Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Ciencia de datos: 4

Jorge Luis Ramos Zavaleta

1 de Julio de 2018

1. Ejercicio

Aunque hay varias extensiones de AdaBoost al caso multiclase, una de las mas usadas es la llamada SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function), ya que esta basada en la caracterización estadística de Friedman et al. Implementa esta version de AdaBoost y verifica su desempeño en un conjunto de datos con más de dos categorias.

Incluye una breve descripcion del método basandote en el artículo: Zhu J, Zou H, Rosset S, and Hastie T (2009). $Multi-ClassAdaBoost$. Statistics and Its Interface,2, 349360. Puedes usar tambien los datos que ahí se muestran para reproducir los resultados.

1.1. Solución

ADAboost es un algoritmo de clasificación binaria que aparece por primera vez a principios de este siglo, y se volvio famoso debido a que tiene una propiedad dada la función de error exponencial que forma parte de su estructura de no generar sobreajuste de los datos, aunque en este sentido hay una controversia ya que en algunos casos se considera que se puede lograr un sobreajuste con muchas iteraciones y en otros con muy pocas, además de la controversia de si el usar árboles con mayor profundidad no hace que se pierda la propiedad de ser Bayes consistente.

El algoritmo es simple de implementar y la idea detrás de el es muy intuitiva: dada una

familia de clasificadores simples, en este caso stumps, queremos encontrar en el espacio funcional de los stumps cual es el que mejor clasifica con un proceso de descenso haciendo uso de pesos, donde se le asigna el mayor peso a aquellas observaciones que fueron mal clasificadas por el stump anterior al que queremos encontrar y con eso minimizar el error de mala clasificación.

Ahora el proceso de elección de estos pesos y del orden del clasificador viene intrínseco en el proceso de minimizar la función de error exponencial que acompaña a este algoritmo. Uno de los problemas que se puede generar con el algoritmo ADAboost es que todos los términos que componen la función de error exponencial deben ser positivos por lo que una vez que el término $\alpha^{(m)} = 0$ que se logra cuando el término $error^{(m)} = 1/2$ entonces los pesos ya no se actualizaran y la clasifición va a mantenerse igual sin importar el número de iteraciones posteriores que se hagan por lo que si se inicializan de manera desafortunada los pesos entonces puede no obtenerse una clasificación correcta.

La idea detrás de ADAboost multiclase es la misma que en el caso de clases aunque ahora se considera una función exponencial multiclase que al minizarse por un proceso de descenso en 2 etapas como en ADAboost se logra encontrar la dirección en la cual se encuentra el mejor clasificador simple, y en este sentido dada la forma de la función de pérdida exponencial multiclase el comportamiento de ADAboost se replica y el algoritmo deja mejorar las clasificaciones una vez que el término $error^{(m)} = 1/k$ donde k es el número de clases a clasificar, por lo que termina formando una extensión natural de ADAboost para dos clases.

Para probar el algoritmo SAME se consideraron 3 datasets el primero es el conjunto de vinos que contiene 3 clases de vinos y 13 atributos como grados de alcohol e intensidad de color, el segundo conjunto es el conjuto iris que contiene 3 clases de plantas iris y cuatro atributos de cada una de ellas como la medidas del tallo y del petalo, y por último el conjunto de letras empleado en el paper de *Zhu et al* para ver la efectividad del algoritmo SAME, sin embargo en este caso no es posible reproducir tal cual los resultados del paper dado que no se indica la profundidad del árbol utilizada solo se indica que se hizo de validación cruzada sobre los árboles que si estaban balanceados para considerar dicha profundidad.

Para el caso del conjunto de vinos se obtuvieron precisiones de 92.23 % y de 77.27 % para los datos de entrenamiento y prueba respectivamente. Las tablas de asignación se encuentran en el cuadro 1.1. El conjunto de datos iris no esta repartido en datos de entrenamiento y prueba por lo que solo se considero solo el conjunto completo para clasificar y se obtuvo un 96 % de precisión en la clasificación y los resultados de la clasificación se pueden ver en el cuadro 4.1. Para ambos conjuntos de datos solo se consideraron 5 iteraciones y una profundidad máxima de 1.

Por último se considero el conjunto de datos *letters* que contiene 26 categorias y 16 atribu-

tos de cada imagen que contiene las letras, las tablas de confusión no se muestran porque son muy grandes pero se obtuvo un nivel de precisión de 97.74 % y de 91. % para los datos de entrenamiento y prueba respectivamente con una profundidad máxima de 7 y 100 iteraciones. En la figura 1.1 se muestra la variación en el nivel de precisión obtenido en los datos de prueba con profundidades máximas 3 y 7, donde se muestra que con mayor profundidad se logra generalmente mejor nivel de precisión.

