UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

ALGORITMOS DE DETECCIÓN DE ESTRELLAS VARIABLES EN IMÁGENES ASTRONÓMICAS BASADAS EN FACTORIZACIÓN NO NEGATIVA DE MATRICES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

EMANUEL ANTONIO BERROCAL ZAPATA

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
PABLO HUIJSE HEISE
JAIME ORTEGA PALMA

SANTIAGO DE CHILE
2015

A mi madre, familia y amigos.

Resumen

En este trabajo se presenta una metodología para de detectar estrellas variables en estampillas de 21 x 21 píxels provenientes de imágenes astronómicas a partir de un pipeline de procesamiento de imágenes para el telescopio Dark Energy Camera (DECam). Para aquello se realizará un aprendizaje con una muestra de imágenes que están etiquetadas como estrellas y no estrellas con el objetivo de detectar nuevas estrellas variables en imágenes desconocidas.

Los objetos astronómicos que se observan en las imágenes, pueden ser agrupados en las categorías de: estrellas variables, estrellas fijas, rayos cósmicos, fuentes puntuales falsas, malas restas, supernovas y objetos desconocidos.

Para la labor de detectar estrellas se ocuparon 2 algoritmos basados en *NMF* (Non-negative matrix factorization) y un tercer algoritmo basado en Análisis de Componentes Principales (*PCA*). El enfoque del primer algoritmo *NMF* es la rapidez, mientras que el segundo se enfoca en la dispersión de información.

Para el uso de los algoritmos anteriores se generó una metodología con la que trabajarán, los cuales ocupan técnicas de optimización no lineal y descomposición de matrices; finalmente se demuestra cual es el mejor de ellos para detectar estrellas gracias a una herramienta llamada *Curva ROC*.

Una vez escogido el mejor método se realiza un estudio de sensibilidad de parámetros para mejorar la detección de estrellas y obtener una representación más genérica a través de la descomposición de matrices.

El resultado final es que utilizando uno de los algoritmos basados en *NMF*, se obtienen los mejores resultados de clasificación, esto se concluye del análisis de la *Curva ROC*.

Tabla de contenido

Capítulo 1

Introducción

En la actualidad es importante tener algoritmos que puedan clasificar cantidades enormes de imágenes debido a los grandes volúmenes de datos existentes. Análisis de Componentes Principales (del inglés *PCA*) ha sido típicamente usado para reducir la dimensionalidad de los datos y así hacer más tratable los problemas. En esta memoria se presentan 2 algoritmos de descomposición de matrices basados en *NMF* [1], los cuales competirán con *PCA* para detectar de manera más eficiente y acertada estrellas variables a partir de datos procesados por el pipeline de procesamiento de imágenes del telescopio DECam (del inglés Dark Energy Camera)[2].

Las imágenes a estudiar son estampillas de 21 x 21 píxels obtenidas del telescopio DECam; esté cuenta con 60 CCD's (del inglés Charge Coupled Device) donde cada CCD es de 4000 x 8000 píxels. DECam obtiene varias imágenes de la misma parte del espacio a tiempos distintos, a continuación se buscan objetos de interés en las imágenes y estos son guardadas en tamaños de 21 x 21 píxels, que es lo que llamamos estampillas. El objeto de estudio en esta memoria serán estas estampillas. Inicialmente se cuenta con dos tipos de estampillas que presentan la particularidad de mostrar la mayor variabilidad en el objeto astronómico a estudiar, luego con estás dos se logra obtener otras 3 más, teniendo finalmente 5 tipos de imágenes (estampillas) referidas al mismo objeto astronómico[2]. Para el trabajo realizado en esta memoria se tienen en total 100000 ejemplares que cuentan con estas 5 estampillas cada uno.

Los algoritmos basados en *NMF* han llegado a ser técnicas prominentes para separación de datos para fuentes ciegas, análisis de bases de datos en imágenes, minería de datos, recuperación de información y aplicaciones en agrupaciones (clustering)[3]. Son muy útiles en separación de datos que poseen escasez de información, es decir, que no se puede ver muy fácilmente distintas clases que se podrían formar dentro de esos datos, en un modo de separación[3].

El objetivo general de esta memoria es entregar una metodología de clasificación de imágenes etiquetadas como estrellas y no-estrellas bajo un aprendizaje semi-supervisado, i.e., se cuenta con una cantidad inicial de imágenes etiquetadas para obtener características que permitirán clasificar imágenes no-etiquetadas.

El objetivo específico de esta memoria, es obtener un error de clasificación menor al 3 % con una tasa de falsos positivos menor al 1 %.

Para lograr los objetivos, esta memoria está organizada de la siguiente forma: En el capítulo 2 se establece el marco teórico acerca de todas las herramientas matemáticas que se ocuparán, dentro de las cuales se encuentra la descomposición de matrices, específicamente *PCA* y algoritmos basados en *NMF*, máxima interacción grupal y maquinas de soporte vectorial. Además se presenta una herramienta que establece criterios para clasificar entre los distintos modelos probados, para saber cuál es más adecuado en la búsqueda de estrellas. En particular en la sección 2.2.6 se muestra un algoritmo nuevo que ocupa *NMF* y que tendrá una restricción para controlar la dispersión de información por parte de las características que describen a las imágenes.

En el capítulo 3 vemos la manera secuencial de ocupar las herramientas descritas en el capítulo 2 para finalmente clasificar estrellas y no-estrellas en imágenes no-etiquetadas, son presentados varios modelos distintos para clasificar y la manera de utilizarlos. Además en este capítulo se ve el pre-procesamiento que deben recibir las imágenes, además se muestran las transformaciones que se utilizaran en las imágenes para ampliar la base de entrenamiento. También se muestra una forma de ajustar parámetros para las herramientas utilizadas.

Luego en el capítulo 4 se presentan los resultados obtenidos luego de haber usado los 3 métodos, dando a conocer cual es el que mejor clasifica en base a la tasa de falsos positivos y tasa de verdaderos positivos. Además se muestran resultados de haber realizado ajustes finos de parámetros para el método escogido y mostrando como poder mejorar aún más los resultados de clasificación.

Para cerrar, se presentan las conclusiones del trabajo hecho, dando a conocer el método más conveniente para el objetivo específico y se plantea el trabajo a futuro.

Capítulo 2

Antecedentes

En este capítulo vemos el marco teórico bajo el que se desarrolla este trabajo, describiendo los algoritmos usados de manera teórica, además de las herramientas matemáticas a ocupar.

Se verán técnicas basadas en descomposición de matrices, específicamente *NMF* (Factorización no-negativa de matrices), *PCA* (Principal Component Analisys), un clasificador basado en optimización no lineal, llamado *SVM* (Maquinas de Soporte Vectorial) y finalmente una herramienta para seleccionar modelos de clasificación óptimos, llamada *Curva ROC*.

2.1. Descomposición de matrices

Dada la gran dimensionalidad de las imágenes, en el análisis y la extracción de características de estudio, es necesario reducir estás para poder entender de manera precisa solo la información relevante que se estudia. En esta sección se presentan 2 técnicas que tratan con la descomposición de los datos originales en un nuevo conjunto, sin necesidad de perder información relevante y sacando a luz la información latente, estas dos técnicas son Factorización No negativa de matrices (*NMF*) y Análisis de Componentes Principales (*PCA*).

2.2. *Non-negative matrix factorization* (*NMF*)

Un problema para los mecanismos actuales en análisis de datos consiste en encontrar una representación adecuada de estos. Una manera útil de representar los datos será aquella que permita reducir las dimensiones de los datos a la vez que muestre ciertas características de ellos que permanecen ocultas a priori; así surge la Factorización No Negativa de Matrices(*NMF*) [1] [4] cuya principal utilidad consiste en encontrar una representación

lineal de los datos, que han de ser no negativos.

Una de las propiedades principales suele ser que proporciona una representación dispersa de los datos que permite codificarlos de forma que el resultado sea bastante sencillo de interpretar.

A diferencia del método *PCA* que se verá luego, usualmente con *NMF* [3] se mejora la interpretación y visualización de los datos sin necesidad de perder sentido físico, además que en comparación con otros métodos de descomposición, es que muchos descomponen la matriz inicial en una serie de matrices de cualquier signo que lleva a una interpretación poco intuitiva y difícil.

Gracias a la no negatividad, se obtiene una representación de los datos a partir de combinaciones aditivas y nunca substractivas [5], lo que se traduce en que cada una de las partes que conforman la suma pueda ser considerado como partes de los datos originales. Así pues, *NMF* permite reducir las dimensiones de los datos a la vez que hace visibles ciertas características de los mismos que en un principio no podían ser observadas.

El objetivo de *NMF* es estimar la matriz que se tiene inicialmente descomponiéndola en la multiplicación de dos matrices, es una técnica de aprendizaje para datos locales de manera no supervisada.

Definición 2.1 *[*NMF*] El modelo en que se basa* NMF *viene dada por la siguiente ecuación:*

$$V = WH + E = AB^T + E, \qquad V \approx \bar{V} = WH \tag{2.1}$$

Donde $V = [v_{ik}] \in \mathbb{R}_+^{M\times P}$ es una matriz de datos conocida, $W = [w_1, w_2, \ldots, w_J] \in \mathbb{R}_+^{M\times K}$ es una matriz que será una mezcla de los datos de la matriz V la que será llamada matriz diccionario y la matriz $H \in \mathbb{R}_+^{K\times P}$ es la de coeficientes.

Finalmente la matriz $\mathrm{E} = [\mathrm{e_{ik}}] \in \mathbb{R}^{\mathrm{M\times P}}$, será la que guarde los errores o ruidos. En el caso de obtenerse una descomposición exacta se tiene $E = 0$.

Las matrices W y H tienen diferente sentido físico en diferentes aplicaciones, por ejemplo en los problemas de clustering (agrupaciones) W representa la matriz base mientras que H denota la matriz de pesos. Otro ejemplo es en separación ciega de fuentes (Blind Signal Separation, BSS) donde W tiene el papel de matriz de mezcla mientras que H representa las fuentes y por último en el análisis acústico, W representa los parámetros base, siendo cada columna de H las posiciones en las que el sonido está activo (ref. [3][6]).

Por otro lado el problema de la ecuación (2.1) podemos expresarlo a través de la suma de los productos entre los vectores:

$$V = \sum_{\mathrm{i}=1}^{K} W_{\mathrm{i}} \cdot H_{\mathrm{i}} + E \tag{2.2}$$

Con W_{i} columna i-ésima de W y H_{i} fila i-ésima de H, en la figura 2.1 podemos ver gráficamente esta ecuación.

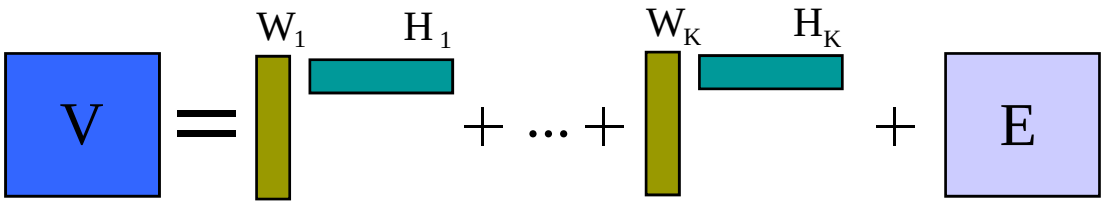


Figura 2.1: Modelo básico *NMF* donde la matriz V es representada como una combinación lineal de matrices no negativas de rango 1 más un error desconocido.

El objetivo de *NMF* es estimar $\bar{V} \in \mathbb{R}^{M \times P}$ y descomponerla en una matriz de dimensión menor, para ello por lo general se fija la dimensión K mas pequeño entre M y P.

Casos Particulares de *NMF*:

Basado en la ecuación (2.1) se pueden definir diferentes casos particulares del modelo básico según las propiedades que presenten las diferentes matrices implicadas, algunos de estos casos son:

- ***NMF* simétrica[6]**: En el caso en que $W = H^t$, se obtiene la ecuación:

  $$V = WW^t + E$$

  Y en caso de existir simetría completa ($E = 0$) se dice que V es completamente positiva y el menor número de columnas de W que satisfacen $V = WW^t$ se denomina rango de la matriz V.

- ***NMF* semi ortogonal[6]**: En este caso se cumple la misma ecuación (2.1) salvo que alguna de las matrices W o H cumplen con la restricción:

  $$W^tW = I \qquad \text{o} \qquad HH^t = I$$

- ***NMF* multicapa[6]**: En este modelo se tiene que la matriz W se reemplaza por un conjunto de matrices (factores) dispuestas en cascada.
  Puede describirse como:

  $$W = W^{(1)}W^{(2)} \cdots W^{(L)}H + E.$$

  Dado que este modelo es lineal, todo podría reagruparse en la única matriz W y volver al modelo básico, sin embargo, la estructura distribuida de este modo de descomposición sirve para mejorar las prestaciones de algunos algoritmos *NMF* y mejorar el problema de los mínimos locales.

2.2.1. Buscando la mejor solución

Para obtener las matrices W y H en la *NMF* básica se necesita una medida acorde para ver la diferencia entre la matriz original V y la aproximación obtenida por el modelo $\bar{V} = WH$.

Así entonces para obtener la descomposición de la ecuación (2.1) se resuelve:

$$\min_{W,H} L(V||WH) \text{ s.a. } W \geq 0, H \geq 0, \tag{2.3}$$

Donde $L(\cdot||\cdot)$ es una función de costo o una métrica de distancia entre matrices y las restricciones son componente por componente.

Las funciones objetivos más comunes usadas en la literatura de *NMF* son la norma de Frobenius [7], información de divergencia [8] y divergencia de Kullback-Leibler[1][9][10]. En los cálculos que vienen se tomará la norma de Frobenius ya que es la más simple y la menos dificultosa de calcular:

$$D_F(V||WH) = \frac{1}{2}\|V - WH\|_F^2 \tag{2.4}$$

$$= \frac{1}{2}\sum_{ij}(V_{ij} - [WH]_{ij})^2. \tag{2.5}$$

La restricción no-negativa es también responsable de la dispersión intrínseca que es usualmente encontrada en las representaciones de *NMF*.
Notar además que no existe un control sobre la dispersión en la ecuación (2.3), por lo que más adelante se verá la modificación realizada a esa ecuación para agregarle un coeficiente que influencie en la dispersión.

El problema de la ecuación (2.3) es convexo con respecto a W o bien H (pero no para ambos), puede tener varios mínimos locales en conjunto (W, H). Además ha sido probado que es un problema NP-complejo[11].

Un enfoque común para resolver *NMF* es usar mínimos cuadrados no-negativos alternados (Alternating Nonnegative Least Squares(*ANLS*))[12][13], en el cual la ecuación (2.3) es dividida en dos sub-problemas los cuales serán mas sencillos de tratar:

1) **Problema 1: Aprendizaje del Diccionario(W)**

$$\min_{W\geq 0} \frac{1}{2}\|H^T W^T - V^T\|_F^2 \quad \text{Con } H \text{ fijo.} \tag{2.6}$$

2) **Problema 2: Matriz de Coeficientes(H)**

$$\min_{H\geq 0} \frac{1}{2}\|V - WH\|_F^2 \quad \text{Con } W \text{ fijo.} \tag{2.7}$$

En el problema 1 los coeficientes (H) son fijos, mientras que en el problema 2 el diccionario está fijo, ambos problemas como se mencionó antes son convexos, por lo tanto las soluciones óptimas pueden ser encontradas para cada uno de ellos.

En la mayoría de los métodos los problemas 1 y 2 son resueltos de manera iterativa hasta que un criterio de parada conocido es alcanzado.

Observar que no hay garantía que se alcance un mínimo global de la ecuación (2.3) ocupando *ANLS* y cuyo funcionamiento es descrito a continuación:

Algoritmo 2.2 *[*ANLS*]*

1. *Se inicializa W de forma aleatoria o mediante alguna estrategia determinista. Una mala elección de la matriz inicial da lugar a lentitud en la convergencia.*

2. *Se estima H de la ecuación $W^t \cdot V = W^t \cdot W \cdot H$ de forma que se resuelve:*

$$\min_H D_F(V||WH) = \frac{1}{2}||V - WH||_F^2, \quad \textit{con } W \textit{ fija}$$

3. *Se impone un valor ε próximo a cero (típicamente 10^{-16}) a todos los elementos de la matriz H que resultaron ser menores que cero.*

4. *Se estima la matriz W de la ecuación $H \cdot V^t = H \cdot H^t \cdot W^t$ de forma que se resuelve:*

$$\min_W D_F(V||WH) = \frac{1}{2}||V^t - H^t W^t||_F^2, \quad \textit{con } H \textit{ fija}$$

5. *Se impone un valor ε proximo a cero (típicamente 10^{-16}) a todos los elementos de la matriz W que resultaron ser menores que cero.*

Observación Este algoritmo no garantiza la convergencia hacia un mínimo global o hacia un punto estacionario, sino que garantiza que la función de costo deja de decrecer [14] [15]. De manera resumida se tiene que *ANLS* se puede escribir como:

$$H \leftarrow \max\left\{\varepsilon, (W^tW)^{-1}W^tV\right\} := \left[(W^tW)^{-1}W^tV\right]_+ \tag{2.8}$$

$$W \leftarrow \max\left\{\varepsilon, VH^t(HH^t)^{-1}\right\} := \left[VH^t(HH^t)^{-1}\right]_+ \tag{2.9}$$

Además en la literatura de *NMF* a *ANLS* se le suele llamar también *ALS*[1] (Alternating Least Squares), pues la manera de utilizarlo en NMF es que condiciona a las matrices resultantes a que sean no-negativas, en los pasos (3) y (5) de *ANLS* se impone la no-negatividad.

Para resolver los pasos (2) y (5) de *ANLS* se ha utilizado o bien gradiente descendente o bien enfoques basado en reglas multiplicativas. Las reglas multiplicativas [1] [9] [10] [16] fueran propuestas como una alternativa a los métodos de gradiente descendente (que corresponden a reglas aditivas).

Las reglas multiplicativas para la función de costo Frobenius son las siguiente:

Para la matriz de coeficientes H:

$$H_{ij} \leftarrow H_{ij}\frac{(W^TV)_{ij}}{(W^TWH)_{ij}}, \tag{2.10}$$

Para la matriz de diccionario W:

$$W_{ij} \leftarrow W_{ij}\frac{(VH^T)_{ij}}{(WH^TH)_{ij}}, \tag{2.11}$$

[1]De acá en adelante a *ANLS* solo se mencionará como *ALS*

Además tenemos que la ecuación (2.10) es equivalente a:

$$H_{ij}^{t+1} = H_{ij}^{t} - \eta_{ij} \nabla_H D_F(V||WH), \tag{2.12}$$

donde:

$$\eta_{ij} = \frac{H_{ij}^{t}}{(W^T W H)_{ij}^{t}}, \tag{2.13}$$

De manera análoga se obtiene la regla de actualización para la ecuación (2.11), por lo que las reglas multiplicativas son un caso especial de gradiente descendente con un paso de tamaño adaptivo. Además en [16] se demuestra que con correcciones menores cualquier punto límite dado para las reglas multiplicativas es un punto estacionario en la ecuación (2.3).

Otra forma exitosa de resolver los 2 sub-problemas anteriores (Problemas 1 y 2) son a través de algoritmos de punto interior [17], gradiente proyectado [12] y métodos de conjunto activo [13]. Todos son más eficientes y convergen mas rápido que las reglas multiplicativas bajo el supuesto que $K < \text{mín}(M, P)$.

2.2.2. Criterio de Detención

Existe mucha variedad en los criterios de parada para los algoritmos iterativos empleados por *NMF*, algunos de estos son:

- La función de coste llega a un valor próximo a cero o por debajo de un umbral establecido ε:
$$D_F(V||\bar{V}^{(j)}) = ||V - \bar{V}^{(j)}||_F^2 \leq \varepsilon$$
- No se producen cambios (o son muy pequeños) en las matrices W y H.
- No se consigue mejora en la función de coste (o son muy pequeños) entre las varias iteraciones sucesivas:
$$\frac{|D_F^{(j)} - D_F^{(j-1)}|}{D_F^{(j)}} \leq \varepsilon$$
- El número de iteraciones empleadas supera el límite pre-establecido.

O también se pueden escoger combinaciones de criterios.

2.2.3. Reconstrucción de imágenes

En esta sección se revisará la aplicación particular de reconstrucción de imágenes a partir de las matrices W y H, la cual tiene directa relación con el trabajo realizado en esta memoria, recordemos que la idea principal de *NMF* es $W \cdot H \approx V$.

En este caso la matriz V representa un conjunto de muestras en donde cada columna es una imagen que contiene sus píxels, en este trabajo el tamaño de la imagen es de 21 x 21 píxels, pero al dejarlo como una sola columna, estos quedan reordenados en 441 píxels, dejando la primera fila de 21 píxels ocupar los primeros 21 espacios en la columna, luego la segunda fila ocupar los siguientes 21 espacios y así hasta llegar a la fila 21, i.e., completando los 441 píxels.

Se puede observar en la figura 2.2, que en la fila superior aparecen los coeficientes de reconstrucción $h_1, \ldots, h_K$ (que corresponde a las columnas de H), y las líneas que unen los $h_j's$ con los $v_j's$ corresponden a los pesos, en este caso a la matriz W; así, finalmente al multiplicar estos $h_j's$ con los $w_{\mathrm{i}j}'s$ se obtiene $\sum_{j=1}^{k} W_{\mathrm{i}j}h_j = v_{\mathrm{i}}$ en la fila inferior, que son las imágenes perteneciente a la matriz V.

Podemos ver que los elementos de la columna $W_{\mathrm{i}j}$ determinan la influencia que la fila de coeficientes de codificación h_j, tiene en la reconstrucción de la i-ésima imagen v_{i}. Como se puede comprobar en la figura, cada h_j tiene repercusión en varios píxeles de la imagen debido a las múltiples conexiones entre los h_j y los v_{i}. Debido a la no negatividad de los elementos $W_{\mathrm{i}j}$, esta influencia será siempre de carácter aditivo en cada píxel por lo que el algoritmo *NMF* controlará las aportaciones adecuadas a una correcta reconstrucción de la imagen.

Para lo que viene se definirá un nuevo algoritmo que es una extensión de *ALS* basado en *NMF*, el cual es más rápido en convergencia y mas eficiente en la obtención de las matrices W y H, este algoritmo se llama Mínimos cuadrados Alternativos Jerarquicos (Hierarchical Alternating Least Squares (*HALS*)[3]).

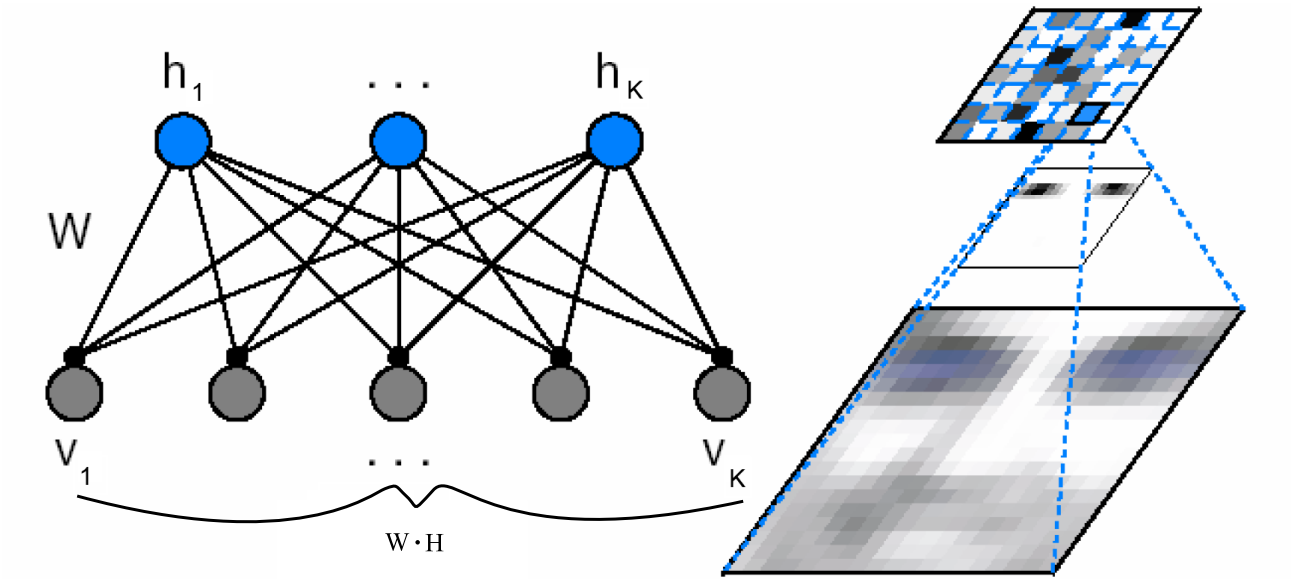


Figura 2.2: Esquema de como se reconstruye una imagen basada en *NMF*. En el diagrama de esta figura, la influencia de h_j sobre v_{i} la determinan las conexiones cuyos pesos vienen dados por los elementos de $W_{\mathrm{i}j}$. Se consideran como 'variables visibles' los v_{i} que son los píxeles de la imagen, mientras que las 'variables ocultas' serán los coeficientes de codificación (los $h_j's$).

