PROYECTO FIN DE CARRERA

COMPRESIÓN DE IMÁGENES MEDIANTE SVM ADAPTATIVA PERCEPTUAL

realizado por
Gabriel Gómez Pérez

dirigido por
Dr. Gustavo Camps i Valls y
Dr. Jesús Malo López

Departament d'Enginyeria Electrònica
UNIVERSITAT DE VALÈNCIA – ESTUDI GENERAL

València – Julio, 2004.

Índice general

Resumen

Las imágenes digitales constituyen hoy en día uno de los principales medios de comunicación de la sociedad actual. Gracias a Internet podemos enviar información de una punta a otra del planeta en cuestión de segundos. Por otro lado cada día más gente tiene algún dispositivo optoeléctrónico capaz de conseguir imágenes digitales del entorno que lo rodea.

Este tipo de imágenes puede llevar asociados volúmenes de información muy elevados, lo cual repercutirá en un trasiego de información más lento y por ello más sujeto a errores de transmisión o almacenamiento. Es en este contexto donde se hace necesario intentar reducir el tamaño ocupado en memoria por las imágenes digitales que obtenemos mediante herramientas como las cámaras digitales integradas en nuestro teléfono móvil.

Por ello, la compresión de imágenes es un campo en continua evolución que busca nuevos algoritmos que permitan almacenar las imágenes, en un espacio de memoria mucho menor y con poca pérdida de calidad. A día de hoy el algoritmo dominador en este campo es claramente JPEG que está basado en la cuantización de los coeficientes de la *Transformada Discreta del Coseno ("Discrete Cosine Transform", DCT).*

Sin embargo, el aprendizaje en máquinas nos proporciona una nueva herramienta que promete ser provechosa para la compresión de imágenes, las *Máquinas de Vector Soporte ("Support Vector Machines", SVM).* En la actualidad existe muy poco trabajo sobre estos algoritmos, por lo que en el presente PFC se va a abordar la compresión de imágenes mediante esta herramienta, a la cual además se le añadirán una serie de parámetros relativos al sistema visual humano, que harán que la calidad de las imágenes que obtengamos sea óptima y se mejoren ampliamente los resultados respecto a JPEG y algoritmos previos basados en SVM.

En el presente PFC nos centraremos en imágenes 2-D monocromáticas por su mayor sencillez, pero todos los resultados que obtendremos serán extrapolables a futuras ampliaciones como puedan ser las imágenes hiperespectrales.

Capítulo 1

Introducción

ESTE primer capítulo lleva a cabo un breve resumen sobre los conceptos de imagen digital y codificación de imágenes. De este modo podremos comprender las distintas técnicas que se han usado históricamente así como los estándar actuales y el nuevo algoritmo que se propone en este proyecto mediante el uso de *Máquinas de Vector Soporte (SVM).*

1.1. Imágenes digitales

Una imagen vale más que mil palabras. Esta frase tan popular es un fiel reflejo de la realidad y de nuestra capacidad cognitiva. De este modo, la información proporcionada por una imagen es fácilmente asimilada por nuestro cerebro, el cual gracias a los impulsos nerviosos que recibe de nuestros ojos nos hace entender la escena que estamos observando.

Por ello, desde siempre, las imágenes han constituido una poderosa herramienta pedagógica y científica. Por esta razón, multitud de ramas tanto tecnológicas como científicas se apoyan cada día más en imágenes, tanto como medio de investigación como de almacenamiento de información. Algunos de estos campos son:

- *Medicina.* Gracias a la gran cantidad de sensores y dispositivos de los cuales dispone el personal médico en el entorno clínico, se pueden obtener multitud de imágenes de determinadas zonas del cuerpo humano. Su principal uso está orientado al diagnóstico y a la monitorización de pacientes.

- *Teledetección.* Las nuevas generaciones de sensores que se instalan en los satélites toman una gran cantidad de imágenes en gran cantidad de bandas del espectro de la superficie terrestre por la que pasan.

- *Multimedia.* Las nuevas tecnologías han inundado el mercado de la "Sociedad de la Informaciónçon nuevos dispositivos de visualización para la cada vez mayor cantidad de imágenes digitales: cámaras de fotos digitales, páginas web, teléfonos móviles con cámara incluida, etc

Además, en concreto, las imágenes digitales presentan otras dos grandes ventajas, como son las siguientes:

- *Procesado Digital.* El procesado digital de señales es un conjunto de técnicas matemáticas que permite el tratamiento de las imágenes como señales 2-D, o el video digital como señales 3-D, etc..

- *Fácil Intercambio y Disponibilidad.* Dado el gran desarrollo que existe en la actualidad de las redes informáticas y otros medios de almacenamiento electrónico, las imágenes digitales gozan de *universalidad*, puesto que pueden transmitirse de un extremo a otro del mundo en cuestión de segundos.

Sin embargo, nos encontramos con que la cantidad de información contenida en una imagen digital puede llegar a ser muy elevada, lo cual nos lleva a tener que considerar archivos de gran tamaño. Esto constituirá un grave incoveniente a la hora de transmitirlos, tratarlos o almacenarlos. Este hecho se agrava más cuando se trata de video digital, que es básicamente una extensión temporal de las imágenes digitales. En cambio, un estudio detallado de la señal que constituye la imagen nos muestra que suele existir una elevada redundancia tanto estadística como perceptual; la estadística hace referencia a la alta correlación entre el valor de un píxel y el de uno contiguo a él, mientras que la perceptual se refiere a la influencia en la percepción de un píxel que tienen sus vecinos. Son estas características de las imágenes digitales las que aprovechan los algoritmos de compresión para reducir de manera considerable el tamaño que ocupan en disco dichas imágenes.

Una imagen digital esta compuesta por $M \times N$ *píxeles* (*M píxeles de altura por N píxeles de anchura*). Así pues, podemos decir que cada píxel representa una porción de imagen. Estableciendo una equivalencia con el ojo humano, diríamos que cada píxel correspondería a una pequeña división de la retina.

Cada píxel lleva asociado un valor distinto de intensidad o varios en función de que la imagen sea monocromática o policromática. En las fotos en color tenemos tres planos distintos (RGB), mientras que en las hiperespectrales podemos llegar a tener hasta N bandas dependiendo de la calidad del sensor.

Los valores de intensidad para cada banda se suelen cuantizar a 8 *bits*, de modo que disponemos de $2^8 = 256$ niveles para caracterizar cada píxel. Por ello, a la hora de almacenar un determinado píxel necesitaremos un *byte*, siendo el total para la imagen de $M \times N$ *bytes*. Este modo de almacenamiento es el que usa el estándar *bitmap (mapa de bits)*, que guarda una matriz de tantos datos como píxeles tiene la imagen con los valores de cada uno de ellos. Pongamos el ejemplo de una imagen captada por una cámara fotográfica digital típica; si la foto obtenida tiene un tamaño de 2024×1024 píxeles, el tamaño en memoria necesario para almacenarla es de aproximadamente 6 Mbytes. Si tenemos en cuenta que las cámaras digitales se suelen servir con tarjetas de memoria de tan sólo 16 Mb resulta evidente que se hace necesaria la introducción de técnicas de compresión que reduzcan el tamaño de las imágenes [34].

Figura 1.1: La imagen que se muestra en la figura (Lena) se ha convertido en un estándar en el campo del tratamiento digital de imágenes.

1.2. Mecanismo de visión humana

Existe cierta similitud entre el sistema visual humano y la manera en que se guardan las imágenes digitales. Podemos ver una representación del ojo humano en la figura 1.2. Este parecido será uno de los puntos claves de nuestro algoritmo, por lo cual cabe detenerse brevemente en estudiar el proceso de visión humana.

A grandes rasgos podemos dividir el proceso de visión humana en tres partes: formación de la imagen, conversión a señal eléctrica, y transmisión al cerebro [48].

- *Formación de la imagen:* La luz que incide sobre el ojo es enfocada en la retina mediante la acción de dos lentes que son la cornea y el cristalino (Fig. 1.3). La primera de ellas es de focal fija, mientras que el cristalino es de focal variable.

- *Conversión a señal eléctrica:* Una vez formada la imagen sobre la retina, los conos y bastones que se encuentran repartidos por ella (aunque de distinto modo) la transforman en señal eléctrica, que pasa a los nervios. Los conos y los bastones son dos tipos distintos de células fotodetectoras que tienen respuestas distintas ante la luz incidente.

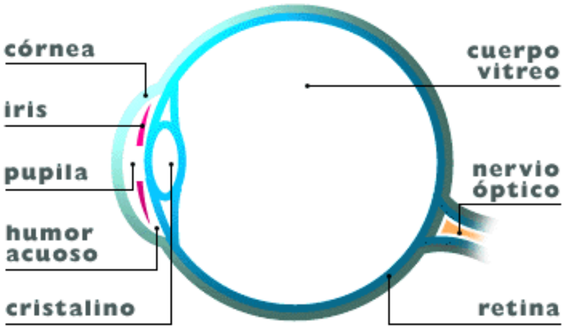


Figura 1.2: Sección de un ojo humano.

- *Transmisión al cerebro:* Por los nervios ópticos que llegan al cerebro se transmite una señal en forma de tren de pulsos modulados. En primera aproximación el número de pulsos por segundo es proporcional a la intensidad del estímulo.

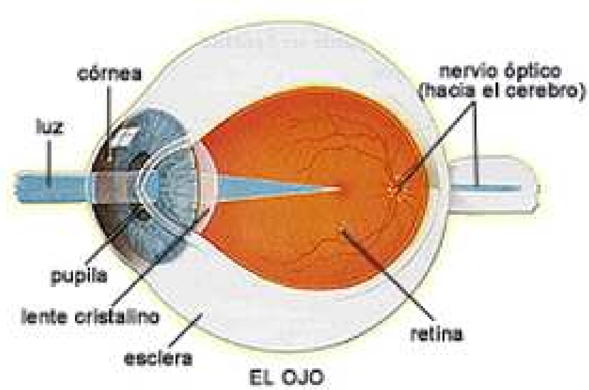


Figura 1.3: Formación de imágenes en el ojo.

La visión humana comprende una serie de fenómenos en cuyo estudio intervienen materias tan diversas como la fisiología, la bioquímica o la psicofísica. La interpretación de la señan se produce en una determinada zona del cerebro llamada córtex visual. No obstante, la parte de información espacial que llega depende de la actuación de, por ejemplo, los conos o las células V1. La distinta sensibilidad de estas células determina nuestra capacidad para percibir distorsiones en la señal. El hecho de que nuestro sistema visual trate de distinta manera ciertas frecuencias será aprovechado por nuestro algoritmo de compresión.

1.3. Técnicas de compresión de imágenes

Tal y como hemos mencionado anteriormente, la información presente en una imagen digital suele ser bastante redundante, lo cual constituye la base de los algoritmos de compresión. En la práctica se distingue entre dos tipos; la compresión sin pérdidas y la compresión con pérdidas [15].

1.3.1. Compresión sin pérdidas

En la compresión sin pérdidas no hay señal de error entre los píxeles originales y los reconstruidos; es decir, una vez descomprimida la imagen es idéntica a la original. También se conoce como *codificación entrópica (entropy coding)* dado que usa técnicas estadísticas basadas en la entropía de la señal a codificar y de descomposición de la propia imagen para, de este modo, eliminar o minimizar la información redundante. El principio fundamental es la alta probabilidad de similitud entre dos píxeles contiguos. Puesto que realmente no perdemos señal, se alcanzan ratios bajos de compresión. El principal campo de aplicación de la codificación sin pérdidas se halla en la medicina dado que la información a comprimir puede llegar a ser de extremada importancia y no se puede dar pérdida de información.

A continuación, vamos a llevar a cabo un breve resumen de los principales algoritmos de codificación entrópica:

- *RLE ("Run length encoding")*. Sustituye series de píxeles contiguos con el mismo valor por símbolos codificados más cortos. Al final, se obtienen varias series de estos símbolos con valores muy similares. En el caso de JPEG, o en el de nuestro algoritmo, un gran número de los coeficientes de la DCT tendrán valores muy bajos o incluso nulos. En lugar de guardarlos todos, simplemente se guarda uno de ellos y a continuación el número de veces que se repite consecutivamente.

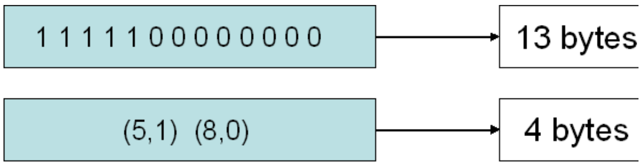


Figura 1.4: Ejemplo de actuación del algoritmo RLE. Observamos como se reduce el tamaño ocupado en memoria, si en lugar de almacenar la secuencia completa sólo se almacenan los símbolos y el número de repeticiones.

- **Codificación lineal predictiva (LPC).** Se basa en que los píxeles de la imagen forman un modelo *autorregresivo* (AR), de modo que el valor de un píxel es combinación lineal de sus vecinos. Se crea una imagen según el modelo AR y restándola a la original, se obtiene la *imagen residual*, que es la que se almacena. La ventaja está en que el rango dinámico de los píxeles de esta última es menor que el original, lo cual facilitará su compresión mediante técnicas estadísticas, como la codificación Huffman o la aritmética que describiremos a continuación.

- **Codificación Huffman.** Se basa en asignar a los símbolos (en nuestro caso niveles de gris de los píxeles) con más probabilidad de ocurrencia secuencias de bits más cortas para su almacenamiento. Por ello, hay que asegurar que el código asignado a un determinado símbolo no coincida con los primeros bits de otro símbolo menos probable.

Símbolo (Nivel de gris)	Probabilidad de ocurrencia	Código Huffman
3	0.1	001
25	0.15	010
72	0.05	0000
124	0.09	0001
158	0.14	011
200	0.27	10
254	0.2	11

Figura 1.5: Ejemplo de codificación Huffman. Se puede observar cómo los símbolos más probables son los que tienen asociados unos códigos más cortos

- **Codificación aritmética:** Esta codificación genera un único código para toda la imagen. En base a la correlación entre píxeles cercanos se genera una *función de probabilidad de densidad cumulativa* (CDF) que nos proporcionará para cada parte de la imagen el símbolo con más probabilidad de ocurrencia.

- **Codificación Lempel-Ziv.** Se construye una tabla con las secuencias de píxeles más comunes. A cada uno de estos conjuntos se le asigna un determinado ínidice de un tamaño menor que el propio conjunto. La imagen comprimida la constituyen dichos índices. Cuando se va a recuperar la información, el algoritmo reconstructor lee los índices y busca en la tabla la secuencia de píxeles que le corresponde, reconstruyendo así la imagen.

1.3.2. Compresión con pérdidas

En estos algoritmos se lleva a cabo una *simplificación* de la señal. Esta simplificación puede ser una *transformación* o un proceso de *cuantización.* Mediante estos procedimientos la imagen reconstruida no es igual a la original, pero si la simplificación ha sido buena, será suficiente para la mayoría de aplicaciones prácticas. Por otro lado, la imagen simplificada suele ser más fácil de codificar mediante las técnicas estadísticas comentadas anteriormente y se consiguen mayores ratios de compresión.

- **Cuantización.** El proceso de cuantización es aquél por el cual se mapea un conjunto de datos continuo a uno discreto, según unos determinados niveles de cuantización establecidos por el usuario. Este proceso introduce un error en la imagen pero disminuye la entropía de la información con la que se trabaja, haciendo así que las técnicas estadísticas de codificación sean más eficientes. De este modo, en las últimas etapas de compresión se usan los métodos de Huffman, aritmético o RLE, aumentando así la compresión sin introducir pérdidas adicionales.

- **Codificación predictiva.** Se trata de una cuantización muy similar al caso sin pérdidas, salvo que ahora la predicción del valor de un píxel se hace sobre los valores reconstruidos (no los originales) de los píxeles vecinos.

- **Codificación por transformación.** Se trata de codificar los coeficientes correspondientes a la transformada de la imagen original. Algunas transformadas comunmente utilizadas son la transformada discreta de Fourier (DFT), la transformada discreta del coseno (DCT) o la transformada Wavelet (WT). Un estudio perceptual de los coeficientes de las transformadas nos dice cuáles son los más significativos, permitiéndonos así descartar el resto sin excesiva degradación en la imagen reconstruida.

1.3.3. Algoritmos reales de compresión

En la práctica lo que realmente se hace es una combinación de técnicas con pérdidas y sin pérdidas destinada a obtener un mayor *ratio* de compresión sin perder calidad de la imagen. Así pues, la manera más usual de trabajar consiste en, a partir de la imagen original, usar una técnica con pérdidas, para después pasar a un proceso de cuantización, y finalizar con una codificación entrópica. Un esquema de este proceso se puede observar en la figura 1.6. En el presente proyecto nos centraremos en la compresión con pérdidas, dado el mayor interés práctico que presenta.

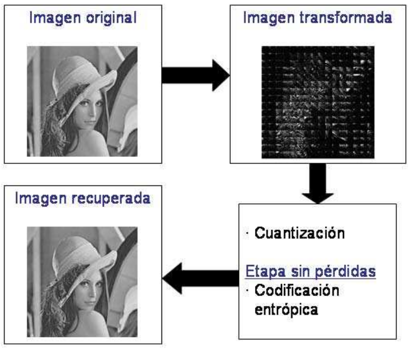


Figura 1.6: Proceso típico de compresión de imágenes

1.4. Estándar actual: *el algoritmo JPEG*

A día de hoy, el algoritmo estándar de compresión universal es *JPEG (Joint Photographic Experts Group)* [47], tanto por su sencilla implementación como por sus altos ratios de compresión para calidades de imagen aceptables en la mayoría de aplicaciones. El comite JPEG comienza su trabajo en 1986, y fruto de éste, han surgido diversos estándar JPEG hasta nuestros días. Una de las principales características de este formato es que ha surgido gracias a la colaboración de muchos expertos en la materia. Por ello, en realidad, existen muchos algoritmos JPEG aunque todos asentados sobre la misma base. Lo que conocemos como estándar JPEG es el ISO/IEC IS 10918-1 — ITU-T Recommendation T.81. A continuación vamos a describir las etapas fundamentales en el funcionamiento de este algoritmo.

1.4.1. Compresión

- **Submuestreo.** Es una etapa opcional que se puede omitir puesto que consiste en reterner una determinada fracción de los píxeles de la imagen.

- **Transformación al dominio espectral.** Se aplica la *transformada discreta del coseno* (DCT) a cada bloque de 8×8 píxeles de la imagen.

- **Cuantización.** Se establecen unos determinados niveles de cuantización, distintos para cada frecuencia según su relevancia perceptual, y se toma para cada valor de coeficiente de la DCT el valor más próximo de los escalones de cuantización. Para fijar estos niveles se hace uso de la *Función de Sensibilidad al Contraste (Constrast Sensitivity Function, CSF)*. Si X es un coeficiente cualquiera de la DCT y q es el escalón de cuantización, tendríamos lo siguiente:

$$\overline{X_q} = round\left(\frac{X}{q}\right)q \tag{1.1}$$

1.4.2. Descompresión

Este proceso consta también de tres pasos:

- **Transformación al dominio espacial.** Se realiza la transformada inversa sobre los coeficientes cuantizados.
- **Interpolación.** Si se había llevado a cabo un submuestreo, se realiza una interpolación seguida de un filtrado paso-bajo que lleva a cabo un suavizado de la imagen.
- **Redondeo.** Los valores obtenidos para cada píxel son redondeados al entero más cercano, de modo que todos los valores estén comprendidos en el rango [0,255].

1.5. Objetivo del proyecto

Tal y como hemos comentado en la sección 1.1, la compresión de imágenes digitales es cada día más importante debido a la espectacular implantación que están teniendo en nuestra sociedad todo tipo de tecnologías digitales (memorias portátiles, pantallas TFT, etc). Por ello, el objetivo de este proyecto será explorar un nuevo camino en este campo.

Para ello, usaremos *Máquinas de Vectores Soporte (Support Vector Machines, SVM)* que nos permitirán estimar el espectro de las imágenes que tratemos. Estudios similares han sido realizados previamente [37] en este contexto, pero en el presente PFC, primero se va a llevar a cabo un estudio exhaustivo sobre la viabilidad de esta nueva familia de algoritmos de compresión de imágenes, y segundo, se propondrán formulaciones elegantes y óptimas desde el punto de vista perceptual a las propuestas en [37].

Las dos principales herramientas con las que se trabajará en este PFC son las siguientes:

1. **Máquinas de Vectores Soporte (SVM)**. Mediante esta novedosa herramienta modelizaremos los coeficientes de la DCT. Las SVM constituyen una reciente aproximación para el problema de aprendizaje artificial. Atacan los problemas de clasificación y regresión desde un pusto de vista geométrico. Las SVM serán presentadas con mayor detalle en el segundo capítulo del presente proyecto.

