Departamento de Ingeniería Eléctrica
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
EL7006-1 Redes Neuronales y Teoría de Información para el Aprendizaje

Informe Final

Clasificación de imágenes de estrellas variables usando NMF y PCA

Integrantes: Diego Campanini
Eduardo Hormazábal
Profesor: Pablo Estevez
Auxiliar: Pablo Huijse
Ayudante: Pablo Huentelemu
Fecha: 11 de diciembre de 2015

Índice

1. Introducción

El proyecto llamado *High Candence Transient Survey* está centrado en la detección en tiempo real de supernovas. Lo que se está intentando encontrar es un fenómeno transiente que ocurre en el instante inicial de la explosión de esta. El fenómeno se conoce como *supernova shock breakout* (SBO) y ocurre dentro de las primeras horas después de comenzado el proceso de la supernova.

La base de datos con la cual se efectuarán las pruebas posee 100.000 muestras, la cual está dividida uniformemente entre las clases positivas candidatas a transiente asociado a variabilidad estelar y la clase negativa, que es la de transientes asociados a rayos cósmicos, además de la detección y substracción de artefactos. Cada muestra corresponde a tres imágenes de 21x21 pixeles cada una, lo que motiva a la reducción de características.

El trabajo que se realizará consiste en efectuar clasificación supervisada para imágenes de estrellas variables. Para esto se extraerán características desde la base de datos usando PCA y NMF. Se utilizarán dos clasificadores: uno basado en SVM y otro será un clasificador *Random Forest* para obtener una amplia visión de la calidad de las características generadas por PCA y NMF.

Para las pruebas en los clasificadores mencionados anteriormente se realizarán tres grandes pruebas, una con características obtenidas sólo con PCA, otra solamente con NMF y por último se combinarán características obtenidas con ambos métodos para mejorar el desempeño del clasificador.

2. Antecedentes

2.1. Supernovas

Una supernova es una explosión estelar, esta radía tanta energía como el sol lo haría en toda su vida útil. La radiación producida en la explosión emite toda o gran parte de la materia de la estrella en cuestión. La velocidad que este material pude alcanzar es de hasta $0{,}1c$, produciendo una onda de choque en el medio que la rodea y una esfera de gas y polvo, llamado remanente de la supernova. En la Figura 1 se muestra el remanente de supernova de Tycho.

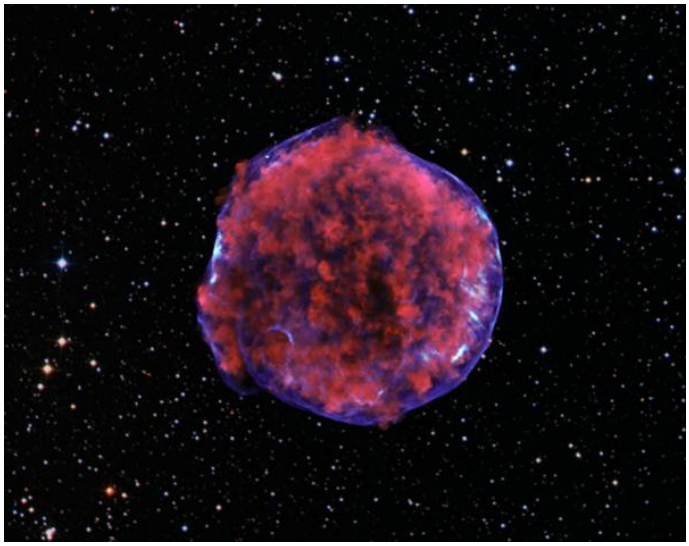


Figura 1: Remanente de la supernova de Tycho (supernova del tipo Ia)

Los astrónomos han clasificado las supernovas de acuerdo a sus curvas de luz y las líneas de absorción de los diferentes elementos químicos que aparecen en su espectro. El primer elemento de división tiene relación con el hidrógeno. Si en el espectro de una supernova aparece una línea de hidrógeno, conocida como serie de Balmer, entonces es de Tipo II, en caso contrario es de Tipo I. Dentro de los dos tipos de supernovas existen sub-clasificaciones, que se realizan dependiendo de la presencia de líneas de espectros de otros elementos o de la forma de las curvas de luz. A continuación se presentan las sub clasificaciones de las supernovas en el Cuadro 1 y sus respectivas curvas de luz en la Figura 2

Clasificación	Sub Clasificación	Característica
Tipo I	Tipo Ia	Presenta un línea de silico ionizado
	Tipo Ib	Presenta una línea de He no ionizado
	Tipo Ic	Debilidad o ausencia de líneas de He
Tipo II	Tipo II P	Alcanza una meseta en su curva de luz
	Tipo II L	Muestra decrecimiento lineal en su curva de luz
	Tipo II LLn	Muestra lineas angostas de sus curvas