2.2.4. Hierarchical Alternating Least Squares (*HALS*)

Denotando las columnas de $W = [w_1, w_2, \ldots, w_K]$ y las filas de $H = [h_1; h_2; \ldots; h_K]$, se puede expresar la ecuación de costo (2.4) como:

$$D_F(V||a_1, \ldots, a_K, h_1, \ldots, h_K) = \frac{1}{2}||V - WH||_F^2 \tag{2.14}$$

$$= \frac{1}{2}||V - \sum_{j=1}^{K} w_j h_j||_F^2 \tag{2.15}$$

La idea básica de *HALS* [3] es definir residuos:

$$V^{(j)} = w_j h_j \tag{2.16}$$

$$= V - \sum_{p \neq j} w_p h_p = V - WH + w_j h_j \tag{2.17}$$

$$= V - WH + w_{j-1}h_{j-1} - w_{j-1}h_{j-1} + w_j h_j \qquad \text{para } j = 1, 2, \ldots, K \tag{2.18}$$

y minimizar alternativamente el conjunto de funciones de costo (con respecto al conjunto de parámetros $\{w_j\}$ y $\{h_j\}$), creando nuevas funciones de costos:

$$D_W^{(j)} = \frac{1}{2}||V^{(j)} - w_j h_j||_F^2, \text{ para } h_j \text{ fijo}$$

$$D_H^{(j)} = \frac{1}{2}||V^{(j)} - w_j h_j||_F^2, \text{ para } w_j \text{ fijo}$$

para $j = 1, 2, \ldots, K$ sujeto a $w_j \geq 0$ y $h_j \geq 0$, respectivamente. En otra palabras, se minimiza alternativamente el conjunto de funciones de costos:

$$D_F^{(j)}(V^{(j)}||w_j h_j) = \frac{1}{2}||V^{(j)} - w_j h_j||_F^2 \tag{2.19}$$

Para $j = 1, \ldots, K$ sujeto a $w_j \geq 0$ y $h_j \geq 0$ respectivamente.

Los gradientes de las funciones de costo de la ecuación (2.19) con respecto a los vectores desconocidos w_j y h_j son:

$$\frac{\partial D_F^{(j)}(V^{(j)}||w_j h_j)}{\partial w_j} = w_j h_j^t h_j - V^{(j)} h_j,$$

$$\frac{\partial D_F^{(j)}(V^{(j)}||w_j h_j)}{\partial h_j} = h_j^t w_j^t w_j - V^{(j)t} w_j.$$

Igualando las componentes de los gradientes a cero y asumiendo que se fuerza a cumplir las restricciones no negativas con la tercera condición impuesta como en el algoritmo *ALS* (2.2) con respecto al ε, se obtienen las ecuaciones de actualización para w_j y h_j y con esto se tiene *HALS*.

Algoritmo 2.3 *[*HALS*]*
Denotando las columnas de $W = [w_1, \ldots, w_K]$ *y las filas de* $H = [h_1; \ldots; h_K] = [h_1^t, \ldots, h_K^t]^t$, HALS *es el algoritmo cuyas reglas de actualización son:*

$$h_j \leftarrow \frac{1}{w_j^t w_j}[w_j^t V^{(j)}]_+, \quad w_j \leftarrow \frac{1}{h_j^t h_j}[V^{(j)} h_j^t]_+ \tag{2.20}$$

para $j = 1, \ldots, K$

Estas mismas reglas de actualización para *NMF* han sido propuestas o redescubiertas independientemente en [18][19][20]. En la ecuación (2.18) se sigue que no se necesita calcular explícitamente la matriz de residuos $V^{(j)}$ en cada paso iterativo solo basta hacerlo de manera mas inteligente [21], pues todos los $V^{(j)}$ pueden ser actualizados simultáneamente.

Ya definido el algoritmo *HALS*, se presentan dos algoritmos basados en *HALS* que son utilizados en esta memoria:

1. *HALS* rápido para *NMF* (***FAST HALS for NMF***).
2. *HALS NMF* con Control de Dispersión (***HALS NMF Sparsity***).

2.2.5. *FAST HALS for NMF*

Este algoritmo mejora la tasa de convergencia y el rendimiento de *HALS*, ya que calcula de manera simultánea las columnas de W, en vez de una por una (ver también [22]).
Por ejemplo, minimizando el conjunto de funciones (2.4) con respecto a h_j y simultáneamente la función (2.19) con normalización de las columnas w_j de largo unitario con $||\cdot||_2$, se genera un algoritmo muy eficiente de aprendizaje ocupando *NMF* en el cual los vectores individuales de H son actualizados localmente (fila por fila) y la matriz W es actualizada globalmente usando *ALS* (todas las columnas w_j simultáneamente).

De la ecuación (2.20), se obtienen la siguiente regla de actualización para h_j:

$$\begin{aligned} h_j &\leftarrow \frac{w_j^t V^{(j)}}{w_j^t w_j} = \frac{w_j^t (V - WH + w_j h_j)}{w_j^t w_j} \\ &= \frac{(w_j^t V - w_j^t WH + w_j^t w_j h_j)}{w_j^t w_j} \\ &= \frac{([W^t V]_j - [W^t W]_j H + w_j^t w_j h_j)}{w_j^t w_j} \end{aligned} \tag{2.21}$$

con $h_j \leftarrow [h_j]_+$ y dado que $||w_j||_2 = 1$, la regla de aprendizaje para h_j queda simplificada a la forma:

$$h_j \leftarrow \left[h_j + [W^t V]_j - [W^t W]_j H\right]_+$$

Y luego de manera análoga a la ecuación (2.21) se obtiene la regla de aprendizaje para w_j, así entonces obtenemos:

Algoritmo 2.4 *[*FAST HALS for NMF*]*
Denotando las columnas de $W = [w_1, \ldots, w_K]$ *y las filas de* $H = [h_1; \ldots; h_K] = [h_1^t, \ldots, h_K^t]^t$, FAST HALS for NMF *es el algoritmo cuyas reglas de actualización son:*

$$\begin{aligned} h_j &\leftarrow \left[h_j + [W^t V]_j - [W^t W]_j H\right]_+ \\ w_j &\leftarrow \left[w_j h_j h_j^t + [V H^t]_j - W[H H^t]_j\right]_+ \\ w_j &\leftarrow \frac{w_j}{||w_j||_2} \end{aligned}$$

para $j = 1, \ldots, K$.

De lo anterior se deduce el pseudo-código del algoritmo *FAST HALS for NMF*, el cual se observa en el *algoritmo* 1.

Algoritmo 1 : ***FAST HALS for NMF***, Para $\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H}$

```
Inicializar las matrices no-negativas W y/o H usando ALS.
Normalizar las columnas de W(W_j) y las filas de H(H_j), bajo ||·||_2.
repetir
  % Actualizar H;
  A = V^t W;
  B = W^t W;
  for j = 1 hasta K hacer:
    H_j ⇐ [H_j + A_j − H · B_j]_+
  end for
  % Actualizar W
  P = V · H;
  Q = V^t · V;
  for j = 1 hasta J hacer:
    W_j ⇐ [W_j Q_jj + P_j − W · Q_j]_+
    W_j ⇐ W_j / ||W_j||_2;
  end for
hasta alcanzar el criterio de detención
```

2.2.6. *HALS NMF Sparsity*

En este algoritmo, se busca controlar la dispersión de los datos. Al agregarle un termino que controla la dispersión de los datos h_j a la ecuación (2.19), se obtiene la función de costo para *HALS NMF Sparsity*:

$$D_F^{(j)}(Y^{(j)}||w_j h_j) = \frac{1}{2}||V^{(j)} - w_j h_j||_F^2 + \alpha_{sp}||h_j||_1. \tag{2.22}$$

Para $j = 1, \ldots, K$ sujeto a $w_j \geq 0$ y $h_j \geq 0$, donde $\alpha_{sp} > 0$ es un parámetro que controla el nivel de dispersión para la matriz H.

Dado que la dependencia de la ecuación (2.22) con respecto a w_j sigue siendo la misma que en la ecuación (2.19) y dado que *HALS NMF Sparsity* no busca actualizar todos los vectores w_j de manera simultanea se tiene que la misma regla de actualización que en (2.20) para w_j, luego para h_j es:

$$\begin{aligned} h_j &\leftarrow \frac{\left[w_j^t V^{(j)} - \alpha_{sp} 1_K\right]_+}{w_j^t w_j}, \qquad \text{e imponiendo que } ||w_j||_2 = 1, \text{ se obtiene} \\ &= \left[w_j^t V^{(j)} - \alpha_{sp} 1_K\right]_+ \end{aligned}$$

Así entonces obtenemos:

Algoritmo 2.5 *[Algoritmo* HALS NMF Sparsity*]*
Denotando las columnas de $W = [w_1, \ldots, w_K]$ *y las filas de* $H = [h_1; \ldots; h_K] = [h_1^t, \ldots, h_K^t]^t$, HALS NMF Sparsity *es el algoritmo cuyas reglas de actualización son:*

$$w_j \leftarrow \frac{1}{h_j^t h_j}[V^{(j)} h_j^t]_+ \tag{2.23}$$

$$w_j \leftarrow \frac{w_j}{||w_j||_2} \tag{2.24}$$

$$h_j \leftarrow \left[w_j^t V^{(j)} - \alpha_{sp} \mathbb{1}_K\right]_+ \tag{2.25}$$

para $j = 1, \ldots, K$ *con* $\mathbb{1}_K$ *vector fila de* K *unos.*

De lo anterior se deduce el pseudo-código del algoritmo *HALS NMF Sparsity*, el cual se observa en el *algoritmo* 2.

Algoritmo 2 : ***HALS NMF Sparsity***, Para $\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H}$

```
Inicializar las matrices no-negativas W y/o H usando ALS.
Normalizar las columnas de W(W_j) y las filas de H(H_j), bajo ||·||_2.
E = V − W;
repetir
  % Actualizar H, W e Y;
  for j = 1 hasta K hacer:
     Y ⇐ E + W_j H_j;
     H_j ⇐ [Y^t W_j − α_sp · 1_K]_+ ;
     W_j ⇐ [Y H_j^t]_+ ;
     W_j ⇐ W_j/(H_j H_j^t);
      W_j ⇐ W_j/||W_j||;
     E ⇐ Y − W_j H_j;
  end for
hasta alcanzar el criterio de detención
```

En cuanto a escoger el parámetro α_{sp} se dice en [6] que típicamente se escoge entre $0{,}1$ y $0{,}5$.

En el Listing 2.1 se tiene la implementación del Algoritmo *HALS NMF Sparsity* en `Matlab` con criterio de convergencia basado en el número de iteraciones, el cual recibe 3 parámetros de entrada:

`V, max_iter, K,` α_{sp}

con:
`V`: matriz de dimensión $M \times P$ normalizada, la cual contiene las imágenes,
`max_iter`: número máximo de iteraciones para el criterio de convergencia,
`K`: el número de representantes obtenidos o prototipos, que representan al conjunto de imágenes.
α_{sp}: control de dispersión.

La salida de esta implementación es:

`W, H`

con:
`W`: matriz diccionario de dimensión $M \times K$,
`H`: matriz de coeficientes de dimensiones $K \times P$,

El algoritmo inicializa los parámetros (líneas 2-5), implementa *ALS* (líneas 6-7), normaliza columnas y filas de `W` y `H` respectivamente, realiza una primera descomposición *NMF* (línea 10), itera y calcula matrices W y H basado en *NMF* control de dispersión (sparsity) (líneas 12-19 y 23), en particular se va obteniendo $\mathtt{V}^{(j)}$ (línea 13) y finalmente verifica el criterio de detención (líneas 20-21)

Listing 2.1: Implementación de Algoritmo *HALS NMF Sparsity*

```
1  function [W,H,V] = hnmf(V,max_iter,K)
2      eps = 1e-16;
3      iter = 2;
4      alpha_sp = 0.1;
5      W = rand(size(V,1),K);
6      H = max((W'*W)\(W'*V),eps);
7      W = max((V*H')/(H*H'),eps);
8      W = normc(W);
9      H = normr(H);
10     E = V - W*H;
11     while(1)
12         for j=1:size(H)
13             Y = E + W(:,j)*H(j,:);
14             H(j,:) = max(Y'*W(:,j) - alpha_sp*ones(size(H(j,:)')),eps)';
15             W(:,j) = max( Y*H(j,:)',eps);
16             W(:,j) = W(:,j)/(H(j,:)*H(j,:)');
17             W(:,j) = W(:,j)/norm(W(:,j));
18             E = Y - W(:,j)*H(j,:);
19         end
20             if(iter >= max_iter )
21                 break;
22             end
23         iter = iter +1;
24     end
25 end
```

Observación En cuanto a la diferencia entre ambos algoritmos, teóricamente se tiene que la rapidez de cálculo por parte de ***FAST HALS for NMF*** es superior a ***HALS NMF Sparsity***[3], pues como se vio anteriormente este calcula de manera simultánea todas las columnas de W y *HALS NMF Sparsity* debe actualizar de manera secuencial cada columna de W y fila de H.

Otro elemento a notar es que a diferencia de ***HALS NMF Sparsity***, ***FAST HALS for NMF*** no tiene un control sobre la dispersión de los datos lo que lo produce que no capture tanta invarianza como los hace ***HALS NMF Sparsity***, esto se verá claramente reflejado en los prototipos obtenidos por cada método.

2.3. Análisis de Componentes Principales (PCA)

Cuando se quiere estudiar las relaciones que se presentan entre p variables linealmente correlacionadas (que poseen información común) se puede transformar el conjunto original de variables en otro conjunto de nuevas variables incorreladas entre sí (i.e., que no tengan repetición o redundancia de la información) llamado conjunto de componentes principales.

Las nuevas variables son combinaciones lineales de las anteriores y se van construyendo según el orden de importancia en cuanto a la variabilidad total que recogen de la muestra. De modo ideal, se busca $m < p$ variables que sean combinaciones lineales de las p originales y que estén incorreladas, recogiendo la mayor parte de la varianza de los datos. En sí, *PCA* consigue obtener una combinación lineal de las variables originales a través de la rotación de los ejes de coordenadas (ref. [23][24]).

Si las variables originales están incorreladas de partida, entonces no tiene sentido realizar un análisis de componentes principales. Y es importante resaltar el hecho de que el concepto de mayor información se relaciona con el de mayor variabilidad o varianza solo en el caso lineal y Gaussiano. Cuanto mayor sea la variabilidad de los datos (varianza) se considera que existe mayor información.

2.3.1. Cálculo de los Componentes Principales

Se considera un conjunto de variables $(x_1,\, x_2, \ldots,\, x_p)$ sobre un grupo de objetos o individuos y se trata de calcular, a partir de ellas, un nuevo conjunto de variables $y_1, y_2, \ldots, y_p$ incorreladas entre sí, cuyas varianzas vayan decreciendo progresivamente.

Cada y_j (donde $j = 1, \ldots, p$) es una combinación lineal de las $x_1, x_2, \ldots, x_p$ originales, es decir:

$$\begin{aligned} y_j &= a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jp}x_p = a_j^t \cdot x \\ \text{con:}\quad a_j^t &= = (a_{1j}, a_{2j}, \ldots, a_{pj}), \end{aligned}$$

un vector de constantes. Y $x^t = (x_1, \ldots, x_p)$, dado que se quiere maximizar la varianza, se puede aumentar los coeficientes a_{ij}, entonces para mantener la ortogonalidad de la transformación se impone que el módulo del vector a_j, sea 1, es decir:

$$a_j^t \cdot a_j = \sum_{k=1}^{p} a_{kj}^2 = 1.$$

El primer componente se calcula eligiendo a_1 de modo que y_1 tenga la mayor varianza posible, sujeta a la restricción de que $a_1^t \cdot a_1 = 1$. El segundo componente principal se calcula obteniendo a_2 de modo que la variable obtenida, y_2 esté incorrelada con y_1. De manera análoga se eligen $y_1, y_2, \ldots, y_p$, independientes entre sí, de manera que las variables aleatorias obtenidas vayan teniendo cada vez menor varianza.

2.3.2. Extracción de los factores

Para elegir a_1 de modo que se maximice la varianza de y_1 sujeta a que $a_1^t \cdot a_1 = 1$.

$$Var(y_1) = Var(a_1 \cdot x) = a_1^t \Sigma a_1, \quad \text{con} \quad \Sigma = E\left[(x - E[x])(x - E[x])^t\right]$$

Donde Σ se conoce como la matriz de covarianza de x. El método habitual que se usa para maximizar este tipo de funciones sujeta a restricciones es por medio de los *multiplicadores de Lagrange*.

El problema consiste en maximizar la función $a_1^t \Sigma a_1$ sujeta a la restricción $a_1^t \cdot a_1 = 1$. Se puede observar que la incógnita es precisamente a_1 (el vector desconocido que nos da la combinación lineal óptima). Así, se construye la función L:

$$L(a_1) = a_1^t \Sigma a_1 - \lambda(a_1^t a_1 - 1)$$

y luego se deriva e iguala a cero:

$$\begin{aligned} \frac{\partial L}{\partial a_1} &= 2\Sigma a_1 - 2\lambda I a_1 = 0 \Longrightarrow \\ (\Sigma - \lambda I)a_1 &= 0 \end{aligned}$$

Lo que genera un sistema lineal de ecuaciones. Luego por el teorema de *Roché-Frobenius* [25], para que el sistema tenga una solución distinta de 0 la matriz $(\Sigma - \lambda I)$ tiene que ser singular, esto es que el determinante debe ser igual a 0:

$$|\Sigma - \lambda I| = 0$$

y así entonces λ es un valor propio de Σ.

La matriz de covarianzas Σ es de orden p y si además es definida positiva, tendrá p valores propios distintos $\lambda_1, \lambda_2, \ldots, \lambda_p$ tales que, se tendrá un orden de por ejemplo, $\lambda_1 > \lambda_2 > \cdots > \lambda_p$.
Se tiene que desarrollando la expresión anterior,

$$\begin{aligned} (\Sigma - \lambda I)a_1 &= 0 \\ \Sigma a_1 - \lambda I a_1 &= 0 \\ \Sigma a_1 &= \lambda I a_1 \end{aligned}$$

y entonces:

$$\begin{aligned} Var(y_1) &= Var(a_1^t \cdot x) = \\ a_1^t \Sigma a_1 &= a_1^t \lambda I a_1 = \\ \lambda a_1^t \cdot a_1 &= \lambda \cdot 1 = \lambda \end{aligned}$$

Luego, para maximizar la varianza de y_1 se tiene que tomar el mayor valor propio, digamos λ_1, y el correspondiente vector propio a_1.

Dado que

$$\begin{aligned} a_1^t &= (a_{11}, a_{12}, \ldots, a_{1p}) \Rightarrow \\ y_1 &= a_1^t \cdot x = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1p}x_p \end{aligned}$$

El segundo componente principal se elige por escoger alguna de las nuevas variables, sea $y_2 = a_2^t \cdot x$, se obtiene de manera análoga a y_1, y dado que se quiere que y_2 sea independiente de y_1 se tendrá que tener:

$$Cov(y_2, y_1) = 0$$

Por lo tanto:

$$\begin{aligned} Cov(y_2, y_1) &= Cov(a_2^t \cdot x, a_1^t \cdot x) = \\ &= a_2^t \cdot E[(x-\mu)(x-\mu)^t] \cdot a_1 = \\ &= a_2^t \Sigma a_1 \end{aligned}$$

es decir se debe tener que $a_2^t \Sigma a_1 = 0$; como se tenía que $\Sigma a_1 = \lambda a_1$, lo anterior es equivalente a:

$$a_2^t \Sigma a_1 \; = \; a_2^t \lambda a_1 \; = \; \lambda a_2^t \cdot a_1 \; = \; 0,$$

de donde se obtiene que $a_2^t \cdot a_1 = 0$, es decir, que los vectores sean ortogonales.

De este modo, se tendrá que maximizar la varianza de y_2 , es decir, $a_2^t \Sigma a_2$, sujeta a las siguientes restricciones:

$$a_2^t \cdot a_2 = 1\,, \qquad a_2^t \cdot a_1 = 0$$

Dando lugar a la función:

$$L(a_2) = a_2^t \Sigma a_2 - \lambda(a_2^t \cdot a_2 - 1) - \delta a_2^t \cdot a_1$$

y al derivarla:

$$\begin{aligned} \frac{\partial L(a_2)}{\partial a_2} &= 2\Sigma a_2 - 2\lambda a_2 - \delta a_1 = 0, \quad \backslash a_1^t \cdot \\ 2a_1^t \Sigma a_2 &= 0 \end{aligned}$$

pues $a_1^t \cdot a_2 = a_2^t \cdot a_1 = 0$ y $a_1^t \cdot a_1 = 1$

Luego

$$\delta \; = \; 2a_1^t \Sigma a_2 \; = \; 2a_2^t \Sigma a_1 \; = \; 0$$

ya que

$$Cov(y_2, y_1) = 0$$

De esta manera, $\frac{\partial L(a_2)}{\partial a_2}$ queda finalmente como:

$$\begin{aligned} \frac{\partial L(a_2)}{\partial a_2} &= 2\Sigma a_2 - 2\lambda a_2 - \delta a_1 = \\ 2\Sigma a_2 - 2\lambda a_2 &= (\Sigma - \lambda I) \cdot a_2 = \\ &= 0 \end{aligned}$$

Usando los mismos razonamientos que antes, se elige λ como el segundo mayor valor propio de la matriz Σ con su vector propio asociado a_2.

De manera análoga se extiende para que a la $j-$ésima componente le corresponda el $j-$ésimo valor propio.

Así finalmente todos los componentes y (en total p) se pueden expresar como el producto de una matriz formada por los vectores propios, multiplicada por el vector x que contiene las variables originales $x_1, x_2, \ldots, x_p$.

$$y = Ax$$

donde:

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix}, \; A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pp} \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

Como:

$$\begin{aligned} Var(y_1) &= \lambda_1 \\ Var(y_2) &= \lambda_2 \\ \vdots &= \vdots \\ Var(y_p) &= \lambda_p \end{aligned}$$

la matriz de covarianzas de y resultara:

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_p \end{pmatrix}$$

ya que la manera de construir $y_1, \ldots, y_p$ fue para que queden independientes.
Entonces, se tiene:

$$\begin{aligned} \Lambda &= Var(Y) = \\ &= A^t \cdot Var(X) \cdot A = \\ &= A^t \cdot \Sigma \cdot A \\ \text{o bien } \Sigma &= A \cdot \Lambda \cdot A^t \end{aligned}$$

ya que A es una matriz ortogonal (pues $a_{\text{i}}^t \cdot a_{\text{i}} = 1$ para todas sus columnas) por lo que $A \cdot A^t = I$.

2.3.3. Porcentaje de información contenida

Se tiene que $Var(y_{\text{i}}) = \lambda_{\text{i}}$, luego al sumar todos los valores propios, se tendrá la varianza total de los componentes, es decir:

$$\sum_{\text{i}=1}^{p} Var(y_{\text{i}}) = \sum_{\text{i}=1}^{p} \lambda_{\text{i}} = traza(\Lambda)$$

pues Λ es diagonal.

Pero, al ocupar las propiedades del operador traza, tenemos:

$$\begin{aligned} traza(\Lambda) &= traza(A^t \Sigma A) \\ &= traza(\Sigma A^t A) \\ &= traza(\Sigma) \end{aligned}$$

porque $A \cdot A^t = I$, ya que A es ortogonal, por lo que:

$$traza(\Lambda) = traza(\Sigma) = \sum_{\mathrm{i}=1}^{p} Var(x_{\mathrm{i}})$$

Es decir, la suma de las varianzas de las variables originales y la suma de las varianzas de las componentes son iguales. Esto permite calcular el porcentaje de varianza total que recoge un componente principal:

$$\frac{\lambda_{\mathrm{i}}}{\sum_{\mathrm{i}=1}^{p} \lambda_{\mathrm{i}}} = \frac{\lambda_{\mathrm{i}}}{\sum_{\mathrm{i}=1}^{p} Var(x_{\mathrm{i}})}$$

Así, también se podrá expresar el porcentaje de variabilidad recogido por los primeros m componentes:

$$\frac{\sum_{\mathrm{i}=1}^{m} \lambda_{\mathrm{i}}}{\sum_{\mathrm{i}=1}^{p} \lambda_{\mathrm{i}}} = \frac{\sum_{\mathrm{i}=1}^{m} Var(x_{\mathrm{i}})}{\sum_{\mathrm{i}=1}^{p} Var(x_{\mathrm{i}})}$$

con $m < p$. Entonces, con esto es posible escoger un numero mucho menor de componentes que posean un porcentaje amplio de información ($\sum_{\mathrm{i}=1}^{m} Var(x_{\mathrm{i}})$). En la parte de resultados se verá cuantas fueron los adecuados a escoger. Para lo que viene se define lo que es la matriz de correlación:

Definición 2.6 *La matriz de correlación $R = (r_{\mathrm{i}j})_{\mathrm{i},j=1,\ldots,n}$ para un conjunto de variables $x_1, x_2, \ldots, x_n$ se define de la siguiente forma:*

$$r_{\mathrm{i}j} = Cor(x_{\mathrm{i}}, x_j) = \frac{Cov(x_{\mathrm{i}}, x_j)}{\sqrt{Var(x_{\mathrm{i}}) Var(x_j)}}, \quad \textit{con} \quad Cov(x_{\mathrm{i}}, x_j) = E[(x_{\mathrm{i}} - E[x_{\mathrm{i}}])(x_j - E[x_j])]$$

y si los x_{i} están estandarizados $(N(0,1))$ entonces la matriz de covarianza y de correlación coinciden.