2. **Función de Sensibilidad al Contraste (CSF)**. Nos dirá en qué zona del dominio espectral podemos cometer mayor error en la modelización sin que se resienta la imagen reconstruida. El ojo humano presenta una determinada respuesta ante los estímulos luminosos que inciden sobre él. A través del estudio de la curva de esta respuesta podemos obtener las zonas en las que el ojo humano es más sensible ante la radiación luminosa, es decir su *sensibilidad*. De este modo podremos elegir unas zonas en las que las condiciones de modelización se podran relajar puesto que no serán de mucha relevancia perceptual.

1.6. Estructura del PFC

Una vez llevada a cabo esta introducción abordaremos el estudio de las SVM en el segundo capítulo. Presentaremos esta nueva familia de algoritmos y nos centraremos en la aplicación para aproximación de funciones. También comentaremos las variantes que usaremos aquí como es el uso de SVM adaptativa.

En el tercer capítulo hablaremos del sistema visual humano. Detallaremos sus principales características, y como éstas son explotadas por los algoritmos de compresión para obtener mejores resultados.

En el cuarto capítulo mostraremos más detenidamente en que consiste la compresión de imágenes mediante SVM, y justificaremos la necesidad de introducir información perceptual al entrenamiento de la SVM que usemos.

En el quinto capítulo mostraremos los resultados obtenidos. Para ello nos apoyaremos en las curvas *Rate-Distortion*, que muestran el error en función de la compresión obtenida. También aquí, analizaremos los parámetros con los que trabajan las SVM.

Finalmente, en el último capítulo expondremos las conclusiones a las que hemos llegado, y las posibles extensiones y campos de aplicación del presente PFC.

Capítulo 2

Máquinas de Vectores Soporte (SVM)

ESTE capítulo se centrará en un estudio detallado de las SVM. Comenzaremos por una somera descripción de su funcionamiento tanto en clasificación como en regresión, para acabar explicando cuál será el uso final que le daremos en este proyecto, que consistirá en estimar el espectro de las imágenes que estudiemos.

2.1. Introducción

2.1.1. Una nueva aproximación: Optimización con restricciones

Las SVM suponen un nuevo concepto en el aprendizaje de máquinas puesto que abordan el problema desde un punto de vista meramente geométrico de separación de muestras pertenecientes a clases diferentes. Por ello, se usa la técnica matemática de los *multiplicadores de Lagrange* para separar correctamente el espacio de datos de entrenamiento. De este modo se construye lo que vendremos a llamar el *Hiperplano óptimo de decisión*, que constituirá la solución a nuestro problema.

El aprendizaje mediante Vectores Soporte fue propuesto por Vapnik en [44] como método para la construcción de hiperplanos separadores con máximo margen pero, en el espacio de unos pocos años, se ha extendido hacia una teoría mucho más general [45, 46]: la aplicación a problemas no lineales mediante *kernels* de Mercer (semidefinidos positivos) tiene lugar en [17]; la implementación en problemas no separables mediante variables de pérdidas en [10]; el planteamiento para problemas de regresión y aproximación en [42]; la incorporación de conocimiento *a priori* y análisis de componentes principales no lineales en [39] y la adecuación de las SVMs al agrupamiento de características en [1]. Además, ha sido aplicado a un considerable número de problemas prácticos, como el reconocimiento de caracteres escritos a mano

[40], el reconocimiento de objetos en tres dimensiones [33], la detección remota [31] y la extensión a problemas de regresión y predicción de series temporales caóticas [29, 39] y predicción de concentraciones de fármacos en el cuerpo [8], clasificación de imágenes hiperespectrales [7], predicción de la eficacia de oligonucleótidos [4].

2.1.2. Ventajas e inconvenientes

Esta nueva técnica presenta como ventajas más importantes las siguientes:

- **Elevada generalización.** Como hemos dicho antes, las SVM consiguen separar correctamente los datos de entrenamiento, pero además, logran que el límite (o margen) trazado entre las distintas zonas sea el máximo posible.
- **Espacio de características.** Las SVM atacan los problemas en los que los datos no son linealmente separables mediante un mapeado del conjunto de datos de entrenamiento a un espacio de dimensión superior llamado *espacio de características* (*feature space*). Gracias a esta transformación se consigue que la separación lineal sea más probable. Las funciones encargadas de llevar a cabo este mapeado se conocen como funciones kernel y deben cumplir una serie de propiedades matemáticas recogidas en el Teorema de Mercer, [11].
- **Interpretación de la solución.** La solución al problema final viene dada en función de un superficie generada por combinación lineal de unos puntos del conjunto de entrenamiento, los vectores soporte (SVs). Su estudio es fundamental para entender y controlar la *capacidad* del modelo.

Por contra, se nos presentan una serie de desventajas entre las que cabe citar las siguientes:

- **Elevado número de restricciones.** Dado que el cálculo del hiperplano óptimo de decisión está sujeto a una serie de restricciones sobre el espacio de entrada, cuando el número de patrones de entrenamiento sea elevado, podemos tener una carga computacional muy alta asociada a la resolución del hiperplano.
- **Sensibilidad numérica.** La matriz Hessiana utilizada en la optimización es muy sensible al redondeo o pérdida de cifras significativas, lo cual puede afectar al resultado final del problema.
- **Parámetros libres.** En las SVM tenemos libertad para elegir los valores de varios de sus parámetros de ajuste: penalización, kernel, error permitido... Los valores adecuados de cada uno de ellos deben obtenerse a través de la experiencia y de un conocimiento detallado del problema al que se aplique o mediante un test en un conjunto de datos representativos del problema.

2.1.3. Multiplicadores de Lagrange

La técnica de los *multiplicadores de Lagrange* es un método procedente del cálculo variacional que nos permite hallar los puntos estacionarios,$\nabla f = 0$, de una función de varias variables. La particularidad de este procedimiento consiste en que podemos añadir ligaduras o restricciones a los puntos extremos de la función.

La resolución de estos sistemas se basa en la construcción de un funcional Lagrangiano, y a continuación, se exige que todas sus derivadas parciales se anulen, obteniendo así los puntos extremos. Sea $f(x_i)$ la función de varias variables de la cual queremos hallar los puntos extremos, $g(x_i) = 0$ la condición de ligadura, y λ_i los multiplicadores de Lagrange, que son asociados a cada restricción o ligadura para, de esta forma, construir el Lagrangiano siguiente:

$$L(x_i, \lambda_i) \equiv f(x_i) + \lambda_i g(x_i), \tag{2.1}$$

que también se le conoce como problema o funcional primal o primario. Obtenemos la solución del problema resolviendo el sistema:

$$\frac{\partial L}{\partial x_i} = 0 \tag{2.2}$$

$$\frac{\partial L}{\partial \lambda_i} = 0 \tag{2.3}$$

Dado que en problemas de clasificación queremos construir un plano con la máxima separación posible entre clases para obtener así una presumible máxima generalización, el método de los multiplicadores de Lagrange nos da un procedimiento válido y que se ajusta muy bien a nuestro problema. Una interpretación geométrica de la resolución de un problema por multiplicadores de Lagrange se puede ver en la figura 2.1:

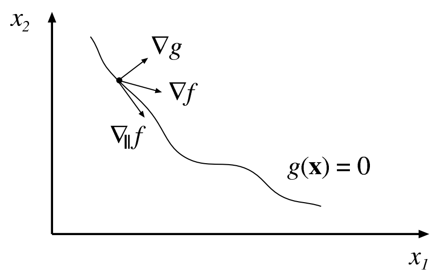


Figura 2.1: Ejemplo de uso de los multiplicadores de Lagrange

2.2. SVM para clasificación de patrones

Dado que históricamente el desarrollo de la teoría de SVM se dirigió a los problemas de clasificación, vamos a estudiar en primer lugar esta aplicación para que una vez analizado el planteamiento básico pasemos a la regresión mediante el uso de la respectiva implemetación, las *Support Vector Regressor, (SVR).*

2.2.1. Hiperplano óptimo de decisión

El hiperplano óptimo de decisión (OHDR) supone una solución a los problemas de reconocimiento con sólo dos clases de datos, y que además son linealmente separables en el espacio de entrenamiento. Futuras ampliaciones permitirán tratar conjuntos de datos no separables linealmente y con varias clases distintas.

Para la resolución del caso más sencillo que hemos nombrado en el párrafo anterior, la resolución del problemas es la siguiente. Sea el conjunto de entrenamiento el dado por los puntos $\{\mathbf{x}_i, y_i\}, i = 1, ..., N$ siendo N el número de muestras e $y_i \in \{-1, +1\}$. Dado que $\mathbf{x}_i \in \mathbb{R}^d$, existe un hiperplano H$\in \mathbb{R}^d$, que delimita las zonas del espacio $\mathbb{R}^d$ pertenecientes a ambas clases.

De este modo, la ecuación que define el hiperplano es:

$$H \equiv \mathbf{w} \cdot \mathbf{x}_i + b = 0 \tag{2.4}$$

siendo $\mathbf{w}$ el vector perpendicular a H, y $\frac{|b|}{||\mathbf{w}||}$ la mínima distancia desde el hiperplano al origen. El margen de separación entre clases será la suma de las distancias más pequeñas entre la muestra de cada clase que más cerca se encuentre del plano; lo que nosotros queremos hacer es que esta distancia sea la máxima posible. Para ello plantearemos las restricciones y el funcional Lagrangiano para obtener la solución según la técnica de los multiplicadores de Lagrange.

Las primeras restricciones que imponemos son las de pertenencia a cada clase:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 0 \ si \ y_i = +1 \tag{2.5}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq 0 \ si \ y_i = -1 \tag{2.6}$$

Si escalamos adecuadamente $\mathbf{w}$ y b podemos reescribir las dos ecuaciones anteriores de manera que podamos combinarla en una única condición de ligadura:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \ si \ y_i = +1 \tag{2.7}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \ si \ y_i = -1 \tag{2.8}$$

Combinando obtenemos:

$$y_i((\mathbf{w}\cdot\mathbf{x}_i+b)-1)\geq 0,\ \ i=1,...,N \tag{2.9}$$

El hiperplano H_1 es aquel formado por los puntos que cumplen la ecuación (2.7), o lo que es lo mismo, $H_1 \equiv \mathbf{w}\cdot\mathbf{x}_i+b=+1$. Igualmente tenemos para los puntos que cumplen la ecuación (2.8) un hiperplano H_2 definido por $H_2 \equiv \mathbf{w}\cdot\mathbf{x}_i+b=-1$. H_1 y H_2 se encuentran a unas distancias del origen de $\frac{|1-b|}{||\mathbf{w}||}$ y $\frac{|-1-b|}{||\mathbf{w}||}$ respectivamente. Ambos planos son paralelos al hiperplano H de (2.4), por lo que se encuentran a una distancia $\frac{1}{||\mathbf{w}||}$ de él. De este modo, concluimos que la mínima separación entre clases será $\frac{2}{||\mathbf{w}||}$. Por ello, minimizar la norma de $\mathbf{w}$, $\|\mathbf{w}\|$ será equivalente a maximizar la distancia entre las clases. En la figura 2.2 podemos ver la construcción tanto del hiperplano H como de H_1 y H_2.

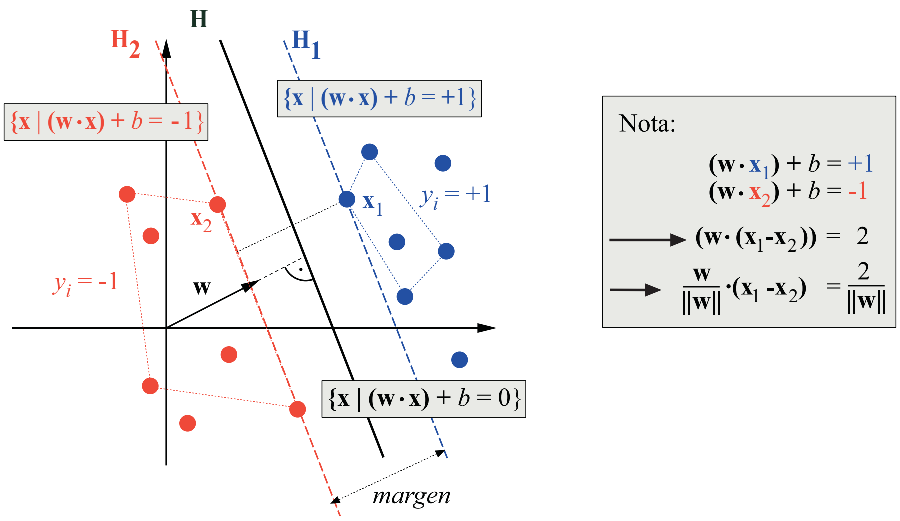


Figura 2.2: Representación geométrica del hiperplano óptimo de decisión

Así pues, la función que queremos minimizar es la norma euclídea de vector $\mathbf{w}$ o lo que resulta equivalente $\|\mathbf{w}\|^2$, sujeto a la restricción de la ecuación (2.9). Con estas condiciones obtenemos el siguiente Lagrangiano primal:

$$L_p=\frac{\|\mathbf{w}\|^{\mathbf{2}}}{2}-\sum_{i=1}^{N}\lambda_i[y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1] \tag{2.10}$$

Como vemos este Lagrangiano es formalmente igual al de la ecuación (2.1), donde se pueden diferenciar la función a minimizar, la ligadura y los multiplicadores de Lagrange λ_i.

Tal y como comentamos, la solución del problema vendría de minimizar L_p respecto a $\mathbf{w}$ y b, teniéndose que cumplir también que las derivadas respecto de λ_i se anularán y $\lambda_i \geq 0$ para todo i. Matemáticamente esto sería:

$$\frac{\partial L_p}{\partial x_i} = 0, \ i = 1, ..., d \tag{2.11}$$

$$\frac{\partial L_p}{\partial \lambda_j} = 0, \ j = 1, ..., N \tag{2.12}$$

En 2.12, el subíndice j se extiende hasta l que no es más que el número de restricciones que impongamos a nuestro problema.

Sin embargo, obtener la solución del sistema anterior resulta ser un problema de programación cuadrática y convexa, de modo que planteando el *problema dual* obtenemos la solución mediante un sistema totalmente equivalente y más sencillo de resolver. Este problema dual consiste en maximizar L_p con las restricciones de que las derivadas respecto de $\mathbf{w}$ y b se anulen y $\lambda_i \geq 0$.

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \ \Rightarrow \ \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \tag{2.13}$$

$$\frac{\partial L_p}{\partial b} = 0 \ \Rightarrow \ \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{2.14}$$

Si ahora sustituimos (2.13) y (2.14) en el Lagrangiano primario (2.10) obtenemos el Lagrangiano del problema dual:

$$L_d = \sum_{i=1}^{N} \lambda_i - \sum_{i,j}^{N} \lambda_i H_{ij} \lambda_j = 0 \tag{2.15}$$

donde hemos introducido la matriz hessiana $H_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$. Recordando que en el problema dual debemos maximizar respecto de λ_i, tiene que cumplirse:

$$\frac{\partial L_d}{\partial \lambda_i} = 0 \ \Rightarrow \ (\mathbf{H}\boldsymbol{\lambda})_i = 1, \ \forall i \tag{2.16}$$

En esta última ecuación comprobamos que para hallar el valor de λ_i y así la solución del problema tenemos que hallar la matriz inversa de la Hessiana.

A partir de aquí vemos en la ecuación (2.13) que el hiperplano surge de una combinación de ciertos puntos de entrenamiento que son los llamados vectores soporte. Estos puntos son aquellos que cumplen la ecuación (2.9), es decir, aquellos puntos de cada clase que más cerca se encuentran al hiperplano separador y, por tanto, son relevantes para la clasificación. Es en estos puntos en los únicos que los λ_i toman valores distintos de cero.

2.2.2. Extensión no lineal de la SVM

El concepto de hiperplano óptimo de decisión se puede extender a problemas no separables y con fronteras no lineales. A continuacón explicaremos brevemente estas modificaciones cuya comprensión será necesaria para el momento en el que abordemos el problema de la regresión.

Clasificación no lineal. Funciones kernel

Para conseguir trazar fronteras de separación no lineales se recurre a funciones de tranformación, $\boldsymbol{\phi}$, que mapean nuestro espacio de entrada de dimensión d a un espacio de dimesión mayor donde sí existe una frontera lineal entre las clases. A este espacio se le denomina *espacio de características* y es ahí donde debe calcularse el OHDR. Esta transformación se puede ver en la figura 2.3.

Para resolver la SVC en problemas separables no linealmente debemos llevar a cabo la siguiente modificación en la formulación del problema: reemplazar $\mathbf{x}_i$ por $\phi(\mathbf{x}_i)$. De este modo la ecuación de ligadura queda:

$$y_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1,\ \ i = 1, ..., N \tag{2.17}$$

Y la solución la obtenemos resolviendo:

$$\max_{\lambda_i} \sum_{i=1}^{N} \lambda_i - \sum_{i,j}^{N} \lambda_i \lambda_j \boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j) \tag{2.18}$$

sujeto a $\lambda_i \geq 0$.

Si observamos la solución dada por (2.18) vemos que la función de transformación $\boldsymbol{\phi}(\cdot)$ solo aparece en la ecuación en forma de producto escalar, por lo que no es necesario conocer de forma explícita la función de mapeo $\boldsymbol{\phi}(\cdot)$, sino sólamente su producto escalar, o según se conoce en la literatura de SVM su *función kernel*, $K(\mathbf{x}_i, \mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j)$. La principal ventaja es que reduce el cálculo computacional, puesto que no se necesita conocer la forma concreta de la función de mapeado, sino solo la matriz de productos escalares, conocida como matriz kernel.

En la bibliografía se han propuesto gran número de funciones kernel, pero en la práctica destacamos las siguiente:

1. Lineal: $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
2. Polinómico: $K(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$
3. Función Gaussiana (RBF): $K(\mathbf{x}_i,\mathbf{x}_j) = \exp\{\text{-}(\mathbf{x}_i \cdot \mathbf{x}_j)^2/2\sigma^2\}$
4. Tangente Hiperbólica (MLP): $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(a(\mathbf{x}_i \cdot \mathbf{x}_j) + r)$

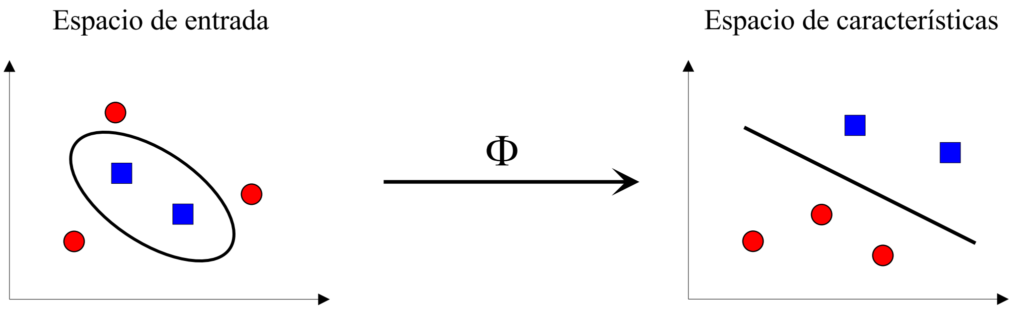


Figura 2.3: Ejemplo de mapeado a una espacio de dimensión superior mediante un función kernel

donde los $\mathbf{x}_i$ son los patrones de entrada y el producto es un producto escalar de vectores.

Los kernels o núcleos RBF y MLP representan una solución equivalente a las redes neuronales correspondientes, con la ventaja de que los parámetros libres (el número de neuronas ocultas equivale al número de SVs y sus pesos equivalen a los asociados a éstos) se ajustan de manera automática.

Conjuntos no separables

El mapeado a un espacio de características transfiere la potencia del método a la matriz de kernel. Sin embargo, el Teorema de Cover [12] que permite ésto no asegura que las muestras sean completamente separables de forma lineal.