Cuadro 1: Tabla clasificación de supernovas

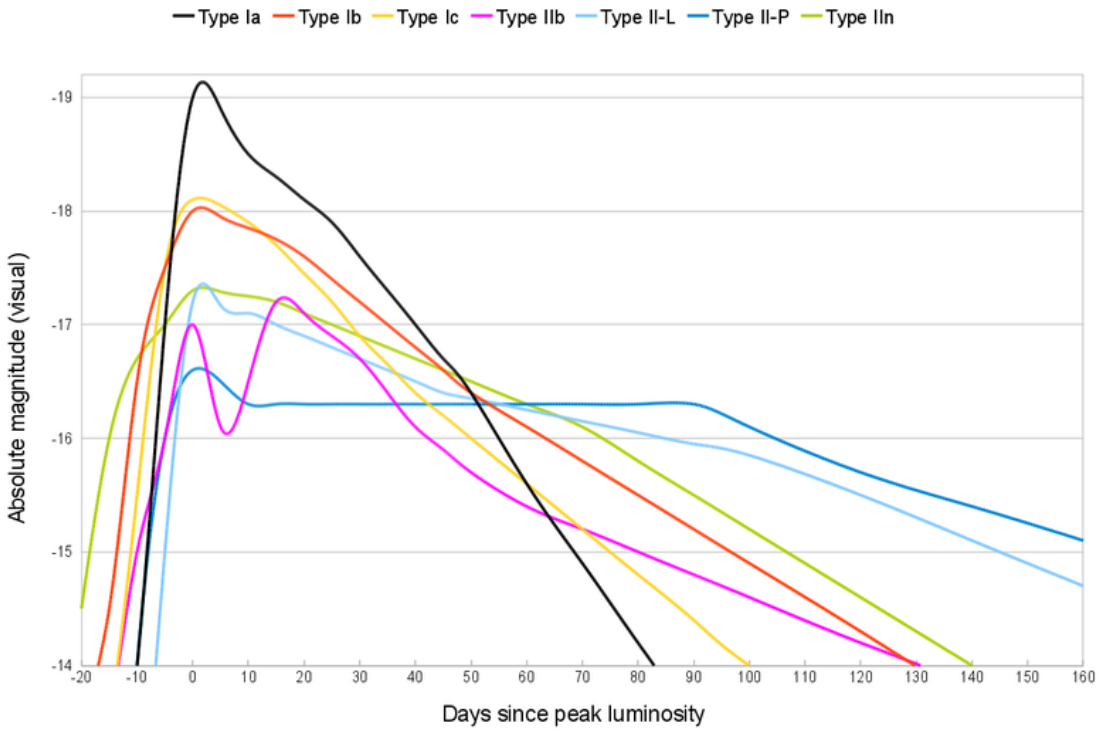


Figura 2: Curvas de Luz según el tipo de supernova

La importancia para los científicos de estudiar las supernovas es que con éstas se pueden estudiar distintos aspectos del universo, como por ejemplo su composición, el origen de planetas como la tierra, y medir distancias en el universo. Un notable hecho sobre el estudio de supernovas es que a través de las tipo Ia se logró determinar la expansión acelerada del universo, ya que se encontraron más de 50 supernovas cuya luz era más débil de lo esperado, lo cual se interpretó como una señal de expansión del universo.

2.2. Base de Datos

La base de datos a utilizar proviene del proyecto astronómico High Caddence Transient Survey (HiTS), que trata de encontrar eventos transientes ocurridos en los primeros instantes de una supernova. En particular se posee la información que data entre los años 2013 y 2015, correspondiente a 100,000 muestras de 3 imágenes de 21x21 pixeles. Cada muestra está etiquetada de acuerdo a dos clases: Transiente por variabilidad estelar y Transiente por artefactos. Estas clases se encuentran en forma balanceada en la base de datos, es decir que aproximadamente un 50 % corresponde a una clase, y el otro 50 % a la otra.

Con el fin de detectar fuentes variables, HiTS calcula en tiempo real una diferencia normalizada por el SNR entre dos imágenes de distintas épocas. Debido a esto, para cada transiente se generan tres imágenes prototipo: de referencia, actual, y diferencia normalizada entre las primeras dos.

Por ejemplo en la Figura 3 se observan dos muestras, en donde una corresponde a un transiente por artefactos ruidosos y la otra indica claramente un cambio en la iluminación entre épocas del sector sen-

sado. Cabe destacar que la diferenciación de las imágenes presenta información relevante pues a simple vista hay casos en que no se podría distinguir la explosión de una supernova.

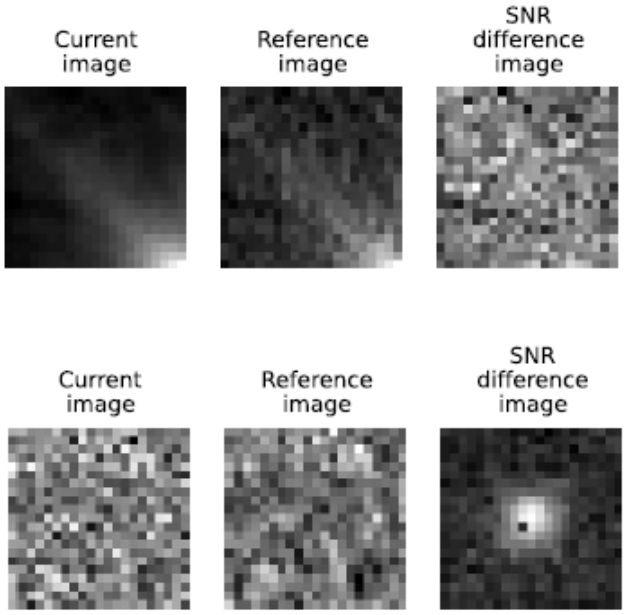


Figura 3: Dos muestras de la base de datos HiTS. Superior: transiente por artefactos. Inferior: transiente por variabilidad estelar.