2.3.4. Cálculo de los componentes principales a partir de la matriz de correlaciones

Si los datos originales están estandarizados $(N(0,1))$ entonces, las componentes principales son vectores propios de la matriz de correlaciones y son distintos de los de la matriz de covarianzas; y así se da igual importancia a todas las variables originales; se tiene que la diagonal solo contiene 1's, y la proporción de varianza recogida por el vector propio $j-$ésimo(componente) es:

$$\frac{\lambda_j}{p}$$

Definición 2.7 (matriz factorial) *Se define la matriz factorial, como* $F = (a_1^*, a_2^*, \ldots, a_p^*)$*, donde* $a_j^* = \lambda_j^{1/2} a_j$.

Dado que $y = A \cdot x$, entonces $x = A^{-1} \cdot y$, de modo que

$$\begin{aligned} Cov(x) &= (A^{-1})^t Cov(y) A^{-1} \\ &= A \cdot \Lambda \cdot A^t \\ &= A \cdot \Lambda^{1/2} \cdot \Lambda^{1/2} \cdot A^t \\ &= F \cdot F^t \end{aligned}$$

ya que al ser A ortogonal, resulta que $A^{-1} = A^t$.

Por lo que dada la matriz factorial F, se pueden calcular las covarianzas de las variables originales, es decir, se puede recuperar la matriz de covarianzas original a partir de la matriz factorial [26]. Si se toma un número menor de factores $m < p$, se podrá reproducir aproximadamente Σ.

Observación Si las variables originales $x_1, \ldots, x_p$ son independientes, entonces carece de sentido hacer *PCA*, pues solo se obtendrían las mismas variables pero reordenadas de mayor a menor varianza.

2.4. Descripción de la base de datos

La base de datos con la que se trabajará esta compuesta por estampillas de imágenes de dimensión 21 $\times$ 21 pixels, luego para poder visualizarlos se utiliza `Matlab`, el cual puede generar una paleta de 256 colores y entonces asignar a cada píxel uno o alguna combinación de estos colores y finalmente graficar en conjunto los 21$\times$21 pixels de la estampilla para poder ver una imagen[2].

Estas imágenes de 21$\times$21 pixels son extraídas de fotografías astronómicas mediante un pipeline que genera candidatos a estrellas [2], a modo general el trabajo realizado por el telescopio DECam es tomar varias fotografías de un mismo lugar espacial a diferentes tiempos. A continuación se seleccionan solo dos imágenes, las cuales entre sí poseen la mayor variabilidad del objeto a estudiar, basado en estas dos imágenes se obtienen otras 3 imágenes más.

Cuando se tiene las dos imágenes iniciales, una de ellas se designa como la imagen de referencia, la cual posea la mejor claridad (menor ruido) y entonces se toma la otra para crear la imagen convolucionada, que básicamente intenta equipararse con la de referencia para una posible comparación; la imagen de diferencia es la resta que se produce entre la de referencia y la convolucionada, puede ser (referencia - convolucionada) o (convolucionada - referencia), eso es dependiendo del nombre del archivo que contenga a dicho

[2]De acá en adelante a estos objetos simplemente los llamaremos imágenes y serán el objeto de estudio principal de esta memoria

objeto astronómico. Finalmente la imagen señal a ruido se obtiene de tomar la imagen de diferencia y dividirla por el ruido de poissoniano que tiene dicha imagen en el modelo matemático que es usado para extraerla[3].

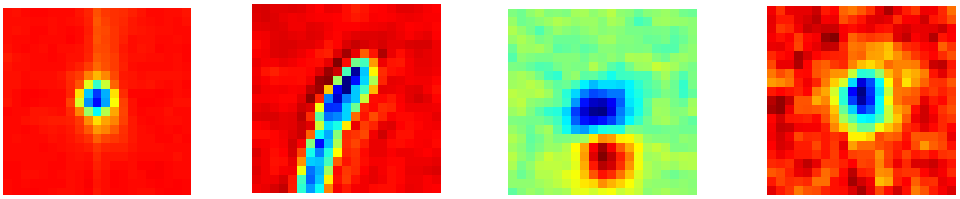
A modo de ejemplo, en la figura 2.5 podemos ver 4 objetos en donde se tiene las 5 imágenes nombradas anteriormente para cada uno.

Cuando se obtiene la imagen de diferencia (y a posterior la imagen señal a ruido) se puede hacer una clasificación en dos grupos de imágenes (tanto para la imagen diferencia como para la imagen señal a ruido), estos son: ***imágenes negativas*** e ***imágenes positivas***, definidas como:

Definición 2.8 (**Imagen negativa**) *Cuando en la imagen de diferencia da como resultado que los pixels de la imagen disminuyen en brillo, entonces se considera negativa, a modo de ejemplo se dejan algunas imágenes negativas en la figura 2.3.*

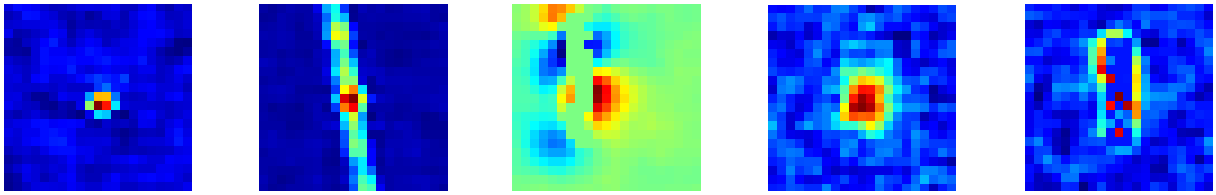
Definición 2.9 (**Imagen positiva**) *Cuando en la imagen de diferencia da como resultado que los pixels de la imagen aumentan en brillo, entonces se considera positiva, a modo de ejemplo se dejan algunas imágenes positivas en la figura 2.4.*

[3]Cada CCD del telescopio registra el n° de fotones que recibe, esto se llama el número de cuentas y son distribuidos en una distribución de Poisson, así el ruido poissoniano es la media de la distribución que en este caso es la raíz cuadrada del n° de cuentas.



(a) (b) (c) (d)

Figura 2.3: Ejemplos de imágenes negativas: (a) una fuente puntual falsa, (b) un rayo cósmico, (c) mala resta y (d) una estrella negativa.



(a) (b) (c) (d) (e)

Figura 2.4: Ejemplos de imágenes positivas: (a) una fuente puntual falsa, (b) un rayo cósmico, (c) mala resta, (d) una estrella y (e) objeto desconocido.

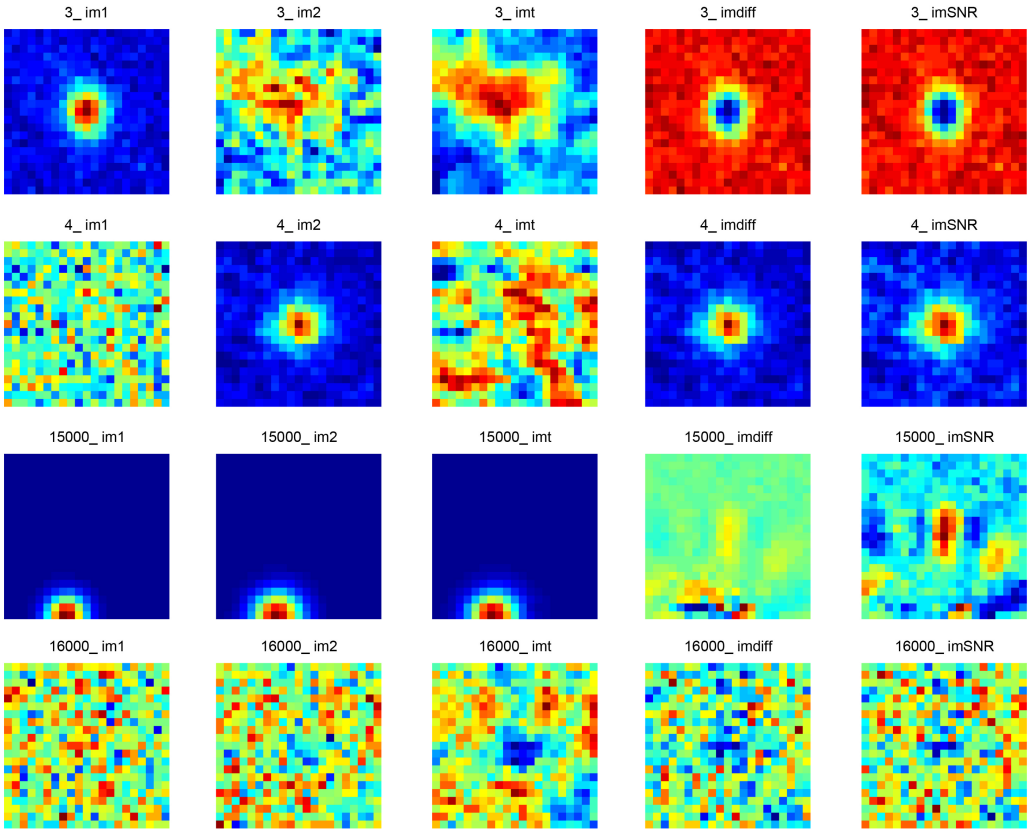


Figura 2.5: Se observan 4 objetos astronómicos con sus respectivas 5 imágenes, de izquierda a derecha: imagen 1, imagen 2, imagen convolucionada, imagen de diferencia e imagen señal a ruido.

Comentario 2.10 *En estricto rigor solo interesaran las imágenes que sean estrellas, y viendo éstas, se puede ver que es muy fácil darse cuenta cuando son negativas y cuando son positivas, pues las negativas se caracterizan por tener un fondo naranjo y las positivas un fondo azul.*
Entonces poco importará si existe dificultad en clasificar imágenes extrañas como las que aparecen en (c) de la figura 2.3 o 2.4, pues lo importante son las estrellas, es ahí donde interesa la clasificación.

2.5. *Curva Receiver Operating Characteristic (Curva ROC)*

La curva *Característica Operativa del Receptor* (*Receiver Operating Characteristic* (***ROC***)), es una herramienta para seleccionar los modelos posiblemente óptimos y descartar modelos sub-óptimos que estudian clasificaciones binarias. Este análisis se relaciona de forma directa y natural con el análisis de coste/beneficio en toma de decisiones diagnósticas (ref. [27] [28]).

La curva *ROC* se desarrolló por ingenieros eléctricos para medir la eficacia en la detección de objetos enemigos en campos de batalla mediante pantallas de radar, a partir de lo cual se desarrolla la Teoría de Detección de Señales. Posteriormente la curva *ROC* se aplicó a medicina, radiología, psicología, entre otras áreas por varias décadas. Actualmente se ha encontrado aplicación en áreas como aprendizaje automático (machine learning), y minería de datos (data mining).

Para entender que hace una curva *ROC*, se debe entender el concepto de matriz de confusión:

Definición 2.11 (Matriz de Confusión) *Será la matriz que para dos o más clases tendrá de manera ordenada los casos obtenidos por el clasificador y lo que realmente debería haber sido según los datos reales.*

Suponiendo que se tienen 2 clases o estados en los que los objetos pueden ser clasificados, sean estos: Positivo y Negativo. Llamaremos ***P*** *a los objetos que realmente son positivos,* ***P'*** *a los que el clasificador dice que son positivos,* ***N*** *a los que realmente son negativos y por ultimo* ***N'*** *a los que el clasificador dice que son negativos. Luego esto se ordenará de la siguiente forma:*

	Positivo(real) = P	*Negativo(real) = N*
Positivo(predicho) = P'	*VP*	*FP*
Negativo(predicho) = N'	*FN*	*VN*

Donde:

- *Verdaderos Positivos (**VP**): Casos que pertenecen a la clase **P** y el clasificador los definió en esa clase (i.e., **P'**).*
- *Falsos Positivos (**FP**): Casos que pertenecen a la clase **N** y que el clasificador no los definió en esa clase (i.e., los definió en **P'**).*
- *Falsos Negativos (**FN**): Casos que pertenecen a la clase **P** y el clasificador no los definió en esa clase (i.e., los definió en **N'**).*
- *Verdaderos Negativos (**VN**): Casos que pertenecen a la clase **N** y que el clasificador los definió en esa clase (i.e., **N'**).*

Si se estuviera variando un parámetro dentro del modelo y se van obteniendo distintos *VP, FN, FP* y *VN*, esto implica que se pueden obtener cada vez, una nueva Matriz de Confusión.

La curva *ROC*, es un gráfico en el que se observa la Tasa de Verdaderos Positivos (**TPR**[4], del ingles True Positive Rate) en el eje **y** y la Tasa de Falsos Positivos (**FPR**[5], del ingles False Positive Rate) en el eje **x**; y por cada Matriz de Confusión que se obtenga se va obteniendo un nuevo punto en la curva.

La **TPR** mide hasta que punto un clasificador es capaz de detectar o clasificar los casos positivos correctamente, de entre todos los casos positivos disponibles durante la prueba y la **FPR** define cuantos resultados positivos son incorrectos de entre todos los casos negativos disponibles durante la prueba.

A veces se suele ocupar además otro indicador para ver que también fue el modelo, este es de Precisión o Exactitud (accuracy (**ACC**[6]))

Se puede ver en la figura 2.6 que el mejor método posible de predicción se situaría

[4]TPR = VP/(VP + FN)
[5]FPR = FP/(FP + VN)
[6]ACC = (VP + VN)/(P + N) = (VP + VN)/(VP + FN + FP + VN)

en un punto en la esquina superior izquierda, de coordenada **(0,1)** del espacio *ROC*, lo cual representa un 100 % de sensibilidad (ningún falso negativo) y un 100 % también de especificidad (ningún falso positivo). Al punto **(0,1)** se llama también una clasificación perfecta. Además en la figura 2.6 si se tuviera un punto como el de **C**, es decir a lo largo de la diagonal, se dice que es una clasificación totalmente aleatoria. La diagonal divide el espacio *ROC* y los puntos por encima representan los buenos resultados de clasificación (al menos mejor que el azar) y los puntos por debajo de la línea los resultados deficientes (peores que el azar).

También se observa que el punto **A** es mejor predictor que el del punto **B**. Y por ultimo se puede ver que si un modelo es deficiente se podría invertir para obtener un buen resultado como lo es en el caso **D** de la figura 2.6. De acuerdo a [27], las ventajas de usar una *Curva ROC* para el análisis, son:

1.- Son una representación fácilmente comprensible de la capacidad de discriminación de la prueba en todo el rango de puntos que se generen en la curva *ROC*.

2.- Son simples, gráficas y fáciles de interpretar visualmente.

3.- No requieren un nivel de decisión particular visualmente.

4.- Son independientes de la prevalencia, ya que la sensibilidad y la especificidad se obtienen en distintos subgrupos. Por tanto, no es necesario tener cuidado para obtener muestras con prevalencia representativa de la población. De hecho, es preferible generalmente tener igual número de individuos en ambos subgrupos.

5.- Proporciona una comparación visual directa entre pruebas en una escala común, mientras que otro tipo de gráficos, como los diagramas de puntos o los histogramas de frecuencia, requieren diferentes gráficos cuando difieren las escalas.

6.- La especificidad y la sensibilidad son accesibles en el gráfico, en contraste con los diagramas de puntos y los histogramas.

7.- Lo principal es que con la curva *ROC* se puede escoger el punto de operación variando un parámetro (teniendo en cuenta que cada clasificador da una curva).

2.6. *Support Vector Machine (SVM)*

Maquina de Soporte Vectorial (Support Vector Machine, SVM) es un modelo de aprendizaje supervisado que clasificación de datos. El objetivo de *SVM* es producir un modelo (basado en los datos de entrenamiento) los cuales predicen los valores objetivos (targets) de los datos a probar dando solo los atributos de estos datos test.

Dado un conjunto de entrenamiento de pares instancias-etiquetas $(x_{\mathrm{i}}, y_{\mathrm{i}}), \mathrm{i} = 1, \ldots, l$ donde $x_{\mathrm{i}} \in \mathbb{R}^n$ y $y \in \{1, -1\}^l$, la maquina de Soporte vectorial (*SVM*) necesita resolver el

siguiente problema de optimización [29][30]:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}w^T w + C\sum_{\mathrm{i}=1}^{l}\xi_{\mathrm{i}} \\ \text{sujeto a} \quad & y_{\mathrm{i}}(w^T\phi(x_{\mathrm{i}}) + b) \geq 1 - \xi_{\mathrm{i}}, \\ & \xi_{\mathrm{i}} \geq 0. \end{aligned}$$

En el problema los vectores de entrenamiento x_{i} son mapeados en un espacio dimensional mas alto (tal vez infinito) mediante la función ϕ. *SVM* encuentra un hiperplano lineal de separación de los datos con el margen máximo en este espacio dimensional mas alto.
$C > 0$ es el parámetro de restricción que determina el compromiso entre dos metas conflictivas: maximizar el margen y reducir el número de errores, es una constante poco intuitiva y no hay manera de seleccionarlo a priori. Además, $K(x_{\mathrm{i}}, x_j) \equiv \phi(x_{\mathrm{i}})^T\phi(x_j)$ is llamado el núcleo de la función (kernel).

Las siguiente 4 kernel son los mas básicos que utiliza *SVM*:

- Lineal: $K(x_{\mathrm{i}}, x_j) = x_{\mathrm{i}}^T x_j$.
- Polinomial: $K(x_{\mathrm{i}}, x_j) = (\gamma x_{\mathrm{i}}^T x_j + r)^{\mathrm{d}}, \gamma > 0$.
- Función de base radial (RBF): $K(x_{\mathrm{i}}, x_j) = exp(-\gamma||x_{\mathrm{i}} - x_j||^2), \gamma > 0$.
- Sigmoideo (sigmoid): $K(x_{\mathrm{i}}, x_j) = tanh(\gamma x_{\mathrm{i}}^T x_j + r)$.

γ, r y d son parámetros exclusivos de cada kernel.

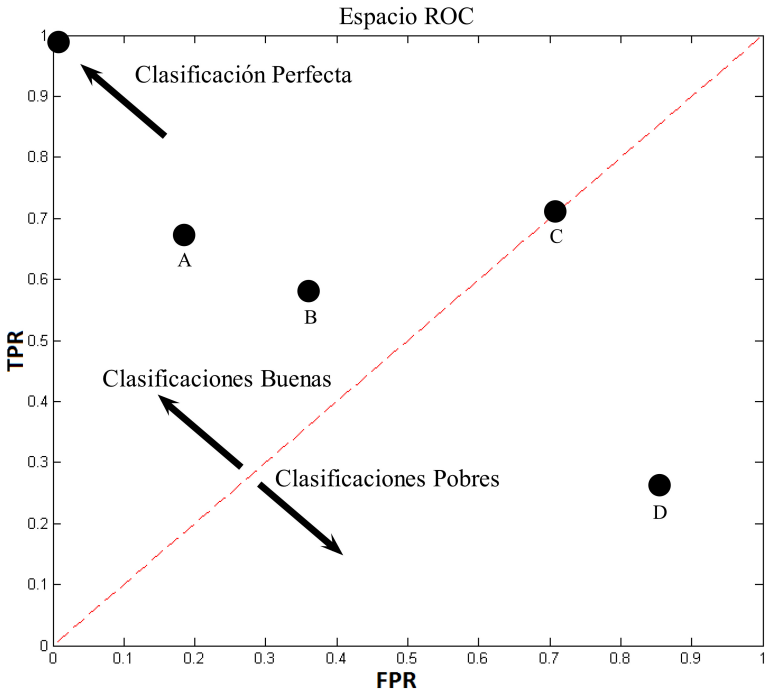


Figura 2.6: La diagonal punteada representa predicciones al azar, se le suele llamar también *línea de no-discriminación*. Los puntos por encima de la diagonal son de buenos modelos (puntos **A, B** y **(0,1)**), los por debajo de la diagonal son de malos modelos (punto **D**) y los de la diagonal son modelos igual que una predicción aleatoria (punto **C**).

Capítulo 3

Metodología

En lo que sigue es importante hacer una diferencia entre que se entiende por método y modelo en esta memoria. Método, es el conjunto de todos los procesos involucrados en orden para obtener una clasificación, son 3 métodos con los que se trabaja: *PCA*, *FAST HALS for NMF* y *HALS NMF Sparsity*. Los 3 métodos tienen en común los procesos de: descomposición matricial[1], reducción de dimensionalidad, proyección y clasificación. En cambio modelo es solo un proceso particular dentro de los 3 métodos[2].

En este capítulo se presenta la metodología utilizada en esta memoria. Los temas a tratar son el tratamiento de los datos antes de ser ingresados a los métodos (pre-procesamiento), ajustes de parámetros para cada modelo ocupado, la forma de como se obtuvieron los resultados, una forma de comparar los métodos propuestos para escoger el que mejor clasifica y finalmente algunas maneras de variar y optimizar el método seleccionado para obtener uno mas sencillo que explique los resultados e inclusive pueda mejorarlos.

Este capítulo se divide en 5 secciones, explicando cada procedimiento, estas son:

1. **Pre-procesamiento:** En esta sección se realiza un preprocesamiento de las imágenes utilizadas, donde se incluye: normalización de los datos y división para tener conjuntos de entrenamiento, validación y prueba.

2. **Ajuste de parámetros:** En esta sección se exponen los criterios y metodología empleada para ajustar parámetros en las descomposiciones (*PCA* y *NMF*) y en los modelos de clasificación.

3. **Metodología de resultados:** En esta sección se explica el uso de la base de datos preprocesada, habiendo previamente ajustado los parámetros de cada modelo, principalmente para mostrar los resultados de las simulaciones.

4. **Comparación de los 3 métodos**, en términos de clasificación.

[1]Bajo el algoritmo respectivo, algoritmo de: descomposición *PCA*, *FAST HALS for NMF* o *HALS NMF Sparsity*.

[2]Ejemplos: clasificación (*SVM* o *KNN*), algún tipo de variación de imagen, descomposición *NMF* según prototipos e iteraciones, descomposición *PCA* según porcentaje de información contenida.

5. **Ajuste fino del mejor método:** Una vez seleccionado el mejor método propuesto, se realizan en éste algunas modificaciones con el fin de mejorar su desempeño.

3.1. Pre-procesamiento

Las imágenes utilizadas fueron 3 de las 5 imágenes descritas en la sección *Descripción de la base de datos* (2.4), y éstas son: imagen de referencia, imagen convolucionada y la imagen señal a ruido. La justificación de porqué se utilizaron estas imágenes se debe a que el experto (astrónomo) utiliza estas tres para clasificar estrellas y no estrellas. Para el experto la imagen de referencia es la mejor entre las dos imágenes iniciales (la 1 o la 2), la imagen convolucionada es un intento de mejora de la que no fue seleccionada como referencia. Finalmente la imagen señal a ruido es una limpieza que se le hace a la imagen de diferencia. Es conveniente mencionar que también se trabajó con las otras dos imágenes descritas en la sección (2.4), sin embargo no entregaron mejores resultados que haber utilizado estás 3 imágenes solamente.

Antes de que las imágenes sean normalizadas se guardan las medias y desviaciones estándar de cada imagen. En cuanto a la normalización de los datos, ésta se realiza por separado para cada una de las imágenes que se ocupan. Por ejemplo, suponiendo que se tiene la matriz V que contiene todas las imágenes señal a ruido[3], la manera de normalizar es local para cada imagen. Se obtiene el mínimo y máximo local, $v_{\mathrm{i}}min$ y $v_{\mathrm{i}}max$ respectivamente de cada imagen $V(:,\mathrm{i})$ y la normalización es:

$$V(:,\mathrm{i}) = \frac{V(:,\mathrm{i}) - v_{\mathrm{i}}min}{v_{\mathrm{i}}max - v_{\mathrm{i}}min}, \quad \text{para todo } \mathrm{i} = 1, 2, \ldots, (\text{n}^\circ \text{ muestras})$$

Esto produce que todos los valores de V estén entre 0 y 1 (incluidos), dejando estandarizadas todas las imágenes en la misma escala.

La cantidad de datos total con los que se cuentan son 100 mil imágenes divididas en dos clases: Estrella y no estrella, y esto es por cada imagen con las que se trabaja (imagen referencia, imagen convolucionada e imagen señal a ruido). Para cada imagen se cuenta con su etiqueta (estrellas o no-estrellas) y su tasa de señal a ruido (*SNR*, del ingles signal to noise rate)[4].

De estos datos se realiza la siguiente división:

- *Conjunto de Entrenamiento (CE)*: 20 mil datos, entre estrellas y no-estrellas en igual cantidad.
- *Conjunto de Validación (CV)*: 40 mil datos, entre estrellas y no-estrellas en igual cantidad.
- *Conjunto de Prueba (CP)*: 40 mil datos, entre estrellas y no-estrellas en igual cantidad.

[3]Las columnas de V son distintas imágenes y las filas de V son cada uno de los píxeles de las imágenes (441 por cada una).

[4]En este trabajo diremos que una imagen es de baja señal si el valor absoluto de su *SNR* esta entre 5 y 6, y diremos es de alta señal si el valor absoluto de *SNR* es mayor que 6.

Todo los conjuntos son disjuntos entre sí y fueron extraídos de manera aleatoria, solo preocupándose que tuvieran igual cantidad de estrellas y no-estrellas cada uno.