Si tenemos un problema de este tipo, como el ejemplo de la figura 2.4, se introduce en la función objetivo (la cual hay que minimizar) un término asociado al coste, C, que mide la importancia de los errores de clasificación cometidos durante el entrenamiento. Esto se lleva a cabo mediante variables de relajación (*slack variables*) ξ_i, $i = 1, ..., L$ en las ligaduras. Haciendo esto, las restricciones quedan del siguiente modo:

$$\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b \geq +1 - \xi_i, \; para \; y_i = +1 \tag{2.19}$$

$$\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b \leq -1 + \xi_i, \; para \; y_i = -1 \tag{2.20}$$

$$\xi_i \geq 0, \; \forall i \tag{2.21}$$

De las ecuaciones (2.19) y (2.20) concluimos que, cuando tenga lugar un error de clasificación ξ_i, será mayor que la unidad, obteniéndose así que $\sum_i \xi_i$ es el límite superior del número de errores de entrenamiento. Así pues, la asignación de la función

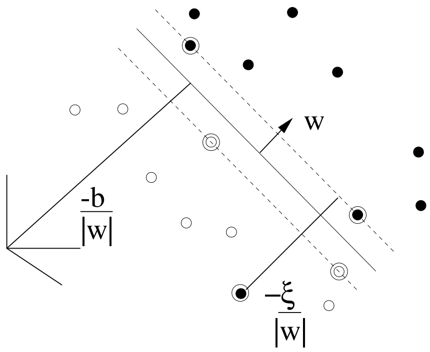


Figura 2.4: Ejemplo de conjunto de datos no separables.

de coste para estos errores es cambiando la función objetivo, que pasará a ser:

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i (\xi_i)^k. \tag{2.22}$$

De nuevo, nos encontramos ante un problema de programación convexa según el valor de k, es decir segúnque valor escojamos tendremos distitna función de coste. Para $k = 2$ y $k = 1$ tenemos un problema de programación cuadrática, siendo $k = 1$ el único valor que nos asegura un prblema de programación cuadrática. En concreto para, $k = 1$ los ξ_i y los multiplicadores de Lagrange λ_i no aparecen en el problema dual, que de este modo pasa a ser:
Maximizar:

$$L_d \equiv \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j \boldsymbol{\phi}(\mathbf{x}_i)\cdot\boldsymbol{\phi}(\mathbf{x}_j) \tag{2.23}$$

sujeto a:

$$0 \leq \lambda_i \leq C, \tag{2.24}$$

$$\sum_i \lambda_i y_i = 0 \tag{2.25}$$

Obteniendo ahora como solución para el hiperplano la siguiente:

$$\mathbf{w} = \sum_{i=1}^{N_S} \lambda_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \tag{2.26}$$

donde el sumatorio se extiende al número total de vectores soporte, N_S. Lo que ocurre ahora es que se ha impuesto un límite superior al valor de los multiplicadores de Lagrange, tal y como se desprende la ecuación (2.24).

Sustituyendo las ecuaciones (2.19), (2.20) y (2.21) en (2.22) obtenemos el Lagrangiano del primal, que queda como sigue:

$$L_p = \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\{y_i(\mathbf{w}_i \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i\} - \sum_{i=1}^{N}\mu_i\xi_i \tag{2.27}$$

donde μ_i es el multiplicador de Lagrange introducido para forzar la positividad de ξ_i. En estas condiciones, las características que debe tener un patrón para ser la solución deseada la función objetivo son las dadas por el Teorema de Karush-Kuhn-Tucker (KKT), las cuales son:

1. Que se cumplan las restricciones lineales (2.19), (2.20) y (2.21).

2. Que se anule el gradiente de L_p en la ecuación (2.27):

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N}\lambda_i y_i \boldsymbol{\phi}(\mathbf{x}_i) = 0 \tag{2.28}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{N}\lambda_i y_i = 0 \tag{2.29}$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \tag{2.30}$$

3. Que los multiplicadores de Lagrange sean positivos:

$$\lambda_i \geq 0, \;\; i = 1, ..., N \tag{2.31}$$

$$\mu_i \geq 0, \;\; i = 1, ..., N \tag{2.32}$$

4. Que cada multiplicador por su restricción sea cero (conjunto ortogonal):

$$\lambda_i\{y_i(\mathbf{w}_i \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i\} = 0, \;\; i = 1, ..., N \tag{2.33}$$

$$\mu_i\xi_i = 0, \;\; i = 1, ..., N \tag{2.34}$$

Como vemos en la ecuación (2.26), la solución viene dada de nuevo únicamente en función de ciertos puntos de entrenamiento (aquellos cuyos λ_i sean no nulos) y de sus respectivos multiplicadores de Lagrange, que actúan como pesos o relevancia de cada muestra. El valor del coste C es asignado por el usuario, de acuerdo al conocimiento a priori del problema o de la distribución del ruido. Si queremos obtener un alto grado de generalización, deberemos usar valores bajos de C para no penalizar demasiado los errores y potenciar así la componente de regularización del funcional a minimizar ($||\mathbf{w}^2||$), mientras que si lo que queremos es una modelización muy ajustada, tendremos que trabajar con valores de C muy elevados, con lo que la suavidad del modelo es menor y corremos el riego de sobreajustar ("overfitting") el modelo a los datos de entrenamiento, sin ninguna (o poca) capacidad de generalización ante datos nunca vistos.

2.3. SVM para regresión y aproximación de funciones

En este punto vamos a presentar la herramienta fundamental de trabajo del presente proyecto: la SVM para problemas de regresión y aproximación de funciones, *Support Vector Regressor* (SVR). Vamos a detallar el funcionamiento de la SVR para el caso no lineal y para el caso con pérdidas.

2.3.1. Función de coste a minimizar: ε-insensitive

Cuando nos enfrentamos a un problema de regresión, observamos que la principal diferencia es que ahora los valores de y_i forman un conjunto continuo e infinito en lugar de ser discreto y finito, como teníamos en el caso de clasificación. Por ello, nos vemos obligados a introducir una función de coste distinta.

Para que el método de resolución sea lo más similar posible al de la SVM para problemas de clasificación (*Support Vector Classifier, SVC*), se usan funciones de coste que hagan que el problema sea el de maximizar un funcional cuadrático. Para ello, dos posibles aproximaciones propuestas en la bibliografía son la función de coste de error absoluto o la función de coste cuadrática [46].

En la práctica la SVR no usa el error absoluto directamente, sino la función de coste de Vapnik (ver figura 2.5) o de error absoluto con zona de insensibilidad ε:

$$f_{Vapnik} = y - f(\mathbf{x}, \mathbf{c})]_{\varepsilon} = \begin{cases} 0 & si \quad |y - f(\mathbf{x}, \mathbf{c})| < \varepsilon \\ |y - f(\mathbf{x}, \mathbf{c})| & en\ caso\ contrario \end{cases} \tag{2.35}$$

Podemos ver que si $\varepsilon = 0$, esta función de coste se corresponde con el valor absoluto.

Existen dos razones por las cuales se trabaja con $\varepsilon \neq 0$, una de orden teórico y otra de orden práctico. La teórica hace referencia a la teoría de aprendizaje; permitiendo un pequeño error ε permitimos que predicciones muy cercanas a la verdadera función, según la zona de insensibilidad, no se consideren como fallos. Desde el punto de vista práctico, $\varepsilon = 0$ implicaría que todas las muestras resultarían ser vectores soporte, lo cual, aparte de poco eficaz computacionalmente, sería una solución trivial al problema. Permitiendo una pequeña desviación ε, obtenemos una disminución

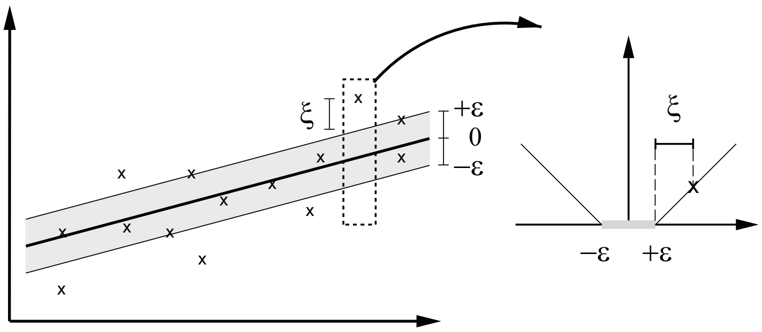


Figura 2.5: Función de coste $\varepsilon - insensitive$ para la SVR. Observamos como aparece un tubo de anchura constante que nos establece el margen de error permitido.

considerable en el número de vectores soporte. Este hecho es muy importante y conduce al hecho de que las SVM ofrecen funciones teóricamente simples o con pocos parámetros. Esta propiedad se conoce como *s*parseness del modelo.

2.3.2. Resolución de la SVR

El modo usual de resolver la SVM considera un margen blando ("soft margin") y se presenta como un problema de minimización cuadrática con restricciones lineales:

$$\min_{\mathbf{w},\xi_i,\xi_i^*,b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i(\xi_i + \xi_i^*) \tag{2.36}$$

con las siguientes restricciones:

$$y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - b \leq \varepsilon + \xi_i, \quad \forall i = 1, ..., n \tag{2.37}$$

$$\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} + b - y_i \leq \varepsilon + \xi_i^*, \quad \forall i = 1, ..., n \tag{2.38}$$

$$\xi_i, \xi_i^* \geq 0, \quad \forall i = 1, ..., n \tag{2.39}$$

donde se usa la misma función sobre las variables auxiliares (ξ_i, ξ_i^*) que para la SVC. De esta manera, según las ecuaciones (2.37) y (2.38), cualquier error menor que ε no llevará asociado un término en ξ_i o ξ_i^* por lo que no aparecerá en la función objetivo. Cada muestra sólo podrá tener asociada una ξ_i o ξ_i^* y por tanto éstas deben ser ortogonales: $\xi_i \cdot \xi_i^* = 0$. De nuevo, tenemos, un modelo no lineal dado por $\hat{y} = \boldsymbol{\phi}(\mathbf{x}_i)\mathbf{w} + b$.

Una vez más, tenemos que resolver la minimización de un funcional cuadrático con restricciones lineales, lo cual, tal y como hemos hecho en puntos anteriores, se lleva a cabo introduciendo dichas restricciones en el funcional a minimizar usando los multiplicadores de Lagrange. Así el Lagrangiano del primal que obtenemos es el siguiente:

$$\begin{aligned} L_P = \; & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\textstyle\sum_i(\xi_i+\xi_i^*) - \sum_i \lambda_i(\varepsilon - y_i + \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} + b) \\ & - \textstyle\sum_i \lambda_i^*(\varepsilon + y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - b) - \sum_i(\mu_i\xi_i + \mu_i^*\xi_i^*) \end{aligned} \tag{2.40}$$

y su única solución viene determinada por las condiciones KKT:

1. Se deben cumplir las restricciones (2.37), (2.38), (2.39).
2. El gradiente de L_p se debe anular en (2.36) con respecto a las variables que debe realizarse la minimización:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N}(\lambda_i - \lambda_i^*)\boldsymbol{\phi}(\mathbf{x}_i) = 0 \tag{2.41}$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{N}(\lambda_i^* - \lambda_i) = 0 \tag{2.42}$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \lambda_i - \mu_i, = 0,\ i = 1, ..., N \tag{2.43}$$

$$\frac{\partial L_p}{\partial \xi_i^*} = C - \lambda_i^* - \mu_i^* = 0,\ i = 1, ..., N \tag{2.44}$$

3. Los multiplicadores de Lagrange deben ser positivos: $\lambda_i, \lambda_i^*, \mu_i, \mu_i^* \geq 0$.
4. Cada multiplicador por su restricción debe ser cero:

$$\lambda_i(\xi_i + \varepsilon - y_i + \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} + b)) = 0,\ \forall i = 1, ..., N \tag{2.45}$$

$$\lambda_i^*(\xi_i^* + \varepsilon + y_i + -\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - b)) = 0,\ \forall i = 1, ..., N \tag{2.46}$$

$$\mu_i\xi_i = 0,\ i = 1, ..., N \tag{2.47}$$

$$\mu_i^*\xi_i^* = 0,\ i = 1, ..., N \tag{2.48}$$

Una vez llevada a cabo la simplificación del primal, el problema dual es maximizar con respecto a λ_i y λ_i^* el siguiente funcional:

$$L_d = \sum_{i=1}^{L} y_i(\lambda_i - \lambda_i^*) - \varepsilon\sum_{i=1}^{L}(\lambda_i + \lambda_i^*) - \frac{1}{2}\sum_{i,j}^{L}(\lambda_i - \lambda_i^*)(\lambda_i + \lambda_i^*)K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.49}$$

sujeto a (2.42) y a $0 \leq \lambda_i, \lambda_i^* \leq C$ que resulta ser un problema de optimización cuadrática con restricciones muy similar al propuesto para el caso de la SVC. La forma usual de resolver estos problemas es mediante la inversión de la matriz Hessiana, como vimos en (2.2).

2.4. SVM Adaptativa

El concepto de SVM adaptativa que usamos aquí no hace referencia a problemas no estacionarios desde un punto de vista temporal, como se hace en filtrado adaptativo o redes neuronales, aunque la formulación se puede adaptar también a estas situaciones. En nuestro caso, lo que se hace es aplicar distintos C y ε a cada muestra de entrenamiento, de forma que el modelo se *a*dapta intrínsecamente a la complejidad de cada muestra en la distribución. En problemas de clasificación solemos tener situaciones en las que se cuenta con muchas muestras de una clase y pocas de otra, por lo que tedremos que usar penalizaciones distintas para cada una de las clases con el fin de evitar en la medida de lo posible falsas predicciones, [23]. Respecto a la regresión, el uso de distintas zonas de insensibilidad en el dominio de la función, permitirá entrenar la SVM siguiendo las variaciones de la función de densidad de probabilidad a lo largo del propio dominio, [32, 3]. En la práctica se suele trabajar variando ε y C de forma inversa, de modo que zonas donde necesitemos una modelización muy precisa se establece un error permitido muy pequeño y una penalización por fallo muy alevada.

Así pues, en el caso de regresiones, podemos conseguir mejores ajustes en las partes del espacio de entrenamiento que más interés tengamos y dar poco importancia a otras zonas que no nos interesen. Por ejemplo, una C grande en zonas en las que necesitemos un alto grado de ajuste, y una C baja en las que nos interese una mayor generalización. El caso de ε es similar ya que un ε grande conduce a permitir mayores errores, y un ε pequeño hace que aumente el número de muestras que caen fuera de la zona de insensibilidad, aumentando así la penalización. Intuitivamente podemos ver C como el coste que asignamos a equivocarnos al predecir una muestra, y ε al error permitido en cada una de ellas. Por tanto, podríamos pensar, con este razonamiento, que $\varepsilon = 0$ y $C = \infty$ sería una excelente solución. El problema con este caso es que el modelo consigue excelentes resultados en el conjunto con el que lo hemos ajustado, pero su capacidad para modelar otros datos se pierde. De esta forma, la elección de C y ε cae en manos del usuario y se realiza en función de los errores que un modelo entrenado (empleando datos de entrenamiento) obtiene en un conjunto de test externo pero proveniente de la misma distribución (datos de validación o test) .

2.4.1. Formulación estándar

La realización de ese algortimo conlleva resolver el siguiente problema de minimización convexo con restricciones:

$$\min_{\mathbf{w},b,\xi_i,\xi_i^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i k_i(\xi_i + \xi_i^*), \;\; k_i = \varepsilon/c_i \tag{2.50}$$

sujeto a $\xi_i \geq 0$, $\xi_i^* \geq 0$ y a

$$y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - b \leq \frac{\varepsilon}{c_i} + \xi_i \qquad \forall i = 1, \ldots, n \tag{2.51}$$

$$\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} + b - y_i \leq \frac{\varepsilon}{c_i} + \xi_i^* \qquad \forall i = 1, \ldots, n \tag{2.52}$$

donde c_i es una función de confianza sobre la muestra i. Se modifica el funcional a minimizar (2.50) y las restricciones (2.51) y (2.52) para pesar de distinta manera el error sobre cada muestra (c_i) y el error permitido asociado a cada una de ellas (insensibilidad variable, ε_i). Normalmente, interesa una penalización alta en muestras con ε pequeño por lo que se puede emplear una relación entre ambas del tipo $k_i = \frac{\varepsilon}{c_i}$.

2.4.2. Resolución de una SVR adaptativa sin bias

Existe un gran número de problemas en los que la serie temporal a modelar o predecir contiene zonas estacionarias, y por tanto, implícitamente existe un sesgo en la regresión. En la teoría de SVM este problema se resuelve usando funciones kernel con el bias implícito (a veces llamadas, funciones de localidad bien definida), como el kernel RBF (Radial Basis Function). En general si tenemos kernels definidos positivos se cumple que su bias está ímplicito.

Partiendo del funcional (2.50) y de las restricciones (2.51) y (2.52), hacemos $b = 0$, obteniendo así:

$$\min_{\mathbf{w},\xi_i,\xi_i^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i k_i(\xi_i + \xi_i^*), \;\; k_i = \varepsilon/c_i \tag{2.53}$$

sujeto a las restricciones:

$$y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} \leq \frac{\varepsilon}{c_i} + \xi_i \qquad \forall i = 1, \ldots, n \tag{2.54}$$

$$\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - y_i \leq \frac{\varepsilon}{c_i} + \xi_i^* \qquad \forall i = 1, \ldots, n \tag{2.55}$$

$$\xi_i, \xi_i^* \geq 0 \qquad \forall i = 1, \ldots, n \tag{2.56}$$

Si ahora introducimos las restricciones lineales en el funcional que se quiere minimizar mediante multiplicadores de Lagrange llegamos a:

$$\begin{aligned} L_P \equiv\; & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\textstyle\sum_i k_i(\xi_i + \xi_i^*) - \sum_i \lambda_i \left(\tfrac{\varepsilon}{c_i} + \xi_i - y_i + \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w}\right) \\ & - \textstyle\sum_i \lambda_i^* \left(\tfrac{\varepsilon}{c_i} + \xi_i^* + y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w}\right) - \sum_i(\mu_i\xi_i + \mu_i^*\xi_i^*) \end{aligned} \tag{2.57}$$

De nuevo la solución quedará definida por las condiciones KKT, que en esta caso son:

1. Se deben cumplir las restricciones (2.54), (2.56), (2.39).

2. El gradiente de L_p se debe anular en (2.53) con respecto a las variables que debe realizarse la minimización:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} (\lambda_i - \lambda_i^*) \boldsymbol{\phi}(\mathbf{x}_i) = 0 \tag{2.58}$$

$$\frac{\partial L_p}{\partial \xi_i} = Ck_i - \lambda_i - \mu_i, = 0, \; i = 1, ..., N \tag{2.59}$$

$$\frac{\partial L_p}{\partial \xi_i^*} = Ck_i - \lambda_i^* - \mu_i^* = 0, \; i = 1, ..., N \tag{2.60}$$

3. Los multiplicadores de Lagrange deben ser positivos: $\lambda_i, \lambda_i^*, \mu_i, \mu_i^* \geq 0$.

4. Cada multiplicador por su restricción debe ser cero (condición de ortogonalidad):

$$\lambda_i \left(\xi_i + \frac{\varepsilon}{c_i} - y_i + \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} + b) \right) = 0, \; \forall i = 1, ..., N \tag{2.61}$$

$$\lambda_i^* \left(\xi_i^* + \frac{\varepsilon}{c_i} + y_i + -\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - b) \right) = 0, \; \forall i = 1, ..., N \tag{2.62}$$

$$\mu_i \xi_i = 0, \; i = 1, ..., N \tag{2.63}$$

$$\mu_i^* \xi_i^* = 0, \; i = 1, ..., N \tag{2.64}$$

Como ahora no tenemos la condición:

$$\frac{\partial L_p}{\partial b} = 0 \tag{2.65}$$

no tiene por qué cumplirse:

$$\frac{\partial L_p}{\partial b} = \sum_i (\lambda_i^* - \lambda_i) = 0 \tag{2.66}$$

Esto supone, además, otra ventaja; a la hora de resolver el problema sólo tenemos que minimizar respecto de un parámetro, $\mathbf{w}$, en lugar de dos, $\mathbf{w}$ y b, lo cual hará que la carga computacional del algoritmo sea menor. En [16], se demuestra que los resultados no cambian significativamente.

2.4.3. Iterative Re-Weighted Least Squares (IRWLS)

Hemos visto cómo para resolver la SVM tanto en clasificación como en regresión el cálculo se reducía a un problema de minimización cuadrática con restricciones lineales. Estos normalmente se suelen resolver mediante algoritmos muy conocidos de programación cuadrática (*QP, "quadratic programming"*). Además, hemos visto cómo en el caso de las SVM, y también las SVR, tenemos que introducir una restricción por cada muestra de entrenamiento, con lo que el coste computacional aumenta con el número de ejemplos empleados [1] Esto hace que si no se dispone de un buen procedimiento para la resolución del problema por bloques, la minimización puede ser muy costosa computacionalmente e inviable mediante ordenadores de sobremesa para problemas de más de 10000 muestras.

Sin embargo, la implementación IRWLS (*Iterative Re-Weighted Least Squares, IRWLS*) permite resolver estos problemas de un modo más eficaz desde un punto de vista computacional. Esta aproximación resuelve la SVM mediante mínimos cuadrados ponderados e iterados, sustituyendo así la programación cuadrática. Además, permite modificar la función de coste y hacerla adaptativa tanto en clasificación como en regresión.

El procedimiento IRWLS se basa en reescribir el Lagrangiano primal hasta dejarlo en forma de un funcional de mínimos cuadrados ponderados. Dicho funcional se resuelve iterativamente hasta que no obtengamos variación, en un cierto margen de tolerancia, de las soluciones del problema $\mathbf{w}$ y b.

El algortimo IRWLS se puede aplicar tanto en problemas de clasificación como en problemas de regresión. Además también permite el uso de la SVM adaptativa. Por todas estas razones está se usará durante todo el proyecto la implementación IRWLS para resolver las SVM.