Cuadro 1.1: Clasificación obtenida en el conjunto de datos de vinos

Cuadro 1.2: Clasificación datos de entrenamiento

	1	2	3
1	45	6	1
2	0	39	0
3	0	6	36

Cuadro 1.3: Clasificación datos de prueba

	1	2	3
1	9	2	0
2	4	14	0
3	0	4	11

Cuadro 1.4: Clasificación obtenida en el conjunto de datos iris

	1	2	3
1	50	0	0
2	0	48	4
3	0	2	46

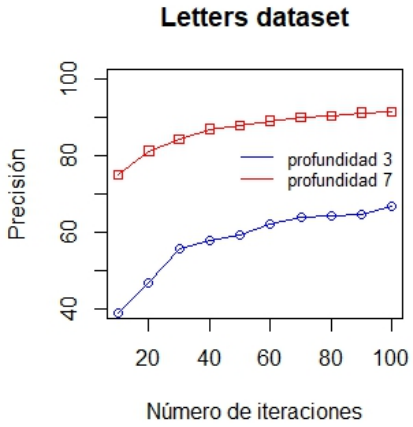

Figura 1.1: Nivel de precisión con distintas profundidades máximas en el conjunto de datos letters

En general ADAboost parece lograr un muy buen trabajo clasificando múltiples clases, aunque debe tener siempre en consideración que si la inicialización de los pesos es mala la clasificación no se va a lograr debido a que los pesos no se estarian actualizando sin importar el número de iteraciones.

2. Ejercicio

Usando los datos de los dígitos escritos a mano y digitalizados, complementa el ejercicio que hiciste en la tarea 1 aplicando métodos de clasicación basados en

- LDA
- QDA
- Redes neuronales
- Máquinas de soporte vectorial
- Árboles de clasificación
- AdaBoost

Utiliza k-fold como criterio para elegir el mejor modelo, así como para compararlos. ¿Qué método elegirías?

Especifica los parámetros que usaste en cada método de clasicación. Incluye graficos informativos sobre el desempeño de cada método. Actualiza tu aplicación interactiva, si es que la implementaste en la primera tarea.

2.1. Solución

Para este ejercicio se utilizo el conjunto de datos *digits* que contiene que 9298 imágenes de números escritos a mano y digitalizados en tamaño 16x16 pixeles repartidas en 7291 imágenes en un conjunto de entrenamiento y 2007 en el conjunto de prueba. El ejercicio consiste en identificar el número escrito en la imagen, para ello en este caso se hará uso de 6 clasificadores: LDA, QDA, redes neuronales, SVM, árboles de clasificación y AdaBoost, además se usara validación cruzada con base en k-folds para buscar la mejor opción de modelo para la clasificación en cada caso.

Para facilitar el calculo de los k-folds y mejorar los tiempos de entrenamiento se hizo uso de las librerias *caret* y *doparallel*. La libreria *caret* es una libreria especializada para hacer uso de herramientas de machine learning y que permite un fácil acceso a diversos algoritmos de clasificación así como también provee extensiones para ellos como es el caso de la posibilidad de hacer k-foldings para dichos clasificadores usando solo un parámetro extra.

La libreria *doparallel* permite paralelizar algunas operaciones dentro del programa con lo que permitiría disminuir los tiempos de entrenamiento en nuestro caso.

Es importante recalcar que comparar los algoritmos solo con respecto a su nivel de precisión final no es necesariamente la manera más objetiva de hacerlo, ya que en algunos casos se debe considerar buscar parámetros para mejorar la precisión y aparte de esto el costo computacional que requieren puede llegar a ser muy importante dependiendo de la aplicación que se requiera. Además de esto los algoritmos que estan basados en supuestos probabilísticos pueden generar niveles de precisión distintos cada vez se entrenan los datos, aunque en este último caso tiende a variar muy poco el nivel de precisión entre algoritmos. Para este ejercicio los niveles de precisión reportados son los más altos obtenidos por el proceso de k-foldings.

Para los casos de LDA y QDA no son necesarios parámetros extra y el tiempo computacional requerido es en general menor que el de los otros 4 clasificadores. Sin embargo, logran en este caso resultados muy parecidos a los obtenidos por los otros clasificadores. En el caso de LDA y QDA se pudo usar una validación cruzada con 10-folding con lo que se obtuvo para el caso de LDA niveles de precisión para el conjunto de entrenamiento y prueba de 92.15 % y 88.54 %, y respectimante para el caso de QDA se obtuvieron 90.04 % y 88.54 %.

Para los otros 4 clasificadores se tuvo que reducir el númerod de k-foldings debido a que el tiempo computacional aumentaba demasiado. Para el caso de redes neuronales y árboles de clasificación se consideraron 5 dobleces, mientras que para los casos de support vector machine y AdaBoost solo se consideraron 3 dobleces, esto también debido a que los cuatro clasificadores tienen parámetros que deben ajustarse.