3.2. Ajuste de Parámetros

De acuerdo a cada herramienta utilizada, se tienen distintos parámetros a considerar para obtener el mejor desempeño, una vez obtenido los datos pre-procesados.

En lo que sigue nos referimos a la matriz que contiene imágenes pre-procesadas, donde las filas son los píxels (441) y las columnas son las observaciones, como **IM**.

3.2.1. Ajustes para *PCA*

Al descomponer con *PCA*, se ingresa **IM**, obteniéndose 2 matrices y un vector:

La primera matriz es la de las componentes principales[5] (W) donde cada columna es una componente principal, ordenadas de manera decreciente por el valor de la varianza calculada individualmente. Cada componente principal posee 441 elementos, por lo que la matriz W es de dimensión[6] 441×441.

La segunda matriz es la de los coeficientes (H), cuya dimensión es n° de observaciones por componentes principales, la cual cumple que $\mathbf{IM} \approx \mathbf{H} \cdot \mathbf{W^t} + \mu$, donde μ es una matriz de igual dimensión que **IM**, que le suma a cada columna de la matriz $\mathbf{H} \cdot \mathbf{W^t}$ la media de esa misma columna.

El vector obtenido es el que posee los valores de las varianzas de cada componente principal, ordenados de manera decreciente. Este es usado para calcular el porcentaje de información contenida (ver sección 2.3.3).

Los prototipos utilizados en la descomposición *PCA* son: 10, 20, 30, 40, 50 y 100, y se escoge el que mejor clasificación obtenga (observando la *FPR* y *TPR*). El conjunto utilizado para descomponer y obtener los prototipos es *CE*, a modo de ilustración en la figura 3.1 podemos ver las 40 primeras componentes principales ordenadas según su varianza. Una vez obtenida la reducción se proyecta el mismo *CE* en estas componentes, resultando una matriz con igual cantidad de observaciones y con número de características reducidas.

El conjunto para ajustar prototipos es *CE*. Una vez seleccionado n° prototipos, se proyecta *CV* en los prototipos escogidos (W), obteniendo la matriz de coeficientes respectiva ($\overline{H}$), la cuál junto con H son ingresada a *SVM* para ajustar parámetros (sección 3.2.4).

[5]También llamadas nuevas variables principales o en efecto para lo que sigue, nuevas características o prototipos.

[6]Recordemos que *PCA* crea nuevas variables para expresar las variables iniciales, quedando en igual n° de variables.

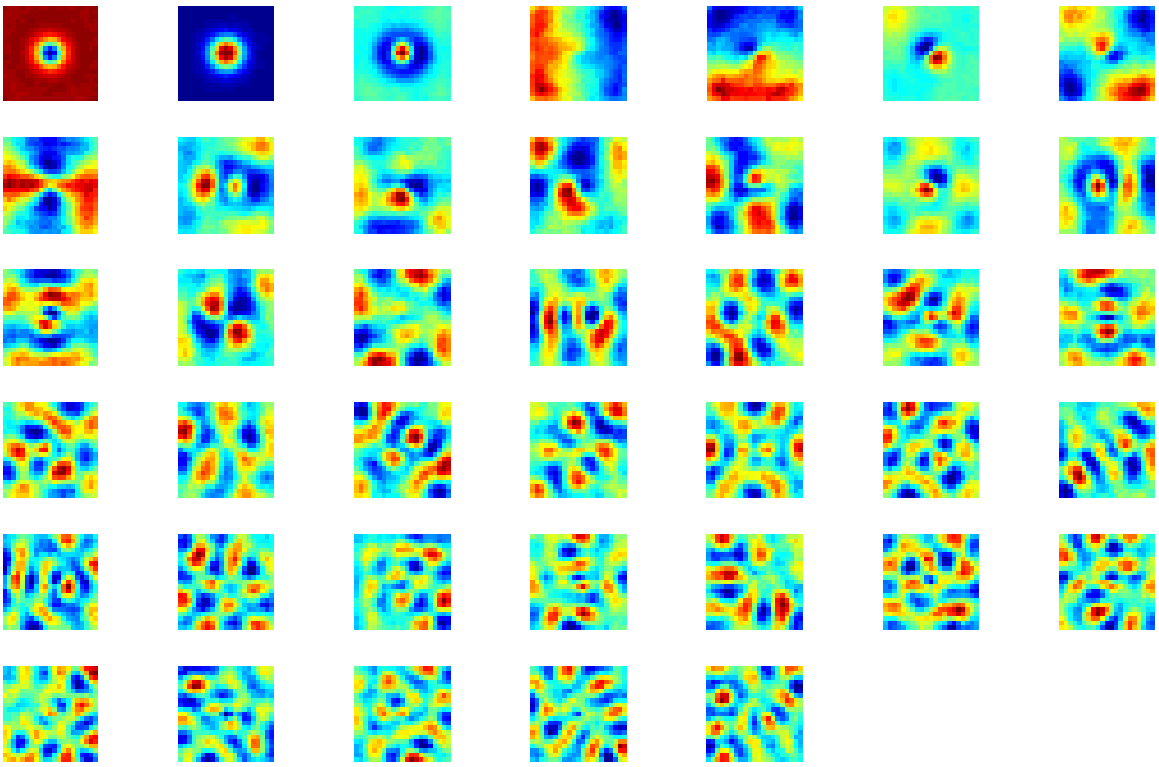


Figura 3.1: Se observan 40 componentes principales, ordenadas de manera decreciente por las varianzas, de izquierda a derecha y de arriba hacia abajo.

3.2.2. Ajustes para *FAST HALS for NMF*

Al descomponer con *FAST HALS for NMF*, se ingresa **IM** al algoritmo *FAST HALS for NMF* y el número de prototipos y número de iteraciones, obteniéndose 2 matrices:

La primera matriz es de prototipos (W), los que representan a todas las imágenes mediante combinaciones lineales. Cada columna es un prototipo de 441 características, la dimensión de W es 441 $\times$ n° prototipos.

La segunda matriz es de coeficientes (H), de dimensión n° de prototipos en filas y n° de observaciones en columnas. Al multiplicar $\mathbf{W} \cdot \mathbf{H}$ obtenemos una reconstrucción de **IM** con un cierto error asociado (2.1).

En el algoritmo *FAST HALS for NMF*, los parámetros a ajustar son el n° de prototipos. El n° de iteraciones que se toma para todos los casos es de 1000 y el n° de prototipos para descomponer son: 10, 20, 30, 40, 50 y 100. Escoger un gran número de prototipos permite obtener un error de reconstrucción relativo bajo, cuya definición es:

$$\textbf{\textit{ERROR CUADRÁTICO RELATIVO}} = \frac{||IM - W \cdot H||^2}{||IM||^2} \tag{3.1}$$

donde el **IM**, es la matriz V de la ecuación (2.1).
Tener un n° de prototipos adecuados, además permite tener una buena representación de las imágenes[1]. La manera de elegir el n° de prototipos es mediante la mejor clasificación mostrada en una curva *ROC*. A modo de ilustración en las figuras 3.2 y 3.3 se observa el

error cuadrático relativo de reconstrucción y los prototipos usados, respectivamente.

El conjunto para ajustar prototipos es *CE*. Una vez seleccionado n° prototipos, se proyecta *CV* en los prototipos escogidos (W), obteniendo la matriz de coeficientes respectiva ($\overline{H}$), la cuál junto con H son ingresada a *SVM* para ajustar parámetros (sección 3.2.4).

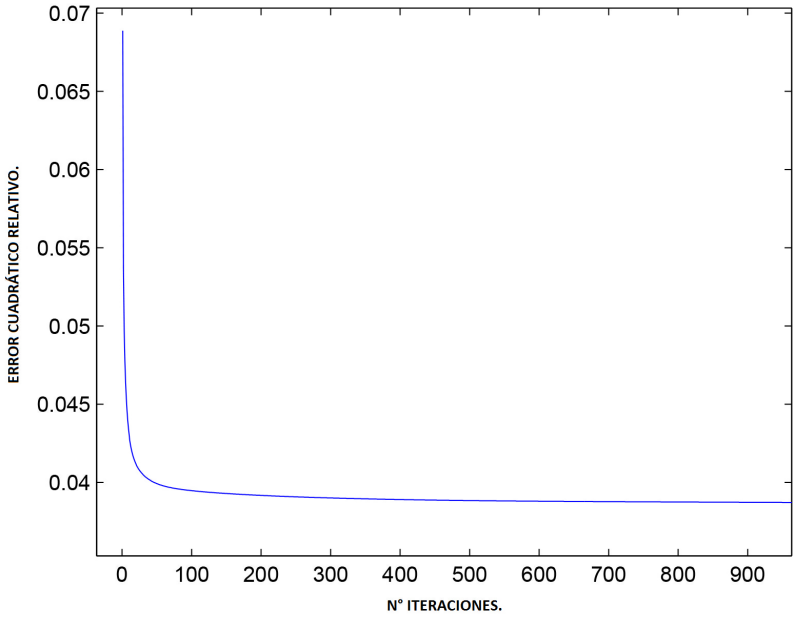


Figura 3.2: Error cuadrático relativo de reconstrucción a lo largo del n° de iteraciones para una descomposición *NMF* de 40 prototipos con 1000 iteraciones.

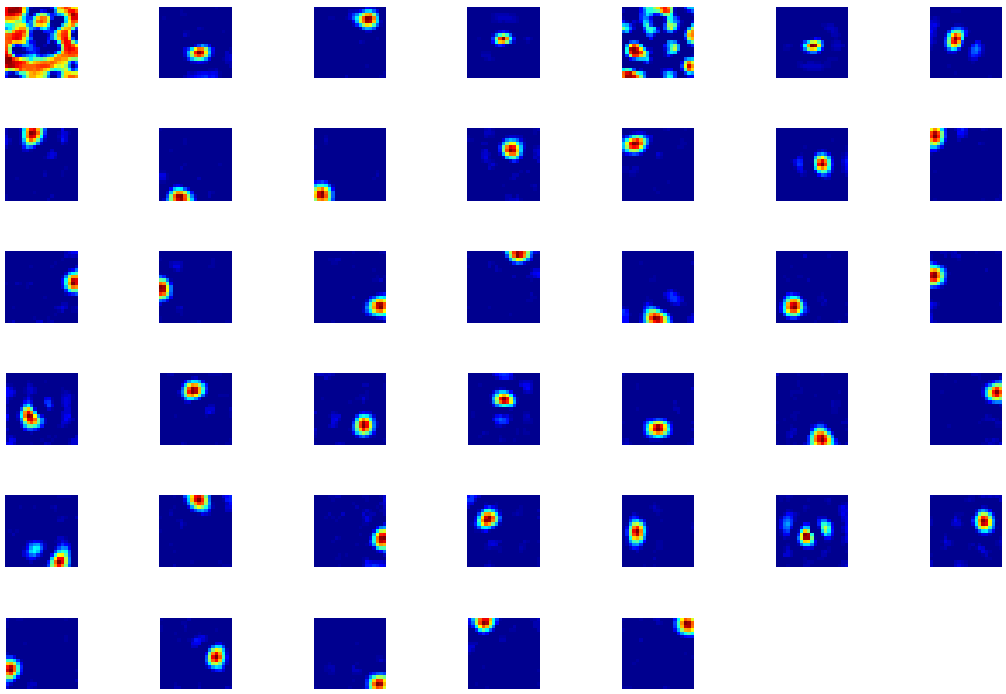


Figura 3.3: Los 40 prototipos entregados por el algoritmo *FAST HALS for NMF*.

3.2.3. Ajustes para *HALS NMF Sparsity*

Al descomponer con *HALS NMF Sparsity* se requiere lo mismo que *FAST HALS for NMF*, además del control de dispersión (α_{sp}). Se obtiene las mismas 2 matrices que *HALS for NMF* (W y H).

El control de dispersión permite a los prototipos cubrir la dispersión y afinidad con los datos reales, esto es, generalizar y obtener mejor similaridad con las imágenes originales pre-procesadas (**IM**), además de mejorar la reconstrucción.

Para el criterio de ajuste de parámetros se considera el error de clasificación. Para ello se crea una malla entre n° de prototipos: 10, 20, 30, 40[7]; y control de dispersión: 0.1, 0.3, 0.5, 1, 2 y 4. El n° de iteraciones es para todos los casos la misma: 1000. En el control de dispersión en [6] se recomienda valores menores a 5, aunque de igual modo se analiza con valores mas altos.

Bajo este algoritmo el criterio para reducir dimensionalidad es escoger n° prototipos y control de dispersión, de manera de obtener los parámetros que resulten con menor *FPR* y más alta *TPR*. A modo de ilustración en la figura 3.4 se observan prototipos para *HALS NMF Sparsity*.

El conjunto para ajustar prototipos y control de dispersión es *CE*. Una vez seleccionado n° prototipos y control de dispersión, se proyecta *CV* en los prototipos escogidos (W), obteniendo la matriz de coeficientes respectiva ($\overline{H}$), la cuál junto con H son ingresada a *SVM* para ajustar parámetros (sección 3.2.4).

[7] Analizar con un n° mayor de prototipos (independiente del control de dispersión) genera *FPR* sobre el 1 %, las cuales no son consideradas.

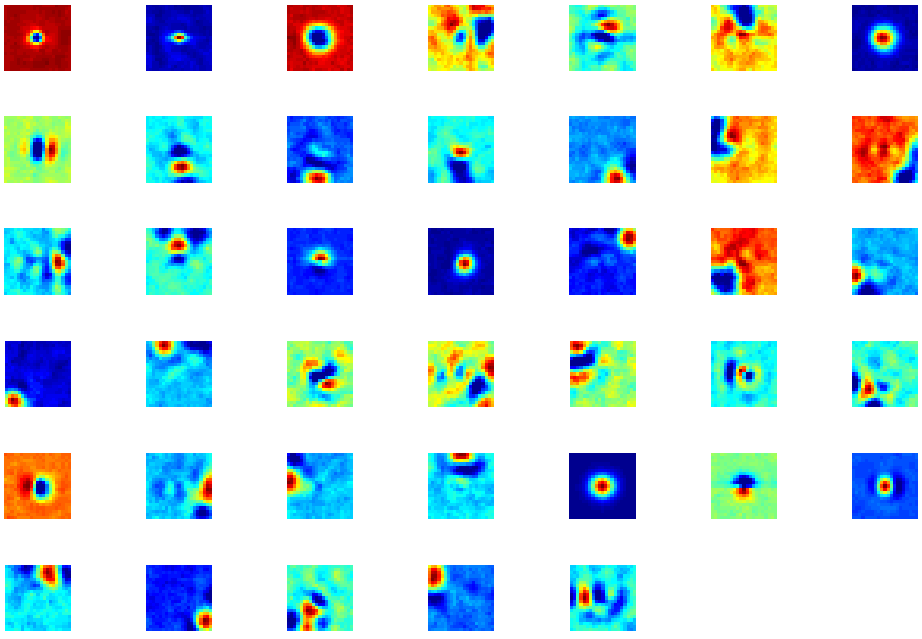


Figura 3.4: Los 40 prototipos entregados por parte de *NMF* bajo el algoritmo *HALS NMF Sparsity*.

3.2.4. Ajustes para *SVM*

La etapa siguiente a la reducción es la clasificación, para ello *SVM* crea un modelo (clasificador) con la base de datos[8], las etiquetas respectivas entre estrellas y no-estrellas, y dos parámetro ν y γ.

El parámetro ν permite tolerancia entre la convergencia de *SVM* y el error de clasificación, teóricamente se debe tener que el error de clasificación debe ser similar al valor de ν. El parámetro γ viene de la función núcleo (kernel) que utiliza el clasificador, la cual corresponde a un función de base radial gaussiana, donde $\gamma = \frac{1}{2\sigma^2}$.

Con el modelo ya creado (aun no ajustado), se clasifica una base de datos distinta a la inicial[9], para ello *SVM* necesita una matriz que en sus filas contenga las observaciones y en sus columnas las características de esas observaciones, y en base a esto entrega una predicción a que corresponde cada observación (estrella o no estrella); luego de acuerdo a la predicciones (etiquetas) entregadas, son comparadas con las etiquetas reales de cada observación.

El análisis post-comparación es calcular la tasa de falsos positivos (***FPR***), tasa de verdaderos positivos (***TPR***) y el error de clasificación (***ERROR***) en base a una matriz de confusión. El criterio para ajustar los parámetros ν y γ es obtener un ***ERROR*** menor del 3 % con una ***FPR*** menor al 1 %, estos resultados son presentados en una curva *ROC*. Se crea una malla con valores de, ν: 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1[10]; y γ: 0.01, 0.02, 0.03, 0.04, 0.05[11].

El conjunto utilizado para ajustar ν y γ es la proyección de *CE* y *CV*, ya sea en: algoritmo *PCA*, algoritmo *FAST HALS for NMF* o algoritmo *HALS NMF Sparsity*. Con la proyección de *CE* se crea el modelo *SVM* (aun no ajustado) y utilizando la proyección de *CV* son encontrados los parámetros requeridos observando la Curva *ROC* obtenida. La forma de utilizar la proyección de *CV* es separarlo en 2 conjuntos de imágenes: las de baja señal y las de alta señal.

3.3. Metodología de Resultados

Con los datos pre-procesados y los parámetros por cada modelo ajustados, en lo que sigue se utilizan los conjuntos *CE*, *CV* y *CP* para obtener los resultados en cada modelo.

La metodología utilizada en el algoritmo *PCA* y los dos algoritmos basados en *NMF* cumplen con las etapas de: descomposición, reducción, proyección y clasificación.

[8] Entiéndase que es la proyección de las imágenes pre-procesadas en cualquiera de los 3 algoritmos vistos anteriormente.

[9] Proyección de nuevas imágenes pre-procesadas en cualquiera de los 3 algoritmos.

[10] Valores basados en los errores de clasificación obtenidos.

[11] Estos valores entre varias simulaciones realizadas resultaron ser los más adecuados, intentar con valores de γ distintos a los propuestos, generan ***FPR***'s por sobre el 1 %

El uso de los conjuntos en cada parte es: descomposición con *CE* en cualquiera de los 3 algoritmos; con ello se obtiene la reducción. En la reducción es proyectado *CE*[12], *CV* y *CP*. Con las proyecciones de *CE* y *CV*, se crea y valida el clasificador *SVM* respectivamente, siendo este el procesos final para cada método. El método que entre los 3 posee mejor validación, i.e., menor ***FPR*** y ***ERROR***, es probado[13] con *CP*. *CP* es proyectado de la misma forma como se utiliza el conjunto *CV*. Además de utilizar el clasificador *SVM* para validar, se usa otro llamado *k-Nearest Neighbors algorithm* (algoritmo de *k* vecinos cercanos) [31], abreviado *knn*, el cual debería dar algo similar y se utiliza con un solo vecino cercano en los 3 métodos.

Para lo que sigue, es necesario dentro de cada conjunto diferenciar los 3 tipos de imágenes que se tiene, se usa la notación:

1) Imagen de referencia: $\mathbf{IM_{ref}}$.

2) Imagen convolucionada: $\mathbf{IM_{conv}}$.

3) Imagen señal a ruido: $\mathbf{IM_{snr}}$.

Para diferenciar los 3 tipos de imágenes entre 2 conjuntos distintos, digamos *CE* y *CV*, se usa la notación:

- $\mathbf{IM_{ref}(ce)}$ para *CE* y $\mathbf{IM_{ref}(cv)}$ para *CV*.
- $\mathbf{IM_{conv}(ce)}$ para *CE* y $\mathbf{IM_{conv}(cv)}$ para *CV*.
- $\mathbf{IM_{snr}(ce)}$ para *CE* y $\mathbf{IM_{snr}(cv)}$ para *CV*.

Con estas notaciones se presenta de manera individual cada metodología con los 3 métodos: *FAST HALS for NMF*, *HALS NMF Sparsity* y *PCA*.

3.3.1. Metodología con *FAST HALS for NMF*

Con las variables:

-) **ite**: n° iteraciones.
-) **K**: n° prototipos.
-) ν, γ: parámetros de *SVM*.
-) $labels(c)$: etiquetas del conjunto C.

Y las funciones:

-) $hals\_nmf$: entrega prototipos (W), coeficientes (H), por descomposición *FAST HALS for NMF* de las imágenes ingresadas.

[12]Por el hecho de haber descompuesto *CE* con los algoritmos basados en *NMF*, automáticamente se obtiene una proyección, la matriz H; sin embargo si se descompone con *PCA*, debe ser proyectado en la reducción de componentes principales que son seleccionadas.

[13]Puede ser más de un método.

-) $\mathrm{add\_media\_desviacion}$: entrega los H's obtenidos, agregándoles una fila extra al final donde va la media y desviación estándar por cada imagen, en el respectivo H.

-) $proyecta\_hals\_nmf$: entrega proyección de las nuevas imágenes ingresadas, en los prototipos fijos, matriz de coeficientes (H).

-) KNN: entrega clasificador *KNN* según las características por observaciones ingresadas con sus etiquetas respectivas.

-) SVM: entrega clasificador *SVM* según las características por observaciones ingresadas con sus etiquetas respectivas.

-) $\mathrm{Predice\_}KNN$: entrega predicción de etiquetas según las características por observaciones ingresadas en el clasificador *KNN*.

-) $\mathrm{Predice\_}SVM$: entrega predicción de etiquetas según las características por observaciones ingresadas en el clasificador *SVM*.

-) $\mathrm{Valida\_}KNN$: entrega TPR, FPR y error de clasificación ($ERROR$), ingresando las predicciones del clasificador *KNN* y las etiquetas reales.

-) $\mathrm{Valida\_}SVM$: entrega TPR, FPR y error de clasificación ($ERROR$), ingresando las predicciones del clasificador *SVM* y las etiquetas reales.

El pseudo-código expuesto en forma de algoritmo para el método *FAST HALS for NMF* lo podemos ver en el *algoritmo* 3.

Teniendo la ***TPR***, ***FPR*** y ***ERROR*** es comparado con los otros 2 métodos.

3.3.2. Metodología con *HALS NMF Sparsity*

Con las variables:

-) **ite**: n° iteraciones.
-) **K**: n° prototipos.
-) α_{sp}: control de dispersión de *NMF*.
-) ν, γ: parámetros de *SVM*.
-) $labels(c)$: etiquetas del conjunto C.

Y las funciones:

-) $hals\_nmf\_sparsity$: entrega prototipos (W), coeficientes (H), por descomposición *HALS NMF Sparsity* de las imágenes ingresadas.

-) $\mathrm{add\_media\_desviacion}$: entrega los H's obtenidos, agregándoles una fila extra al final donde va la media y desviación estándar por cada imagen, en el respectivo H.

-) $proyecta\_hals\_nmf\_sparsity$: entrega proyección de las nuevas imágenes ingresadas,

en los prototipos fijos, matriz de coeficientes (H).

-) Con las funciones: KNN, SVM, $Predice\_KNN$, $Predice\_SVM$, $Valida\_KNN$ y $Valida\_SVM$ cumpliendo lo mismo que en *FAST HALS for NMF*.

El pseudo-código expuesto en forma de algoritmo para el método *HALS NMF Sparsity* lo podemos ver en el algoritmo 4.

Teniendo la ***TPR***, ***FPR*** y ***ERROR*** es comparado con los otros 2 métodos.

3.3.3. Metodología con *PCA*

Con las variables:

-) **n**: n° que representa a los **n** primeros prototipos de mayor varianza que cubren el 85 % de información contenida.
-) **m**: n° prototipos entregados por *NMF*.
-) ν, γ: parámetros de *SVM*.
-) $labels(c)$: etiquetas del conjunto C.

Y las funciones:

Algoritmo 3 : **Método *FAST HALS for NMF***

1: Obtención de $\mathbf{W}$ y $\mathbf{H}$ usando $hals\_nmf$ para CE.
2: $[\mathbf{W_1}, \mathbf{H_1(ce)}] = hals\_nmf(\mathbf{ite}, \mathbf{K}, \mathbf{IM_{ref}(ce)})$;
3: $[\mathbf{W_2}, \mathbf{H_2(ce)}] = hals\_nmf(\mathbf{ite}, \mathbf{K}, \mathbf{IM_{con}(ce)})$;
4: $[\mathbf{W_3}, \mathbf{H_3(ce)}] = hals\_nmf(\mathbf{ite}, \mathbf{K}, \mathbf{IM_{snr}(ce)})$;
5: $[\mathbf{H_1(ce)}, \mathbf{H_2(ce)}, \mathbf{H_3(ce)}] = a\text{dd\_}media\_desviacion(\mathbf{H_1(ce)}, \mathbf{H_2(ce)}, \mathbf{H_3(ce)})$;
6: Proyección de los $\mathbf{W}$ para CV;
7: $\mathbf{H_1(cv)} = proyecta\_hals\_nmf(\mathbf{ite}, \mathbf{W_1}, \mathbf{IM_{ref}(cv)})$;
8: $\mathbf{H_2(cv)} = proyecta\_hals\_nmf(\mathbf{ite}, \mathbf{W_2}, \mathbf{IM_{con}(cv)})$;
9: $\mathbf{H_3(cv)} = proyecta\_hals\_nmf(\mathbf{ite}, \mathbf{W_3}, \mathbf{IM_{snr}(cv)})$;
10: $[\mathbf{H_1(cv)}, \mathbf{H_2(cv)}, \mathbf{H_3(cv)}] = a\text{dd\_}media\_desviacion(\mathbf{H_1(cv)}, \mathbf{H_2(cv)}, \mathbf{H_3(cv)})$;
11: Agrupación de los $\mathbf{H}$ (todos de las mismas observaciones);
12: $\mathbf{H(ce)} = [\mathbf{H_1(ce)}; \mathbf{H_2(ce)}; \mathbf{H_3(ce)}]$;
13: $\mathbf{H(cv)} = [\mathbf{H_1(cv)}; \mathbf{H_2(cv)}; \mathbf{H_3(cv)}]$;
14: Creación de los clasificadores con $H(ce)$;
15: $\mathbf{model_1} = KNN(\mathbf{H(ce)}, \mathbf{label(ce)}, \nu, \gamma)$;
16: $\mathbf{model_2} = SVM(\mathbf{H(ce)}, \mathbf{label(ce)}, \nu, \gamma)$;
17: Predicción de los clasificadores con $H(cv)$;
18: $\mathbf{prediccion_1} = Predice\_KNN(\mathbf{model_1}, \mathbf{H(cv)})$
19: $\mathbf{prediccion_2} = Predice\_SVM(\mathbf{model_2}, \mathbf{H(cv)})$
20: Validación de los clasificadores con $label(cv)$;
21: $[\mathbf{TPR_1}, \mathbf{FPR_1}, \mathbf{ERROR_1}] = Valida\_KNN(\mathbf{prediccion_1}, \mathbf{label(cv)})$
22: $[\mathbf{TPR_2}, \mathbf{FPR_2}, \mathbf{ERROR_2}] = Valida\_SVM(\mathbf{prediccion_2}, \mathbf{label(cv)})$

-) pca: entrega prototipos (W), coeficientes (H) y el vector de varianzas (L), por descomposición *pca* de las imágenes ingresadas.