3. Marco Teórico

3.1. NMF

Non-negative Matrix Factorization (NMF)[1] es un método de reducción de dimensionalidad a K prototipos en base a la construcción automática de un diccionario W cuya combinación lineal se ajusta a las muestras originales V. También se destaca el manejo de la *sparsidad* de los coeficientes del diccionario, H, que es producto de la condición de no-negatividad dentro de la combinación lineal WH.

Así, la condición a satisfacer para optimizar el desempeño de esta descomposición es el siguiente:

$$\min_{W,H} \|V - WH\|_F^2 + \lambda \|H\|_1 \quad s.t. W \geq 0, H \geq 0 \tag{1}$$

donde $V \in \mathbb{R}^{MxN}$ (N ejemplos y M dimensiones), $W \in \mathbb{R}^{MxK}$, $H \in \mathbb{R}^{KxN}$, y λ corresponde al coeficiente de *sparsidad* que se encarga de establecer un *trade-off* entre el error de reconstrucción y qué tan *sparse* es esta reconstrucción.

Una de las características positivas de NMF es que genera vectores base que representan partes de la información de entrada, lo que en términos de NMF aplicado a imágenes se obtendrán máscaras que indican distintas características en la imagen y, como también se busca *sparsidad*, la cantidad de máscaras para representar una imagen en particular debiera ser reducida.

3.2. PCA

El Análisis de Componentes Principales (PCA) tiene como principal característica la reducción de dimensión de los datos desde d a K, minimizando la información perdida mediante la maximización de la varianza entre componentes (criterio con momento de segundo orden).

La idea general de obtener las características consiste en lo siguiente:

- Computar el vector de medias de d-dimensiones, es decir las medias para cada dimensión de los datos y.
- Computar la matriz de covarianza de los datos.
- Computar los d vectores y los d valores propios.
- Ordenar los valores propios de forma decreciente y elegir los K mayores, formando una matriz W de dimensiones $d\mathbf{x}K$ (considerando los vectores propios respectivos). Esto se resume en una aproximación de los datos $\hat{y} = W^T \mathbf{x} x$, en donde x es una muestra de tamaño d e $\hat{y}$ es la transformada de tamaño K.

3.3. SVM

En el contexto de aprendizaje el *Suppor Vector Machine* es un método que permite la clasificación de clases. En términos generales el SVM encuentra un hiperplano que separa dos clases, tal que el margen de separación se maximice. Lo anterior se aplica a casos linealmente separables y se extiende para los que no lo son a través del aumento de dimensionalidad con un kernel.

Para separar las dos clases se puede definir la ecuación del hiperplano que las separa, la cual se expresa como sigue:

$$\mathbf{w}^t \cdot \mathbf{x} + b = 0 \tag{2}$$

Donde $\mathbf{w}$ es un vector normal al hiperplano (vector de pesos), $\mathbf{x}$ es un vector de entrada y b es el bias que corresponde a un valor constante. Para un cierto $\mathbf{w}$ y b, la separación entre el hiperplano y el conjunto de datos más cercano se denomina *margen*. Cuando los datos son no separables se pueden seleccionar dos planos tal que entre ellos no se encuentre ningún dato, para seleccionar toda la porción de los datos de una clase o la otra, estos dos planos se pueden definir como:

$$\mathbf{w}^t \cdot x + b \geq 1 \tag{3}$$
$$\mathbf{w}^t \cdot x + b \leq -1 \tag{4}$$

En la figura 4 se pueden observar los hiperplanos con línea segmentada que definen el margen y el hiperplano óptimo, para un problema de dos clases.

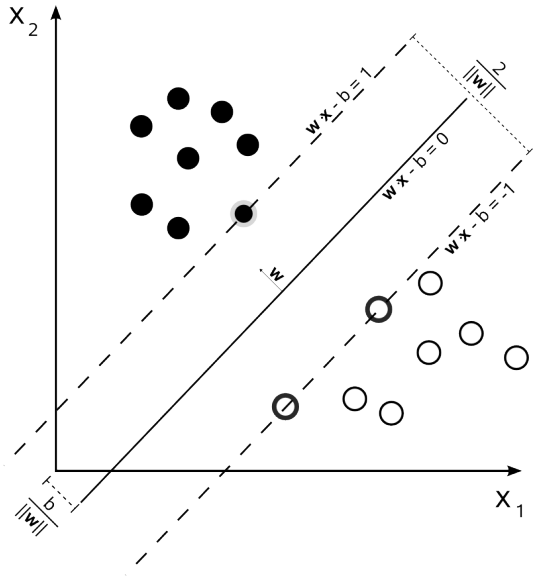


Figura 4: Separación de dos clases (círculos blancos y negros) mediante hiperplanos

3.4. Random Forest

Un Random Forest (RF) es un predictor compuesto por un conjunto de árboles de decisión que promedia los valores entregados por éstos. Cada árbol de decisión es construido utilizando un subconjunto aleatorio de los datos de entrenamiento.

La ventaja que tiene un clasificador RF con respecto a otros es que es un buen algoritmo de aprendizaje para conjuntos de datos grandes [4], ya sea como su desempeño como su eficiencia en el entrenamiento.