Para el caso de redes neuronales se usaron 3 capas de 10 nodos cada una y se obtuvo un nivel de precisión para datos de entrenamiento y prueba de 91.08 % y 86.15 % respectivamente. Para el caso de arboles de clasificación se busco el mejor parámetro de complejidad, para ello se uso una sucesión de valores posibles desde 0.00005 hasta 0.05 con saltos de 0.005, y se obtuvo un nivel de precisión para los datos de entrenamiento de 92.76 % y para un nivel de precisión para lo datos de prueba de 83.66 % conseguidos estos con el parámetro de complejiidad elegido durante la validación cruzada que fue de 0.00005, que como se puede observar en la figura 2.1 es el que mejor nivel de precisión generaba.

Particularmente para support vector machine es posible hacer uso de un kernel para mejorar la precisión sin embargo esto puede aumentar aún mas el costo computacional por lo que se decidió hacer uso de un kernel lineal y se calculo el nivel de margén suave que en este caso se conoce como costo, para ello se busco el mejor parámetro de costo en una sucesión de 2^i con $i \in [-3, 3]$ con saltos de 1 unidad para el caso de i. Con esto se obtuvo un nivel de precisión para los datos de entrenamiento de 92.76 % y para los datos de prueba de 83.66 %. La búsqueda del parámetro de costo se puede resumir en la figura 2.1.

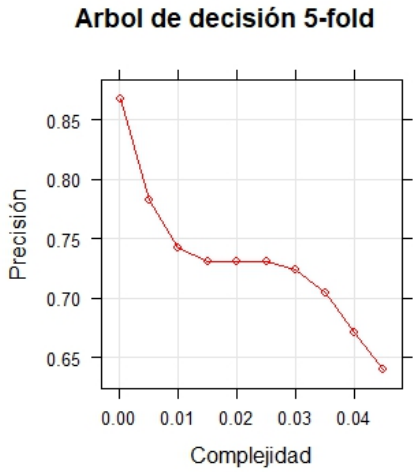

Figura 2.1: Validación cruzada y parámetro de complejidad para arboles de clasificación

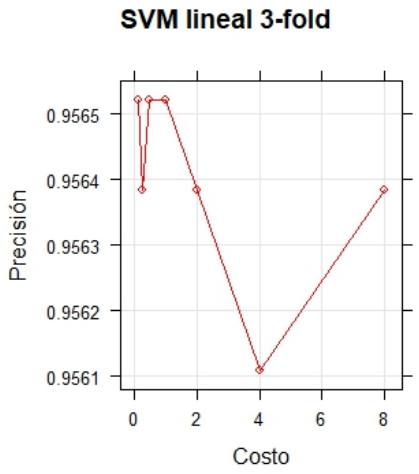

Figura 2.2: Validación cruzada y parámetro de coosto para support vector machine con kernel lineal

Por último se considero el clasificador AdaBoost multiclase el cual permite como parámetro la elección de la profundidad máxima que pueden tener los árboles clasificadores de los que hace uso, en este sentido se consideraron solo 3 profundidades máximas para mejorar los tiempos de ejecución, se usaron árboles de 6, 8 y 10 como máxima profundidad. Además de esto se consideraron 18, 24 y 30 iteraciones. Con esto se obtuvo con un profundidad máxima de 10 y 30 iteraciones un nivel de precisión para los datos de entrenamiento de

100 % y de 92.38 % para el caso de los datos de prueba.

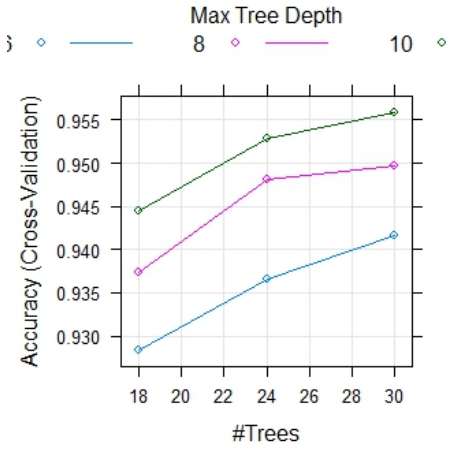

Figura 2.3: Validación cruzada con profundidad máxima y número de iteraciones para AdaBoost multiclase

Los resultados obtenidos se resumen en el siguiente cuadro

Cuadro 2.1: Cuadro resumen de los resultados obtenidos por los clasificadores

	Entrenamiento	Prueba	k-foldings	Parámetros óptimos
LDA	92.15	88.54	10	No aplica
QDA	90.04	88.54	10	No aplica
Redes neuronales	91.08	86.15	5	3 capas ocultas de 10 nodos cada una
Arboles de clasificación	92.76	83.66	5	Parámetro de complejidad=0.00005
SVM	99.98	92.97	3	Costo=0.125
AdaBoost	100	92.38	3	Maxdepth=10, iteraciones=30

En este sentido la elección del mejor algoritmo clasificador basándonos solo en el nivel de precisión obtenido sería un SVM con el parámetro encontrado. Aunque si se considera además el costo computacional sería mejor optar por usar LDA.