-) $calculo\_info\_contenida$: entrega el $\mathbf{n}$, ingresando vector de varianzas L.

-) $reduccion\_pca$: entrega una versión reducida de prototipos mediante ingresar $\mathbf{n}$ o bien $\mathbf{m}$ y W.

-) $proyecta\_pca$: entrega proyección de las nuevas imágenes ingresadas, en los prototipos reducidos, matriz de coeficientes (H).

-) $add\_media\_desviacion$: entrega los H's obtenidos, agregándoles una fila extra al final donde va la media y desviación estándar por cada imagen, en el respectivo H.

-) Con las funciones: KNN, SVM, $Predice\_KNN$, $Predice\_SVM$, $Valida\_KNN$ y $Valida\_SVM$ cumpliendo lo mismo que en *FAST HALS for NMF*.

El pseudo-código expuesto en forma de algoritmo para el método *PCA* lo podemos ver en el algoritmo 5.

Teniendo la ***TPR***, ***FPR*** y ***ERROR*** es comparado con los otros 2 métodos.

Algoritmo 4 : **Método *HALS NMF Sparsity***

Obtención de $\mathbf{W}$ y $\mathbf{H}$ usando $hals\_nmf$ para CE.
$[\mathbf{W_1}, \mathbf{H_1(ce)}] = hals\_nmf\_sparsity(\mathbf{ite}, \mathbf{K}, \mathbf{IM_{ref}(ce)}, \alpha_{\mathbf{sp}})$;
$[\mathbf{W_2}, \mathbf{H_2(ce)}] = hals\_nmf\_sparsity(\mathbf{ite}, \mathbf{K}, \mathbf{IM_{con}(ce)}, \alpha_{\mathbf{sp}})$;
$[\mathbf{W_3}, \mathbf{H_3(ce)}] = hals\_nmf\_sparsity(\mathbf{ite}, \mathbf{K}, \mathbf{IM_{snr}(ce)}, \alpha_{\mathbf{sp}})$;
$[\mathbf{H_1(ce)}, \mathbf{H_2(ce)}, \mathbf{H_3(ce)}] = add\_media\_desviacion(\mathbf{H_1(ce)}, \mathbf{H_2(ce)}, \mathbf{H_3(ce)})$;
Proyección de los $\mathbf{W}$ para CV;
$\mathbf{H_1(cv)} = proyecta\_hals\_nmf\_sparsity(\mathbf{ite}, \mathbf{W_1}, \mathbf{IM_{ref}(cv)}, \alpha_{\mathbf{sp}})$;
$\mathbf{H_2(cv)} = proyecta\_hals\_nmf\_sparsity(\mathbf{ite}, \mathbf{W_2}, \mathbf{IM_{con}(cv)}, \alpha_{\mathbf{sp}})$;
$\mathbf{H_3(cv)} = proyecta\_hals\_nmf\_sparsity(\mathbf{ite}, \mathbf{W_3}, \mathbf{IM_{snr}(cv)}, \alpha_{\mathbf{sp}})$;
$[\mathbf{H_1(cv)}, \mathbf{H_2(cv)}, \mathbf{H_3(cv)}] = add\_media\_desviacion(\mathbf{H_1(cv)}, \mathbf{H_2(cv)}, \mathbf{H_3(cv)})$;
Agrupación de los $\mathbf{H}$ (todos de las mismas observaciones);
$\mathbf{H(ce)} = [\mathbf{H_1(ce)}; \mathbf{H_2(ce)}; \mathbf{H_3(ce)}]$;
$\mathbf{H(cv)} = [\mathbf{H_1(cv)}; \mathbf{H_2(cv)}; \mathbf{H_3(cv)}]$;
Creación de los clasificadores con $H(ce)$;
$\mathbf{model_1} = KNN(\mathbf{H(ce)}, \mathbf{label(ce)}, \nu, \gamma)$;
$\mathbf{model_2} = SVM(\mathbf{H(ce)}, \mathbf{label(ce)}, \nu, \gamma)$;
Predicción de los clasificadores con $H(cv)$;
$\mathbf{prediccion_1} = Predice\_KNN(\mathbf{model_1}, \mathbf{H(cv)})$
$\mathbf{prediccion_2} = Predice\_SVM(\mathbf{model_2}, \mathbf{H(cv)})$
Validación de los clasificadores con $label(cv)$;
$[\mathbf{TPR_1}, \mathbf{FPR_1}, \mathbf{ERROR_1}] = Valida\_KNN(\mathbf{prediccion_1}, \mathbf{label(cv)})$
$[\mathbf{TPR_2}, \mathbf{FPR_2}, \mathbf{ERROR_2}] = Valida\_SVM(\mathbf{prediccion_2}, \mathbf{label(cv)})$

3.4. Comparación de los 3 métodos

Los resultados de cada modelo, se comparan en base a la clasificación realizada. Esta comparación es observada en una curva *ROC*, donde además se agrega un 4 método que es clasificar con todos los píxels (441) en *SVM*, observando que reducir dimensión es necesario. La comparación es hecha en base a la mejor configuración de cada método, esto es escoger el n° de prototipos para *PCA*, *FAST HALS for NMF*, y n° de prototipos junto con el control de dispersión para *HALS NMF Sparsity*.

Algoritmo 5 : **Método *PCA***

1: Obtención de $\mathbf{W}$, $\mathbf{H}$ y $\mathbf{L}$ usando pca para CE_1;
2: $[\mathbf{W_1}, \mathbf{H_1}(\mathbf{ce}), \mathbf{L_1}(\mathbf{ce})] = pca(\mathbf{IM_{ref}}(\mathbf{ce}))$;
3: $[\mathbf{W_2}, \mathbf{H_2}(\mathbf{ce}), \mathbf{L_2}(\mathbf{ce})] = pca(\mathbf{IM_{con}}(\mathbf{ce}))$;
4: $[\mathbf{W_3}, \mathbf{H_3}(\mathbf{ce}), \mathbf{L_3}(\mathbf{ce}))] = pca(\mathbf{IM_{snr}}(\mathbf{ce}))$;
5: Calculo de $\mathbf{n}$;
6: $\mathbf{n} = calculo\_info\_contenida(\mathbf{L})$;
7: Reducción de los prototipos $\mathbf{W}$, con $\mathbf{j} = \mathbf{n}$ ó $\mathbf{m}$;
8: $\overline{\mathbf{W}}_\mathbf{1} = reduccion\_pca(\mathbf{j}, \mathbf{W_1})$;
9: $\overline{\mathbf{W}}_\mathbf{2} = reduccion\_pca(\mathbf{j}, \mathbf{W_2})$;
10: $\overline{\mathbf{W}}_\mathbf{3} = reduccion\_pca(\mathbf{j}, \mathbf{W_3})$;
11: Proyección de los $\overline{\mathbf{W}}$ para CE;
12: $\overline{\mathbf{H}}_\mathbf{1}(\mathbf{ce}) = proyecta\_pca(\overline{\mathbf{W}}_\mathbf{1}, \mathbf{IM_{ref}}(\mathbf{ce}))$;
13: $\overline{\mathbf{H}}_\mathbf{2}(\mathbf{ce}) = proyecta\_pca(\overline{\mathbf{W}}_\mathbf{2}, \mathbf{IM_{con}}(\mathbf{ce}))$;
14: $\overline{\mathbf{H}}_\mathbf{3}(\mathbf{ce}) = proyecta\_pca(\overline{\mathbf{W}}_\mathbf{3}, \mathbf{IM_{snr}}(\mathbf{ce}))$;
15: $[\overline{\mathbf{H}}_\mathbf{1}(\mathbf{ce}), \overline{\mathbf{H}}_\mathbf{2}(\mathbf{ce}), \overline{\mathbf{H}}_\mathbf{3}(\mathbf{ce})] = a\text{dd\_}media\_desviacion(\overline{\mathbf{H}}_\mathbf{1}(\mathbf{ce}), \overline{\mathbf{H}}_\mathbf{2}(\mathbf{ce}), \overline{\mathbf{H}}_\mathbf{3}(\mathbf{ce}))$;
16: Proyección de los $\overline{\mathbf{W}}$ para CV;
17: $\mathbf{H}_\mathbf{1}(\mathbf{cv}) = proyecta\_pca(\overline{\mathbf{W}}_\mathbf{1}, \mathbf{IM_{ref}}(\mathbf{cv}))$;
18: $\mathbf{H}_\mathbf{2}(\mathbf{cv}) = proyecta\_pca(\overline{\mathbf{W}}_\mathbf{2}, \mathbf{IM_{con}}(\mathbf{cv}))$;
19: $\mathbf{H}_\mathbf{3}(\mathbf{cv}) = proyecta\_pca(\overline{\mathbf{W}}_\mathbf{3}, \mathbf{IM_{snr}}(\mathbf{cv}))$;
20: $[\mathbf{H}_\mathbf{1}(\mathbf{cv}), \mathbf{H}_\mathbf{2}(\mathbf{cv}), \mathbf{H}_\mathbf{3}(\mathbf{cv})] = a\text{dd\_}media\_desviacion(\mathbf{H}_\mathbf{1}(\mathbf{cv}), \mathbf{H}_\mathbf{2}(\mathbf{cv}), \mathbf{H}_\mathbf{3}(\mathbf{cv}))$;
21: Agrupación de los $\mathbf{H}$(son de las mismas observaciones);
22: $\mathbf{H}(\mathbf{ce}) = [\overline{\mathbf{H}}_\mathbf{1}(\mathbf{ce}); \overline{\mathbf{H}}_\mathbf{2}(\mathbf{ce}); \overline{\mathbf{H}}_\mathbf{3}(\mathbf{ce})]$;
23: $\mathbf{H}(\mathbf{cv}) = [\mathbf{H}_\mathbf{1}(\mathbf{cv}); \mathbf{H}_\mathbf{2}(\mathbf{cv}); \mathbf{H}_\mathbf{3}(\mathbf{cv})]$;
24: Creación de los clasificadores con $H(ce)$;
25: $\mathbf{model_1} = KNN(\mathbf{H}(\mathbf{ce}), \mathbf{label}(\mathbf{ce}), \nu, \gamma)$;
26: $\mathbf{model_2} = SVM(\mathbf{H}(\mathbf{ce}), \mathbf{label}(\mathbf{ce}), \nu, \gamma)$;
27: Predicción de los clasificadores con $H(cv)$;
28: $\mathbf{prediccion_1} = Predice\_KNN(\mathbf{model_1}, \mathbf{H}(\mathbf{cv}))$
29: $\mathbf{prediccion_2} = Predice\_SVM(\mathbf{model_2}, \mathbf{H}(\mathbf{cv}))$
30: Validación de los clasificadores con $label(cv)$;
31: $[\mathbf{TPR_1}, \mathbf{FPR_1}, \mathbf{ERROR_1}] = Valida\_KNN(\mathbf{prediccion_1}, \mathbf{label}(\mathbf{cv}))$
32: $[\mathbf{TPR_2}, \mathbf{FPR_2}, \mathbf{ERROR_2}] = Valida\_SVM(\mathbf{prediccion_2}, \mathbf{label}(\mathbf{cv}))$

3.5. Ajuste fino del mejor método

Con el método final seleccionado se emplean modificaciones en alguno de los procesos de este a fin de minimizar aun mas la ***FPR*** y el ***ERROR*** y aumentar la ***TPR***.

La primera modificación es con las imágenes pre-procesadas, consiste en probar transformaciones del tipo: rotación[14], traslación[15] y la combinación de estas. Todas estas transformaciones son hechas sólo en la imagen señal a ruido[16] del conjunto *CE*, y así obtener más prototipos que capten la invarianza en la representación de estrellas y no-estrellas.

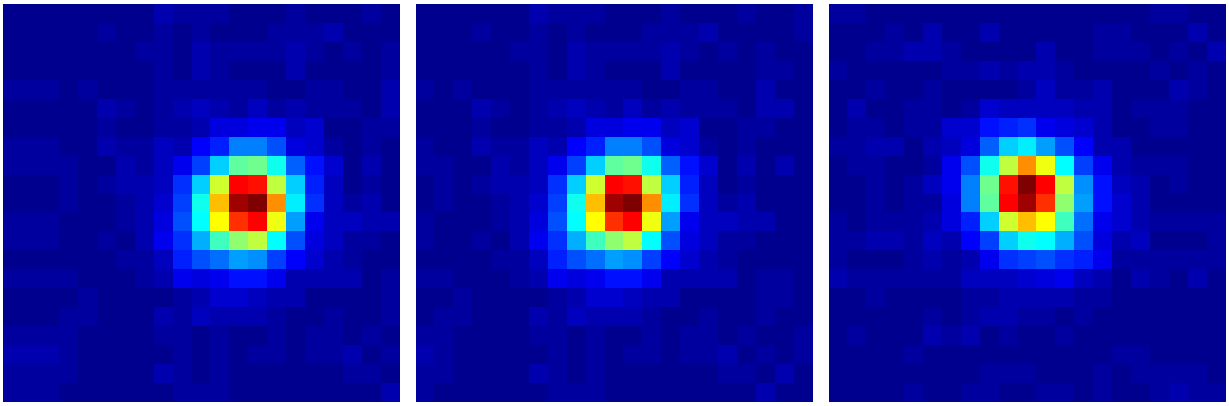
Para igualar en cantidad con las imágenes de referencias y las convolucionadas, a las imágenes IM_{SNR} artificiales se les agrega las misma de referencia y convolucionada que corresponden a la original de IM_{SNR}. En la figura 3.5 se observa los 2 tipos de transformaciones[17] junto con la imagen original.

[14]Con 3 rotaciones hechas: 90°, 180° y 270°

[15]Traslación de un píxel en las 4 direcciones, intentar con mas píxels no supera al anterior.

[16]Hacerlo en las imágenes de referencia o la convolucionada, no tiene sentido, pues no siempre éstas se encuentran centradas.

[17]Para notar un real efecto visual, se realiza con 2 píxels hacia la derecha.



(a) (b) (c)

Figura 3.5: (a) imagen trasladada en 2 píxel, (b) imagen original y (c) imagen rotada en 90°

Capítulo 4

Resultados

En este capítulo se presentan los resultados de haber usado los 3 métodos descritos en la metodología y la comparación entre estos por medio de la clasificación. Una vez seleccionado el mejor método se realiza ajustes finos para mejorar los resultados, los que incluye transformaciones en las imágenes para lograr invarianza y por consecuencia aumentar la base de datos.

Los resultados de este capítulo son presentados en 4 secciones:

1. **Resultados de ajustar *SVM*:** En esta sección son mostrados los errores de clasificación (***ERROR***, ***FPR***) al entrenar *SVM*, para seleccionar los parámetros adecuados.

2. **Resultados de ajustar *NMF* y *PCA*:** En esta sección vemos los prototipos seleccionados para *PCA* y *FAST HALS for NMF* en cuanto a la clasificación entregada por ambos. Además se obtiene la combinación entre n° de prototipos y el control de dispersión en *HALS NMF Sparsity* para obtener la clasificación requerida.

3. **Resultados de comparación entre los 3 métodos:** En esta sección vemos los resultados de clasificación por cada método, habiéndolo ajustado previamente. Son comparados las mejores versiones de los 3 métodos ajustado.

4. **Resultados de ajustar el método seleccionado y prueba final:** En esta sección se presentan los resultados al realizar el ajuste fino del modelo seleccionado, de manera de bajar aun más el error de clasificación y la tasa de falsos positivos. En la última parte se muestra los resultados de haber aplicado estos ajustes a un conjunto de prueba (conjunto no utilizado en ninguna parte anteriormente).

4.1. Resultados de ajustar *SVM*

Dado que *SVM* necesita recibir las proyecciones por parte de los 3 algoritmos para ser ajustado, los parámetros ν y g son independientes para cada método.

Índice	ν	g	*FPR*	*TPR*
6	0.02	0.09	0.304 %	94.72 %
8	0.02	0.08	0.328 %	95.568 %
9	0.02	0.04	0.36 %	98.016 %
15	0.01	0.03	0.392 %	98.36 %
16	0.01	0.06	0.392 %	97.024 %
17	0.02	0.06	0.392 %	97.032 %
18	0.01	0.05	0.4 %	97.6 %
19	0.02	0.03	0.408 %	98.368 %
20	0.02	0.02	0.464 %	98.76 %
21	0.02	0.01	0.568 %	99.04 %

Tabla 4.1: Curva *ROC* para los valores de ajuste de *SVM* en la proyección entregada por *HALS NMF Sparsity*. Se muestran los valores con mejores *FPR* y *TPR*. La columna Índice es puesta a propósito para que el punto en cuestión sea observado en la figura 4.1

Partiendo con *HALS NMF Sparsity*, al reducir a 20 características (i.e., 20 prototipos en cada una de las 3 descomposiciones *NMF* realizadas) de acuerdo a los valores para ν y g planteados en la metodología se observa la curva *ROC* en la figura 4.1 con los datos de la tabla 4.1. De los observado y de varias simulaciones hechas, los parámetros de *SVM* escogidos para la proyección *HALS NMF Sparsity* son $\nu = 0{,}01$ $g = 0{,}03$. Este mismo análisis fue realizado al separar el conjunto proyectado entre las que son de baja señal y las de alta señal, las que se pueden observar en las figuras 4.2 y 4.3 respectivamente. Basado en lo anterior, probando distinto número de características (prototipos), se obtuvo que para la proyección de *HALS NMF Sparsity* una buena clasificación en promedio se obtiene con $\nu = 0{,}01$ $g = 0{,}03$. Del mismo modo esto fue realizado para *FAST HALS for NMF* y *PCA* dando los mismos parámetros. Además en los ajustes de *NMF* y *PCA* se mostrará que es más importante el n° de prototipos escogido mas que los parámetros de *SVM*[1].

4.2. Resultados de ajustar *NMF* y *PCA*

4.2.1. *NMF*

Para ajustar el n° de prototipos y n° de iteraciones de *NMF*, es que siempre se están calculando 3 descomposiciones sea cual sea el conjunto con el que se trabaje, estas son para la imagen de referencia, la imagen convolucionada y la imagen señal a ruido. Dada la complejidad del problema en cuanto a combinaciones posibles entre prototipos y los 3 *NMF* uno por cada una de las 3 imágenes, se opta solo por ajustar *NMF* con la imagen señal a ruido y a posterior se verificó que esto era suficiente para lograr los objetivos específicos. También fue verificado que asignar los parámetros ajustados de *NMF* con la imagen señal a ruido a los otros *NMF* de la imagen de referencia y la convolucionada dieron resultados que cumplieron los objetivos específicos.

[1]Dentro de los planteados en la metodología.

N° de prototipos	*ERROR*
10	2.13 %
20	1.87 %
30	1.70 %
40	1.60 %
50	1.48 %
100	1.25 %

Tabla 4.2: En la columna dos se observan los errores cuadráticos relativos para cada n° de prototipos respectivo, utilizando 1000 iteraciones.

El error de reconstrucción relativo (ecuación (3.1)) va disminuyendo a medida que la descomposición es con mas prototipos, aunque lo que es tomado en cuanto para seleccionar n° de prototipos es obtener una buena clasificación. Se pueden observar errores de reconstrucción en *FAST HALS for NMF* para el n° de prototipos en la tabla 4.2. De igual modo se observan algunos de los errores obtenidos para *HALS NMF Sparsity* en las combinaciones de n° de prototipos y control de dispersión en la tabla 4.3.

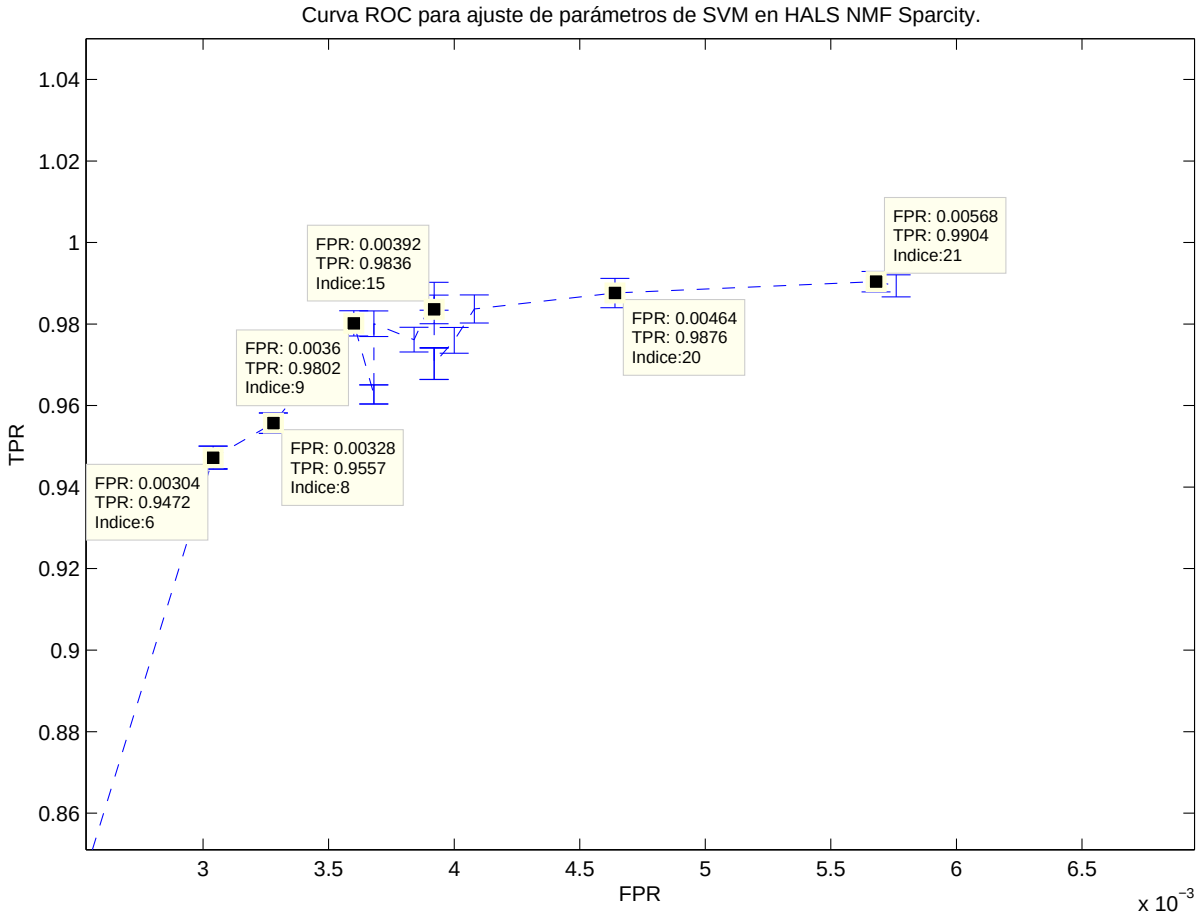


Figura 4.1: Curva *ROC* de los mejores valores en la búsqueda de parámetros para *SVM* con la proyección de *HALS NMF Sparsity*.

