelos.

5.3. Clasificación SVM con parámetros obtenidos y 100% de las muestras

Una vez definidos los algoritmos de selección y extracción de características, encontrados (búsqueda “grid-search”) los parámetros del clasificador C-SVM y del modelo propuesto por la combinación del mismo que mejor funcionan para el reconocimiento de las clases, se realizó una etapa de verificación final para determinar el modelo definitivo a utilizar con el set de test final.

5.3.1. Utilizando un sólo clasificador C-SVM

La prueba que se detalla a continuación muestra la precisión que presenta el clasificador C-SVM para clasificar las muestras con los parámetros encontrados y estrategia de validación cruzada dividiendo el espacio muestral en 10 grupos. Para esta prueba se consideró el 100% de las muestras, tomando 90% para entrenamiento y 10% para test.
El resultado obtenido fue siguientes:

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma) Estrategia: validación cruzada con el 100% de muestras y kfolds=10		
kernel= ‘rbf’	C= 5	gamma= 0.5
crossval-accuracy: **90,7 %**		

5.3.2. Utilizando combinación de clasificadores C-SVM

Para verificar el rendimiento del modelo propuesto (figura 5.1) por la combinación de clasificadores C-SVM se realizó una búsqueda de parámetros óptimos y luego un ensayo tomando el 100% de las muestras y estrategia de validación cruzada tomando grupos de a 10 para evaluar la exactitud en el punto óptimo encontrado.
El resultado obtenido fue el siguiente:

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: 2xC-SVM (kernel, C, gamma) Estrategia: validación cruzada con el 100% de muestras y kfolds=10		
kernel= 'rbf'	C= 1500	gamma= 1.0
crossval-accuracy: **90.37 %**		

De los resultados obtenidos en los ensayos anteriores se deduce finalmente que el modelo constituido por un único clasificador C-SVM configurado con los parámetros óptimos es el que presenta mayor exactitud al clasificar las muestras de la base de datos fashion-MNIST.

5.4. Ajuste preciso de parámetros del clasificador C-SVM

Una vez encontrado el punto óptimo de trabajo aproximado del clasificador C-SVM (kernel= 'rbf', C=5, gamma=0.5) utilizando el 10% de las muestras, se realiza un re-ajuste de los parámetros dentro de un entorno al punto encontrado con el objetivo de disminuir el sobreajuste que pueda existir en este punto. Para ello se realiza una segunda instancia de búsqueda "grid-search" dividiendo el set de muestras de entrenamiento en 70-30%, para entrenamiento y validación respectivamente.
Luego de realizada la prueba se identificaron dos puntos como alternativas posibles para definir el clasificador, uno de ellos presentó la exactitud en test más alta a cambio de un alto margen entre exactitud en entrenamiento y test (un poco sobre ajustado), el otro punto de interés presenta apenas menos exactitud en test pero disminuye el gap entre la exactitud en entrenamiento y test (mejora el sobreajuste). Los resultados para los puntos en cuestión fueron los siguientes:

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma) Estrategia: set de 42.000 - 18.000 muestras para entrenamiento y validación respectivamente		
kernel= 'rbf'	C= 1	gamma= 0.6
Train - accuracy: 92.51 %		
Test - accuracy: **89.10 %**		
kernel= 'rbf'	C= 5	gamma= 0.7
Train - accuracy: 97.65%		
Test - accuracy: **90.48 %**		

*La prueba “grid-search” se realizó para los parámetros kernel, C y gamma variando entre:

- Kernel= [’rbf’]
- C= [1, 3, 5, 7, 10, 15]
- gamma=['auto', 0.1, 0.3, 0.5, 0.6, 0.7, 1.0]

Luego de identificar estos dos puntos se realizó otra prueba de entrenamiento y validación con una proporción de 90-10%. Estos son los resultados obtenidos para los puntos de interés:

<table>
<tr><td colspan="3">extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma)
Estrategia: set de 54.000 - 6.000 muestras para entrenamiento y validación respectivamente</td></tr>
<tr><td>kernel= ‘rbf’</td><td>C= 1</td><td>gamma= 0.6</td></tr>
<tr><td colspan="3">Train - accuracy: 92.40%</td></tr>
<tr><td colspan="3">Test - accuracy: 90.10 %</td></tr>
<tr><td>kernel= ‘rbf’</td><td>C= 5</td><td>gamma= 0.7</td></tr>
<tr><td colspan="3">Train - accuracy: 97.50 %</td></tr>
<tr><td colspan="3">Test - accuracy: 91.15 %</td></tr>
</table>

De los resultados obtenidos se identificó el punto de trabajo “kernel= ‘rbf’, C= 5 y gamma= 0.7”, como el punto que resulta en el mejor accuracy de validación a cambio de un mayor sobreajuste del clasificador.