3. Ejercicio

Repite el ejercicio 2 para los datos de frutas que usaste en la tarea 2. Utiliza la representación en el espacio HSV con la mediana y los cuartiles centrales.

Puntos extra: verifica el desempeño en el clasificador que elegiste en ejemplos reales". Toma algunas fotos de frutas y realiza un preprocesamiento básico para clasicarlas. Puedes usar el código en C (cortesía de Karen) para quitar el fondo de tu foto. Lee las instrucciones que vienen documentadas.

3.1. Solución

El conjunto de frutas contiene 13 frutas diferentes en diferentes posiciones contando con 1300 imágenes en total, con 100 de ellas por cada fruta. Para la realización del ejercicio se eligieron de manera arbitraria 20 imágenes de cada fruta para formar un conjunto de prueba y las restantes 80 se usaron para formar el conjunto de entrenamiento.

Para la clasificación se tranformo las imágenes al espacio de color HSV y se consideraron 2 conjuntos uno integrado solo por las medianas en dicho espacio y otro integrado por los cuartiles centrales para observar cual de estos espacios muestra una mejor clasificación, aunque dado que el conjunto de los cuartiles centrales contiene al de la mediana se espera que este conjunto permita una mejor clasificación.

Tanto LDA como QDA mostraron una buena clasificación para ambos casos. En el caso de LDA se obtuvieron 87.31 % y 93.08 % de niveles de precisión para los casos de prueba con la represión en la mediana y en los cuartiles centrales respectivamente. Así también para el caso de QDA se obtuvieron 96.15 % y 100 % de clasificación en los casos de prueba para las representaciones de mediana y cuartiles centrales respectivamente.

Para el caso de árboles de decisión se encontró con la validación cruzada que el parámetro óptimo de complejidad es de **0.00505** y se obtuvo un nivel de precisión para el conjunto de prueba de 95.77 % y de 98.46 % para las representaciones de mediana y cuartiles centrales. En las figuras 3.1 y 3.2 se detalla el nivel de precisión obtenido con respecto al parámetro de complejidad en ambas representaciones.

De igual manera con SVM se encontró que el parámetro de margén o costo fue de 8 para el caso de la mediana y de 2 para el caso de los cuartiles centrales y se obtuvieron niveles de precisión para el conjunto de prueba de 96.92 % y 99.62 % para las representaciones de mediana y cuartiles centrales respectivamente. En las figuras 3.3 y 3.4 se detallan distintos niveles de costo y el nivel de precisión que se obtuvo con ellos en cada representación usando SVM.

Por último se utilizó AdaBoost multiclase y en ambas representaciones se obtuvo que el mejor nivel de precisión se lograba con una profundidad máxima de 8 y 30 iteraciones. En este caso se obtuvieron 99.23 % y 100 % de niveles de precisión en las representaciones de la mediana y los cuartiles centrales respectivamente. Las figuras 3.5 y3.6 detallan la elección de los parámetros de máxima profundidad y número de iteraciones para conseguir el mejor

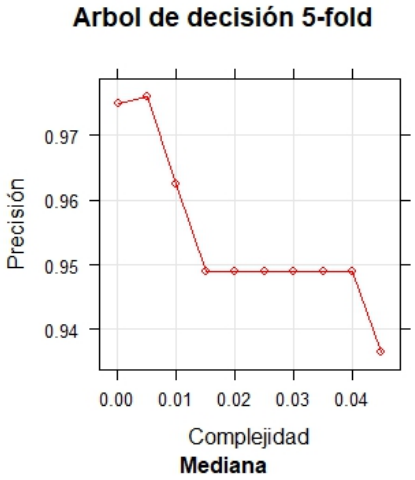

Figura 3.1: Gráfica de nivel de precisión Vs parámetro de complejidad para un árbol de decisión con representación de las frutas en el espacio HSV con respecto a la mediana

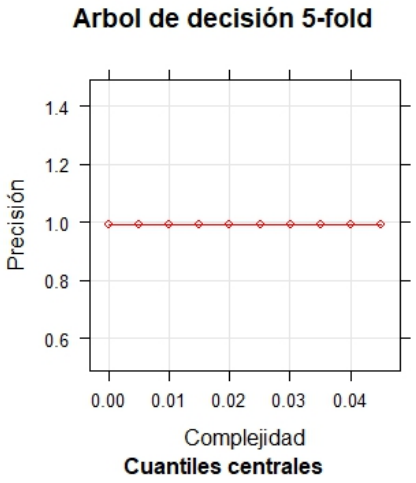

Figura 3.2: Gráfica de nivel de precisión Vs parámetro de complejidad para un árbol de decisión con representación de las frutas en el espacio HSV con respecto a los cuartiles centrales

nivel de precisión.