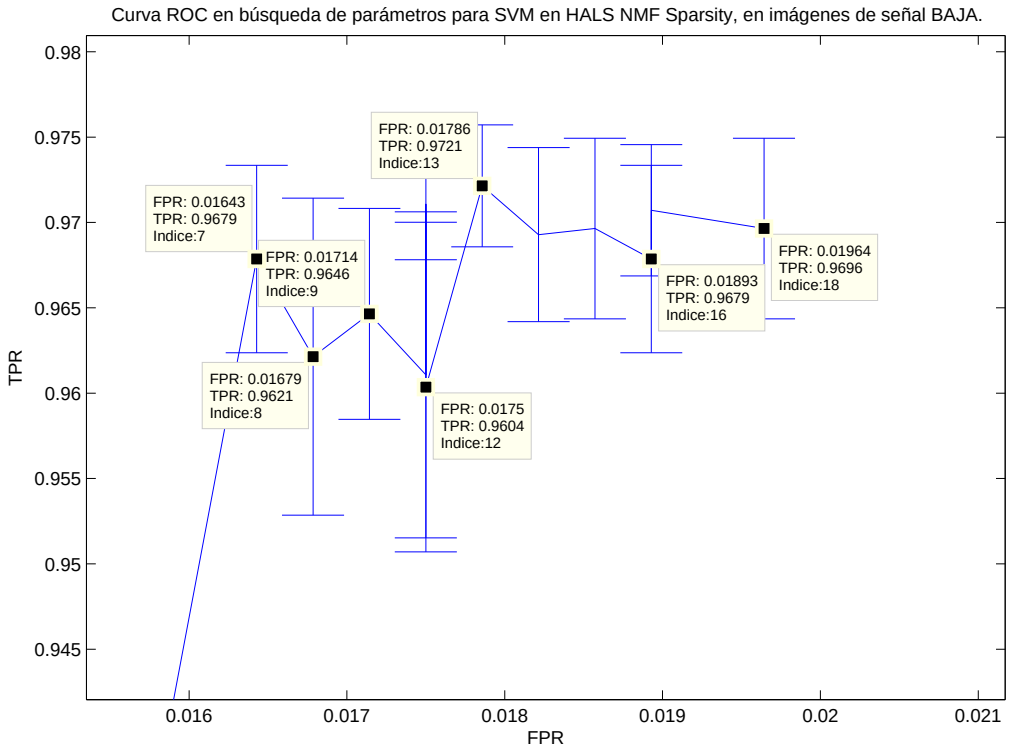


Figura 4.2: Curva *ROC* de los mejores valores en la búsqueda de parámetros para *SVM* cuando se proyecta con *HALS NMF Sparsity* con las imágenes de señal baja.

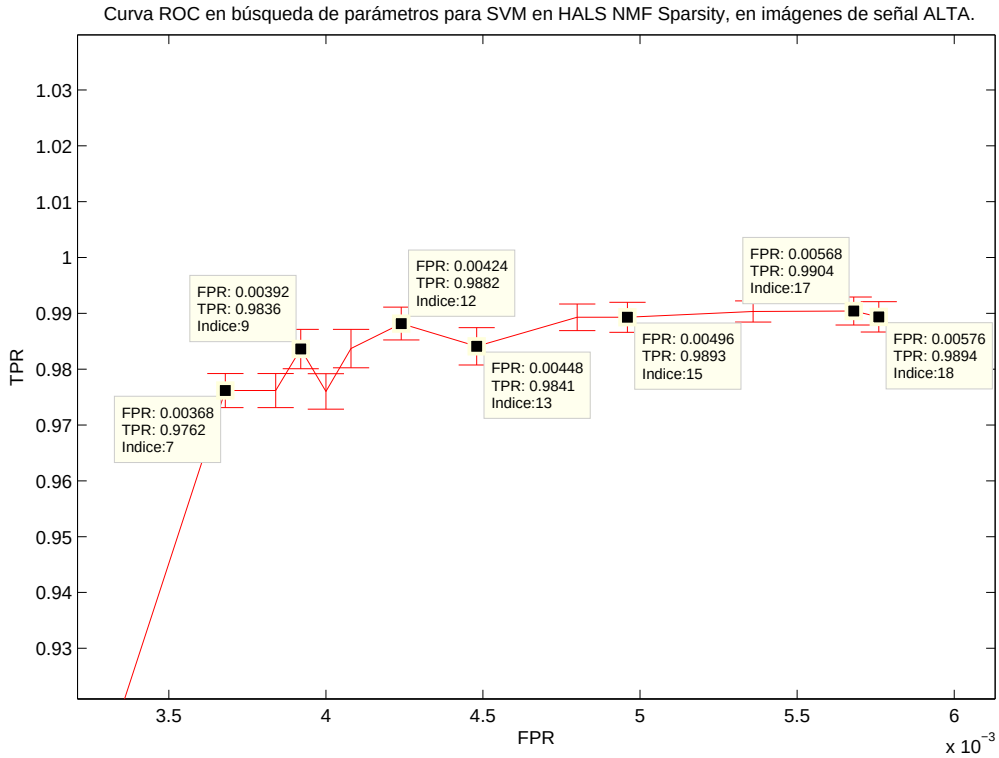


Figura 4.3: Curva *ROC* de los mejores valores en la búsqueda de parámetros para *SVM* cuando se proyecta con *HALS NMF Sparsity* con las imágenes de señal alta.

N° prototipos	α_{sp}	ERROR
10	0	2.12 %
10	0.3	2.54 %
20	0	1.85 %
20	0.3	4.23 %
30	0	1.703 %
30	0.3	2.48 %
40	0.5	4.19 %

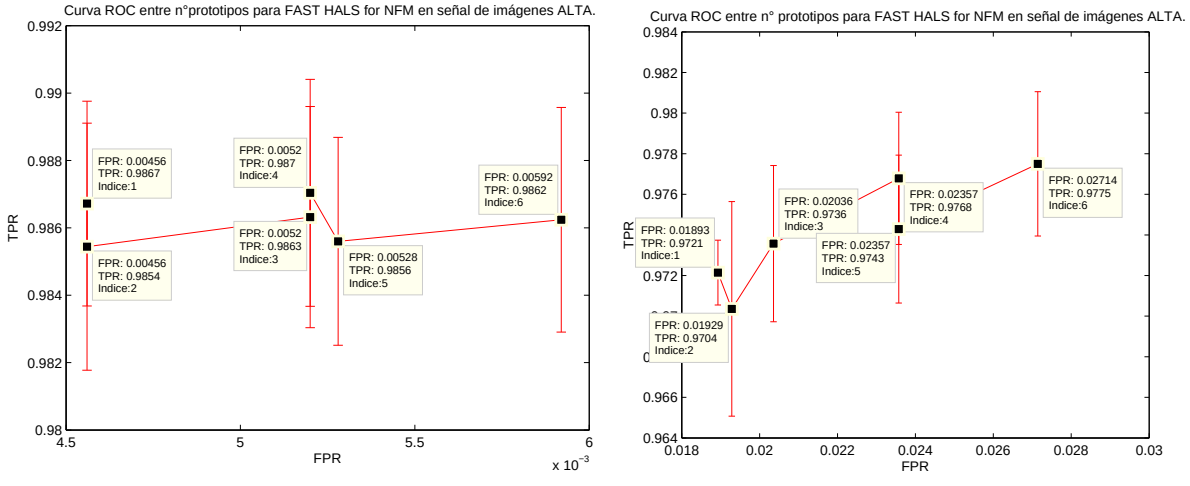
Tabla 4.3: Valores del error cuadrático relativo de reconstrucción para *HALS NMF Sparsity* con las combinaciones prototipos-control de dispersión.

Índice	N° prototipos	FPR	TPR
1	20	0.456 %	98.672 %
2	40	0.456 %	98.544 %
3	50	0.52 %	98.632 %
4	100	0.52 %	98.704 %
5	30	0.528 %	98.56 %
6	10	0.592 %	98.624 %

Índice	N° prototipos	*FPR*	*TPR*
1	20	1.892 %	97.21 %
2	10	1.920 %	97.03 %
3	30	2.035 %	97.35 %
4	50	2.357 %	97.67 %
5	40	2.357 %	97.42 %
6	100	2.714 %	97.75 %

Tabla 4.4: Errores de clasificación con distintos N° de prototipos en *FAST HALS for NMF*. En la tabla de la izquierda es con las imágenes de señal alta y la derecha es para las de señal baja. La columna **Índice** tiene directa relación con los puntos de los gráficos mostrados en la figura 4.4.

Para seleccionar el n° de prototipos en *FAST HALS for NMF* se probaron 10, 20, 30, 40, 50 y 100 prototipos, los resultados en cuanto a *FPR* y *TPR* se observan en la tabla 4.4 y en la figura 4.4. Entre los gráficos observados y simulaciones realizadas se escogen 10 prototipos para la reducción *FAST HALS for NMF* como mejor configuración.



(a) (b)

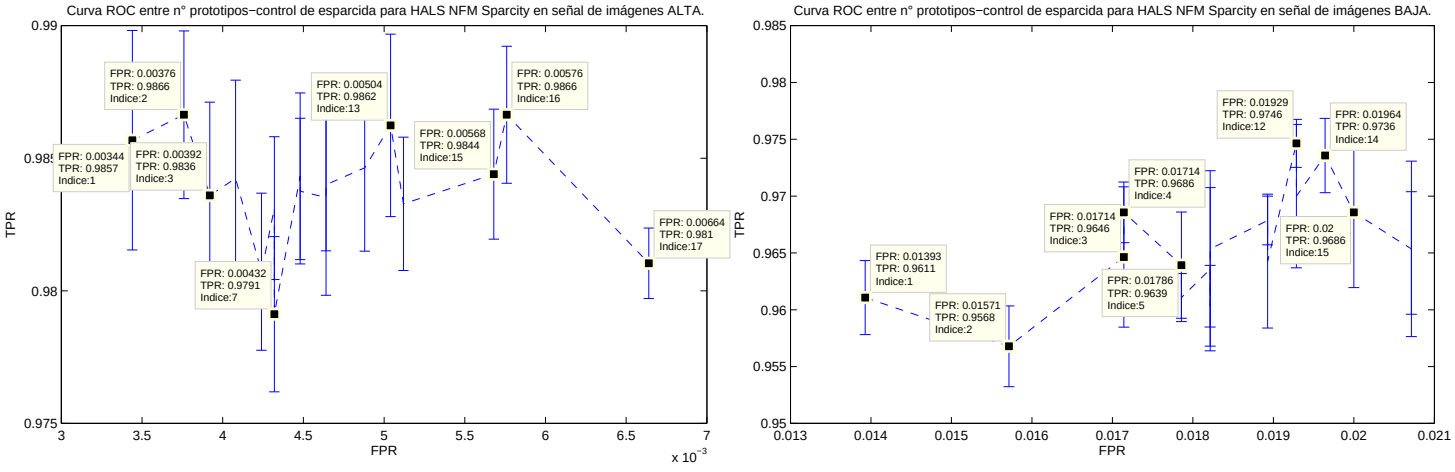
Figura 4.4: Curva *ROC* para ajustar n° de prototipo en *FAST HALS for NMF*, con imágenes de: (a) Alta señal y (b) Baja señal.

Índice	n°P	α_{sp}	*FPR*	*TPR*
1	20	0.1	0.344 %	98.568 %
2	30	0.1	0.376 %	98.664 %
3	20	0.3	0.392 %	98.36 %
4	40	0.3	0.408 %	98.424 %
5	20	1	0.424 %	98.072 %
6	20	0.5	0.432 %	98.312 %
7	30	1	0.432 %	97.912 %
8	30	0.3	0.448 %	98.432 %
9	30	0.5	0.448 %	98.376 %
10	10	0.3	0.464 %	98.352 %
11	10	0.5	0.464 %	98.4 %
12	10	0.1	0.488 %	98.464 %
13	20	0	0.504 %	98.624 %
14	40	0.5	0.512 %	98.328 %
15	10	1	0.568 %	98.44 %
16	10	0	0.576 %	98.664 %
17	10	2	0.664 %	98.104 %

Índice	n°P	α_{sp}	*FPR*	*TPR*
1	10	0.5	1.39 %	96.10 %
2	10	2	1.57 %	95.67 %
3	20	0.3	1.71 %	96.46 %
4	30	0.5	1.71 %	96.85 %
5	30	0.3	1.78 %	96.39 %
6	30	1	1.78 %	96.10 %
7	10	0.3	1.82 %	96.35 %
8	20	0.5	1.82 %	96.03 %
9	30	0.1	1.82 %	96.53 %
10	20	0.1	1.89 %	96.78 %
11	10	1	1.89 %	96.42 %
12	20	0	1.92 %	97.46 %
13	40	0.5	1.92 %	97 %
14	40	0.3	1.96 %	97.35 %
15	10	0	2 %	96.85 %
16	10	0.1	2.07 %	96.53 %
17	20	1	2.07 %	96.5 %

Tabla 4.5: Errores de clasificación con distintos combinaciones de N° de prototipos (n°P)-control de dispersión (α_{sp}) en *HALS NMF Sparsity*. En la tabla de la izquierda es con las imágenes de alta señal y la derecha es para las de baja señal. La columna **Índice** tiene directa relación con los puntos de los gráficos mostrados en la figura 4.5.

En cuanto a *HALS NMF Sparsity* podemos observar los errores de clasificación tanto en las imágenes de baja y alta señal en la tabla 4.5 y en la figura 4.5. Entre los gráficos observados y simulaciones realizadas se escogen 20 prototipos con $\alpha_{sp} = 0{,}3$ de control de dispersión para la reducción *HALS NMF Sparsity* como mejor configuración.



(a) (b)

Figura 4.5: Curva *ROC* para ajustar n° de prototipo-control de dispersión en *HALS NMF Sparsity*, con imágenes de: (a) Alta señal y (b) Baja señal.

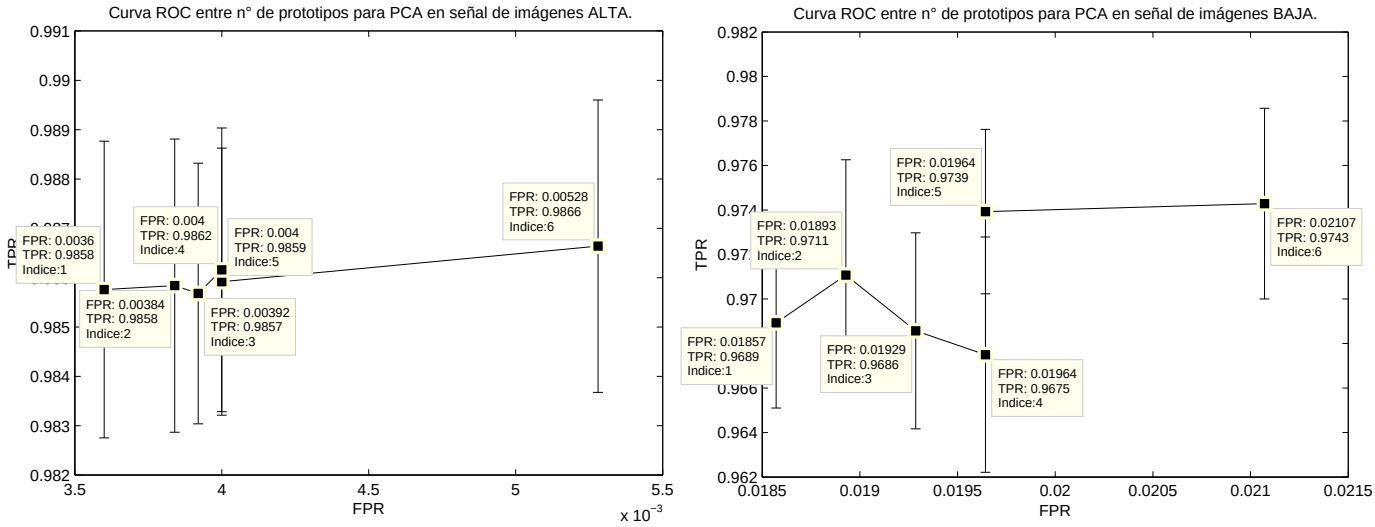
Índices	N° prototipos	*FPR*	*TPR*
1	30	0.36 %	98.576 %
2	40	0.384 %	98.584 %
3	50	0.392 %	98.568 %
4	20	0.4 %	98.616 %
5	100	0.4 %	98.592 %
6	10	0.528 %	98.664 %

Índices	N° prototipos	*FPR*	*TPR*
1	20	1.85 %	96.89 %
2	40	1.89 %	97.10 %
3	30	1.92 %	96.85 %
4	10	1.96 %	96.75 %
5	50	1.96 %	97.39 %
6	100	2.10 %	97.42 %

Tabla 4.6: Errores de clasificación con distintos combinaciones de N° de prototipos en *PCA*. En la tabla de la izquierda es con las imágenes de alta señal y la derecha es para las de baja señal. La columna **Índice** tiene directa relación con los puntos de los gráficos mostrados en la figura 4.6.

4.2.2. *PCA*

Los errores de clasificación en imágenes de alta y baja señal para un n° distinto de prototipos en *PCA* pueden ser observadas en la tabla 4.6 y figura 4.6. De los gráficos observados y simulaciones realizadas se escogen 10 prototipos para la reducción *PCA* como mejor configuración.



(a) (b)

Figura 4.6: Curva *ROC* para ajustar n° de prototipo-control de dispersión en *HALS NMF Sparsity*, con imágenes de: (a) Alta señal y (b) Baja señal.

4.3. Resultados de comparación entre los 3 métodos

A continuación se presentan los resultados por parte de los 3 métodos con sus parámetros ajustados. Recordando que para entrenar se utilizó el conjunto ***CE*** y para validar

CV. El conjunto ***CP*** es dejado para los métodos que mejor clasifican y se utilizará en la siguiente sección.

4.3.1. Resultados para *PCA*

Con la configuración seleccionada para *PCA*, se tienen que sus prototipos son los de la figura 4.7. En cada proyección se guardo el valor máximo para crear histogramas basados en los prototipos que mas predominan en las descomposiciones. Se observan histogramas de los máximos en la descomposición de la imagen señal a ruido en la figura 4.8, se observa que tanto para estrellas como para no estrellas los prototipos que más se repiten son el 1 y 2.

De igual manera se observan los histogramas para la descomposición *PCA* de la imagen de referencia y la imagen convolucionada en las figuras 4.9 y 4.10 respectivamente. Lo que podemos observar en general es que *PCA* no genera una discriminación tan explícita con respecto a los prototipos, pues dentro de los prototipos de la imagen señal a ruido, los

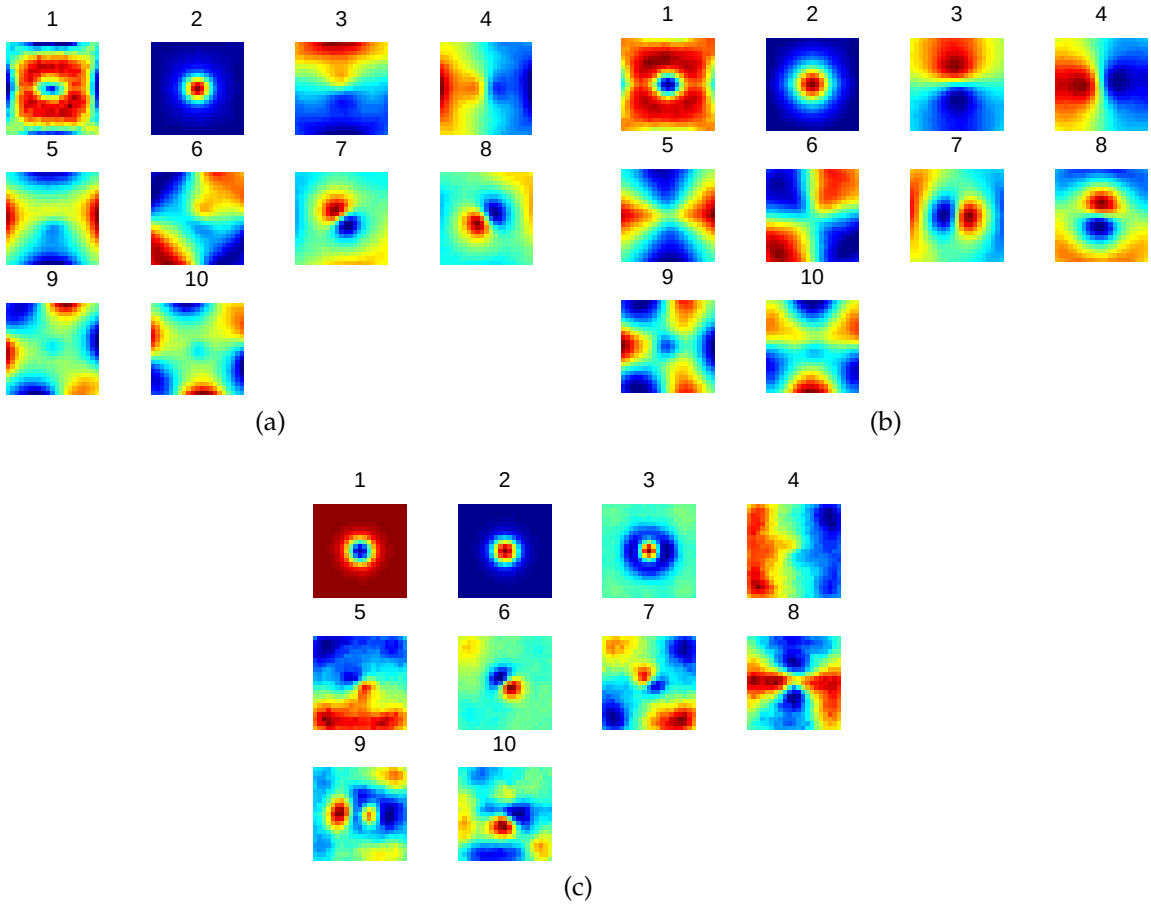


Figura 4.7: Prototipos de *PCA*, con 10 en: (a) la Imagen de referencia,(b) Imagen Convolucionada e (c) Imagen señal a ruido. Cada prototipo es numerado a propósito para a posterior observar histogramas en las figuras 4.8, 4.9, 4.10.

que poseen forma de estrella segura son el 1, 2 y podría ser el 3, sin embargo tanto la mayoría de estrellas como no-estrellas poseen sus máximos en el prototipo uno. De la misma forma en los histogramas de la imagen de referencia se observa que el prototipo uno tiene una fuerte tendencia tanto en estrellas como no-estrellas y para el histograma de la imagen convolucionada también se repite mayoritariamente el prototipo uno[2]. Sin embargo también se muestra que existen mas prototipos que hacen la diferencia entre estrellas y no-estrellas, como máximos repetidos.

[2]Notar que el prototipo uno de la imagen de referencia y la imagen convolucionada son distintos.