Así pues, vamos a ver como se resuelve el funcional de la SVR adaptativa sin bias (2.57) mediante este algoritmo. Lo primero que se hace es reorganizar el primal L_p, acumulando los factores dependientes de ξ_i y ξ_i^* en un sumatorio puesto que en la solución del problema ambos términos deben ser nulos:

$$\begin{aligned} L_P = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \lambda_i(y_i - \frac{\varepsilon}{c_i} - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w}) - \sum_i \lambda_i^*(\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - \frac{\varepsilon}{c_i} - y_i\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w}) \\ + \sum_i \xi_i(Ck_i - \lambda_i - \mu_i) + \sum_i \xi_i^*(Ck_i - \lambda_i^* - \mu_i^*) \end{aligned}$$

[1] Este es el caso opuesto al de las redes neuronales en que el coste computacional aumenta con el número de entradas o dimensión del espacio de entrada.

con lo que tenemos:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \lambda_i \left(y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - \frac{\varepsilon}{c_i} \right) - \sum_i \lambda_i^* \left(\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - y_i - \frac{\varepsilon}{c_i} \right) \quad (2.67)$$

Este funcional se puede minimizar planteando su equivalente RLS:

$$\begin{aligned} L'_P =& \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}\sum_i \frac{2\lambda_i}{y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - \frac{\varepsilon}{c_i}} \left(y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - \frac{\varepsilon}{c_i} \right)^2 \\ &+ \frac{1}{2}\sum_i \frac{2\lambda_i^*}{\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - y_i - \frac{\varepsilon}{c_i}} \left(\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - y_i - \frac{\varepsilon}{c_i} \right)^2 \\ =& \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}\sum_i \left(a_i e_i^2 + a_i^* (e_i^*)^2 \right) \end{aligned} \quad (2.68)$$

donde se ha definido

$$e_i = y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - \frac{\varepsilon}{c_i} \quad (2.69)$$

$$e_i^* = \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - y_i - \frac{\varepsilon}{c_i} \quad (2.70)$$

$$a_i = \frac{2\lambda_i}{y_i - \boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - \frac{\varepsilon}{c_i}} = \frac{2\lambda_i}{e_i} \quad (2.71)$$

$$a_i^* = \frac{2\lambda_i^*}{\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{w} - y_i - \frac{\varepsilon}{c_i}} = \frac{2\lambda_i^*}{e_i^*} \quad (2.72)$$

Podemos entender e_i y e_i^* como una medida del error en la modelización siendo a_i y a_i^* sus correspondientes pesos. Como $a_i = a_i(e_i)$ y $a_i^* = a_i^*(e_i^*)$ el funcional (2.68) se deberá resolver de forma iterativa teniéndose que minimizar únicamente respecto a $\mathbf{w}$ con lo que llegamos a:

$$\frac{\partial L'_P}{\partial \mathbf{w}} = \mathbf{w} - \boldsymbol{\Phi}^T \mathbf{D_a}[\mathbf{y} - \boldsymbol{\Phi}\mathbf{w} - \varepsilon] + \boldsymbol{\Phi}^T \mathbf{D_{a^*}}[\boldsymbol{\Phi}\mathbf{w} - \varepsilon - \mathbf{y}] = \mathbf{0} \quad (2.73)$$

donde:

$$\begin{aligned} \mathbf{y} &= [y_1, ..., y_n] \\ \boldsymbol{\Phi} &= [\phi(x_1), ..., \phi(x_n)] \\ \mathbf{w} &= [w_1, ..., w_n] \\ (\mathbf{D_{a^*}}) &= a_{ij}\delta(i-j) \quad \forall i, j = 1, ..., N \\ \varepsilon &= \left[\frac{\varepsilon}{c_1}, ..., \frac{\varepsilon}{c_n} \right] \end{aligned}$$

El procedimiento para la resolución consta de tres pasos:

1. Minimizar L'_P respecto a $\mathbf{w}$ supuestos a_i fijos. Por tanto, ahora tendremos que resolver un sistema de una ecuación ($\frac{\partial L'_P}{\partial \mathbf{w}} = 0$) con una sola incógnita ($\mathbf{w}$). Agrupando la ecuación (2.73):

$$\mathbf{w} \cdot \left[\mathbf{\Phi}^T \mathbf{D}_{(\mathbf{a}+\mathbf{a}^*)} \mathbf{\Phi} + \mathbf{I}\right] = \mathbf{\Phi}^T (\mathbf{D}_{(\mathbf{a}+\mathbf{a}^*)} \mathbf{y} + \mathbf{D}_{(\mathbf{a}^*-\mathbf{a})} \varepsilon)$$

2. Actualizar a_i y a_i^* con el resultado anterior. Este segundo paso acarrea el calculo de a_i y a_i^* de manera que se cumplan las condiciones KKT con la nueva solución $\mathbf{w}$. Para ello hay que verificar:

$$a_i = \begin{cases} 0, & e_i < 0 \\ \dfrac{2k_i C}{e_i}, & e_i \geq 0 \end{cases} \tag{2.74}$$

$$a_i^* = \begin{cases} 0, & e_i^* < 0 \\ \dfrac{2k_i C}{e_i^*}, & e_i^* \geq 0 \end{cases} \tag{2.75}$$

3. Repetir hasta que no varíe $\mathbf{w}$.

En realidad, la forma de resolver este problema no es así (en el primer paso se necesita conocer $\boldsymbol{\phi}$), sino que se fuerza a trabajar con kernels:

$$K(\cdot,\cdot) = \mathbf{\Phi}^T(\cdot)\mathbf{\Phi}(\cdot)$$

Para ello, se fuerza que la solución $\mathbf{w}$ sea la combinación lineal de las muestras de entrenamiento:

$$\mathbf{w} = \sum_i \beta_i \boldsymbol{\phi}(\mathbf{x}_i) = \mathbf{\Phi}^T \boldsymbol{\beta} \tag{2.76}$$

Con lo que el primer paso del algoritmo queda:

$$\left[\mathbf{H} + \mathbf{D}^{-1}_{(\mathbf{a}+\mathbf{a}^*)}\right] \cdot \boldsymbol{\beta} = \mathbf{y} - \mathbf{D}_\varepsilon \mathbf{E} \tag{2.77}$$

que no depende de $\phi(\cdot)$, sino de $K(\cdot,\cdot)$, donde:

$$(\mathbf{H})_{ij} = \phi^T(x_i) \cdot \phi(x_j) \equiv K(x_i, x_j) \tag{2.78}$$

En este capítulo hemos hecho una revisión en profundidad sobre la naturaleza de las SVM, centrándonos en la aplicación para regresión de funciones SVR. También hemos descrito la implementación con la que trabajaremos a lo largo del PFC. Tal y como se comentó al inicio del capítulo, la SVR será usada para estimar espectros de imágenes naturales. Como la visión humana es un fenómeno subjetivo en el siguiente capítulo abordaremos el estudio de la percepción visual.

Capítulo 3

Modelos Perceptuales

3.1. Percepción visual humana: modelo general

El proceso de percepción visual humana tiene lugar en una parte del cerebro conocida como el córtex visual. En él las neuronas que lo componen responden de distinta manera a los estímulos que les llegan desde los nervios ópticos. En concreto presentan respuestas distintas según sea la frecuencia, la localización espacial y su orientación angular del estímulo. Este comportamiento es el que hace posible descartar información dada su poca relevancia perceptual.

Los modelos más recientes sobre el sistema visual humano asumen que las imágenes que percibimos son primero mapeadas a un dominio de frecuencia local, a través de un banco de filtros lineales de ganancia variable [28, 49], para luego producir una respuesta según una nueva transformación no lineal [18, 51].

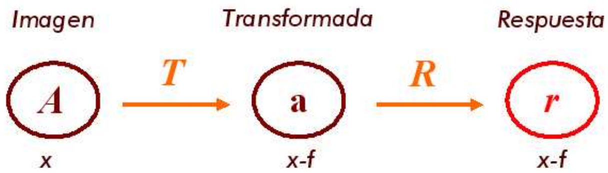


Figura 3.1: Proceso de percepción visual.

Estas transformaciones se pueden observar en la figura 3.1. La transformación al dominio de frecuencias locales mapea la imagen original, $\mathbf{A_x}$, a un nuevo espacio donde la representaremos como combinación de unas funciones base, $\mathbf{T_{p,x}}$, de modo que:

$$\mathbf{a_f} = \sum_x T_{p,x} \mathbf{A_x} \tag{3.1}$$

Como en la mayoría de las ocasiones la transformación es ortogonal, podemos volver al dominio original mediante la matriz traspuesta:

$$\mathbf{A_x} = \sum_p T_{x,p}\mathbf{a_p} \tag{3.2}$$

Gráficamente podemos ver todo el proceso que lleva a cabo el sistema visual humano en la figura 3.2:

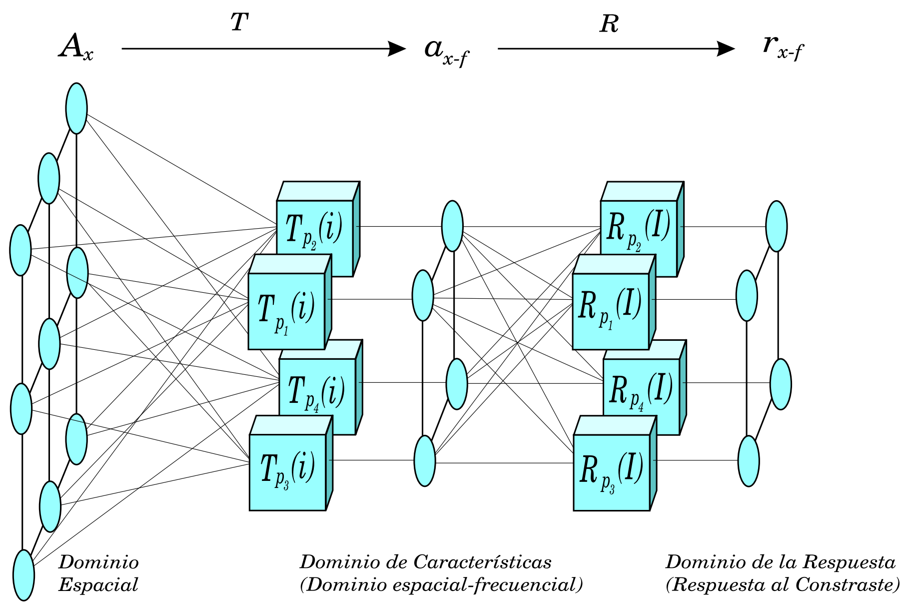


Figura 3.2: Esquema representativo de las transformaciones que lleva cabo el sistema visual humano.

Estas funciones base están localizadas tanto en el dominio espacial como en el dominio frecuencial. Los coeficientes según esta base representan la salida de cada filtro cuando se hace pasar a través de cada uno de ellos una determinada imagen. Por ello la matriz T que define esta transformación esta compuesta por las respuestas impulsionales de cada uno de los filtros (ver figura 3.3).

Estas transformaciones llevan a cabo el siguiente recubrimiento del dominio de Fourier mediante las bandas de paso de los filtros de la tranformada T (ver figura 3.4) :

No obstante, las transformaciones frecuenciales locales como la Block-DCT (ver sección 3.4) tiene propiedades cualitativas similares, es decir, sus funciones base están localizadas simultáneamente tanto en el dominio frecuencial como espacial. Por ello, pueden utilizarse como primera aproximación para modelizar la transformada T [50, 48]. En la segunda etapa, R, las respuesta de los sensores lineales se modifica

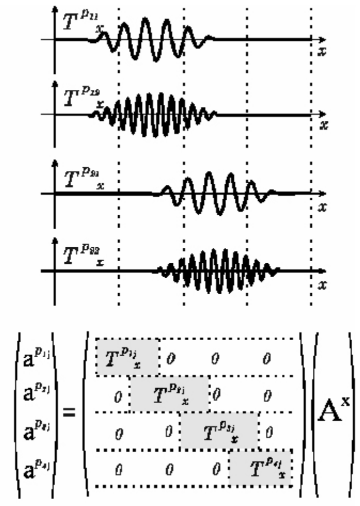


Figura 3.3: Ejemplo de la transformada al dominio de frecuencia local. Observamos como para cada elemento del dominio espacial se lleva a cabo la transformación según distintas funciones base para cada sub-banda frecuencial.

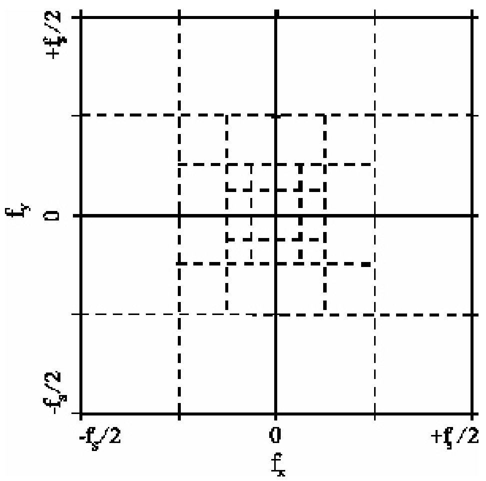


Figura 3.4: Recubrimiento del dominio de Fourier.

dando lugar a una respuesta final. Para pasar a este dominio, $\mathbf{r_i}$, se han propuesto varias aproximaciones:

- **Comportamiento lineal [2, 27, 51].** La salida de cada filtro se multiplica por su respectiva ganacia F_i, obteniendo así directamente la respuesta del sistema. A esta ganancia se le llama *Función de Sensibilidad al Constraste (Contrast*

Sensitivity Function, CSF) y su forma se puede ver en la figura 3.5.

$$r_i = F_i \cdot a_i \tag{3.3}$$

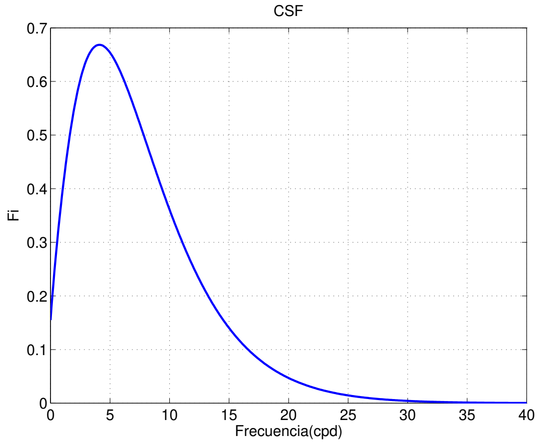


Figura 3.5: CSF acromática. En el eje x se muestra la frecuencia en ciclos por grado, mientras que en eje y se representa la ganancia, F_i , de los filtros lineales del sistema visual humano para cada frecuencia.

- **Modelo no lineal sin interacción [21].** En esta aproximación la respuesta para una determinada frecuencia no sólo depende de su sensibilidad al constraste dada por la CSF, puesto que también se rectifica y normaliza según unos valores γ y β_i calculados experimentalmente.

$$r_i = \frac{(F_{ii} \cdot a_i)^\gamma}{\beta_i + (F_{ii} \cdot a_i)^\gamma} \tag{3.4}$$

- **Modelo no lineal con interacción [18, 51].** Para el cálculo de la respuesta a una determinada frecuencia no se considera solo el coeficiente asociado a dicha frecuencia sino todos los coeficientes de la señal.

$$r_i = \frac{(F_{ii} \cdot a_i)^\gamma}{\beta_i + \sum_j H_{ij}(F_{jj} \cdot a_j)^\gamma} \tag{3.5}$$

En la figura 3.6 podemos ver las curvas de respuesta para cada uno de los casos presentados anteriormente.

Así pues, son dos las carácterísticas fundamentales del sistema visual humano, la transformación al dominio de frecuencia local y la respuesta dependiente de la frecuencia, ya sea en un modelo lineal sencillo o con modelos no lineales de diferente sofisticación. Estas dos propiedades serán claves a la hora de introducir distorsiones en el proceso de compresión, porque debido a estas características algunos coeficientes aceptarán más error que otros, tal y como vamos a mostrar a continuación.

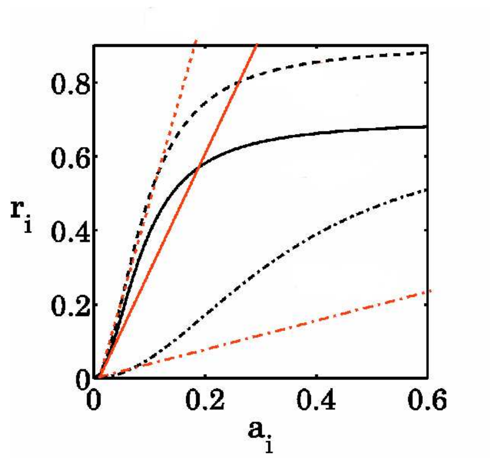


Figura 3.6: Respuestas al estímulo según los distintos modelos: modelo lineal (líneas color naranja) y modelo no lineal sin interación (lineas negras). El modelo con interacción también tiene un comportamiento no lineal tipo sigmoide pero a diferencia del modelo sin interacción, la sigmoide depende de la entrada. El tipo de lineas hace referencia a la fracuencia de muestreo, 6 cpd (líneas rayadas), 2 cpd (líneas contínuas) y 16 cpd (líneas punto-raya).

3.2. Medida de distancias perceptuales y métricas perceptuales

La medición del error cometido en la codificación de la imagen se suele llevar a cabo en el dominio espacial mediante el cálculo de valores como son el RMSE o la SNR. Sin embargo esto no es del todo correcto puesto que nuestro cerebro interpreta la información en el dominio transformado y no en el espacial.

En el siguiente ejemplo vamos a demostrar que esto es así y vamos a ver cuales son las características básicas de la percepción de distorsiones. Para ello, tomaremos errores de distinta naturaleza frecuencial pero de la misma energía (ver figura 3.7) y los aplicaremos sobre distintas imágenes (ver figura 3.8). De este modo, todas ellas presentarán los mismos valores de RMSE. Sin embargo, comprobaremos que la distorsión perceptual será distinta dependiendo de la señal de error que hayamos introducido y de la imagen sobre la que la hayamos aplicado, como se ve en la figura 3.9.

De la observación de la figura 3.9 obtenemos principalmente dos conclusiones:

- **Dependencia frecuencial.** El efecto más importante es que, independientemente del tipo de imagen sobre el que se aplica el ruido, su efecto depende de su composición frecuencial. Este efecto, puede explicarse mediante el modelo

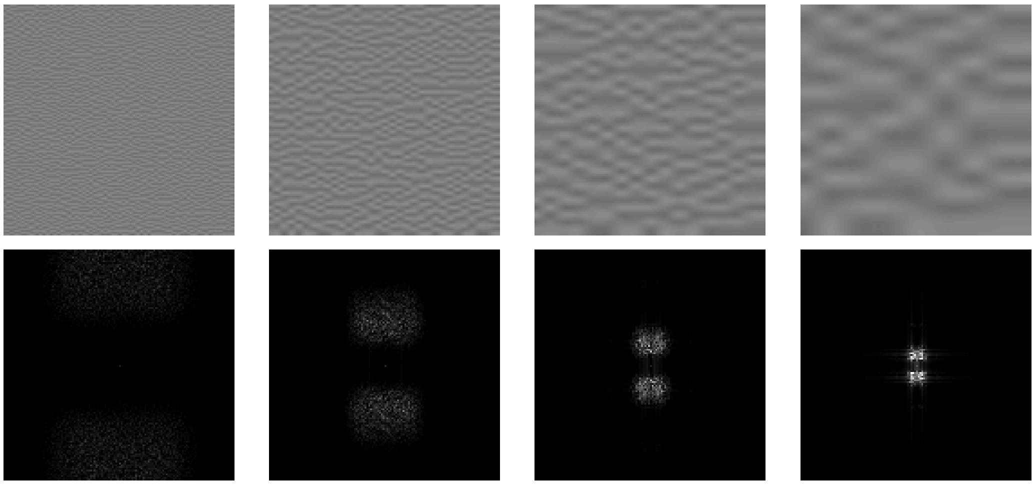
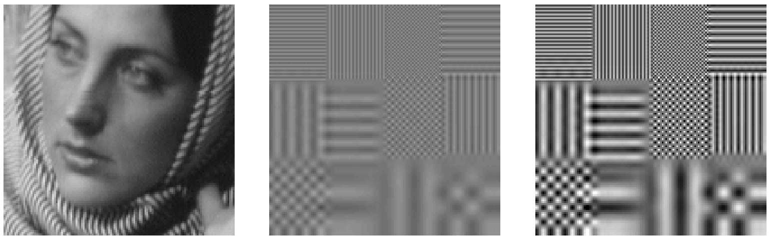


Figura 3.7: Señales de ruido (fila superior) y sus respectivas composiciones frecuenciales (fila inferior). Las frecuencias centrales de las señales de cada columna son 24 cpd, 12 cpd, 6 cpd y 3 cpd respectivamente.