Por último se detalla en los cuadros 3.1 y 3.2 a manera de resumen todos los parámetros y niveles de precisión encontrados en el proceso. Dado que Adaboost multiclase reporto el mejor desempeño en ambas representaciones se utilizo para clasificar imágenes reales sin embargo debido a algunos problemas técnicos con respecto a OpenCv se tuvo que realizar el recorte y segmentación de las imágenes usadas de una forma alternativa por lo que no se lograron muy buenos resultados, en este sentido solo se logró obtener 3 clasificaciones correctas de las 13 imágenes utilizadas, es decir se logró un nivel de precisión de aproximadamente 23 % aproximadamente para ambos casos.

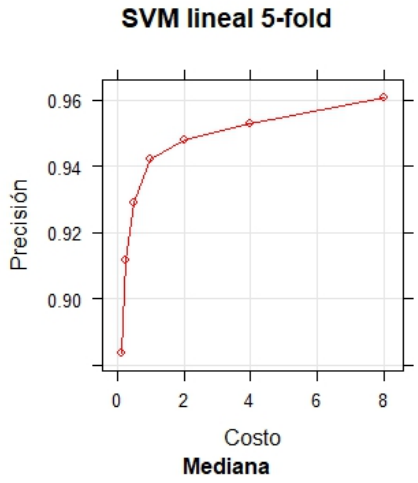

Figura 3.3: Gráfica de nivel de precisión Vs parámetro de costo para SVM con representación de las frutas en el espacio HSV con respecto a la mediana

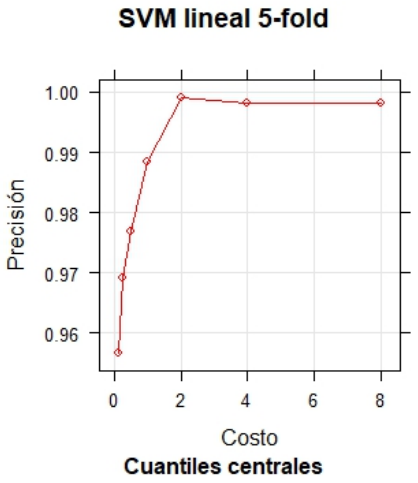

Figura 3.4: Gráfica de nivel de precisión Vs parámetro de costo para SVM con representación de las frutas en el espacio HSV con respecto a los cuartiles centrales

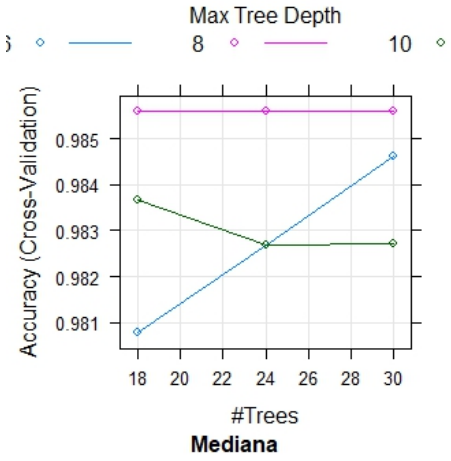

Figura 3.5: Gráfica de nivel de precisión Vs máxima profundidad Vs número de iteraciones en la representación de mediana

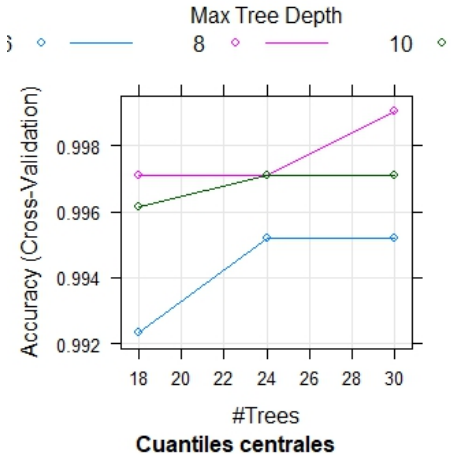

Figura 3.6: Gráfica de nivel de precisión Vs máxima profundidad Vs número de iteraciones en la representación de cuartiles centrales

Cuadro 3.1: Cuadro resumen de los resultados obtenidos por los clasificadores en la representación de mediana