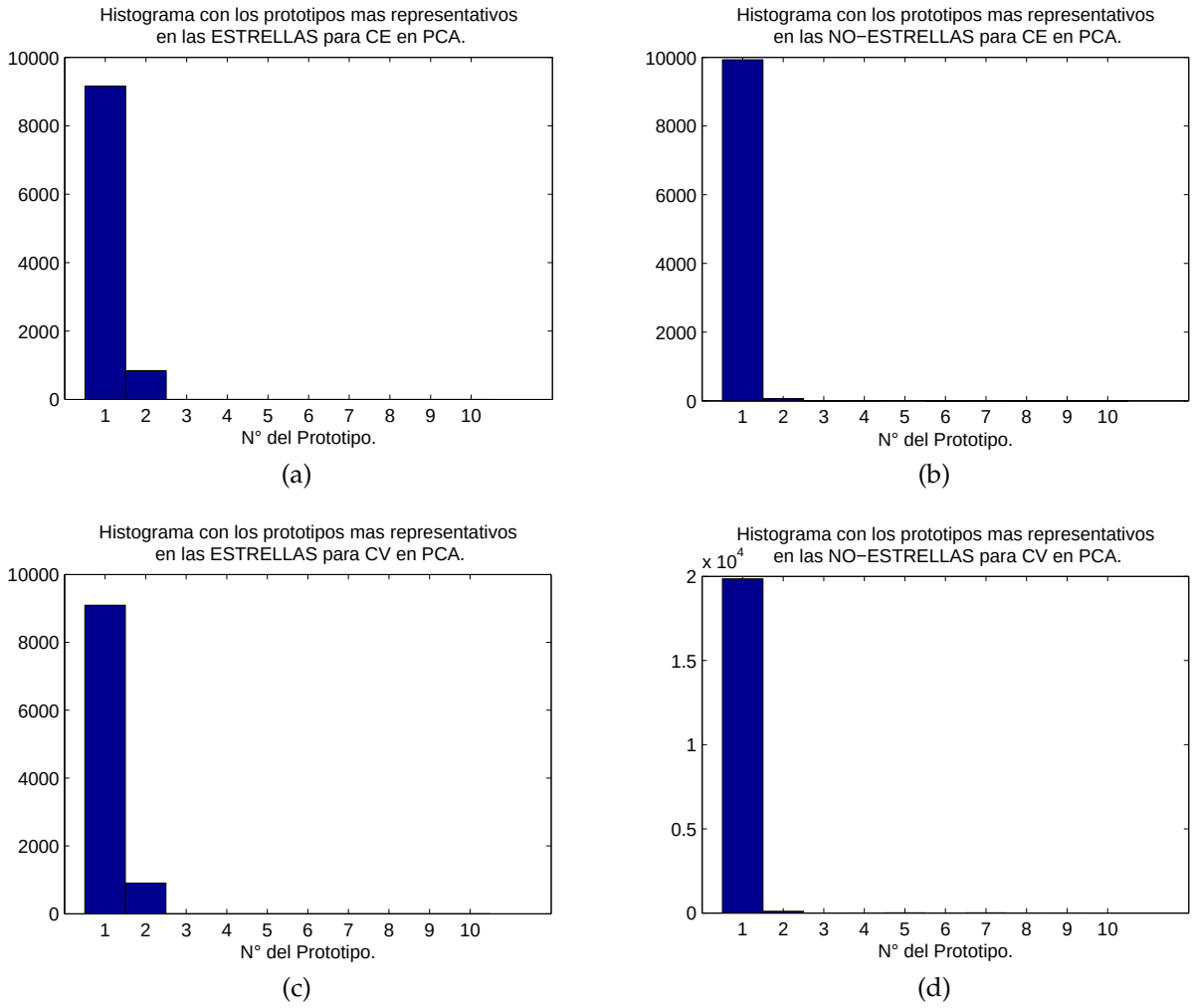
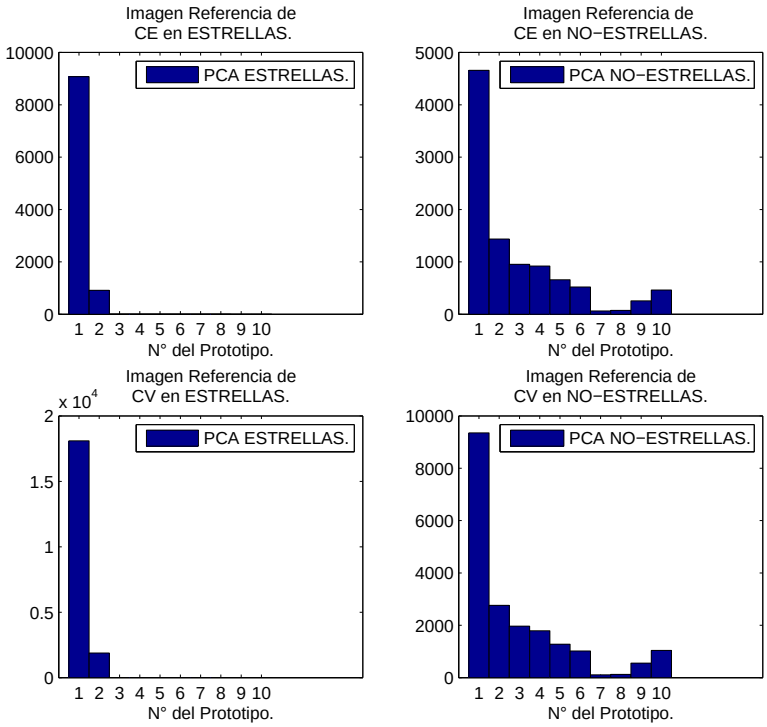
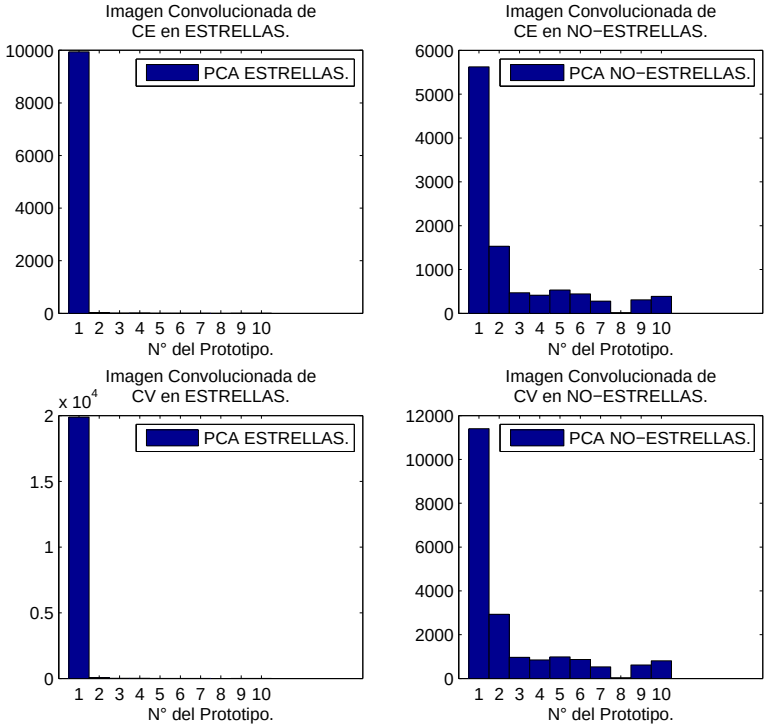


Figura 4.8: Histogramas con los prototipos mas repetidos como máximos en las proyecciones de *PCA* en la imagen señal a ruido, los histogramas son en: (a) Estrellas de *CE*,(b) No-estrellas de *CE*, (c) Estrellas de *CV* y (d) No-estrellas de *CV*



(a)

Figura 4.9: Histogramas para la imagen de referencia en la descomposición *PCA*.



(a)

Figura 4.10: Histogramas para la imagen convolucionada en la descomposición *PCA*.

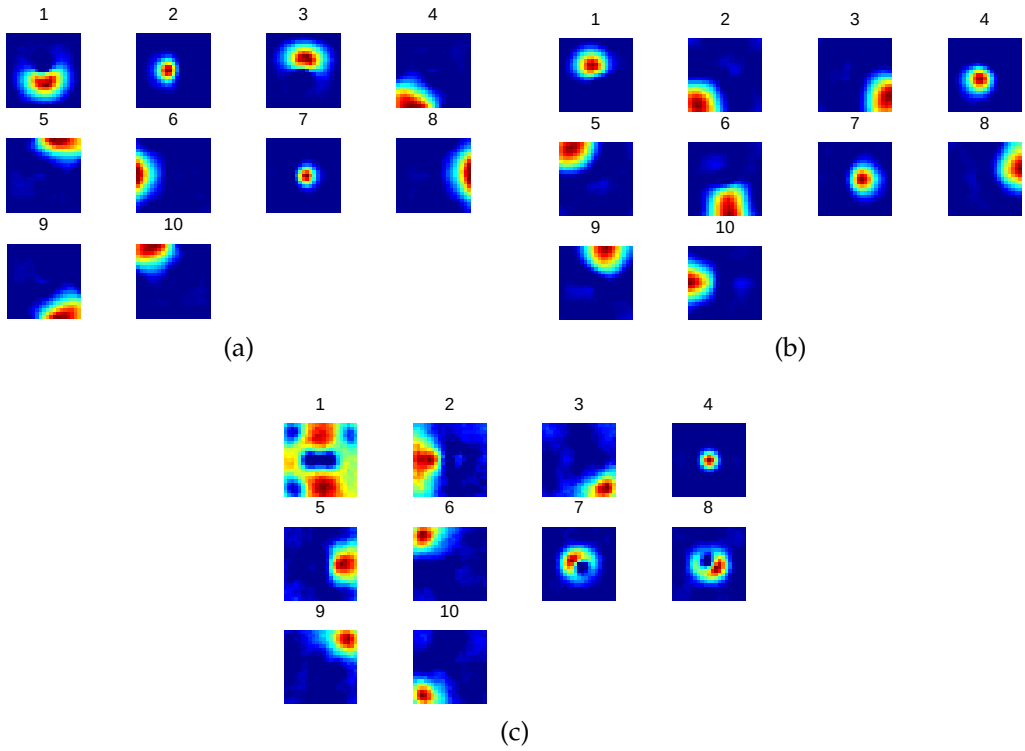


Figura 4.11: Prototipos de *FAST HALS for NMF*, con 10 en: (a) la Imagen de referencia,(b) Imagen Convolucionada e (c) Imagen señal a ruido. Cada prototipo es numerado a propósito para a posterior observar histogramas en las figuras 4.12, 4.13, 4.14.

4.3.2. Resultados para *FAST HALS for NMF*

Con la configuración seleccionada para *FAST HALS for NMF*, se tienen que sus prototipos son los de la figura 4.11. En cada proyección se guardo el valor máximo para crear histogramas basados en los prototipos que mas predominan en las descomposiciones.

Se observa histograma de los máximos en la descomposición de la imagen señal a ruido en la figura 4.12. Lo que podemos observar para *FAST HALS for NMF* es que un prototipo predomina tanto para estrellas como para no-estrellas, sin embargo el prototipo 4 hace la diferencia entre estrellas y no-estrellas y esto se observa en los conjuntos *CE* y *CV*.

De igual manera se observan los histogramas para la descomposición *FAST HALS for NMF* de la imagen de referencia y la imagen convolucionada en las figuras 4.13 4.14 respectivamente. Con respecto al histograma de la imagen de referencia se observa que hay una diferencia bastante notoria entre los prototipos que son predominantes en las no-estrellas y los de las estrellas, por ejemplo el prototipo 7 es el más alto en las estrellas, sin embargo llega tan solo a la mitad en las no-estrellas y el prototipo 4 es el más alto en las no-estrellas y es uno de los más bajos en las estrellas. Para el histograma de la imagen convolucionada se puede observar que hay bastante distribución entre todos los prototipos, aunque los prototipos 3 y 4 marcan una diferencia entre estrellas y no estrellas.

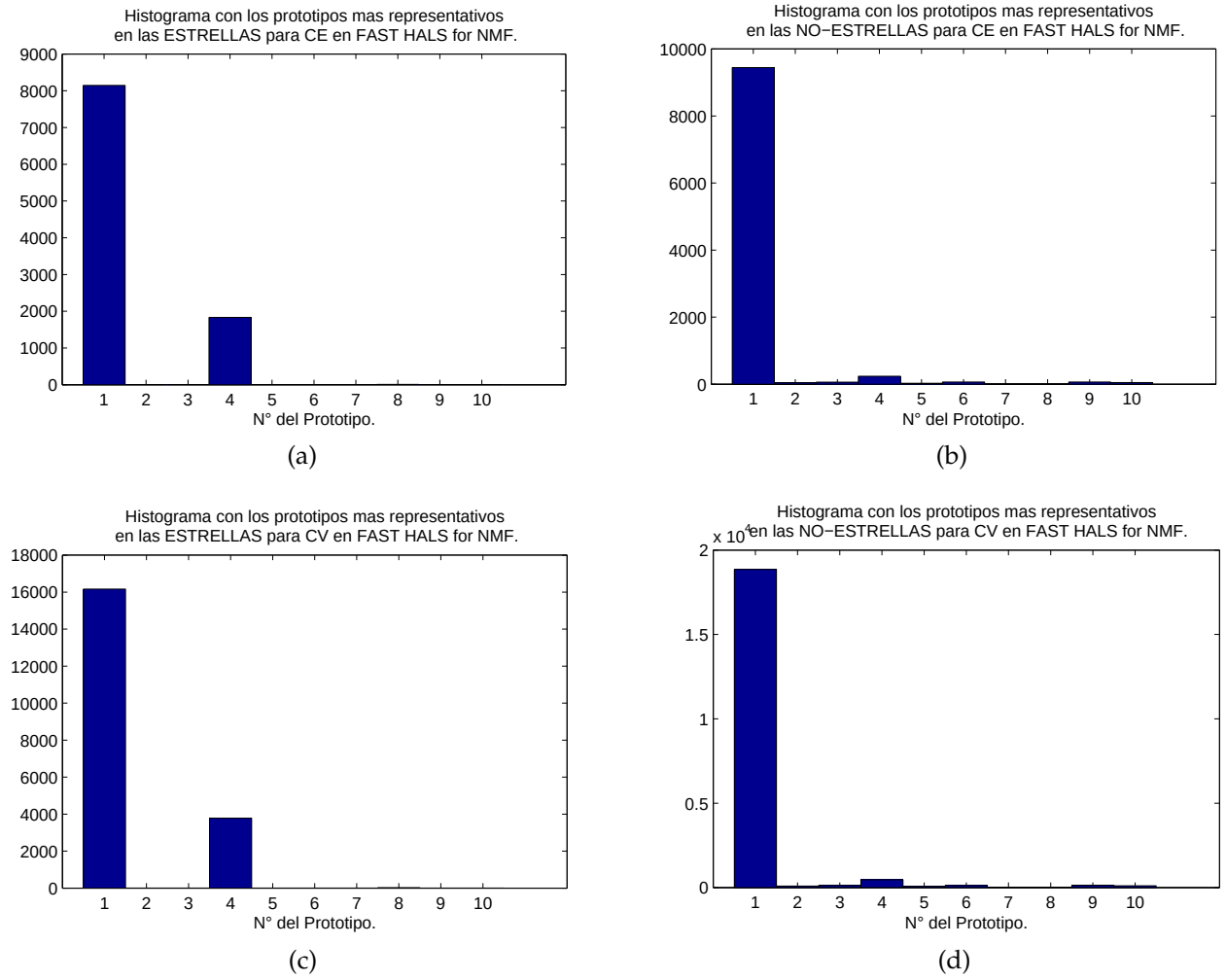
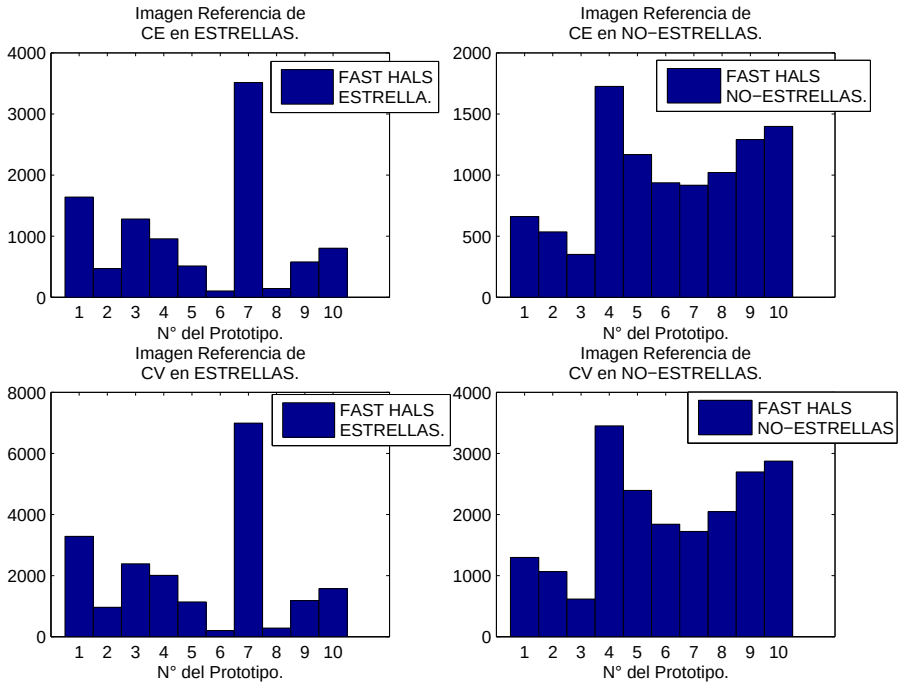


Figura 4.12: Histogramas con los prototipos mas repetidos como máximos en las proyecciones de *FAST HALS for NMF* en la imagen señal a ruido, los histogramas son en: (a) Estrellas de *CE*,(b) No-estrellas de *CE*, (c) Estrellas de *CV* y (d) No-estrellas de *CV*

4.3.3. Resultados para *HALS NMF Sparsity*

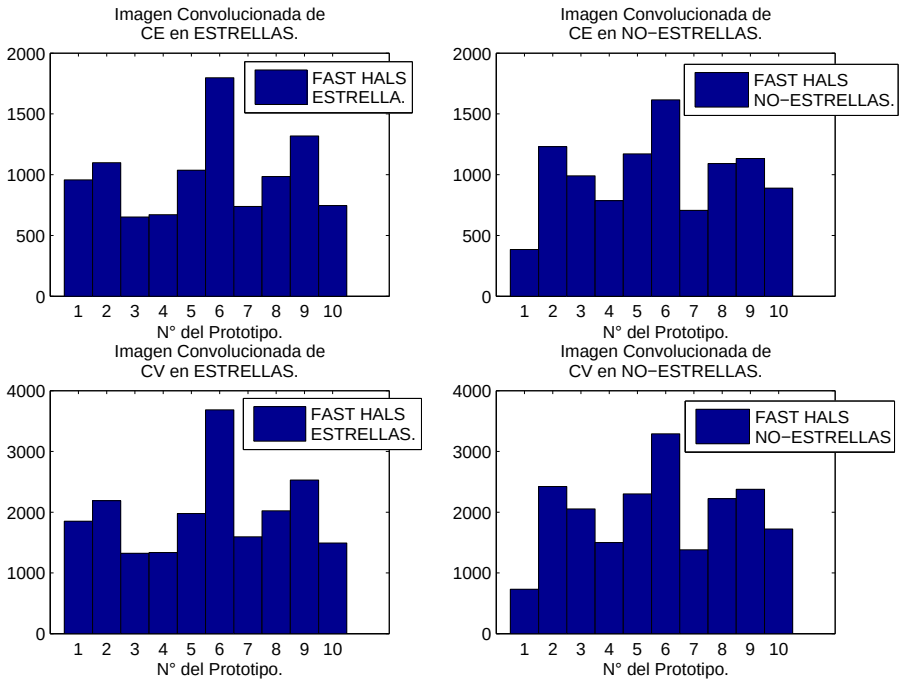
Con la configuración seleccionada para *HALS NMF Sparsity*, se tienen que sus prototipos son los de la figura 4.15. Los histogramas de las 3 descomposiciones fueron construidos de manera análoga a los histogramas de los métodos nombrados anteriormente, se pueden observar en las figuras 4.16 y 4.17.

En el histograma de la imagen señal a ruido se ve claramente una distribución distinta para estrellas y no estrellas, por ejemplo el prototipo 4 es alto en las estrellas y solo de un tercio en las no-estrellas, el prototipo 1 se observa que es casi el doble de valor en las no-estrellas de lo que se repite en las estrellas. El prototipo 19 también muestra una predominancia para las estrellas. Con respecto a los prototipos 1, 4 y 19 al observar la figura 4.15, vemos que corresponden a formas de estrellas. Lo otro interesante es que en las no-estrellas se repiten varios prototipos como máximos de las proyecciones que en las estrellas ni siquiera una sola vez son máximos.



(a)

Figura 4.13: Histogramas para la imagen de referencia en la descomposición *FAST HALS for NMF*.



(a)

Figura 4.14: Histogramas para la imagen convolucionada en la descomposición *FAST HALS for NMF*.

En el histograma de la imagen de referencia se observa claramente una tendencia del prototipo 1 y 2 hacia las estrellas y del prototipo 3 para las no-estrellas, además al ver los prototipos de la imagen de referencia se tiene que estos corresponden a formas de estrellas. Y en el histograma de la imagen convolucionada se observa que en las no-estrellas se repiten varios prototipos en menor medida, que no predominan en la estrellas.

Finalmente para comparar los 3 métodos se realizó una Curva *ROC*, en donde cada punto de la curva es una variación en el umbral de corte que utiliza *SVM* para clasificar, así se pueden comparar distintas *TPR*'s y *FPR*'s para los métodos en varios puntos, además se agrega la curva de clasificación de haber utilizado los 441 píxels para clasificar. Se realizaron 2 Curvas *ROC* para lo anterior, una en las imágenes de alta señal y la otra en las de baja señal, se pueden observar en las figuras 4.18 y 4.19 respectivamente. De los gráficos se concluye que los algoritmos basados en *NMF* superan mayoritariamente a la clasificación en la señal alta a *PCA* y a las 441 características, y en menor medida para la imagen de señal baja.

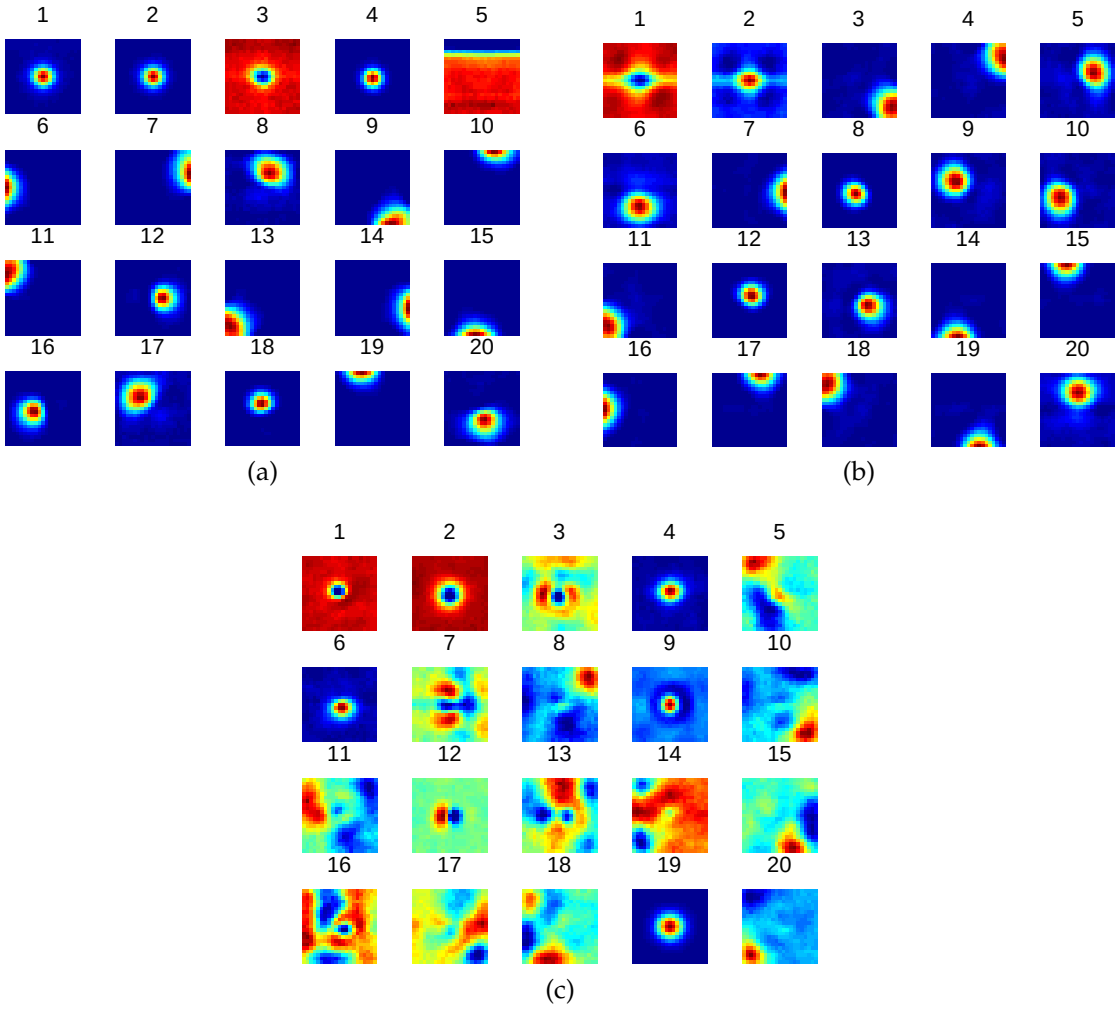


Figura 4.15: Prototipos de *HALS NMF Sparsity*, con 20 en: (a) la Imagen de referencia,(b) Imagen Convolucionada e (c) Imagen señal a ruido. Cada prototipo es numerado a propósito para a posterior observar histogramas en las figuras 4.16, 4.17.

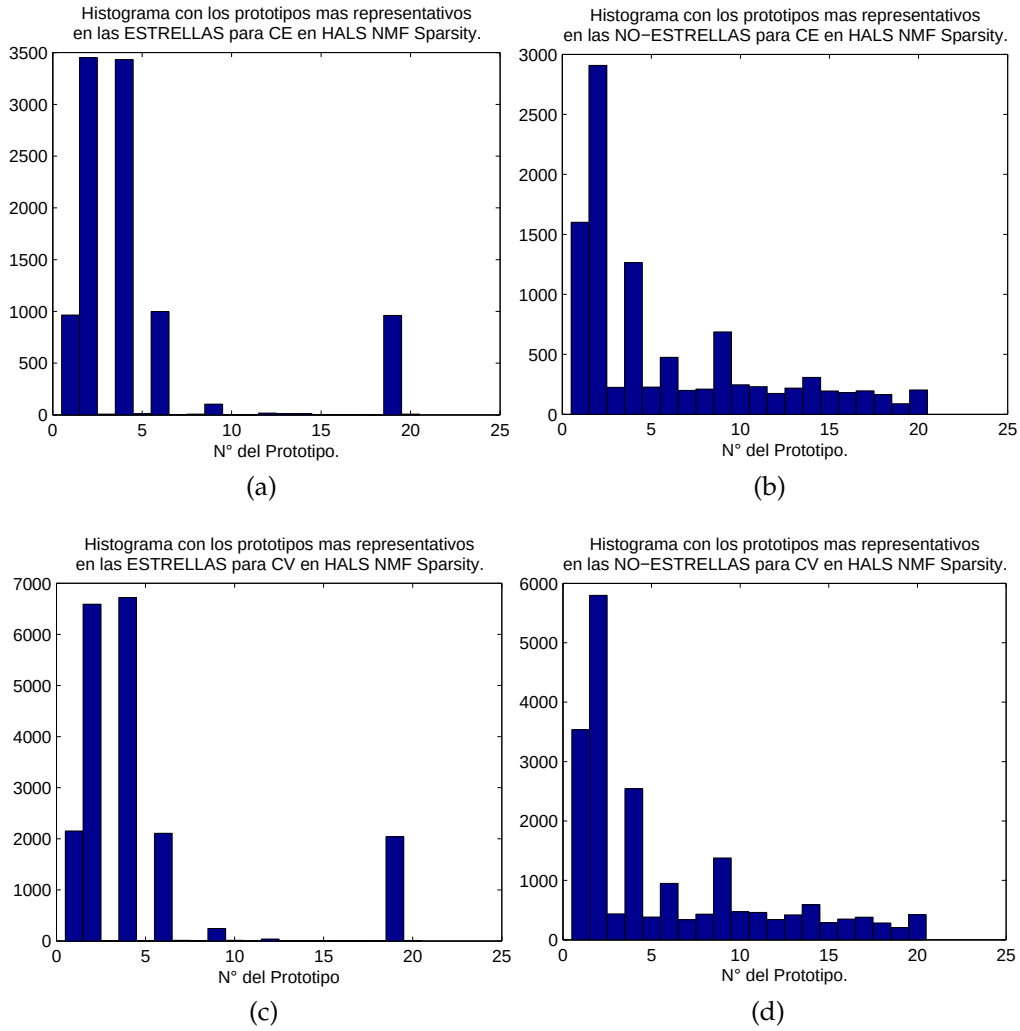


Figura 4.16: Histogramas con los prototipos mas repetidos como máximos en las proyecciones de *HALS NMF Sparsity* en la imagen señal a ruido, los histogramas son en: (a) Estrellas de *CE*,(b) No-estrellas de *CE*, (c) Estrellas de *CV* y (d) No-estrellas de *CV*

4.4. Resultados de ajustar el método seleccionado y prueba final

Finalmente se escoge los modelos basados en *NMF* como los que mejor clasifican. A continuación se presentan los resultados obtenidos de haber realizado los 2 métodos en un conjunto que no se ha utilizado antes, el *CP* (ver tabla 4.7).