(a) (b) (c)

Figura 3.8: Imágenes sobre las que aplicaremos las señales de ruido vistas en la figura 3.7. Utilizamos una imagen natural (Barbara) (a) y dos imágenes sintéticas, una de bajo contraste (b) y otra de alto contraste (c). En el caso de las imágenes sintéticas están formadas por zonas con frecuencias definidas coincidentes con las de las señales de ruido.

de respuesta lineal: la CSF, figura 3.5). La sensación de distorsión depende de la banda frecuencial del ruido porque la pendiente de la respuesta de los sensores depende de la frecuencia, como se comprueba al movernos en las filas de la figura 3.9.

- **Enmascaramiento.** El enmascaramiento se refiere a la disminución de la sensibilidad al ruido debido al estímulo sobre el que el ruido es añadido. El enmascaramiento tiene dos vertientes ejemplificadas en la figura 3.9: *enmascaramien-*

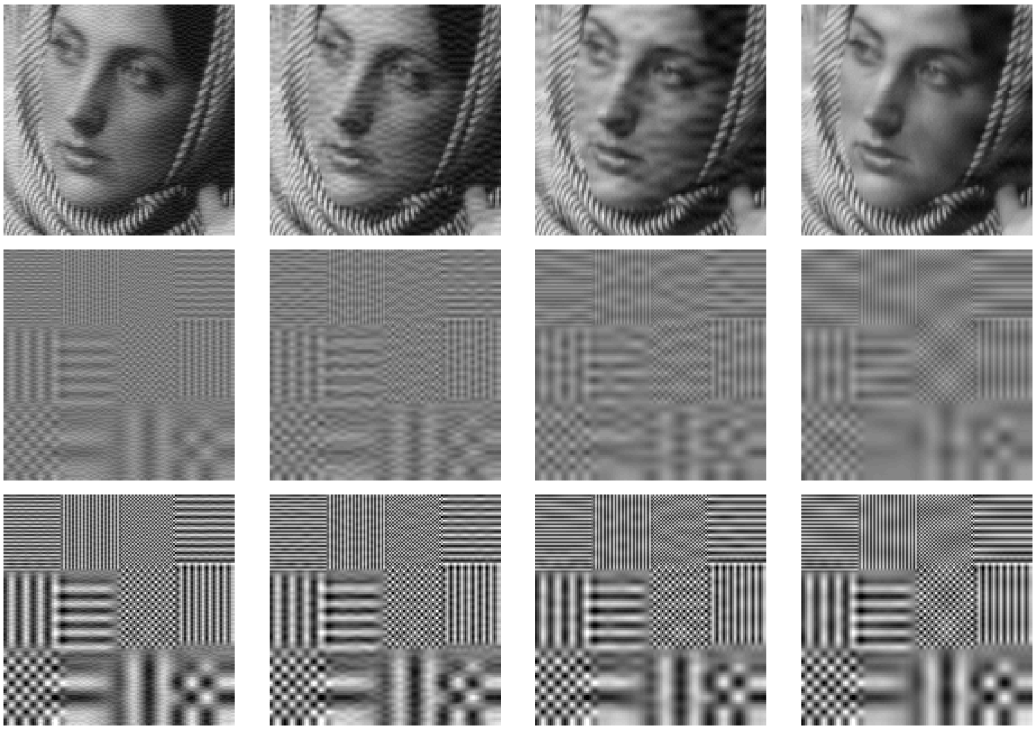


Figura 3.9: Imágenes ruidosas con los mismos valores de RMSE. Notese como la distorsión perceptual es distinta en todas las imágenes.

to por contraste y *enmascaramiento por coincidencia espacio-frecuencial de señal y ruido*. Por un lado, se observa que el ruido es menos perceptible cuando se aplica sobre señales de alto contraste (fila 3 frente a fila 2 en la figura 3.9). Este efecto se puede explicar mediante un modelo no-lineal sin interacción de las respuestas de los sensores (3.6). Por otro lado, la visibilidad se atenúa aun más en las regiones espaciales en las que la composición frecuencial del ruido coincide (o es próxima) a la de la imagen. Por ejemplo, en las imágenes sintéticas, resulta evidente que el ruido de alta frecuencia es mas perceptible en las regiones de baja frecuencia y viceversa. Este efecto de interacción o ausencia de interacción entre componentes frecuenciales es lo que motivó el refinamiento de los modelos escalares.

La medida de distorsión perceptual debería poder explicar estos efectos. Para ello, veamos cómo se expresa matemáticamente la distancia perceptual entre una imagen dada, A_0, y una imagen reconstruida con su correspondiente error, $A_0 + \Delta A$ [14]. La distancia perceptual entre dos imágenes en cualquier dominio de representación de los citados anteriormente vendrá dada, en principio, por la expresión estándar de una distorsión no euclidea [13]:

$$
\begin{aligned}
d_r(r, r+\Delta r)^2 &= \Delta r^T \cdot W_r(r) \cdot \Delta r && (3.6)\\
d_c(a, a+\Delta a)^2 &= \Delta a^T \cdot W_a(a) \cdot \Delta a && (3.7)\\
d_A(A, A+\Delta A)^2 &= \Delta A^T \cdot W_A(A) \cdot \Delta A && (3.8)
\end{aligned}
$$

donde las W representa la matriz de la métrica perceptual, en cada uno de los dominios. En W los elementos de la diagonal principal representan la contribución de cada coeficiente a la distorsión perceptual, mientras que el resto de elementos representa la interacción perceptual entre coeficientes vecinos. De este modo, la diagonalidad de W representa cuantitativamente el concepto de dependencia o independencia perceptual de los coeficientes de cada una de las representaciones.

Si esta métrica, W, fuera la identidad los errores cometidos en el dominio transformado serían iguales a los del dominio espacial (el MSE o el SNR). Sin embargo, tal y como se estudió en la sección 3.1, la respuesta final del sistema depende fuertemente de la frecuencia, por lo que la distancia perceptual entre dos imágenes no tiene por qué ser proporcional al error cometido en el dominio espacial. La expresión de W en los dominios espacial y transformado se puede calcular, suponiendo que en el dominio de la respuesta W_r es la identidad [43, 24] y que la distancia percepual es invariante bajo un cambio de representación. En concreto W_A sería:

$$\Delta A^T \cdot W_A(A) \cdot \Delta A = \Delta r^T \cdot I \cdot \Delta r \tag{3.9}$$

y, considerando que si la distorsión introducida es suficientemente pequeña , podemos aproximar el cambio de la respuesta por la pendiente local de la propia respuesta

$$r + \Delta r = R(T \cdot (A + \Delta A)) = R(A) + \nabla R(A) \cdot T \cdot \Delta A \tag{3.10}$$

con lo que finalmente obtenemos:

$$W_A(A) = T^T \cdot \nabla R(A)^T \cdot \nabla R(A) \cdot T \tag{3.11}$$

quedando la distancia perceptual:

$$d_A(A, A+\Delta A)^2 = \Delta A^T \cdot T^T \cdot \nabla R(A) \cdot \nabla R(A)^T \cdot \Delta A \tag{3.12}$$

Si ahora consideramos el modelo lineal que mostramos en la ecuación (3.3), tenemos que $\nabla R_{ii} = F_{ii}$ y $\nabla R_{ij} = 0$, para $i \neq j$, con lo que W_A queda:

$$W_A(A) = T^T \cdot F^2 \cdot T \tag{3.13}$$

donde en nuestro caso la función F representa la *CSF*. Con esto vemos que, asumiendo un modelo lineal, la métrica en el dominio transformado resulta diagonal. Así pues, la distancia perceptual quedará:

$$d_A(A, A + \Delta A)^2 = \int CSF_f^2 \cdot \Delta a_f^2 \cdot df \qquad (3.14)$$

Esta métrica basada en un modelo lineal es la que se propuso empíricamente en [30], y será la que usemos para medir las distorsiones perceptuales de las imágenes comprimidas. Naturalmente, este modelo tan sencillo tan solo reproducirá el efecto de la dependencia frecuencial de la percepción de distorsiones. No obstante, el uso de modelos más sofisticados tanto para evaluar los resultados como para entrenar las SVMs, se plantea como un desarrollo futuro para mejorar los resultados presentados en este proyecto.

3.3. Redundancia estadística y perceptual

Para que un algoritmo de codificación de imágenes sea efectivo debe conseguir regenerar las imágenes en un dominio en el que tanto la redundancia estadística como la perceptual sean mínimas, porque de este modo se consiguen unos mayores ratios de compresión y una menor distorsión de la imagen [15].

3.3.1. Eliminación de la redundancia estadística

Este tipo de redundancia hace referencia a la correlación existente entre los símbolos que se escogen para representar la imagen. En imágenes naturales la dependencia estadística es muy elevada, es decir, un píxel tiene una probabilidad muy elevada de tener un valor similar a los píxeles adyacentes. En este caso la cuantización escalar es ineficiente (ver figura 3.10 (a) y (b)).

La supresión de esta dependencia se consigue mediante la transformación de la imagen a un nuevo dominio donde los coeficientes no están correlacionados (o lo están muy poco). Si se consigue la independencia estadística de los coeficientes se justifica el uso de cuantización escalar. El efecto de la transformación es reorientar los ejes de nuestra representación según unas determinadas funciones base (diferentes para cada transformada), de modo que se elimina la dependencia entre los coeficientes y se consigue una cuantización más eficaz y por tanto unos *ratio* de compresión más elevados. Un claro ejemplo de esta reorientación lo podemos ver en la figura 3.11

En la extensa bibliografía existente sobre este asunto se proponen multitud de transformaciones que en mayor o menor medida consiguen eliminar la dependencia estadística del dominio espacial [15] . Históricamente se abordó este problema mediante el Análisis de las Componenetes Principales (PCA), que consiste en la diagonalización de la matriz de covarianza de la señal. Sin embargo se demuestra que

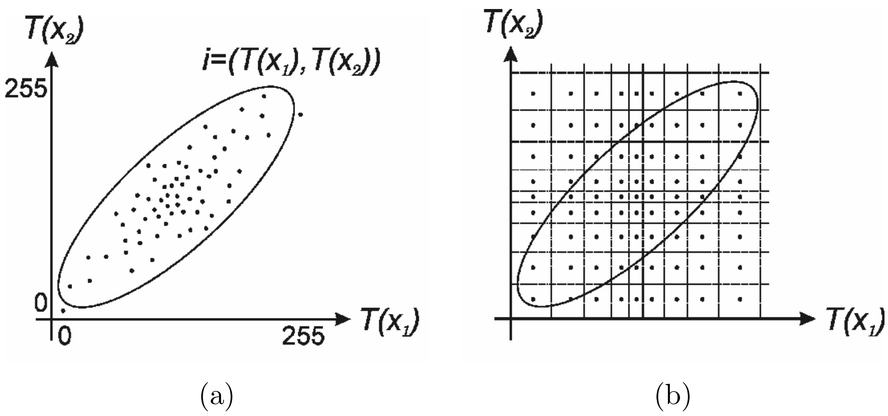


Figura 3.10: (a) Ejemplo de correlación entre los valores de píxeles adyacentes. En el eje vertical se representa el valor de luminancia de un determinado píxel y en el horizontal el de su vecino. Cada punto del plano representa una pareja de píxeles adyacentes. En (b) vemos cómo el esquema de cuantización escalar es muy ineficiente en el dominio espacial.

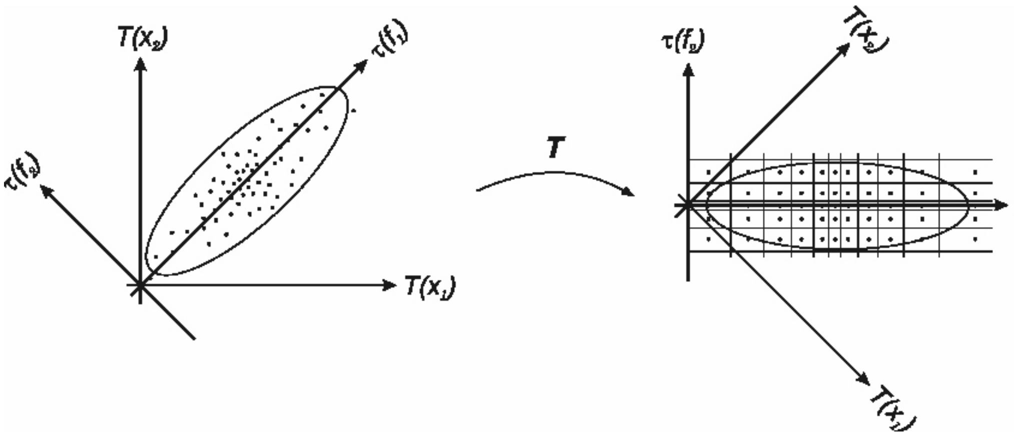


Figura 3.11: Cuantización escalar sobre el dominio transformado. En esta ocasión vemos como el esquema de cuantización escalar resulta mucho más eficiente.

asumiendo modelos AR sencillos estacionarios, la PCA tiende a la Transformada Discreta del Coseno (DCT) [9], que no es más que una extensión de la Transformada de Fourier (FT). Mientras que en la FT las funciones base son combinaciones lineales de senos y cosenos, en la DCT las funciones base se forman sólo con cosenos. Ambas transformaciones asumen funciones base totalmente localizadas en frecuencia y por ello totalmente deslocalizadas en el espacio.

Sin embargo, tal y como hemos descrito anteriormente, el proceso perceptivo humano no usa este tipo de funciones base, sino otras que están tanto localizadas en el dominio frecuencial como en el espacial. Una transformada que se aproxima mejor a este comportamiento es la transformada wavelet [49, 41].

En el presente proyecto, siguiendo una aproximación similar a [50, 48], nos centraremos en el estudio de la block-DCT, que sí presenta una cierta lozalización espacial aunque no recubre el dominio frecuencial como se representa en la figura 3.4. No obstante supone un modelo espacio-frecuencial sencillo, y sobre el cual podremos comprobar el correcto funcionamiento de nuestro algoritmo.

La teoría estandar de la codificación de la transformada requiere que la medida de distorsión global sea una suma de las distorsiones en cada coeficiente, es decir, requiere que la métrica sea diagonal, o lo que es lo mismo, debemos representar la señal en un dominio donde no haya redundancia perceptual.

3.3.2. Eliminación de la redundancia perceptual

Por redundancia perceptual entendemos el hecho de que la percepción de una muestra esté influida por el valor que tenga una muestra vecina. Si esto ocurre en el dominio a cuantizar llegaremos a que cometer un error en la cuantización de un determinado coeficiente nos supondrá un efecto perceptual en los coeficientes cercanos a él, con lo cual aumentará la distorsión de la imagen. Por ello, hemos de buscar un dominio donde cada coeficiente sea perceptualmente independiente de sus vecinos. Esto se conseguirá representando la señal en el dominio de la respuesta $\mathbf{r}$, donde la métrica de distorsión perceptual de dicho dominio W_{ij} es diagonal (vease seccion 3.2).

Una manera de igualar la relevancia perceptual de todos los coeficientes del dominio a cuantizar es aplicar a cada uno su respectiva ganancia F_i dada por la función de sensibilidad al constraste que vimos en la figura 3.5. De forma aproximada la aplicación conjunta de la DCT y de la CSF consigue eliminar tanto la redundancia perceptual como la estadística, aunque el uso de modelos más avanzados revelan que existe interacción entre los coeficientes de la DCT, porque la métrica $W_{i,j}$ no es diagonal.

En cualquier caso, en el presente proyecto usaremos este modelo porque el grado de no diagonalidad de la métrica es muy pequeño, lo cual nos permitirá obtener resultados satisfactorios. El uso de modelos más completos se dejará como trabajo futuro.

3.4. Transformación: Transformada discreta del coseno (DCT)

La transformada discreta del coseno es una variante de la Transformada Discreta de Fourier (DFT) en la que se usa como núcleo de la tranformación tan solo funciones coseno, en lugar de senos y cosenos. Para el procesamiento de imágenes, se necesitan transformaciones bidimensionales. En este caso, por cada frecuencia horizontal se lleva a cabo una búsqueda de todas las frecuencias verticales posibles. Por otro lado la DCT es separable, lo que significa que es posible obtener la DCT bidimensional mediante su cálculo en cada dimensión de manera separada. La expresión matemática que permite su cálculo es la siguiente:

$$T[i,j] = c(i,j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} A[x,y] \cdot cos\frac{(2y+1)i\pi}{2N} \cdot cos\frac{(2x+1)j\pi}{2N} \tag{3.15}$$

donde:

$$\begin{aligned} c(i,j) &= \frac{2}{N}, & i,j \neq 0 \\ c(i,j) &= \frac{1}{N}, & i,j = 0 \end{aligned}$$

Para aplicar la DCT, se divide la imagen en bloques cuadrados de $N \times N$ píxeles, siendo los valores más usuales N=8 ó N=16. La DCT es una transformación a frecuencias totalmente localizadas, por lo que supone que las imágenes que trata son estacionarias. Esta aproximación es correcta para bloques de pocos píxeles, pero deja de serlo para tamaños mayores de bloque.

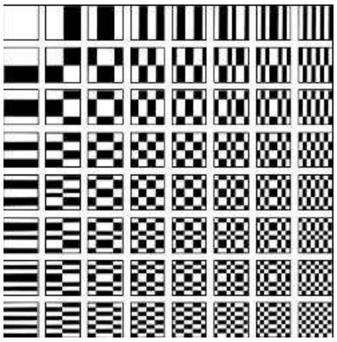


Figura 3.12: Ejemplo de funciones base de la DCT para bloques de 8×8 píxeles

Las funciones base de la DCT son combinaciones lineales de cosenos que se reparten para todo el rango frecuencial. Podemos ver un ejemplo de las funciones base para un bloque de 8×8 píxeles en la figura 3.12. Como vemos en la figura 3.12 el término de la esquina superior izquierda corresponde al termino de continua del bloque, es decir su nivel medio de luminancia. Conforme nos alejamos de dicho punto encontramos las componentes de frecuencias más elevadas. En general, cuanto más distanciado de la esquina superior izquierda se encuentre el coeficiente, menor será su magnitud por término medio, lo que muestra una característica muy importante de la DCT, que es la alta concentración de la energía en bajas frecuencias. La ganancia de codificación (reducción del número de bits necesarios) se obtiene aprovechando los coeficientes cero y de bajo valor para reducir los datos necesarios.

La DCT es la transformada que más se ha usado hasta el momento puesto que presenta una seríe de características que la hacen muy útil desde el punto de vista práctico. El hecho de que en la DCT se requieran solamente cálculos reales, facilita su aplicación en los circuitos prácticos. Como resultado, la DCT está disponible ampliamente en circuitos VLSI (circuitos integrados a muy alta escala) de propósito especial, que los hace atractivos para el uso en tiempo real.

3.5. Respuesta: Función de Sensibilidad al Constrate (CSF)

Tal y como ya hemos comentado en varias ocasiones, el cerebro no responde igual a toda la radiación incidente. En concreto, la percepción presenta una gran dependencia de la frecuencia, de la extensión espacial, y de la orientación angular de un determinado estímulo.

Una de las características principales de la percepción visual humana es el contraste, que se define según:

$$C = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{3.16}$$

donde L_{max} y L_{min} representan las luminancias máxima y mínima de la escena que se observe. De este modo, la sensibilidad al contraste es una medida de la capacidad de nuestro sistema visual para discernir patrones. Un ejemplo clásico de la sensibilidad al constraste es el test de Campbell-Robson [2] que se puede ver en la figura 3.13.

En dicho test se representa un estímulo sinusoidal que aumenta en frecuencia espacial logarítmicamente hacia la derecha, y disminuye en contraste, también logarítmicamente, hacia arriba. De la observación de la imagen se extraen principalemte dos conclusiones:

1. Las barras del entramado sinusoidal tienden a desaparecer conforme nos desplazamos en sentido ascendente por la imagen, es decir conforme disminuimos

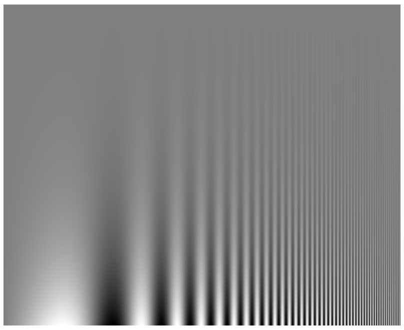


Figura 3.13: Test de Campbell-Robson para la sensibilidad al contraste.

el constraste. En concreto el punto en el que parecen desaparecer se conoce como constraste umbral.

2. En ambos extremos del eje de frecuencias las barras desaparecen más abajo que en las frecuencias intermedias, donde permanecen visibles hasta cotas superiores en el eje de contrastes. Es decir, el contraste umbral en las frecuencias medias es más pequeño que en las frecuencias altas y bajas. Es decir obtenemos que nuestro sistema visual implementa un filtro pasabanda, que es lo que se conoce como función de sensibilidad al contraste.