4. Metodología

El proceso que consiste en llevar a cabo la clasificación binaria de las imágenes se puede separar en una secuencia de bloques funcionales: normalización, extracción de características, clasificación. El primero se realiza mediante una transformación lineal de los datos, mapeándolos en el intervalo [0.0, 1.0]. El segundo bloque se construye de dos formas distintas: Análisis de Componentes Principales (PCA) y Non-negative Matrix Factorization (NMF), con el fin de comparar sus desempeños [2]. El último bloque es entrenado con Support Vector Machine (SVM) y Random Forest (RF), con el objetivo de obtener el mejor desempeño posible con respecto a las características extraídas en el segundo bloque.

4.1. Normalización

La normalización consiste en realizar un mapeo lineal hacia el intervalo [0.0, 1.0] de cada muestra, para cada una de las 3 imágenes disponibles. Esto se debe a que el algoritmo de extracción de características NMF requiere datos con componentes no negativos, además de acotar la varianza de las muestras a 1,0.

La normalización utilizada está dada por:

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})} \tag{5}$$

4.2. Extracción de características

4.2.1. Análisis de Componentes principales

Este algoritmo se utilizó desde el *toolbox* de Matlab, en donde el parámetro de extracción es la cantidad K de componentes principales. Se computaron resultados para distintos $K \in \{1, 10, 20, 50, 100, 200\}$ con el fin de comparar el mejor de éstos con el mejor extractor de características para cada algoritmo (NMF y combinación PCA-NMF).

4.2.2. Non-negative Matrix Factorization

NMF se compone de dos partes. La primera es obtener los diccionarios utilizando los datos de entrenamiento, lo que se realizó con el *toolbox* de Matlab. La segunda es calcular la proyección de los datos de prueba en los diccionarios obtenidos. Esto último se programó en Matlab de acuerdo a lo visto en la sección 3.1. El parámetro de extracción es la cantidad $K \in \{1, 10, 20, 50, 100, 200\}$ de vectores prototipo, de manera similar al algoritmo PCA visto en el inciso anterior.

4.2.3. Combinación PCA-NMF

La última propuesta de extracción de características consiste en utilizar los componentes obtenidos tanto en PCA como en NMF. Para ello, se seleccionan combinaciones de estas componentes de forma heurística hasta lograr resultados convincentes.

Gracias a la combinación de ambas se puede establecer un *trade-off* entre la solución óptima para una norma L2 de PCA y la robustez frente al ruido no Gaussiano de NMF. Esto podría ser equivalente a un

"kernel"no lineal sobre los momentos que considera PCA y NMF, y como se ha visto en clases, es posible llegar a mejores resultados ajustando los parámetros de este "kernel".

4.3. Clasificador

Los dos clasificadores a utilizar se obtienen del *toolbox* de Matlab, en donde el entrenamiento se hace mediante los vectores de características de las imágenes obtenidos en la sección 4. Este proceso no se detalla en mayor profundidad ya que el objetivo es comparar NMF versus otros extractores de características. La razón por la que se utilizarán 2 clasificadores diferentes es para explorar mejores resultados y así evitar un posible sesgo de un extractor de características para un clasificador en particular.

5. Resultados y Análisis

Para realizar la clasificación se utilizó el 20 % de la base de datos HiTS, es decir, se emplearon 20.000 curvas. Se seleccionó al azar un 70 % (14000 datos) de las curvas para formar el conjunto de entrenamiento, dentro de este conjunto se tomó un 15 % para emplearlo como set de validación para el clasificador SVM. El 30 % (6000 datos) restante se utilizó como conjunto de prueba

5.1. Clasificación con SVM

5.1.1. Clasificación con características obtenidas con PCA

La primera clasificación que se realizó con SVM fue para un kernel gaussiano con desviación estándar igual a 2. Se consideró la selección de características usando PCA, tomando 6 distintos números de componentes principales K (1, 10, 20, 30, 50, 100 y 200) para cada una de las tres imágenes. Además se clasificó usando la media y la desviación estándar de las imágenes no normalizadas.

El peor resultado de clasificación en términos de la TPR y FPR del conjunto de prueba, se obtuvo para K=200 (figura 5 (b)), se tiene que los falsos positivos alcanzan un 14.93 %, mencionar que se obtieen un buen TPR, el resulta ser de un 99.49 %, sin embargo, el FPR alcanza un 29.47 %. Por otra parte el mejor caso se obtuvo para K=10 (figura 5 (a)), obteniéndose un TPR=99.39 % y FPR=4.9 %.

		Clase predicha	
		Positiva	Negativa
Clase Real	Positiva	2942 49.03 %	18 0.30 %
	Negativa	149 2.48 %	2891 48.18 %

(a) Clasificación para K=10

		Clase predicha	
		Positiva	Negativa
Clase Real	Positiva	2945 49.08 %	15 0.25 %
	Negativa	896 14.93 %	2144 35.73 %

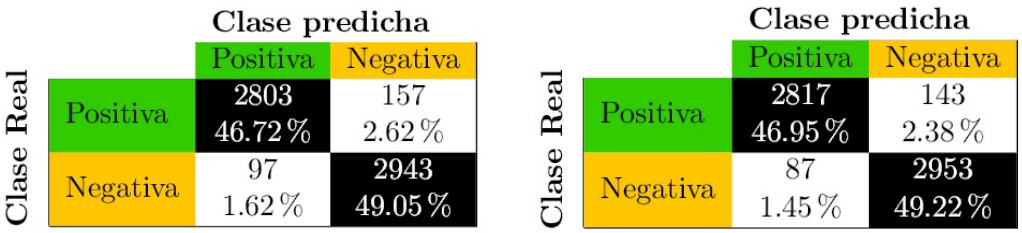
(b) Clasificación para K=200

Figura 5: Matrices de confusión para selección de características con PCA para el conjunto de prueba

5.1.2. Clasificación con características obtenidas con NMF

Se procede a entrenar con SVM para un kernel gaussiano con desviación estándar de 2, al igual que los resultados expuestos en la subsección anterior. Se entrenó con las matrices de coeficientes H_i de dimensión Kx14000 y se probó con las matrices de coeficientes proyectadas H_i' de dimensión Kx6000.