	Entrenamiento	Prueba	k-foldings	Parámetros óptimos
LDA	88.94	87.31	10	No aplica
QDA	96.44	96.15	10	No aplica
Arboles de clasificación	98.17	95.77	5	Parámetro de complejidad=0.00505
SVM	97.02	96.92	5	Costo=8
AdaBoost	100	99.23	3	Maxdepth=8, iteraciones=30

Cuadro 3.2: Cuadro resumen de los resultados obtenidos por los clasificadores en la representación de cuartiles centrales

	Entrenamiento	Prueba	k-foldings	Parámetros óptimos
LDA	94.23	93.08	10	No aplica
QDA	99.42	100	10	No aplica
Arboles de clasificación	99.52	98.46	5	Parámetro de complejidad=0.00505
SVM	99.9	99.62	5	Costo=2
AdaBoost	100	100	3	Maxdepth=8, iteraciones=30

4. Ejercicio

Considera los datos contenidos en $img_expression.zip$, que corresponden a fotos (256x256 pixeles) de mujeres japonesas con diferentes tipos de expresión.

El archivo class_img_exp.dat contiene las etiquetas para cada imagen. En este caso, tenemos dos tipos de etiquetas:

- $file.expression$: corresponde basicamente a la expresión que se le pidió hacer a la persona. La etiqueta NEU es un rostro inexpresivo. Estas etiquetas son:

  HAP (happiness), SAD (sadness), SUR (surprise) ANG (anger), DIS (disgust) y NEU (neutral)

- *semantic.expression* : corresponde a una calicación semántica asignada de acuerdo a un experimento psicológico donde se le pidió a varias personas clasificar cada imagen. En este caso, la clase neutral desaparece, ya que se asignó a alguna de las otras etiquetas. La etiquetación se realizó según la calicación máxima.

Para mas detalles, puedes consultar el archivo README, que describe los datos originales.

1. Repite el ejercicio 1 para estos datos usando Eigenfaces y las etiquetas *semantic.expression*. Especica además cuántos componentes principales usaste y el critero que adoptaste.

2. Ahora hazlo considerando las etiquetas $file.expression$ ¿Qué diferencias notas en el desempeño?

3. Prueba el clasicador que elegiste en imágenes tuyas para estimar tu expresión. Prueba con distintos tipos de fondo, luminosidad y posición para verificar que tan sensible es a las características del entorno. ¿Qué recomendarías para mejorar el clasificador?

4.1. Solución

El conjunto de imágenes class_img_exp.dat contiene fotografias de 10 mujeres con diferentes expresiones faciales. El número de imágenes por mujer es variable entre 17 y 19 imágenes, y también se presenta variabilidad en el número de fotos por expresión facial en cada mujer.

Se separo el conjunto de datos en imágenes de entrenamiento y prueba, para el conjunto de entrenamiento se eligió una imagen de cada expresión facial por cada mujer por lo que se consideraron 60 imágenes para el entrenamiento y 120 para el conjunto de prueba. Aunque claramente el nivel de precisión aumentaría si se usaran más fotografias pero se considera más importante probar el poder del método para reconocer gestos faciales aún cuando el nivel de información que tiene para clasificar es menor de lo usual que de lograr una buena clasificación indicaría que el método captura elementos esenciales de las expresiones faciales y no esta categorizando por rasgos mucho más globales de la cara.

Una vez generada la matriz de imágenes se procede a calcular la media de dicha matriz para hacer que cuestiones como pequeñas traslaciones de los ojos o la boca no afecten el proceso. Después de esto se procede a calcular su factorización SVD para obtener una base de eigenfaces, que son combinaciones lineales de las imágenes centradas con lo que se puede obtener algunos atributos particulares de las imágenes que permitirán clasificarlas con respecto a estos. En la figura 4.1 se pueden apreciar las primeras 5 eigenfaces. Cabe recalcar que el algoritmo SVD nos regresa atributos globales de las imágenes por lo que no se reconocen cosas específicas como los labios sino un conjunto completo como la expresión de los ojos, los labios y ka nariz así como también el cabello.

Figura 4.1: Primeras 5 eigenfaces.

Para la clasificación se usaron 22 eigenfaces (componentes principales) debido a que permiten explicar cerca del 88 % de la varianza, lo cual presenta una reducción a casi una tercera parte de la dimensión del espacio original. Para la clasificación se hizo uso del algoritmo

SAME (AdaBoost multiclase) sobre las dos categorías: *semantic* y *file*.

Usando 10 iteraciones y una profundidad máxima de dos se obtuvieron para la categoria *semantic* 83.3 % y 40.83 % de precisión en los conjuntos de entrenamiento y prueba respectivamente, y en el caso de la categoría $file$ se obtuvieron 86.67 % y 43.33 % para los datos de entrenamiento y prueba respectivamente. Se hizo una prueba más con una profundidad máxima de 10 y 100 iteraciones de donde se obtuvieron 65.3 % y 72.5 % en los casos de prueba de las categorias semantic y file respectivamente.