La CSF corresponde al caso en que se supone que la respuesta del sistema visual es lineal con los coeficientes del dominio transformado, tal y como vimos en la ecuación (3.3).

En este capítulo hemos llevado a cabo una presentación del funcionamiento del sistema visual humano. En el siguiente capítulo vamos a mostrar como se puede aprovechar lo aprendido aquí para la compresión de imágenes mediante SVM.

Capítulo 4

Compresión de imágenes mediante SVM adaptativa perceptual

EN este capítulo revisaremos trabajos previos de SVM en compresión de imágenes y expondremos nuestra aportación, que básicamente se puede definir como usar una SVM sensitiva perceptual, frente a la SVM insensitiva y euclidea [37].

4.1. Trabajos previos

Las primeras referencias al uso de las SVM para compresión de imágenes se encuentran en [36]. El procedimiento que ahí se presentaba consistía en atacar directamente la compresión sobre el dominio espacial, es decir, las SVR eran entrenadas con bloques de 8×8 píxeles de la imágenes. Tal y como se explicó cuando presentamos los modelos perceptuales, está aproximación no resulta correcta, puesto que en el dominio espacial los datos presentan una alta dependencia estadística, siendo así difícil alcanzar ratios de compresión elevados.

Los mismos autores abordan el problema desde otro punto de vista en [37]. En esta ocasión, ya usan la SVM sobre el dominio frecuencial, mediante el uso de la DCT. El algoritmo que proponen a grandes rasgos se basa en usar una SVR con una zona de insensibilidad ε constante para todos los coeficientes de la DCT. Esto tiene sentido en el dominio espacial ya que, a priori, todos los píxeles alrededor de uno en concreto tienen la misma importancia para su modelado pero este hecho no es extrapolable al dominio frecuencial ya que:

- Las frecuencias altas de una imagen se asocian normalmente a detalles de ésta y/o varianciones bruscas de intensidad.

- Los modelos perceptuales nos muestran que dado que nuestro sistema visual se comporta como filtro pasa-banda y por tanto lo correcto es tratar de distinto modo los coeficientes del dominio transformado.

Además, con esta aproximación, no se explotan las características de las imágenes naturales, la teoría bien fundada de la percepción humana, o la adaptatividad de los modelos de regresión. Es por todo ello, que este PFC realiza una mejora al trabajo [37], en el que se profundiza en el modelado adaptativo de los coeficientes de la DCT en base a perfiles que penalicen y permitan mayor o menor error en función de aproximaciones a la sensibilidad al contraste tomadas de los modelos perceptuales del sistema visual humano.

4.1.1. El algoritmo RKi

El algoritmo RKi, que se propone en [37], comprende los pasos que se detallan en la figura 4.1.

1. *Normalización de la imagen entre 0 y 1.*
2. *División de la imagen en bloques.*
3. *Mapeado zig-zag con el bloque actual.*
4. *Almacenamiento por separado del primer coeficiente.*
5. *Entrenamiento de la SVM con ε constante.*
6. *Selección sólo de los primeros 8-16 pesos obtenidos de la SVM para su almacenamiento.*
7. *Cuantización de los pesos.*
8. *Uso de RLE y codificación Huffman para generar el archivo binario.*
9. *Reconstrucción de la imagen.*
10. *Cálculo de los errores cometidos.*

Figura 4.1: Pseudocódigo del funcionamiento del algortimo RKi.

4.1.2. Deficiencias

Como hemos comentado en la sección anterior, la SVM que se usa en el algoritmo usa un tubo de error ε uniforme en todo el dominio frecuencial. Es decir, permite el mismo error en frecuencias altas, medias y bajas. Como ya vimos esto no es correcto puesto que nuestro sistema visual responde de distinto modo ante estímulos con distintas frecuencias espaciales. Por ello, este algoritmo presenta el fallo de no considerar distintas ganancias para cada frecuencia.

Además, eliminan *ad-hoc* los últimos coeficientes de la DCT. Esto lo justifican en base a que el sistema visual humano presenta una respuesta menor a altas frecuencias. Aunque como hemos visto esto es cierto, podría conseguirse el mismo efecto de forma más elegante utilizando un perfil adaptado para ε. La ventaja del uso de un modelo adaptativo es que no siempre se eliminarán esos coeficientes, de modo que en situaciones de contraste elevado de dichos coeficientes obtendremos una ganancia en la calidad de la imagen al haberlos conservado.

En resumen, el algoritmo RKi presenta la grave falta de no tener en cuenta las propiedades más basicas del sistema visual humano, por lo cual en las imágenes que reconstruye se observan distorsiones elevadas.

4.2. SVM adaptativa y perceptual

Con el fin de superar las deficiencias que presenta el algoritmo RKi vamos a desarrollar una nueva aplicación en la cual usaremos las herramientas que hemos explicado ampliamente en los capítulos anteriores, es decir la SVM adaptativa sin bias y los modelos perceptuales para fijar de forma adaptativa la cota de error en el dominio de la DCT.

En la siguiente tabla podemos ver un comparativa entre RKi y el modelo que se propone en este proyecto.

	RKi	**SVM Adaptativa Perceptual**
Implementación SVM	SMO	IRWLS
Zona de Insensibilidad ε	Constante	Adaptativo según CSF
Eliminación de coeficientes	Sí (Ad-Hoc)	No

Cuadro 4.1: Tabla comparativa entre el algortimo RKi y el modelo que se propone en este PFC.

Toda la revisión que se ha realizado tanto de las SVM como de los modelos perceptuales estaba orientado a entender las ventajas de usar estos sistemas de aprendizaje en la compresión de imágenes. Por ello, para mayor claridad, vamos a llevar a cabo un pequeño ejemplo de cómo será usada una SVR para la compresión de imágenes.

Nuestro objetivo final será modelar los coeficientes de la DCT, con el menor número posible de parámetros y exigiendo que la imagen reconstruida sea de una calidad aceptable. En la figura 4.2 se puede ver la forma típica de un bloque de coeficientes de la DCT.

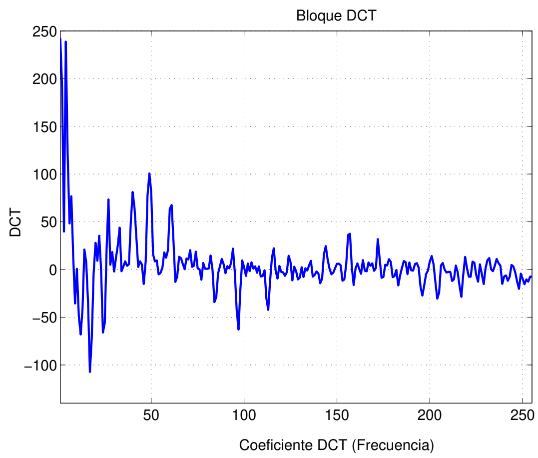
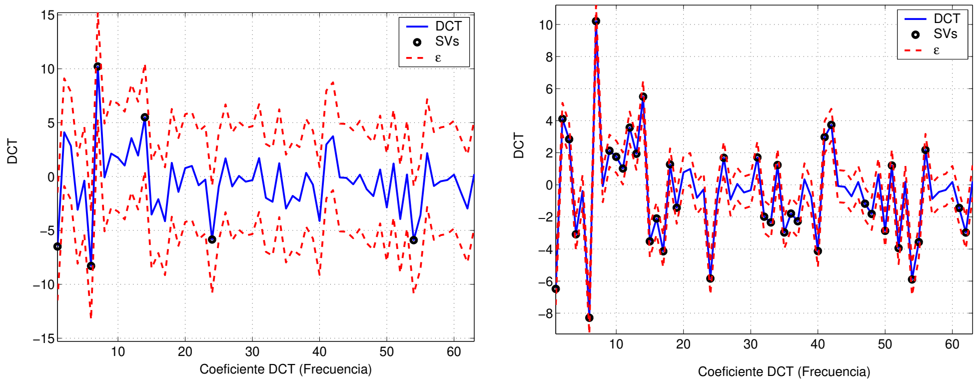


Figura 4.2: Ejemplo de un bloque de coeficentes de la DCT de 16×16 píxeles zigzageado (ver sección 4.3.1).

La SVR nos permite un modelado no lineal, gracias a la inclusión de las funciones kernel, teniendo además un comportamiento numérico muy robusto. Por otro lado, mediante la adaptación para cada muestra de la anchura del tubo ε podremos hacer que la máquina seleccione un número alto o bajo de vectores en una determinada zona de interés soporte, lo cual será una medida indirecta de la compresión. A continuación vemos un ejemplo de regresión en la figura 4.3.

Sin embargo, en las gráficas de la figura 4.3 (a) y (b) el tubo ε es constante para todas las frecuencias (eje horizontal). Tal y como se explicó en el capítulo anterior, el sistema visual humano tiene respuestas muy distintas según sea el estímulo presente, por lo cual será más adecuado usar una zona $\varepsilon - insentitive$ adaptada a la curva de respuesta de nuestro sistema perceptual. Por ello pesaremos nuestro factor ε por la función de sensibilidad al constraste (CSF) que fue presentada en el capítulo anterior (Fig. 3.5). El resultado se puede ver en la figura 4.4. El efecto de la CSF en la SVR es que elige los vectores soporte en la zona de baja frecuencia.

Por otro lado, si observamos la figura 4.2, vemos que la zona de altas frecuencias se alcanza un régimen estacionario a valor nulo. Esto constituye una de las principales características de la DCT. De este modo, podemos aplicar lo que vimos en 2.4.2 y trabajar con bias implícito ($b = 0$) lo cual además constituirá una ventaja computacional puesto que en la solución sólo hay que resolver una ecuación.



(a) (b)

Figura 4.3: En (a) vemos un ejemplo de regresión con pocos SVs (bloque de 8×8 píxeles). Como hemos usado un tubo de $\varepsilon = 5$ requerimos pocos vectores soporte, seis en concreto. En (b) tenemos un ejemplo de regresión con muchos SVs (bloque de 8×8 píxeles). En esta ocasión tenemos un tubo más estrecho ($\varepsilon = 1$) por lo que el algoritmo ha requerido más vectores soporte para una correcta modelización, en este caso en concreto se seleccionan 40 vectores soporte.

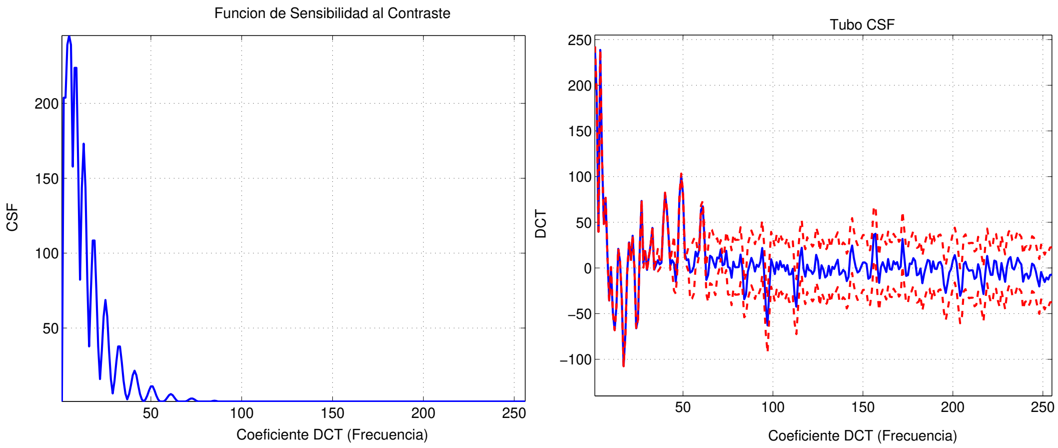


Figura 4.4: En (a) vemos un ejemplo de CSF 2-D que mediante un mapeado en zig-zag (ver sección 4.3.1) ha sido tranformada a 1-D y en (b) como se adapta el tubo ε según la misma función.

4.3. SVM adaptativa perceptual en compresión de imágenes

Llegados a este punto disponemos de todas las herramientas necesarias para el desarrollo de un algoritmo de codificación de imágenes mediante SVM adaptativa. Por ello, a modo de ejemplo se va a detallar su funcionamiento para dos bloques de la DCT; uno correspondiente a altas frecuencias y otro a bajas. También aprovecharemos esta sección para hacer algunas consideraciones sobre los valores de los parámetros libres, los perfiles usados o el tipo de función kernel.

4.3.1. Programación

Vamos a ver los diferentes pasos que se llevan acabo para la codificación de la imagen.

1. **División de la imagen en bloques de** 16×16 **píxeles.** Este paso es necesario dado que la eficacia computacional de la DCT disminuye con el tamaño de la imagen. Por ello, es necesario hacer la transformada sobre pequeñas agrupaciones de píxeles. Es por esta razón por la cual JPEG trabaja con bloques de 8×8 píxeles, sin embargo esto hace que la resolución espectral de la DCT sea baja apareciendo el fenómeno de *spectral leakage*. Como nuestra aplicación pesa los coeficientes de la DCT con la CSF será necesaría que dicha resolución espectral sea mayor, viéndonos así obligados a trabajar con bloques de 16×16 píxeles, aunque esto suponga una pequeña penalización desde el punto de vista computacional. En la figura 4.5 vemos un ejemplo de la actuación de la DCT sobre un bloque de 4×4 píxeles.

242	227	209	235
59	194	113	188
155	116	157	45
124	5	202	103

DCT-2D

594	-16	-18	47
167	-1	9	-133
80	-14	49	78
38	94	-1	55

Figura 4.5: Ejemplo de una DCT sobre una matriz de 4×4 píxeles.

2. **Mapeado zig-zag.** Se aplica a la matriz de los coeficientes de la DCT la transformación que se muestra en la figura 4.6. Así conseguimos que la matriz de píxeles se convierta en una serie de datos unidimensional, con lo que la SVM a entrenar también será unidimensional.

3. **Almacenar por separado el primer coeficiente de la DCT.** Tal y como se explicó cuando se presentó la DCT, sección 3.4, el primer coeficiente es el nivel medio de intensidad de cada bloque, y por tanto un error cometido en su almacenamiento o modelización repercute ostensiblemente en la calidad final de la imagen. En realidad lo que se hace es almacenar tan solo el primer

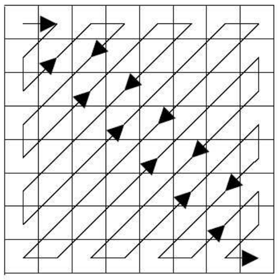


Figura 4.6: Ejemplo del mapeado en zig-zag.

coeficiente del primer bloque de la imagen, y para el resto se guarda la diferencia con respecto al anterior, puesto que según vimos en la introducción así se consigue una disminución de la entropía de la serie, permitiendo de este modo un aumento de la compresión.

4. **Entrenar la SVM adaptativa con el resto de coeficientes.** Dados unos valores iniciales de ε y C se lleva acabo el entrenamiento de la SVM con una ε_i para cada muestra, donde $\varepsilon_i \propto \frac{1}{CSF_i}$. Su resultado, los w_i, se pasarán a la etapa de cuantización.

5. **Cuantización de los pesos w_i.** Se mapea un conjunto infinito de valores a uno discreto según el número de niveles de cuantización que escojamos.

6. **Medición de la entropía.** Una vez calculada, se copmpará con el valor fijado como deseado por el usuario. Dado un umbral de tolerancia, si la entropía obtenida no está dentro de dicho umbral, se reajusta la constante ε y se vuelve a entrenar la SVM. Este proceso se repite hasta que se obtiene la compresión deseada. La entropía total es la suma de las entropías asociadas a los pesos w_i, a sus signos y a los coeficientes iniciales de cada bloque de la DCT.

Una descripción a nivel de flujo de programa se puede ver en la figura 4.7.

Esta es una descripción global del proceso sobre la que hay que destacar dos puntualizaciones:

- Realmente la SVM no se entrena directamente con los valores de los coeficientes de la DCT, sino que se hace con los valores absolutos de ellos, por lo que además tenemos que guardar los signos de cada uno. La razón de hacer esto es que así se disminuye el rango dinámico de los pesos w_i, lo cual permitirá en la cuantización tener una mayor resolución, que repercutirá en una mejora en la calidad de la imagen.

1. *Fijar $k = cte$, $Tol = cte.$ y $\varepsilon = \varepsilon_0$.*
2. *División de la imagen en bloques de 16×16 píxeles.*
3. *Mapeado zig-zag con el bloque actual.*
4. *Almacenamiento por separado del primer coeficiente.*
5. *Entrenamiento de la SVM usando ε adaptada perceptualmente para una determinada entropía deseada programada por el usuario (H_d).*
6. *Cuantización de los pesos obtenidos del entrenamiento para número de niveles de cuantización fijados por el usario.*
7. *Cálculo de la entropía de los resultados del entrenamiento obtenidos (H_o).*
   - *Si $|H_o - H_d| \leq Tol$ salta a 9.*
   - *Si $|H_o - H_d| > Tol$ sigue.*
8. *Calcula la nueva ε según:*

$$\varepsilon_{n+1} = \varepsilon_n + (H_o - H_d) \cdot k.$$

   y vuelve al punto 5.

9. *Reconstrucción de la imagen.*
10. *Calculo de los errores cometidos.*

Figura 4.7: Pseudocódigo del funcionamiento de la SVM adaptativa perceptual.

- Se probará el algoritmo con distintas cantidades de niveles de cuantización. De este modo la disminución de entropía de la señal vendrá dada por una colocación más adecuada de los vectores soporte. En concreto el tubo sensitivo será capaz de, con una distribución más adecuada de los vectores soporte, obtener imágenes de mayor calidad que el tubo fijo, para una misma tasa de compresión.

4.3.2. Parámetros libres

Con la SVM que estamos usando tenemos tres parámetros libres que tenemos que ajustar según nuestro conocimiento del problema; la penalización C, la constante de insensibilidad ε, y el tipo de kernel a utilizar.

Penalización C

Según se explicó en el segundo capítulo la C actúa como límite al valor de los pesos w_i, por lo que es un parámetro que regula la capacidad de generalización o de sobreajuste de la máquina. Como en nuestro problema lo que buscamos es una modelización los más ajustada posible, usaremos valores muy altos para este parámetro, en concreto del orden de $C = 1000$. En cualquier caso este valor irá pesado por la CSF, de igual modo que sucederá con ε.

Constante ε

Dado que vamos a usar una SVM adaptativa la zona de insensibilidad será variable para cada muestra de entrenamiento según nos marque la CSF. Sin embargo, tenemos la libertad de elegir una constante de escala que deberemos ajustar según los valores típicos de los coeficientes de la DCT. Lo que se hace en este proyecto es permitir que el propio algoritmo seleccione el valor de ε para el valor de entropía final deseada por el usuario.

Función kernel

En este proyecto solo hemos trabajado con kernels RBF. Existen muchas razones para escoger una función kernel RBF, como sus pocas disficultades numéricas y que sólo haya que fijar un parámetro, la anchura de la Gaussiana σ. Así pues, el kernel RBF convierte la SVM en una red neuronal RBF regularizada con la ventaja añadida de que el centro de las Guassianas se establece automáticamente. Por otro lado, los kernel sigmoidales se comportan como los RBF para ciertos parámetros [20, 22] pero por contra no son kernels definidos positivos para todos los casos, lo cual limita su aplicación práctica [38]. Además desde el punto de vista de la aplicación emplear un kernel RBF implica que el modelado de un coeficiente de la DCT tendrá influencia de sus vecinos de forma local.

Estos tres parámetros hacen referencia exclusivamente al entrenamiento de la SVM. El algoritmo global tiene un parámetro libre más, que es el número de niveles cuantización.

Niveles de cuantización

Los niveles de cuantizción son los números a los que redondearemos los pesos que nos devuelva la SVM. Cuanto mayor número de niveles pongamos, menor será la distosión de dichos pesos pero mayor será la entropía de los resultados. Por el contrario, si ponemos muy pocos niveles la entropía será baja pero los pesos pueden quedar afectados por un error de cuantización muy elevado. En la práctica exigiremos un compromiso de ambos factores. Para ello probaremos nuestro algoritmo con distinto número de niveles y escogeremos empíricamente la opción óptima.

4.3.3. Ejemplo de aplicación

Vamos a ver qué es lo que se obtiene cuando se realiza todo el proceso descrito en el apartado 4.3.1 sobre un bloque de baja frecuencia y otro de alta frencuencia. Haremos esta serie de pruebas comparando también los resultados obtenidos mediante la SVM insensitiva, el algoritmo RKi y la SVM adaptativa perceptual.

Baja frecuencia

Escogemos como bloque de 16×16 píxeles de entrenamiento la esquina inferior izquierda de Barbara. Una ampliación de esa zona se observa en la figura 4.8.

Figura 4.8: Bloque de entrenamiento para bajas frecuencias.