El peor caso fue cuando K=10 (figura 6 (a)) , obteniendo un TPR=94.7 % y un FPR=3.19 %, lo cual resulta bueno, pero inferior a la clasificación con PCA. El mejor caso resultó para K=200 (figura 6 (b))con un TPR=95.17 % y un FPR=2.86, estos resultados continúan siendo inferior a los obtenidos al clasificar con característica obtenidas con PCA.



(a) Clasificación para K=10

(b) Clasificación para K=200

Figura 6: Matrices de confusión para selección de características con NMF para el conjunto de prueba

Una visión general del comportamiento de este clasificador para el kernel antes mencionado, se muestra en la curva ROC de la figura 7, la cual muestra TPR versus FPR para los seis K utilizados en el análisis, esto para PCA y NMF. Destacar que claramente para PCA el mejor resultado se obtiene para K=10, además NMF resulta muy inferior (curva roja Figura 7), sobre todo en lo que respecta a la TPR. Estos resultados serán contrastados con los obtenidos en la siguiente sección para un clasificador Random Forest.

Se analizó también la clasificación para los mismos conjuntos y valores de K, pero esta vez para distintos sigma, resultando que para sigma menores a 1 el rendimiento del clasificador usando NMF mejoraba alcanzando TPR entre 96 y 97 %, por otra lado, para desviaciones estándar superiores a 1 el rendimiento empeoraba, por ejemplo con desviación estándar de 10 se alcanzó un TPR=86 % y FPR=25 %. Para el uso de un kernel lineal el rendimiento caía bruscamente alcanzando TPR=15.6 % y FPR=81.7 %. Similar comportamiento se obtuvo para la clasificación sólo con características calculadas con PCA, destacar que en todos los casos PCA resultó tener un mejor porcentaje de correcta clasificación.

En la siguiente sección se comprobará que el desempeño de las características calculadas con NMF tienen relación con el clasificador que se emplea. Además se tiene que considerar que se está trabajando con una base de datos reducida de la original, por lo que los resultados no son totalmente comparables a los resultado previos [2].

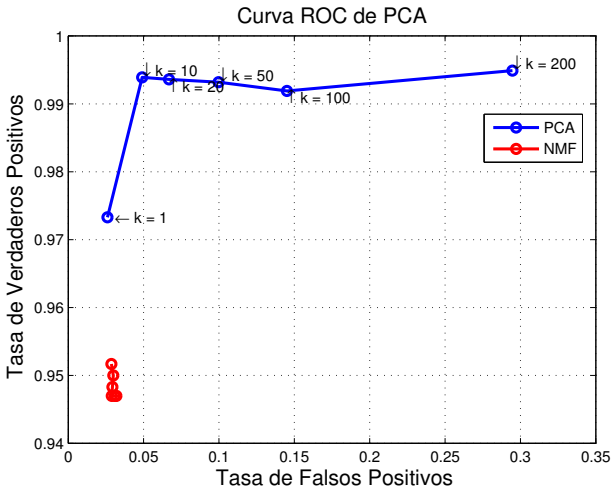


Figura 7: Curva ROC para clasificación SVM con características PCA y NMF para K=1,10,20,50,100 y 200

5.2. Clasificación con Random Forest

5.2.1. Clasificación con características obtenidas con PCA

El segundo tipo de clasificador se efectuó con el algoritmo de Random Forest utilizando 100 árboles de decisión, al igual que el *paper* en que se basa este trabajo [2]. Se consideró la selección de características usando PCA para 6 distintos números de componentes principales K (1, 10, 20, 50, 100 y 200), al igual que el clasificador de la sección 5.1.

El peor caso de clasificación con respecto a la TPR y FPR del conjunto de prueba fue para K=200, con un 1.46 % de error de clasificación (figura 8(b)). Mientras que el mejor caso es para K=10, con un 0.97 % de error de clasificación (figura 8(a)).

Clase Real \ Clase predicha	Positiva	Negativa
Positiva	2939 48.98 %	21 0.35 %
Negativa	37 0.62 %	3003 50.05 %

(a) Clasificación para K=10

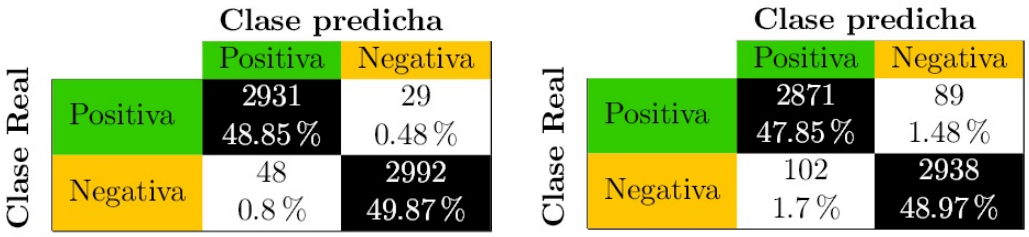
Clase Real \ Clase predicha	Positiva	Negativa
Positiva	2937 48.95 %	23 0.38 %
Negativa	65 1.08 %	2975 49.58 %