Los resultados de este último ejercicio se resúmen en los cuadros 4.1 y 4.2, lo que parece indicar en ambos casos que las expresiones de felicidad y sorpresa son las más fáciles de encontrar.

A su vez estos resultados anteriores nos sugieren que la presencia de la categoria neutral tiende a afectar bastante los resultados de clasificación a la alza.

Cuadro 4.1: Clasificación datos de prueba, clasificación tipo semantic

	ANG	DIS	HAP	SAD	SUR
ANG	7	2	2	0	1
DIS	8	17	3	5	0
HAP	0	2	21	4	0
SAD	1	2	4	17	3
SUR	0	0	4	0	17

Cuadro 4.2: Clasificación datos de prueba, clasificación tipo file

	ANG	DIS	HAP	NEU	SAD	SUR
ANG	10	0	0	2	0	1
DIS	1	12	0	0	2	0
HAP	2	2	11	0	3	0
NEU	2	0	4	14	2	3
SAD	4	5	3	4	14	0
SUR	1	0	2	2	0	17

Para corroborar este último hecho se hicieron pruebas con diferentes iteraciones y tamaños de profundidad. En las figuras 4.2 y 4.3 se hicieron pruebas con iteraciones del 1 al 100 y profundidad máxima de 2 aquí se logra apreciar que el mismo efecto se mantiene para los datos de prueba, así también en las figuras 4.5 y 4.7 se puede apreciar este mismo fenómeno por lo que podríamos asegurar que la presencia de este expresión neutral tiende a afectar el nivel de clasificación, lo cual puede deberse a que las características establecidas para esta expresión son significativas realmente. Cabe observar que a mayor profundidad el nivel de precisión tiende a mejorar nuestro nivel de precisión, pero puede generar problemas de complejidad del modelo por lo que no servirá cuando se presenten variaciones en nuevos conjuntos de prueba.

Por último se considero probar el último módelo de 50 iteraciones con profundidad máxima de 10 con 19 fotos reales para ver que tan buena clasificación se puede obtener considerando una mayor variabilidad en las imágenes debido a distintos ángulos de foco e iluminación.

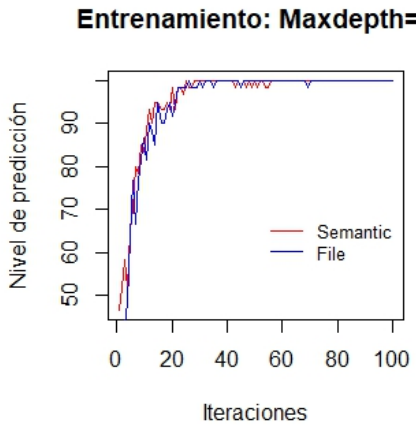

Figura 4.2: Primeras 100 iteraciones con profundidad máxima igual a 2 para datos de entrenamiento en ambas categorías

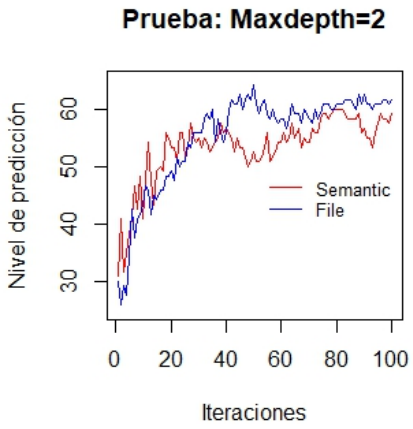

Figura 4.3: Primeras 100 iteraciones con profundidad maxima igual a 2 para datos de prueba en ambas categorías

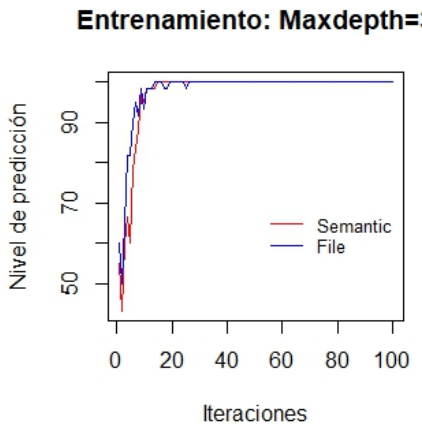

Figura 4.4: Primeras 100 iteraciones con profundidad máxima igual a 3 para datos de entrenamiento en ambas categorías