Sobre esta sección aplicamos la DCT, almacenamos su primer coeficiente, y trabajamos con los valores absolutos. Por lo que la función que tendremos que aproximar será la mostrada en la figura 4.9 (a).

A continuación se entrenan las distintas SVMs, lo cual nos da como resultados la posición de los vectores soporte y el valor de los pesos w_i en cada una de las configuraciones en empleadas. Los resultados se pueden observar en la figura 4.9 (b) para SVM con perfil uniforme, 4.9 (c) para RKi y 4.9 (d) para SVM adaptada perceptualmente.

Como vemos al ser un bloque de baja frecuencia, los vectores soporte que hemos obtenido en los tres algoritmos se encuentran en las bajas frecuencias, no existiendo mucha diferencia entre los resultados obtenidos por los distintos métodos.

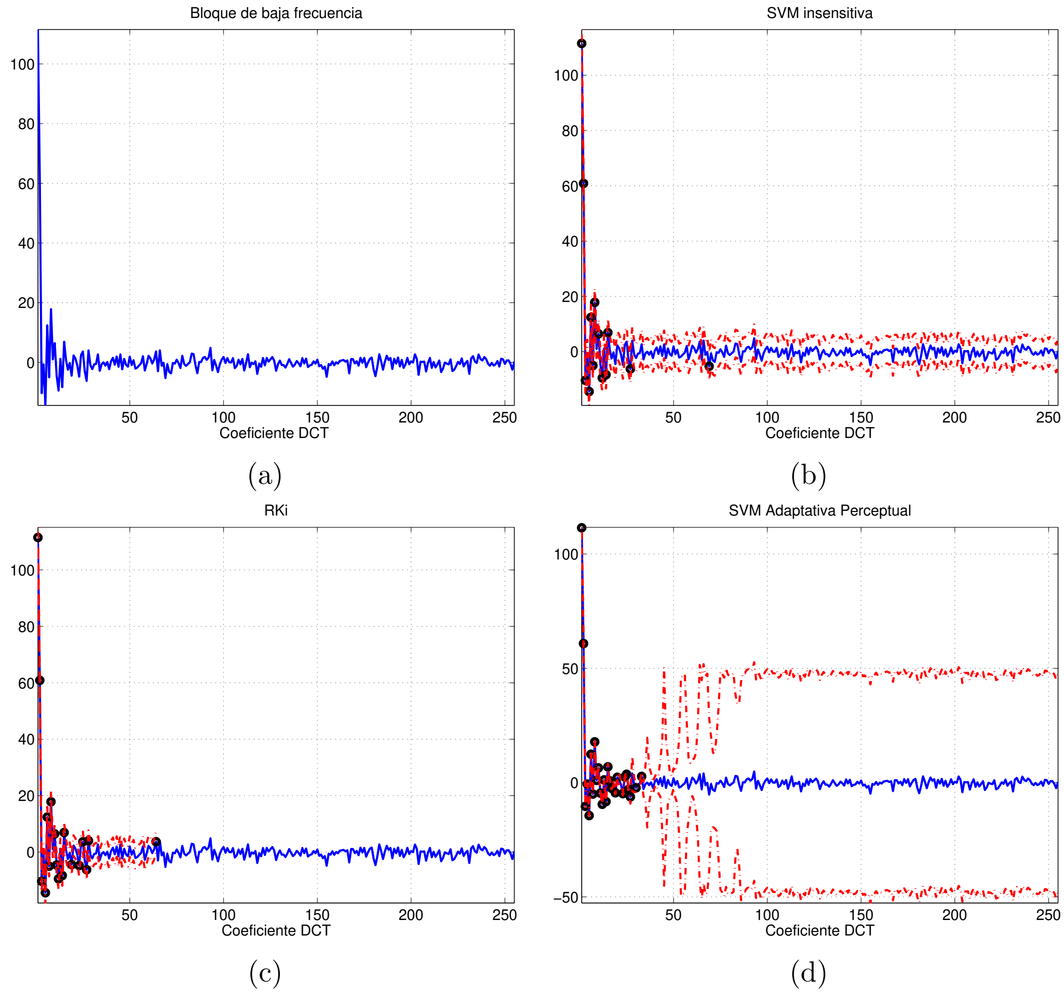


Figura 4.9: Bloque de entrenamiento en el espacio de la DCT y Vectores soporte para un bloque típico de bajas frecuencias. En (a) observamos la señal representada en el dominio de la DCT una vez llevada a cabo la transformación zig-zag. En (b), (c) y (d) vemos los resultados del entrenamiento según la SVM insensitiva, Rki y SVM adaptativa perceptual respectivamente.

Alta frecuencia

Escogemos como bloque de 16×16 píxeles de entrenamiento una zona central de Barbara, que corresponde a una de las zonas en las que está el pañuelo (zona rayada), es decir con componentes apreciables de alta frecuencia. Una ampliación de esa zona se observa en 4.10.

Aplicando los mismos pasos que en la sección anterior, obtenemos la repre-

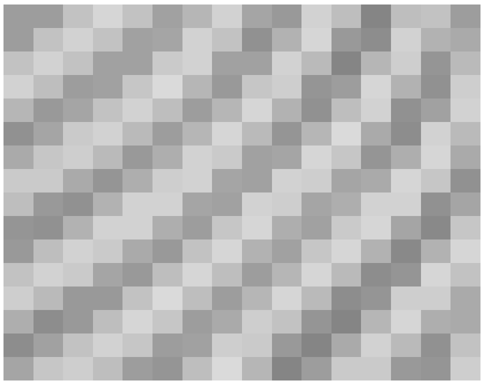
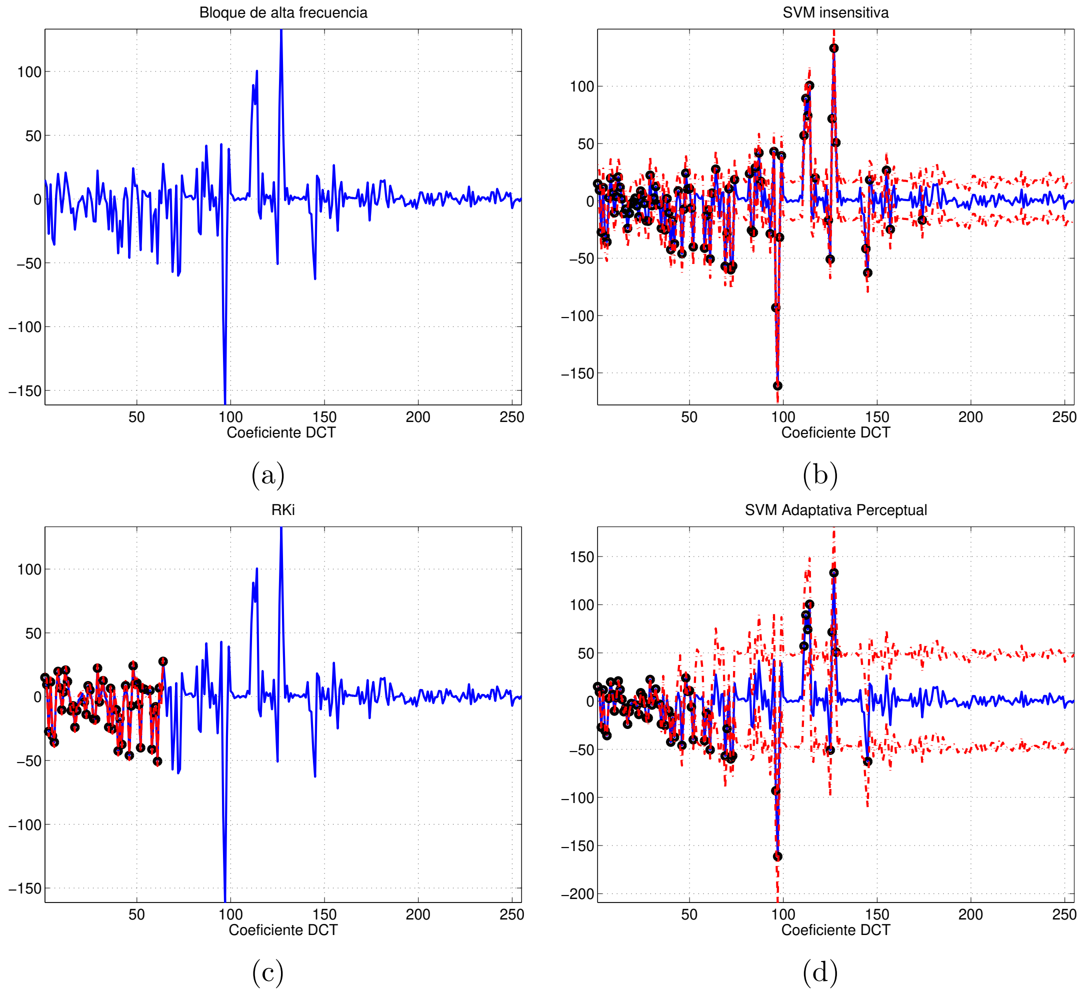


Figura 4.10: Bloque de entrenamiento para bajas frecuencias.

sentación de la DCT en la figura 4.11 (a). A continuación se entrenan la SVM lo que nos da como resultados la posición de los vectores soporte (ver figura 4.11 (b), (c) y (d)) y el valor de los pesos w_i para cada algoritmo. En esta ocasión observamos cómo se colocan algunos vectores soporte en la zona de frecuencias elevadas.

En este bloque con alto contenido en frecuencias espaciales elevadas, observamos como los algoritmo basados en SVM insentiva y adaptativa han conservado dichas componentes espectrales, mientras que RKi las ha eliminado. Dado que la energía de estas frecuencias en este bloque es elevada, la supresión de los coeficientes devendrá en una distorsión apreciables de la imagen.

Hasta aquí hemos presentado nuestro algoritmo, sus principales novedades y puntos clave. En el capítulo siguiente aplicaremos lo estudiado hasta ahora a imágenes naturales completas y comprobaremos la eficiencia de nuestra propuesta.



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

Figura 4.11: Bloque de entrenamiento en el espacio de la DCT y Vectores soporte para un bloque típico de altas frecuencias. La principal diferencia con la figura anterior, es que ahora sí tenemos vectores soporte en las frecuencias elevadas. El algoritmo RKi al cortar en al coeficiente número 64, no ha considerado estas componentes frecuenciales.

Capítulo 5

Resultados

UNA vez llevada a cabo la explicación detallada de las herramientas con las que vamos a trabajar se va a describir la metodología de trabajo seguida y los resultados que se han obtenido. Para ello nos apoyaremos en la curvas *Rate-Distortion* y en las imágenes comprimidas obtenidas.

5.1. Imágenes empleadas

Existe una serie de imágenes estándar con las cuales se trabaja en el campo de la codificación y compresión de imágenes. Cada una de estas imágenes tiene una descomposición espectral distinta, es decir en unas las componentes de baja frecuencia son más significativas, mientras que en otras son las componentes de alta frecuencia las más características de la imagen. Típicamente imágenes con alto contenido en baja frecuencia son aquellas con áreas de color muy suaves y uniformes, mientras que las de alta frecuencia son zonas con cambios abruptos como puedan ser areas rayadas o punteadas.

Todas las imágenes con las que vamos a trabajar están en escala de grises codificadas con 8 bits por píxel, por lo que cada píxel puede tomar un valor entre 0 y 255. La extensión del algoritmo a imágenes en color es inmediata, puesto que el plano de luminancia, que es con el que trabajamos en escala de grises, es el que más información contiene. Por ello, los planos de color serán más sencillos de codificar.

Cuando se comprueba la fiabilidad de un algoritmo de compresión se tiene que aplicar sobre distintas imágenes para ver su comportamiento sobre todo el espectro frecuencial. En nuestro caso trabajaremos con cuatro imágenes: *Lena, Barbara, Boats y Cameraman*. Estas imágenes están disponibles en un gran número de programas informáticos para el tratamiento de imágenes como MATLAB, VISTA, etc. Para cada una de ellas fijaremos un nivel de compresión y mediremos la señal de error, para obtener así las curvas de Rate-Distortion. De este modo podremos comprobar el funcionamiento de nuestro algoritmo sobre varias imágenes de características muy distintas.

Cameraman representa una imagen sencilla de tratar dado que presenta dos zonas muy uniformes como son el cielo y el traje del fotógrafo. La imagen original será *Cameraman.tiff* con un tamaño de 256×256 píxeles, por lo que trabajaremos con la imagen al completo puesto que no supone un coste computacional elevado.

Por su parte *Lena* representa una imagen de dificultad media dado que presenta zonas de alta frecuencia como pueden ser los ojos o las plumas del sombrero y zonas de baja frecuencia como la piel de la cara. La imagen estándar tiene unas dimensiones de 512×512 píxeles, por lo que escogeremos la parte central de 256×256 píxeles para la comparación de resultados.

Finalmente, para estudiar imágenes más complicadas usaremos *Barbara* y *Boats*. Ambas presentan dos zonas con elevadas componentes de altas frecuencias; *Barbara* en la zona del pañuelo y *Boats* en las zona de los mástiles. De este modo, constituyen un buen test para verificar el comportamiento de nuestro algoritmo. Las dos imágenes tienen un tamaño de 512×512 píxeles por lo que trabajaremos con divisiones de ambas. En el caso de Barbara trabajaremos con la zona de la cara, y en Boats con la esquina superior derecha, la que comprende los mástiles de los barcos.

Para obtener una medida del error cometido a lo largo de todo el proceso de compresión usaremos el *Root Mean square error (RMSE).*

5.2. Medida del error y del volumen

- **RMSE.**

  El RMSE es la raíz cuadrada del error cuadrático medio, que se calcula como la diferencia al cuadrado de cada pixel entre la imagen original y la reconstruida. Para nuestros cálculos llevamos a cabo una media para toda la imagen.

$$RMSE = \sqrt{\frac{1}{N \times M} \left[\sum_{j=1}^{M} \sum_{i=1}^{N} \left(I_{i,j} - \hat{I}_{i,j} \right)^2 \right]} \tag{5.1}$$

  donde $I_{i,j}$ y $\hat{I}_{i,j}$ representan las imágenes original y reconstruida respectivamente, y N y M son las dimensiones en píxeles de la imagen.

En cualquier caso, hay que recordar que estas medidas en el dominio espacial son un tanto irrelevantes, puesto que el efecto perceptual debe medirse en el dominio de la respuesta. Por ello la comparación realmente importante debe ser la subjetiva, es

Cameraman

Lena

Barbara

Boat

Figura 5.1: Imágenes digitales con las que realizaremos la experimentación.

decir la comparación visual directa entre las imágenes que obtengamos por los diversos procedimientos que vamos a emplear. Para poder realizar estas comparaciones mostraremos las imágenes que vamos obteniendo con cada algoritmo.

Sin embargo, como ya dijimos en la sección 3.2, vamos a emplear una métrica perceptual para mayor completitud de los resultados. Las curvas de Rate-Perceptual Distortion que obtengamos mediante está medida deberán corresponderse con la impresión subjetiva que obtengamos de la comparación visual de las imágenes.

- **Distancia de Nill.** Vamos a usar una de las métricas perceptuales más sencillas de las que se han propuesto [30], que es la distancia de Nill. En esta métrica se considera como ganancia de los filtros la CSF y no se incluye ningún efecto no lineal. Una descripción más detallada ya se hizo en la sección 3.2 (ecuación (3.14)).

- **Medida de la compresión.**

  Tal y como se explicó en el primer capítulo, la entropía de una señal hace referencia a la uniformidad de los datos que la componen. Para imágenes digitales, la entropía se mide con el número de bits por píxel (bpp), es decir, el número de bits necesarios por cada píxel de la imagen para almacenarla sin pérdida de información (puesto que la pérdida ha tenido lugar en la etapa de modelización y cuantización), por métodos como Huffman o codificación aritmética.

  Si la imagen está en formato original, es decir cuantizada a ocho bits, tenemos 255 símbolos, y la entropía de la imagen se calcula como:

  $$H = -\sum_{i=0}^{255} p_i log_2 p_i \tag{5.2}$$

  Si aplicamos esta fórmula directamente a una imagen obtenemos valores muy cercanos a 8 bpp es decir, apenas hay compresión. Sin embargo, en cuanto aplicamos la modelización y la cuantización, los valores bajan hasta 0.5 bpp o incluso menores.

  La entropía de Shannon (ecuación (5.2)) es el límite teórico de compresión que se podrá alcanzar mediante los algoritmos de codificación como Huffman. Para pasar de la medida de la entropía en bpp a la razón de compresión *("Compression Ratio", CR)* tendremos que generar un archivo binario con todos los datos necesarios para la reconstrucción de la imagen.

5.3. Curvas Rate-Distortion para imágenes naturales

Las curvas Rate-Distortion muestran en ordenadas la entropía obtenida de la imagen y en abcisas el error obtenido el cual, tal y como se explicó en la sección 5.1 será el RMSE. Estas gráficas se presentarán para las cuatro imágenes estándar que también se vieron en el apartado 5.1. Sin embargo, la mejor prueba será la comparación visual directa de las imágenes entre sí, para comprobar, de este modo, cuál de todas es la más similar perceptualmente a la original. Estas gráficas se obtendrán para distinto número de niveles de cuantización y para distintos algoritmos de compresión basados todos en las SVM.

Como punto de inicio se fija el número de níveles de cuantización y a partir de ahí se prueban las tres variantes del algoritmo: ε constante, RKi y ε adaptada perceptualmente con la CSF. Tal y como se comentó en la sección 4.3.1 el programa ajustará el valor inicial de ε para conseguir el valor de entropía deseada.

Los resultados que se van a mostrar se organizan del siguiente modo, en primer lugar se presentarán las gráficas obtenidas para el RMSE en los casos en los que los pesos se codificaban con nueve, seis y tres bits. A continuación se presentará una gráfica en la que se juntaran las anteriores para poder comparar en una misma figura lo obtenido. Finalmente se presentarán todas las imágenes obtenidas para la comparación subjetiva, que será la que realmente nos diga cuál es la mejor imagen obtenida.

Las curvas Rate-Distortion para todas las configuraciones y todas las imágenes se pueden ver en las figuras 5.3 (Lena), 5.6 (Cameraman), 5.10 (Boats) y 5.14 (Barbara), respectivamente. Para cada configuración (número de niveles de cuantización) se presentan seis imágenes: las obtenidas para cada uno de los algoritmos que estamos probando en los niveles de 0.3 bpp y 0.5 bpp. Esto se repite para todas las imágenes, con lo que la comparación subjetiva se debe hacer con las siguientes figuras:

- Lena: Las imágenes correspondientes a nueve bits se pueden contemplar en la figura 5.3, seis bits en la figura 5.4 y tres bits en la figura 5.5.
- Cameraman: Las imágenes correspondientes a nueve bits se pueden observar en la figura 5.7, seis bits en la figura 5.8 y tres bits en la figura 5.9.
- Boat: Las imágenes correspondientes a nueve bits se pueden ver en la figura 5.11, seis bits en la figura 5.12 y tres bits en la figura 5.13.
- Barbara: Las imágenes correspondientes a nueve bits se observan en la figura 5.15, seis bits en la figura 5.16 y tres bits en la figura 5.17.