(b) Clasificación para K=200

Figura 8: Matrices de confusión para características PCA y clasificación con Random Forest para el conjunto de prueba

5.2.2. Clasificación con características obtenidas con NMF

Con respecto al extractor de características NMF, el mejor caso se obtuvo para K=10 con un 1.28 % de error de clasificación (figura 9(a)), mientras el peor fue para K=100 con un 3.18 % de error (figura 9(b)). Cabe destacar que el extractor PCA funciona mejor que el NMF para el clasificador RF, que difiere del *paper* de referencia [2]. Esto probablemente se debe a la utilización de un porcentaje de la base de datos real.



(a) Clasificación para K=10

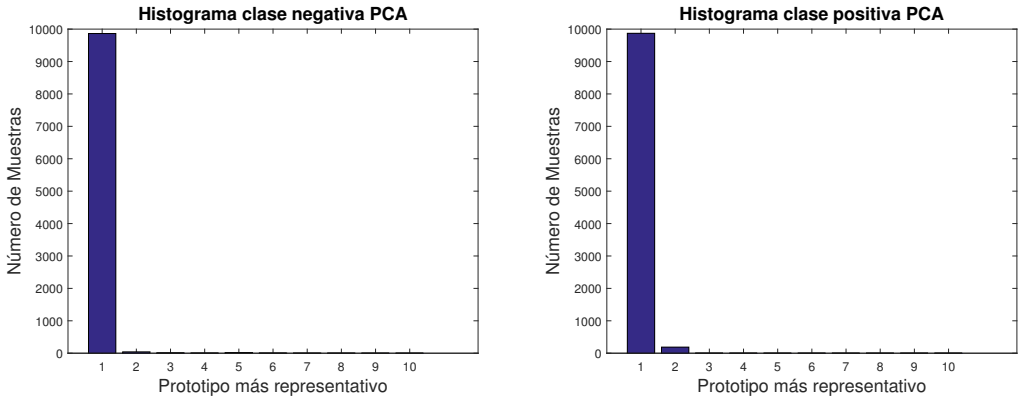
(b) Clasificación para K=100

Figura 9: Matrices de confusión para características NMF y clasificación con Random Forest para el conjunto de prueba

5.2.3. Clasificación con características obtenidas con NMF y PCA mezcladas

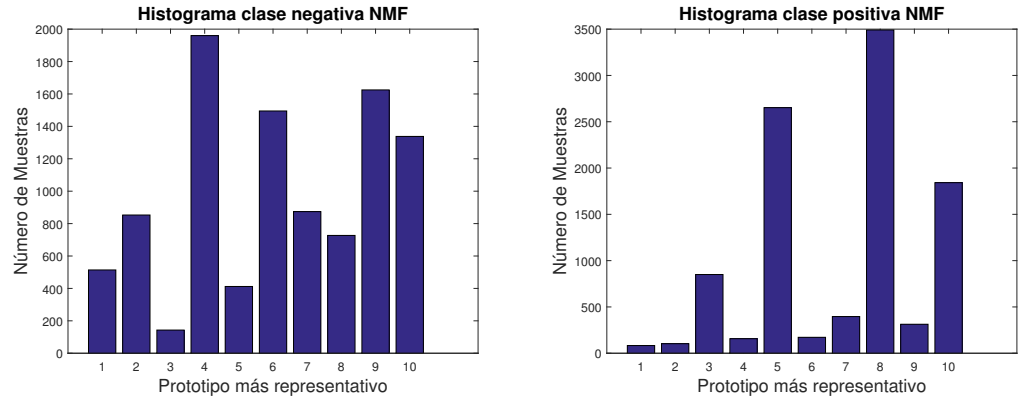
Una de las incógnitas que se tuvo era la posibilidad de mejorar el desempeño de un clasificador utilizando una combinación de características generadas por PCA y NMF. Para verificar que las características de ambos extractores difieren en su comportamiento se graficaron los histogramas del prototipo más influyente para cada dato con respecto a su clase, tanto de entrenamiento como de prueba para un K=10, pues este valor corresponde al mejor extractor de características encontrado en las secciones.

En la figura 10 se observan los histogramas de la clase positiva y negativa, en donde ambos poseen una similitud con respecto al prototipo más predominante. Por otro lado en la figura 11 se aprecia claramente una distribución diferente para ambas clases, lo que indica que NMF logra separar en *clusters* a las dos etiquetas, por lo tanto es posible utilizar la información del mayor prototipo obtenido de la proyección de los datos para mejorar la clasificación.



(a) Clase negativa (b) Clase positiva

Figura 10: Histogramas del prototipo más representativo para PCA, con K=10



(a) Clase negativa (b) Clase positiva

Figura 11: Histogramas del prototipo más representativo para NMF, con K=10

Como la separación de las clases es distinta para PCA y NMF (de acuerdo a los histogramas), y además ambos extractores arrojan resultados similares al momento de clasificar sus características, se mezclaron características de ambos métodos, obteniendo los resultados del Cuadro 2.