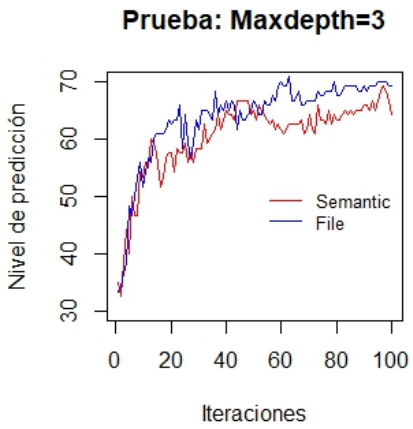

Figura 4.5: Primeras 100 iteraciones con profundidad maxima igual a 3 para datos de prueba en ambas categorías

Para el proceso de las imágenes se uso un software simple para cortar partes innecesarias de las fotografias y a su vez reescalarlas para dejarlas lo más parecidas posibles a las usadas pa-

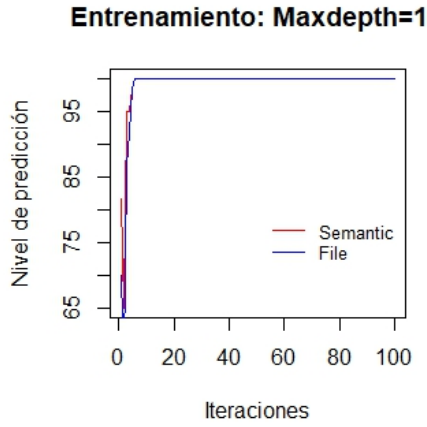

Figura 4.6: Primeras 100 iteraciones con profundidad máxima igual a 10 para datos de entrenamiento en ambas categorías

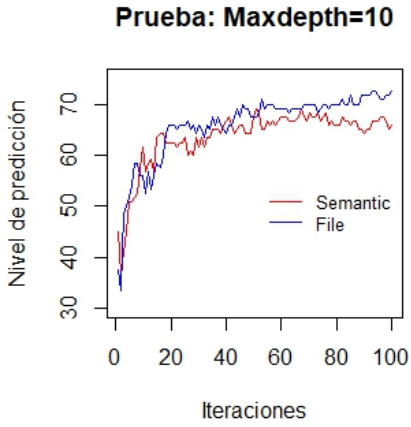

Figura 4.7: Primeras 100 iteraciones con profundidad maxima igual a 10 para datos de prueba en ambas categorías

ra el entrenamiento. Una muestra de las imágenes se puede observar en las figuras 4.8 y 4.9.

Figura 4.8: Feliz

Figura 4.9: Sorprendido

El nivel de precisión obtenido es de 21.05 % y 15.79 % para las clasificaciones *semantic* y *file* respectivamente. En los cuadro 4.3 y 4.4 se resúmen los resultados de la clasificación. Estos resultado nos muestran una gran falla en nuestro clasificador debido a que no logra encontrar expresiones y tiende a desaparecer tipos de expresión completas esto se deja claro

ya que en ninguno de los dos casos se encontraron las expresiones de enojo, esto se puede deber a muchas cuestiones, que pueden y deben resolverse si se desea construir un mejor clasificador.

Cuadro 4.3: Clasificación de imágenes reales usando el etiquetado tipo semantic

	ANG	DIS	HAP	SAD	SUR
ANG	0	0	0	0	0
DIS	0	0	0	2	1
HAP	1	1	3	0	2
SAD	0	2	0	0	0
SUR	1	0	2	3	1

Cuadro 4.4: Clasificación de imágenes reales usando el etiquetado tipo file

	ANG	DIS	HAP	NEU	SAD	SUR
ANG	0	0	0	0	0	0
DIS	0	0	0	0	0	0
HAP	1	0	0	1	0	1
NEU	1	3	4	3	2	3
SAD	0	0	0	0	0	0
SUR	0	0	0	0	0	0

Entre estas cuestiones debe considerarse que las imágenes de entrenamiento utilizadas son de mujeres asiáticas por lo que los rasgos faciales difieren en este caso tanto por sexo como por raza, por lo que para mejorar el clasificador deben considerar integrar imágenes variadas por género y raza para aumentar la variabilidad en los datos.

Otra cuestión importante es que las fotografías reales fueron tomadas de manera amateur por lo que los focos y distancias de las fotos presentan una gran variación y aún cuando se centraron con la media de las fotos de entrenamiento puede terminar difiriendo bastante de las originales. Para esto una opción es contruir distintas bases de eigenfaces una por cada tipo de expresión incluyendo en ella imágenes tomadas desde diferentes ángulos y establecer un criterio de elección por error mínimo para su clasificación. Para el problema de la iluminación se puede usar un proceso de segmentación previo para generar mascaras y solo considerar la imagen de la cara o implementar métodos que disminuyan el efecto de la iluminación sobre la imagen.