Los resultados de clasificar *CP* agregando invariancia son: para la rotación[3] (ver tabla 4.8), para la traslación[4] (ver tabla 4.9) y para la combinación de ambos (ver tabla 4.10). Se observa que para la traslación y la combinación entre rotación y traslación se obtienen las menores *FPR* con una *TPR* sobre el 98 %.

[3]En los ángulos: 90°, 180° y 270°

[4]En las cuatro direcciones: un píxels hacia: arriba, abajo, izquierda y derecha

Método	FPR_{svm}	TPR_{svm}	$ERROR_{svm}$	FPR_{knn}	TPR_{knn}	$ERROR_{knn}$
FAST HALS for NMF	0.93 %	98.54 %	2.385 %	2.62 %	98.34 %	4.28 %
HALS NMF Sparsity	0.86 %	98.155 %	2.705 %	2.11 %	98.32 %	3.79 %

Tabla 4.7: Clasificación por parte de las mejores configuraciones de ambos métodos. Clasificación validada por 2 clasificadores: *SVM*, *KNN*.

Método	FPR_{svm}	TPR_{svm}	$ERROR_{svm}$	FPR_{knn}	TPR_{knn}	$ERROR_{knn}$
FAST HALS for NMF	0.955 %	98.545 %	2.41 %	2.375 %	98.31 %	4.065 %
HALS NMF Sparsity	0.845 %	98.155 %	2,55 %	2.015 %	98.35 %	3.665 %

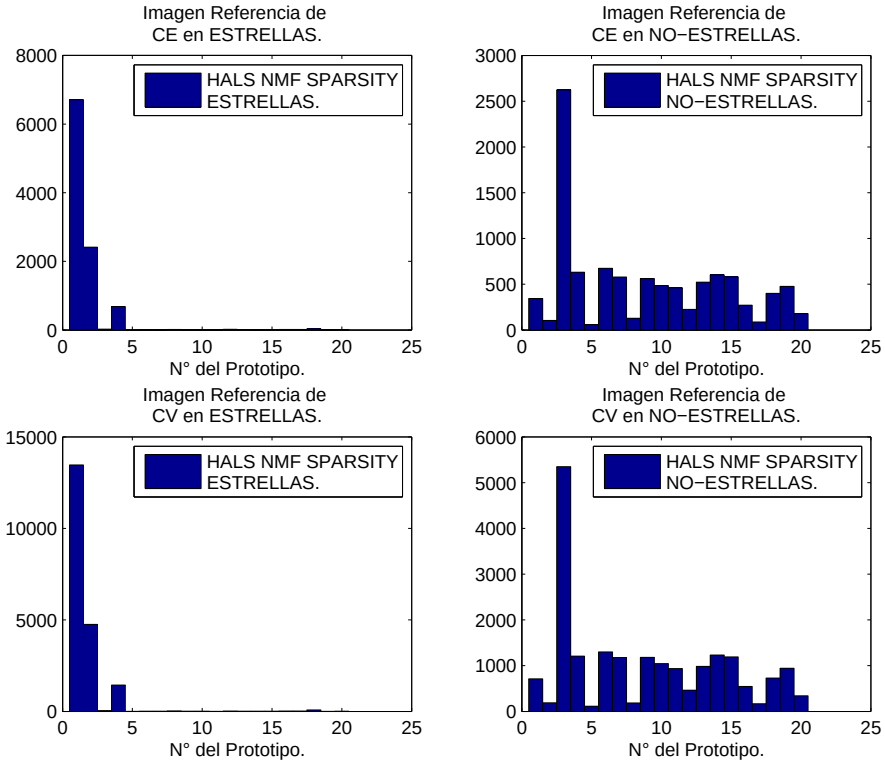
Tabla 4.8: Clasificación por parte de las mejores configuraciones de ambos métodos, agregándole invariancia de rotación. Clasificación validada por 2 clasificadores: *SVM*, *KNN*.

Método	FPR_{svm}	TPR_{svm}	$ERROR_{svm}$	FPR_{knn}	TPR_{knn}	$ERROR_{knn}$
FAST HALS for NMF	0.955 %	98.605 %	2.35 %	2.565 %	98.345 %	4.22 %
HALS NMF Sparsity	0.81 %	98.44 %	2.36 %	1.77 %	98.315 %	3.455 %

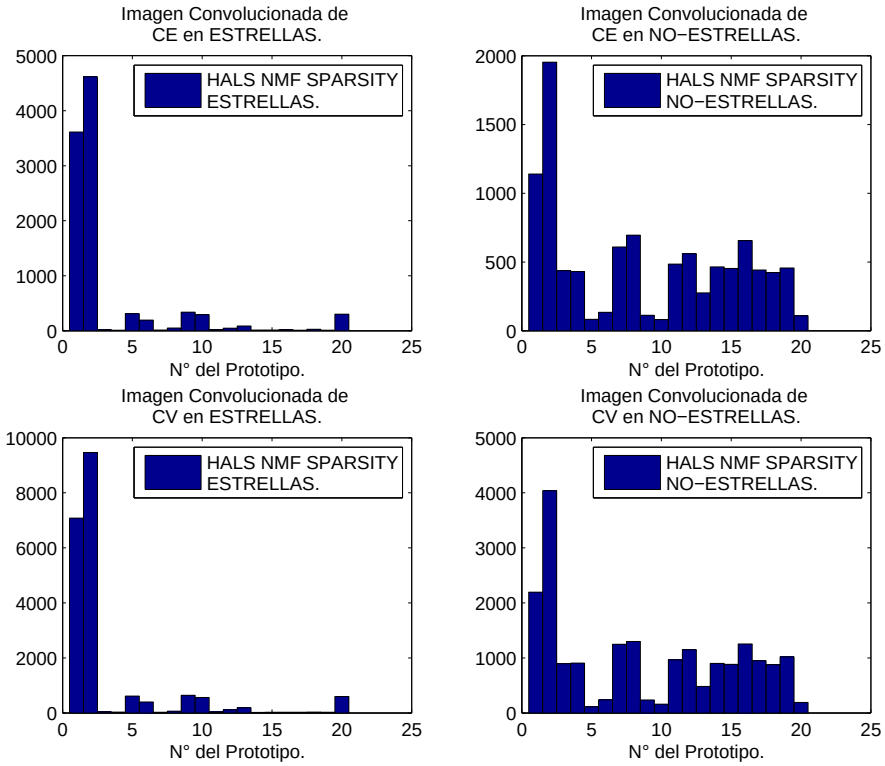
Tabla 4.9: Clasificación por parte de las mejores configuraciones de ambos métodos, agregándole invariancia de traslación . Clasificación validada por 2 clasificadores: *SVM*, *KNN*.

Método	FPR_{svm}	TPR_{svm}	$ERROR_{svm}$	FPR_{knn}	TPR_{knn}	$ERROR_{knn}$
FAST HALS for NMF	0.93 %	98.54 %	2.385 %	2.62 %	98.34 %	4.28 %
HALS NMF Sparsity	0.86 %	98.155 %	2.705 %	2.11 %	98.32 %	3.79 %

Tabla 4.10: Clasificación por parte de las mejores configuraciones de ambos métodos *NMF*, agregándole invariancia de rotación y traslación simultáneamente. Clasificación validada por 2 clasificadores: *SVM*, *KNN*.



(a)



(b)

Figura 4.17: Histogramas para la imagen de referencia (a) y la imagen convolucionada (b) en la descomposición *HALS NMF Sparsity*.

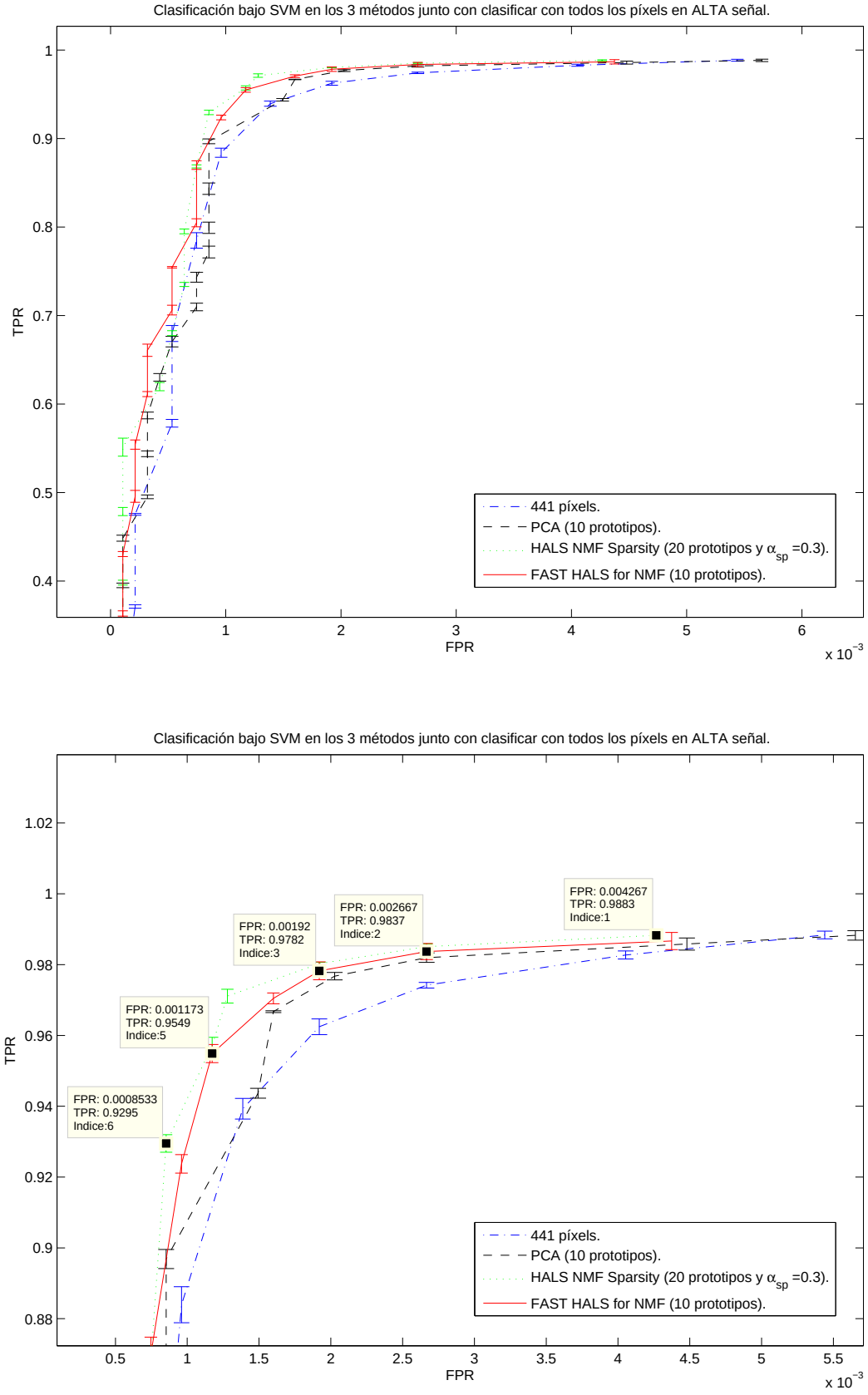


Figura 4.18: Comparación de los 3 métodos con sus mejores configuraciones junto con la clasificación en base a los 441 píxels en la Alta señal. Zoom en la imagen de abajo.

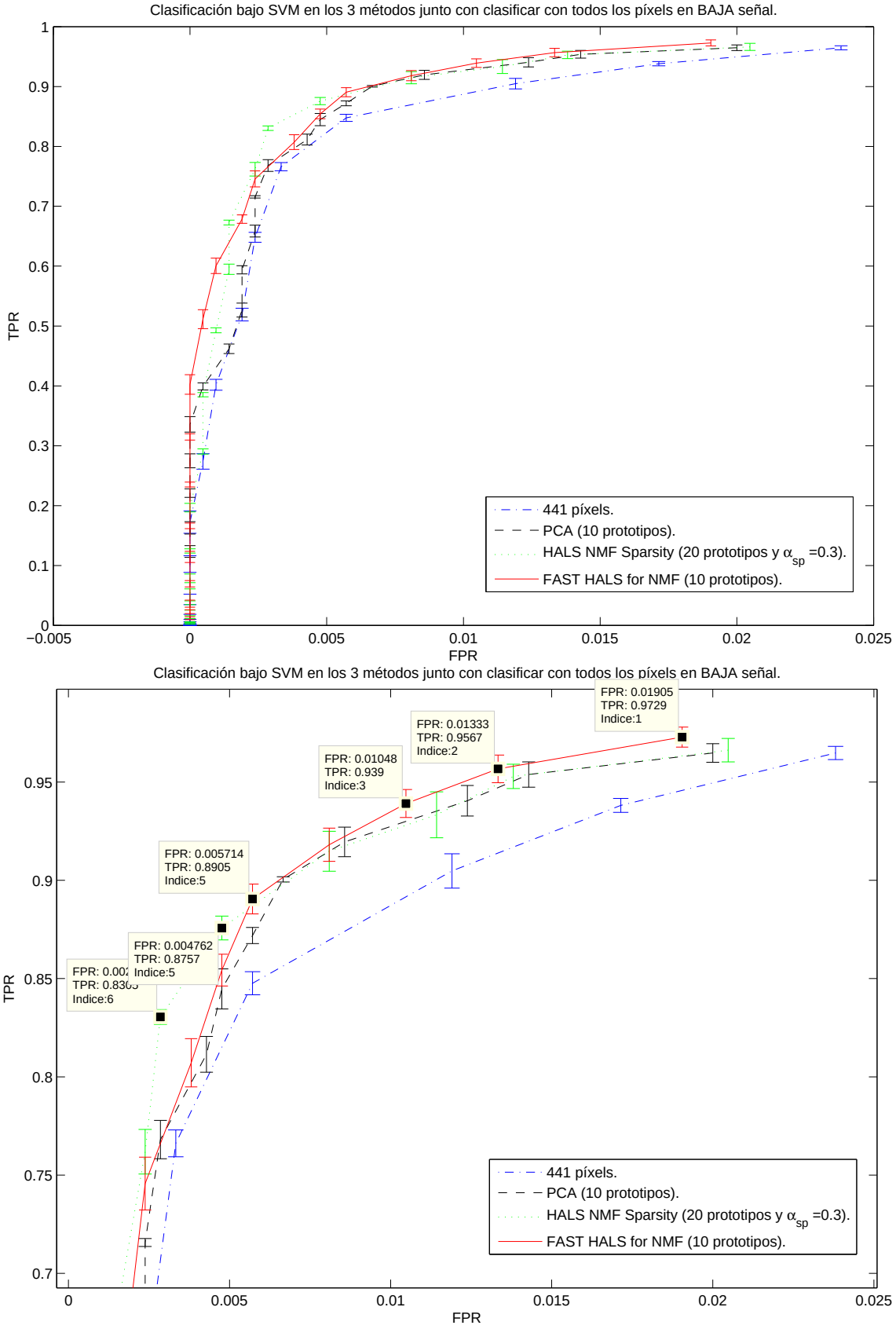


Figura 4.19: Comparación de los 3 métodos con sus mejores configuraciones junto con la clasificación en base a los 441 píxels en la Baja señal. Zoom en la imagen de abajo.

Conclusión

En este trabajo hemos ocupado aprendizaje semi-supervisado para detectar estrellas en imágenes no-etiquetadas, para esto hemos estudiado 3 algoritmos que permiten reducir dimensionalidad a través de la descomposición matricial.

Hemos visto que entre los tres algoritmos, el que ofrece mejor reconstrucción de las imágenes es *HALS NMF Sparsity*, pues este posee un factor que controla la dispersión de información, en este caso de las imágenes, esto da a lugar a que se obtengan representantes (prototipos) que son similares a las imágenes que son descompuestas; en cambio *PCA* crea variables nuevas, incorreladas y ortogonales entre sí, esto genera prototipos distintos entre si y distintos con las imágenes descompuestas. Adicionalmente *PCA* no produce una descomposición en matrices positivas. En cuanto a la reconstrucción entregada por *FAST HALS for NMF*, no es tan buena como la que genera *HALS NMF Sparsity*, esto se observo en los prototipos que generan cada uno.

En cuanto a la clasificación por parte de *SVM* se utilizó el criterio de la *Curva ROC* para escoger parámetros. Utilizar *SVM* permitió lograr los objetivos específicos planteados en este trabajo. Basados en los resultados de clasificación al utilizar *SVM*, podemos decir que entre menos características se le ingresen mejor clasificación logra. Esta es la razón del porque cuando se descompone en muchos prototipos, no se obtienen mejores clasificaciones que con menos prototipos, esto ultimo se observo en la comparación de la curva *ROC* entre los 3 métodos junto con *SVM*.

Finalmente con la elección de *HALS NMF Sparsity* tomando su ecuación de costo:

$$D_F^{(j)}(Y^{(j)}||w_j h_j) = \frac{1}{2}||V^{(j)} - w_j h_j||_F^2 + \alpha_{sp}||h_j||.$$

vimos que al hacer muy grande el factor α_{sp} que controla la dispersión, se generaba reconstrucciones muy poco similares a las imágenes. Sin embargo con un α_{sp} adecuado se generan prototipos que pueden reconstruir muy bien a las imágenes originales, tanto así que dentro de los mismos prototipos se pueden formar clases bien definidas de imágenes, entre ellas: estrellas, malas restas, rayos cósmicos, fuentes puntuales falsas entre otras.

En cuanto a las comparaciones entre los tres métodos se concluye que *HALS NMF Sparsity* junto con *FAST HALS for NMF* logran las mejores clasificaciones, es difícil decir cual es mejor entre ellos, pues cuando se clasificaron imágenes de baja señal gano *FAST HALS for NMF*, pero cuando fueron imágenes de alta señal gano *HALS NMF Sparsity*. Al descomponer con *PCA*, si bien es cierto que obtuvo buenas clasificaciones, el hecho de no marcar

una fuerte diferencia entre los prototipos que son máximos en las proyecciones, produce que el clasificador no pueda discriminar correctamente. A diferencia de los algoritmos *NMF*, estos lograban establecer ciertos prototipos que predominaban como valores máximos en las proyecciones implicando que el clasificador pueda tener características más predominantes para clasificar estrellas de no-estrellas.

En cuanto a los dos algoritmos basados en *NMF* lo importante es que el algoritmo *HALS NMF Sparsity* es muy efectivo a la hora de agrupar imágenes y generar una reducción, cuyos prototipos son muy similares a las imágenes originales. En cambio *FAST HALS for NMF* tiene la rapidez de calculo como ventaja y también entrega buenas características para estrellas. También es posible agregar un control de dispersión a *FAST HALS for NMF* por lo que no es tan correcto decir cual de entre los dos es mejor.

Con los 2 métodos finales seleccionados, se realizó la prueba con un conjunto que no ha sido manipulado anteriormente, este es *CP*. Se mostró una mejora en cuanto a la clasificación, además que se obtuvo una $FPR < 1\,\%$ (lo cual era lo esperable). En cuanto a agregar invarianza a los modelos se observo que mejora casi en un 0.05 % menos de *FPR* cuando se mezclan las traslaciones (i.e.,las 4 direcciones: 1 píxel arriba, abajo, derecha e izquierda) o se mezclan las rotaciones (i.e., las 3 rotaciones: 90°, 180° y 270°) o se mezclan las traslaciones con rotaciones. De estas pruebas se mostró que *HALS NMF Sparsity* obtuvo la ventaja en la clasificación validada por *SVM* y *KNN*.

En resumen, este trabajo como resultado final da a entender que los algoritmos basados en *NMF* son mejores clasificando que *PCA* y que es levemente mejor *HALS NMF Sparsity*. Además que agregando invarianza a *CE*, puede mejorar aun mas la clasificación. Así en esta memoria se cumplen el objetivo general, que es dar una metodología para clasificar estrellas y no-estrellas en imágenes no etiquetas; y se cumple con el objetivo específico que es obtener una tasa de falsos positivos menor a 1 %.

Trabajo Futuro

Dada la complejidad del problema se pueden crear muchas más combinaciones entre n° de prototipos, n° de iteraciones y control de dispersión, y esto por cada descomposición de imagen: imagen de referencia, imagen convolucionada e imagen señal a ruido. Además se pueden agregar mas transformaciones de imágenes para capturar mas invarianza, estas pueden ser: erosión, dilación, apertura y cerradura.

Bibliografía

[1] D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[2] F. Foster, "Pipeline en extracción de imágenes (en preparación para publicar)," *CMM*.

[3] A. Cichocki and A. H. Phan, "Fast local algortithms for large scale nonnegative matrix and tensor factorizations," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E92-A, no. 3.

[4] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 6, pp. 1336–1353, 2013.

[5] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, 2004.

[6] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorization. Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Tokyo: John Wiley and Sons, 1st ed., 2009.

[7] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, final ed., 1985.

[8] J. A. T. Thomas M. Cover, *Elements of Information Theory*. final ed., 1991.

[9] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *In NIPS*.

[10] Z. Yang and E. Oja, "Unified development of multiplicative algorithms for linear and quadratic nonnegative matrix factorization," *Trans. Neur. Netw.*, vol. 22, pp. 1878–1891, 2011.

[11] S. A. Vavasis, "On the complexity of nonnegative matrix factorization," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1364–1377, 2009.

[12] C. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[13] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281,

2008.

[14] A. L. P. P. M. Berry, M. Browne and R. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics and Data Analysis*, vol. 52.

[15] S. C. R. P. A. Cichocki, R. Zdunek and S. Amari, "Nonnegative tensor factorization using alpha and beta divergencies," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP07), Honolulu*, vol. 52.

[16] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *Neural Networks, IEEE Transactions on*, vol. 18, no. 6, pp. 1589–1596, 2007.

[17] M. Merritt and Y. Zhang, "Interior-point gradient method for large-scale totally nonnegative least squares problems," *Journal of Optimization Theory and Applications*, vol. 126, pp. 191–202, 2005.

[18] N.D.Ho, P. Dooren, and V. Blondel, "Descent algorithms for nonnegative matrix factorization," *Numerical Linear Algebra in Signals, Systems and Control*.

[19] M. Biggs, A. Ghodsi, and S. Vavasis, "Nonnegative matrix factorization via rank-one downdate," *ICML-2008*.

[20] N. Gillis and F. Glineur, "Nonnegative matrix factorization and underapproximation," *SIAM conference on Optimization*.

[21] A. Phan and A. Cichocki, "Multi-way nonnegative tensor factorization using fast hierarchial alternating least squares algorith (hals)," *Proc. of the 2008 International Symposium on Nonlinear Theory and its Applications*.

[22] A.Cichocki. and R. Zdunek, "Regularized alternating least squares algorithms for non-negative matrix/tensor factorizations," *Springer LNCS*, vol. 4493, pp. 793–802, 2007.

[23] J. I.T., *Principal Component Analysis*. Springer, 2010.

[24] A. Hivärinen, E. Oja, and J. Karhunen, *Independent Component Analysis*. New York: John Wiley and Sons, final ed., 2001.

[25] M. F. D. L. T. S. V. G. Barutello, Viviana; Conti, *Analisi matematica. Con elementi di geometria e calcolo vettoriale*. Milano: APOGEO, 2 ed., 2008.

[26] H. H. Harman, *Modern Factor Analysis*. University of Chicago, 1976.

[27] M. B. no, J. García-Bastos, and J. González-Buitrago, "Las curvas roc en la evaluación de las pruebas diagnósticas," *Med. Clin (Barc)*.

[28] Z. MH and C. G., "Receiver-operating characteristic (roc) plots: A fundamental eva-

luation tool in clinical medicine," *Clin Chem*, vol. 39.

[29] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[30] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 4th edition ed., 2009.

[31] T. Mitchell, *Machine Learning*. McGraw-Hilll, final ed., 1997.

[32] J. W. Sammon and JR., "A nonlinear mapping for data structure analysis," *IEEE TRANSACTIONS ON COMPUTERS*, vol. C-18, no. 5, pp. 401–409, 1969.

[33] M. R. Spiegel, *Estadística*. Mc Graw Hill, 1991.