Número de prototipos K por imagen	Combinación de características por imagen	TPR [%]	FPR [%]
K=10	K-PCA + K-NMF + mean + std	99.39	1.25
	K-PCA + K-NMF	98.92	1.51
	K/2-PCA + K/2-NMF + mean + std	99.32	1.12
	K/2-PCA + K/2-NMF	98.95	1.55
K=20	K-PCA + K-NMF + mean + std	99.39	1.38
	K-PCA + K-NMF	98.95	2.01
	K/4-PCA + K/4-NMF + mean + std	98.68	2.04
	K/4-PCA + K/4-NMF	98.61	2.11

Cuadro 2: Tasas de sensibilidad y especificidad para un clasificador RF con características mezcladas de PCA y NMF

En el Cuadro 2 se puede observar que se alcanza una mayor tasa de verdaderos positivos (TPR) y una menor tasa de falsos positivos (FPR) que cualquiera de los resultados anteriores a mezclar las características. En particular si se extraen características con K=10, y se elige la mitad de las de PCA y la mitad de las de NMF, junto con la media y la desviación estándar, se logra un TPR=99.32 % y un FPR=1.12 %. Es decir, se logra una mejora al combinar características de los diferentes métodos.

También cabe destacar que la inclusión de la media y desviación estándar mejoran el desempeño del clasificador. Esto confirma la hipótesis de combinar características de distintos métodos, pues la media y la desviación estándar son extracciones de características heurísticas utilizadas en gran parte de la literatura, reforzando la clasificación de datos que poseen determinadas características.

6. Conclusión

Destacar que la cantidad de datos a procesar no fue menor, por lo que resultó de gran relevancia optimizar los algoritmos implementados, trabajar con arreglos generados previamente y que no se actualicen su dimensión en cada iteración. En este sentido la extracción de características con NMF resultó más costosa que PCA desde el punto de vista computacional.

Resultó mejor clasificar utilizando Random Forest con 100 árboles de decisión que con SVM para cualquier kernel probado (lineal y gaussiano con distintas desviaciones estándar). Lo anterior tanto en lo que respecta al rendimiento del clasificador, como al costo computacional de los dos métodos de clasificación empleados.

El mejor resultado de clasificación se logró al mezclar características de NMF y PCA. Esto se obtuvo para K=10 considerando las 5 primeras componentes de cada método, más la desviación estándar y la media, lográndose un TPR=99.32 % y FPR=1.12 %

Se propone investigar la calidad de los vectores de características al incluir estadísticos de diferente orden, pues se verificó que al agregar la media y desviación estándar al conjunto de características se mejoran los resultados de clasificación, con lo que podría esperarse un resultado similar al incluir por ejemplo el skewness y la kurtosis.

Bibliografía

[1] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401:788–791, 1999.

[2] Pablo Huijse, Pablo A. Estévez, Francisco Förster, Emanuel Berrocal, Discriminating Variable Star Candidates in Large Image Databases from the HiTS Survey Using NMF. *Procedia Computer Science, Volume 53, 2015, Pages 29-38*, ISSN 1877-0509.

[3] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. Nonnegative matrix andtensor factorizations: applications to exploratory multi-way data analysis and blind source separation. John Wiley & Sons, 2009.

[4] Rich Caruana; Nikos Karampatziakis; Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

7. Anexos

Listing 1: Random Forest PCA

```
clear
addpath('Funciones')
load('HiTS2000')
%% Extracci n de caracter sticas con PCA y armado de conjuntos
k=200; % n mero de componentes seleccionadas
[ H_all,P_all,pH1] = makePca2( entH1,entH2,entHs,pH1,pH2,pHs,k); % Conjunto de
    entrenamiento
H_all=[H_all m_ent std_ent];
P_all=[P_all m_test std_test];

%%  rbol
nTrees=100;
B=TreeBagger(nTrees,H_all,classE,'Method','classification');
out_pca_ent=B.predict(H_all);
out_pca_ent=str2double(out_pca_ent);

out_pca_test=B.predict(P_all);
out_pca_test=str2double(out_pca_test);


%% C lculo de FPR y TPR
[FPRe,TPRe,FNRe,TPe,FPe,FNe,TNe] = rendimiento( classE, out_pca_ent);
[FPRt,TPRt,FNRt,TPp,FPp,FNp,TNp] = rendimiento( classP, out_pca_test);
```

Listing 2: Random Forest NMF

```

clear
addpath('Funciones')
load('HiTS2000')
%%  Extracci n de caracter sticas con NMF
k=200;% filas matriz de coeficientes
[H_all,D1,D2,D3,W1,W2,W3] = makeNmf(entH1,entH2,entHs,k,m_ent,std_ent);
[P_all,e1,e2,e3] = makeNmf_h(pH1,pH2,pHs,W1,W2,W3,k,m_test,std_test);


%%  rbol
nTrees=100;
B=TreeBagger(nTrees,H_all,classE,'Method','classification');
out_nmf_ent=B.predict(H_all);
out_nmf_ent=str2double(out_nmf_ent);

out_nmf_test=B.predict(P_all);
out_nmf_test=str2double(out_nmf_test);


%% C lculo de FPR y TPR
[FPRe,TPRe,FNRe,TPe,FPe,FNe,TNe] = rendimiento( classE, out_nmf_ent );
```

```
[FPRt,TPRt,FNRt,TPp,FPp,FNp,TNp] = rendimiento( classP , out_nmf_test );
```