5.5. Ensayo comparativo con y sin extracción para clasificadores SVM y RandomForest

En esta última etapa, antes del ensayo final del sistema definitivo con las muestras de test propuestas, se realizará un estudio comparativo de rendimiento entre el clasificador C-SVM ajustado con los parámetros óptimos (encontrados en 5.4) y el clasificador Random Forest con parámetros según benchmarks. La misma prueba comparativa se realizará con y sin extracción de características, lo que permitirá también observar el efecto de la extracción una vez que se encontraron los parámetros óptimos de funcionamiento para el clasificador C-SVM.

Resultados de las pruebas sin extracción:

Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma) Estrategia: set de 54.000 - 6.000 muestras para entrenamiento y validación respectivamente		
kernel= ‘rbf’	C= 5	gamma= 0.7
Train - accuracy: no fue posible obtener un resultado con estos parámetros y espacio de muestras considerado, es posible se deba al alto costo computacional que supone esta prueba.		
Test - accuracy: - - -		

Normalización [-1,1] | clasificador: RandomForest(criterion, max_depth, n_estimators) Estrategia: set de 54.000 - 6.000 muestras para entrenamiento y validación respectivamente		
criterion= ‘entropy’	max_depth= 50	n_estimators= 100
Train - accuracy: 100 %		
Test - accuracy: **88.8 %**		

Resultados de las pruebas con extracción:

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma) Estrategia: set de 54.000 - 6.000 muestras para entrenamiento y validación respectivamente		
kernel= ‘rbf’	C= 5	gamma= 0.7
Train - accuracy: 97.4 %		
Test - accuracy: **91.1 %**		

extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: RandomForest Estrategia: set de 54.000 - 6.000 muestras para entrenamiento y validación respectivamente		
criterion= ‘entropy’	max_depth= 50	n_estimators= 100
Train - accuracy: 100 %		
Test - accuracy: **87.0 %**		

5.6. Prueba con datos de test proporcionados

Los datos utilizados en los ensayos anteriores, que sirvieron tanto para encontrar el método de selección y extracción como el clasificador SVM y sus parámetros óptimos de funcionamiento, fueron los datos de entrenamiento proporcionados al inicio del proyecto (compuesto por las 60.000 muestras).
Para realizar el ensayo final de evaluación del modelo encontrado se entrenará el mismo con el set de datos de entrenamiento constituido por las 60.000, y se testeará con los 10.000 datos de test proporcionados especialmente para realizar esta prueba final.

<table>
<tr><td colspan="3">extracción: PCA + SelectKBest | Normalización [-1,1] | clasificador: C-SVM (kernel, C, gamma)
Estrategia: set de 60.000 - 10.000 muestras para entrenamiento y test respectivamente</td></tr>
<tr><td>kernel= ‘rbf’</td><td>C= 5</td><td>gamma= 0.7</td></tr>
<tr><td colspan="3">Train - accuracy: 97.45 %</td></tr>
<tr><td colspan="3">Test - accuracy: 90.98 %</td></tr>
</table>

6. Conclusiones

La forma en que se avanzó en el estudio permitió obtener resultados por etapas y de acuerdo a estos decidir cómo seguir hacia la etapa siguiente, así hasta obtener un modelo final que optimice la exactitud en el reconocimiento de los patrones fashion-MNIST.
De esta manera se estudió el método de selección y/o extracción más conveniente a utilizar, comparando entre PCA, SelectKbest (mejores características), KernelPCA y paper encontrado, identificando finalmente la metodología PCA combinada con KBest (“k” mejores características a utilizar) como la técnica que mejor funciona.
Al realizar los ensayos descritos con la metodología utilizada en el paper se concluyó finalmente que no obtiene los rendimientos esperados en comparación a las demás herramientas de extracción, y dada su complejidad de utilización se decidió dejarla al margen en este estudio.
En efecto, la mejor configuración de extracción que mejor funcionó es; PCA con n_componentes=175 + SelectKBest con k=25.
Aplicar la selección y extracción de características con la configuración elegida permite bajar considerablemente la complejidad temporal de los ensayos, lo que permitió realizar las campañas de búsqueda de parámetros necesarias para encontrar el punto óptimo de configuración para el clasificador elegido.
Se plantearon dos modelos diferentes para clasificar las muestras, el primero compuesto por un único clasificador C-SVM y el segundo (modelo 2) por una combinación de clasificadores