Listing 3: SVM PCA

```
%%
clear
addpath('Funciones')
load('HiTS2000')

%% Extracci n de caracter sticas con PCA y armado de conjuntos
k=10; % n mero de componentes seleccionadas
[ H_all , P_all ,pH1] = makePca2( entH1 ,entH2 ,entHs ,pH1 ,pH2 ,pHs ,k); % Conjunto de
    entrenamiento
H_all=[H_all m_ent std_ent];
P_all=[P_all m_test std_test];
%[ P_all] = makePca2( pH1 ,pH2 ,pHs ,k);% COnjunto de Prueba

%% Entrenamiento
svmStruct=fitcsvm(H_all ,classE ,'Holdout',0.15,'KernelFunction','rbf','BoxConstraint'
    ,10,'ClassName',[1 ,0],'KernelScale',2);

%% C lculo clase de salida
CompactSVMModel = svmStruct.Trained{1};
%%
out_ent = predict(CompactSVMModel, H_all); % Clasificaci n con conjunto de entrenamiento
out_test = predict(CompactSVMModel, P_all); % Clasifiaci n con conj de test
[FPRe,TPRe,FNRe,TPe,FPe,FNe,TNe] = rendimiento( classE , out_ent);
[FPRt,TPRt,FNRt,TPp,FPp,FNp,TNp] = rendimiento( classP , out_test);
```

Listing 4: SVM NMF

```
load('HiTSall','imdata','clases')

%%
clear
addpath('Funciones')
load('HiTS2000')
% %%Extracci n de caracter sticas con NMF
k=10;% filas matriz de coeficientes
[H_all ,D1,D2,D3,W1,W2,W3] = makeNmf(entH1 ,entH2 ,entHs ,k,m_ent ,std_ent);
[P_all ,e1 ,e2 ,e3] = makeNmf_h(pH1 ,pH2 ,pHs ,W1,W2,W3,k, m_test , std_test);

%% Entrenamiento
svmStruct=fitcsvm(H_all ,classE ,'Holdout',0.15,'KernelFunction','rbf','BoxConstraint'
    ,0.1,'ClassName',[1 ,0],'KernelScale',0.5);

%% C lculo clase de salida
CompactSVMModel = svmStruct.Trained{1};

%%
out_nmf_ent = predict(CompactSVMModel, H_all); % Predicci n (clasificaci n) para
                                               % el conjunto de
```

```
21 out_nmf_test = predict(CompactSVMModel, P_all); % Predicci n (clasificaci n) para
22                                                  % el conjunto de
23 %% C lculo de FPR y TPR
24 [FPRe,TPRe,FNRe,TPe,FPe,FNe,TNe] = rendimiento( classE, out_nmf_ent );
25 [FPRt,TPRt,FNRt,TPp,FPp,FNp,TNp] = rendimiento( classP, out_nmf_test );
```

Listing 5: Random Forest mezcla

```
1  % Solo la componentes (todas)
2  H_join1=[H_all_pca_rfK20(:,1:66) H_all_nmf_rfK20(:,1:66)];
3  P_join1=[P_all_pca_rfK20(:,1:66) P_all_nmf_rfK20(:,1:66)];
4
5
6  % 5 primeras componentes
7  % H_join1=[H_all_pca_rfK20(:,1:5) H_all_pca_rfK20(:,11:15)...
8  %     H_all_pca_rfK20(:,21:25) H_all_nmf_rfK20(:,1:5)...
9  %     H_all_nmf_rfK20(:,11:15) H_all_nmf_rfK20(:,21:25)];
10 %
11 % P_join1=[P_all_pca_rfK20(:,1:5) P_all_pca_rfK20(:,11:15)...
12 %     P_all_pca_rfK20(:,21:25)  P_all_nmf_rfK20(:,1:5)...
13 %     P_all_nmf_rfK20(:,11:15) P_all_nmf_rfK20(:,21:25)];
14
15
16 % 5 primeras componentes + media y varianza
17 % H_join1=[H_all_pca_rfK20(:,1:5) H_all_pca_rfK20(:,11:15)...
18 %     H_all_pca_rfK20(:,21:25) H_all_nmf_rfK20(:,1:5)...
19 %     H_all_nmf_rfK20(:,11:15) H_all_nmf_rfK20(:,21:25) ...
20 %     H_all_pca_rfK20(:,31:36) H_all_nmf_rfK20(:,31:36)];
21 %
22 % P_join1=[P_all_pca_rfK20(:,1:5) P_all_pca_rfK20(:,11:15)...
23 %     P_all_pca_rfK20(:,21:25)  P_all_nmf_rfK20(:,1:5)...
24 %     P_all_nmf_rfK20(:,11:15) P_all_nmf_rfK20(:,21:25)...
25 %     P_all_pca_rfK20(:,31:36) P_all_nmf_rfK20(:,31:36)];
26
27 %  rbol
28 nTrees=100;
29 B=TreeBagger(nTrees,H_join1,classE,'Method','classification');
30 out_join_ent=B.predict(H_join1);
31 out_join_ent=str2double(out_join_ent);
32
33 out_join_test=B.predict(P_join1);
34 out_join_test=str2double(out_join_test);
35
36
37 % C lculo de FPR y TPR
38 [FPRe,TPRe,FNRe,TPe,FPe,FNe,TNe] = rendimiento( classE, out_join_ent );
39 [FPRt,TPRt,FNRt,TPp,FPp,FNp,TNp] = rendimiento( classP, out_join_test );
```