C-SVM (uno para separar todas las muestras y otro para especializado en separar las 4 clases más complejas). Luego de las pruebas realizadas se determinó que el modelo 2 no mejora en cuanto a exactitud con respecto al clasificador SVM funcionando solo. Por tal motivo se resolvió descartar el modelo 2 en este estudio, lo cual no significa que se pueda mejorar el rendimiento de este modelo a futuro, si se considera combinar diferentes clasificadores, encontrando parámetros óptimos de funcionamiento para cada uno.

Del resultado de las pruebas tipo “grid-search” para encontrar los parámetros del clasificador C-SVM, se encontraron dos puntos óptimos en los que el mismo funciona mejor, estos puntos fueron:
Punto 1- “kernel= ‘rbf’, C= 5 y gamma= 0.7”
Punto 2- “kernel= ‘rbf’, C= 1 y gamma= 0.6”
Para estos dos puntos se obtuvieron valores similares de accuracy, presentando el punto 1 una diferencia del 1% por encima del punto 2, sin embargo este último (punto 2) acorta considerablemente el margen entre el accuracy de entrenamiento y validación (clasificador no sobre entrenado).
Para la prueba de test final con las 10.000 muestras proporcionadas se decidió darle prioridad al valor de score de validación obtenido en los resultados con los puntos 1 y 2, por ello se eligió el primero para configurar el clasificador SVM en esta última instancia.
La prueba de test final se ejecutó entrenando el clasificador C-SVM con los parámetros óptimos de funcionamiento encontrados, entrenando con las 60.000 muestras de entrenamiento y testeando finalmente con las 10.000 muestras de test.
El resultado fue el siguiente:
Train - accuracy: 97.45 %
Test - accuracy: **90.98 % ,**

Lo que indica que, si bien el sistema se encuentra en cierta medida sobre entrenado, el objetivo del estudio con la configuración encontrada se cumple ya que el score de accuracy obtenido en test está por encima del 90%.
Entendemos sin embargo, que este valor podría incrementarse al menos una unidad porcentual si se mejora el modelo del clasificador, por ejemplo haciendo combinación de clasificadores SVM, ó combinando y optimizando (parámetros óptimos) clasificadores en el modelo propuesto como alternativa (modelo 2).

7. Anexo

7.1. Detalle archivos y carpetas

Carpeta de códigos contiene:

- 4-1_4-2_4-3_extraccion.py
- 4-4_matrizExtraccion.py
- 4-4_featureUnion.py
- 4-5_clasifPaper_01.py
- 4-5_clasifPaper_06.py
- 4-5_clasifPaper_06_RF.py
- 5-2-1_ensayos_grid_search_con_un_solo_clasificador_C-SVM.ipynb
- 5-2-2_prueba_grid_search_modelo2.ipynb
- 5-4_ajuste_preciso_parámetros_clasificador_C-SVM_grid_search.ipynb
- 5-4_seleccion_y_pruebas_ptos1y2.ipynb
- 5-5_ensayo_con_sin_extracción_clasifi.SVM_y_ RandomForest.ipynb
- 5-6_prueba_modelo_final_con_datos_test.ipynb
- norm_datos.ipynb
- modelo2.py

*nomenclatura utilizada para numeración de archivos según capítulos correspondientes

7.2. Links con información de clases y librerías python utilizadas

Librería scikit-learn: http://scikit-learn.org/stable/
Librería SVM para python: http://scikit-learn.org/stable/modules/svm.html

7.3. Bibliografía y fuentes consultadas

Base de datos fashion-MNIST:
https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/

Documento paper consultado:
"A Novel Feature Selection and Extraction Technique for Classification" de Goel, Vohra y Bakshi.