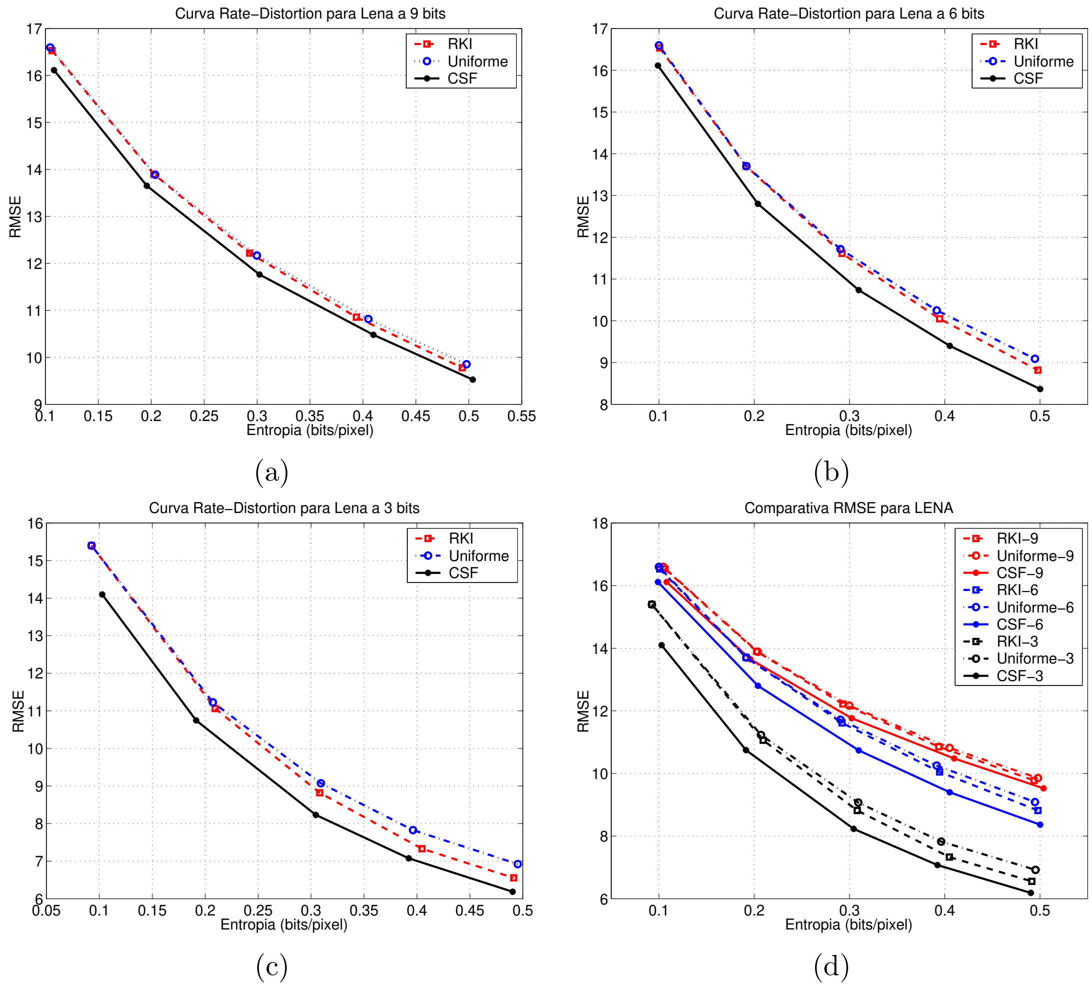


Figura 5.2: Curvas Rate-Distortion para Lena (a) nueve bits, (b) seis bits, (c) tres bits. En (d) vemos la comparación entre los tres casos anteriores. En las tres primeras gráficas observamos como las curvas con valores más bajos del RMSE pertenecen a la SVM adaptativa perceptual, mientras que apenas existe diferencia entre las curvas obtenidas mediantes los algoritmos de perfil uniforme y RKi. Finalmente en las comparación entre las tres configuraciones observamos como los mejores resultados se obtienen con tan sólo tres bits de codificación.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.3: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 9 bits de cuantización. Se aprecia una mejora en el uso del perfil perceptual puesto que es donde menos patente se hace el bloqueado de la imagen tanto en 0.3 com en 0.5 bpp. Por otro lado se ve como las imágenes obtenidas para RKi y el perfil uniforme son prácticamente iguales.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.4: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 6 bits de cuantización. También aquí se repite el comportamiento de la figura 5.3. La SVM adaptativa es de nuevo la que mejores imágenes nos ofrece.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

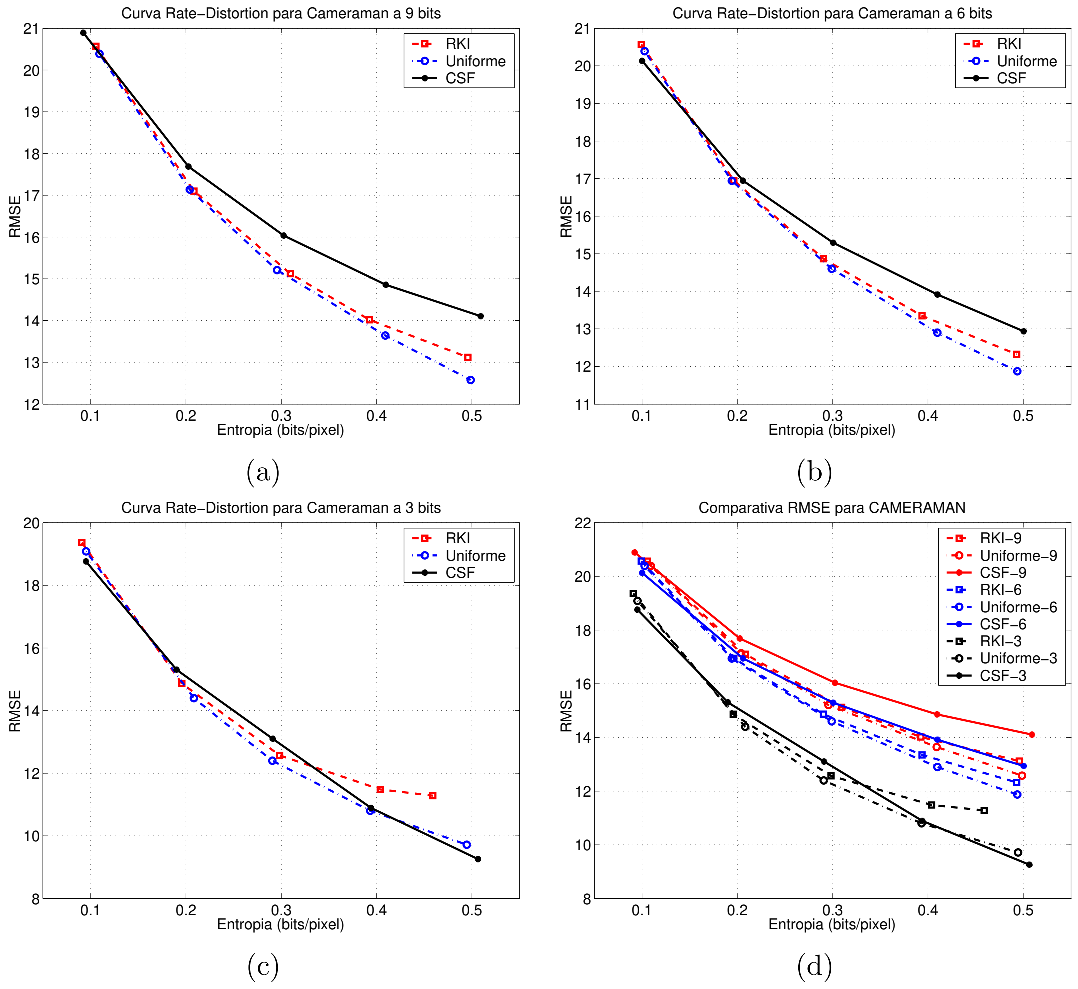
(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.5: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 3 bits de cuantización. En esta ocasión la mejora no es tan apreciable en la columna de imágenes a 0.5 bpp, pero sí se hace más patente en la columna de 0.3 bpp, donde de nuevo la imagen obtenida mediante la SVM adaptativa es la menos afectada por la distorsión.



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

Figura 5.6: Curvas Rate-Distortion para Cameraman (a) nueve bits, (b) seis bits, (c) tres bits. En (d) vemos la comparación entre los tres casos anteriores. En esta ocasión mientras que el algoritmo RKi y el basado en perfil uniforme siguen obteniendo aproximadamente las mismas curvas R-D, la SVM adaptativa perceptual obtienes valores mayores del RMSE en los casos de nueve y seis bits. Sin embargo, a tres bits las tres curvas presentan comportamientos muy similares. Respecto a la comparación entre niveles de nuevo observamos como la configuración que obtiene menores errores es la que usa tan sólo tres bits.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.7: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 9 bits de cuantización. Aunque las gráficas de la figura 5.6 mostraban errores mayores usando el perfil CSF, la comparación visual nos dice que la imagen menos distorsionada es la que correponde justamente a dicho perfil. Esto ocurre tanto a 0.3 como a 0.5 bpp.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.8: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 6 bits de cuantización. De nuevo observamos como la imagen obtenida mediante el perfil perceptual es la que menos distorsión presenta tanto en 0.3 como en 0.5 bpp.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.9: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 3 bits de cuantización. Una vez más observamos como la imagen más semejante a la original es la conseguida mediante la SVM adaptativa perceptual, tanto en la columna de 0.3 como en la 0.5 bpp.

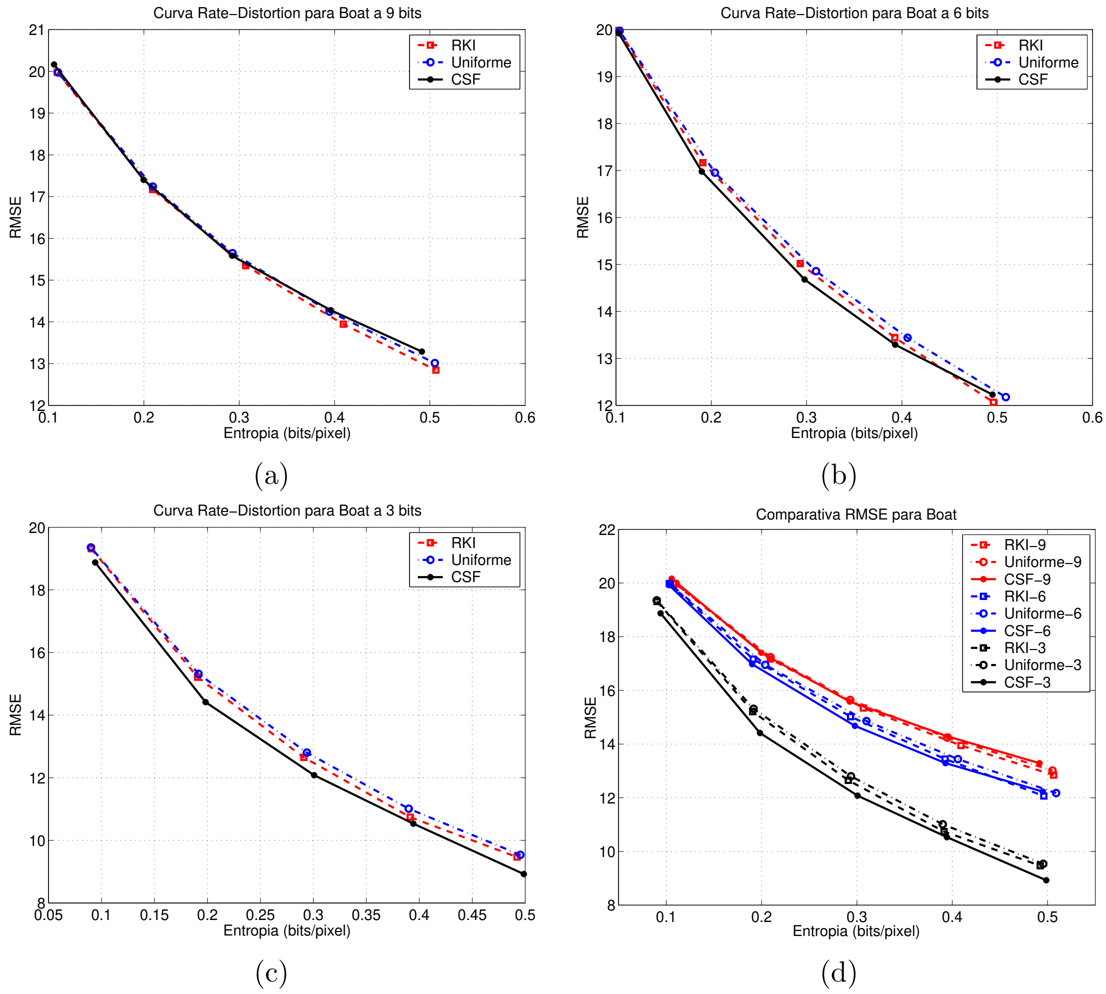


Figura 5.10: Curvas Rate-Distortion para Boats (a) nueve bits, (b) seis bits, (c) tres bits. En (d) vemos la comparación entre los tres casos anteriores. Ahora lo que ocurre es que tenemos prácticamente el mismo comportamiento y los mismos valores de RMSE en los tres casos y para los tres algoritmo distintos, esto no hace más que confirmarnos que la comparación realmente importante es la subjetiva. En la comparativa entre los resutlados obtenidos según el distinto número de niveles de cuantización volvemos a obtener el resultado que ya vimos en las figuras 5.3 y 5.6, es decir que la mejor configuración resulta ser la de 3 bits.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.11: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 9 bits de cuantización. En esta imagen también obtenemos los mejores resultados mediante el uso de la SVM adaptativa.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKI 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.12: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 6 bits de cuantización. De nuevo se observa como las imágenes menos distorsionadas son las asociadas al uso de un perfil perceptual.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.13: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 3 bits de cuantización. Mientras que 0.5 bpp no se aprecia mucha diferencia entre los casos RKi, uniforme y CSF en 0.3 bpp sí que se ve más claramente como una vez más el perfil CSF es el que menos error origina en la imagen reconstruida.

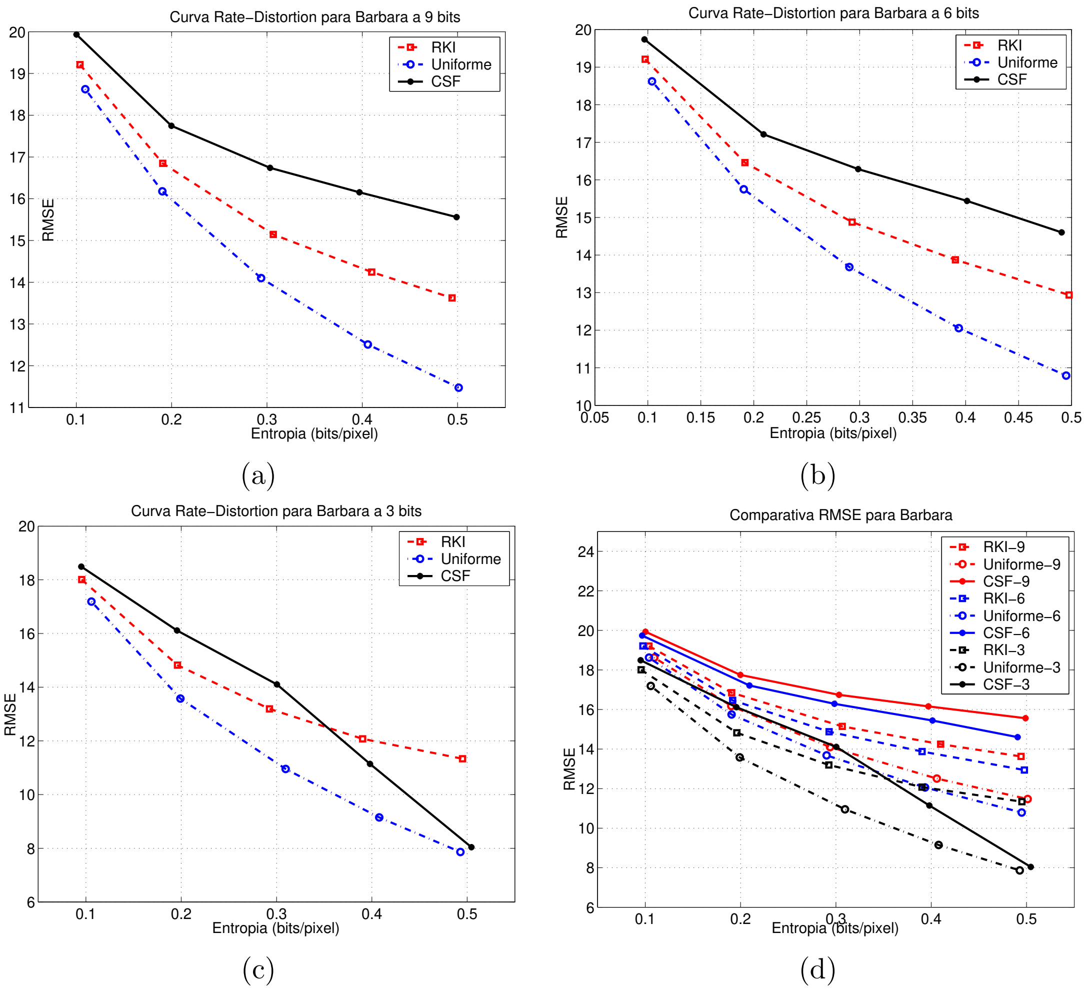


Figura 5.14: Curvas Rate-Distortion para Barbara (a) nueve bits, (b) seis bits, (c) tres bits. En (d) vemos la comparación entre los tres casos anteriores. Es en esta imagen donde mayor separación vemos entre las curvas correspondientes a distintos métodos de compresión, sobre todo en las configuraciones de nueve y seis bits. Además en ambos casos es nuestro algoritmo es que mayor distorsión objetiva presenta. En la configuración de sólo tres bits las curvas vuelven a ser más similares.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.15: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 9 bits de cuantización. Las imágenes obtenidas mediante al perfil CSF son las que menos han considerado las componentes de alta frecuencia, por ello en la zona del pañuelo no se aprecia el rayado. Por contra, es la que presenta un aspecto más suavizado, menos afectado por la división en bloques de 16×16 píxeles de la imagen.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.16: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 6 bits de cuantización. Se obtiene prácticamente lo mismo que en 5.15, por lo que las conclusiones a las que se llega son las mismas.

(a) $\varepsilon = cte$ 0.3 bits/píxel

(b) $\varepsilon = cte$ 0.5 bits/píxel

(c) RKi 0.3 bits/píxel

(d) RKi 0.5 bits/píxel

(e) CSF 0.3 bits/píxel

(f) CSF 0.5 bits/píxel

Figura 5.17: Comparación de las imágenes obtenidas con los distintos perfiles usados usando 3 bits de cuantización. En esta configuración el perfil CSF ya es capaz de reproducir las componentes de alta frecuencia, por lo que ya podemos ver el rayado del pañuelo. Tal y como ocurría para las imágenes anteriores es en el nivel de 0.4 bpp donde más se aprecia la mejora en la imagen proveniente del uso del perfil perceptual.

De todas las imágenes que hemos presentado anteriormente se puede concluir lo siguiente:

- La calidad relativa de las imágenes obtenidas es la siguiente:

$$C_{CSF} > C_{RKI} > C_{\varepsilon}$$

  El principal efecto que se observa es que el algoritmo basado en la SVM adaptativa perceptual es que minimiza el efecto de los bloques en la imagen, mientras que en los otros dos algoritmos dichos bloques se hacen más patentes.

- Las agrupaciones de imágenes que presentan tanto menor distorsión objetiva (RMSE) como subjetiva (perceptual), son las que se obtienen usando tan sólo ocho niveles de cuantización. Esto es así dado que al usar un número bajo de estos niveles, la SVM puede tomar valores más pequeños para la zona de insensibilidad ε, y de este modo modelizar mejor los coeficientes de la DCT. Lo que realmente ocurre en esta configuración es que tenemos más pesos, w_i, pero más afectados por el error de cuantización, es decir más distorsionados. Por el contrario, en los casos donde se trabaja con seis y nueve bits para codificar los niveles de cuantización, la SVM se ve obligada a escoger poco vectores soporte, pero los pesos asociados a ellos se cuantizan con muy poco error. En cualquier caso, tal y como se ha visto los menores errores, se obtienen usando tan sólo tres bits.

- La diferencia relativa entre los tres métodos es mayor cuando aumenta el número de niveles de cuantización, es decir cuando la disminución de entropía viene dada por una colocación de los vectores soporte más adecuada. Por ello la ganancia conseguida mediante el uso de la SVM adaptativa perceptual se observa más claramente en situaciones de menor entropía tal y como se puede ver en todas las imágenes que hemos presentado en el nivel de 0.3 bpp, donde se aprecía una mayor calidad de nuestras imágenes.

- Tal y como ya hemos comentado en ocasiones anteriores, las curvas de Rate-Distortion puede llevarnos a engaño, puesto que la medida del error en el dominio espacial no es necesariamente igual al error en la percepción de la imagen. Este efecto se observa más acusadamente en la figura 5.14, donde las curvas que mayores RMSE presentan son las correspondientes al uso del perfil CSF, mientras que la comparación visual de la imágenes de las figuras 5.15, 5.16 y 5.17 nos muestra como las imágenes mejor reconstruidas son las obtenidas por ese perfil.

5.3.1. Curvas Rate-Peceptual Distortion

Según hemos obtenido de la comparación visual de las imágenes obtenidas, el algoritmo de compresión basado en la SVM adaptativa perceptual ha sido siempre el que presentaba menor distorsión subjetiva, no obstante, esto no está reflejado en las curvas de distorsión de RMSE. Este es un resultado esperado debido a la ineficiencia de las medidas euclídeas para reproducir resultados perceptuales, tal y como ya explicamos en la sección 3.2. Para confirmar esto vamos a calcular las distancias perceptuales, es decir la distorsión subjetiva (ecuación (3.14)), de las imágenes obtenidas mediante los tres algoritmos con los que hemos trabajado. Los resultados se pueden ver la figura 5.18.

Como vemos en la figura 5.18 la SVM-CSF es la que menor distorsión perceptual presenta en todos los casos. Esto coincide con la comparación visual que hemos hecho previamente con las imágenes comprimidas.

Con el fin de analizar las mejoras derivadas del uso de nuestro algoritmo vamos a llevar una seríe de estudios sobre los parámatros de entrenamiento de las SVM que estamos usando.

5.3.2. Histogramas de Vectores Soporte

Los vectores soporte constituyen las muestras más importante del espacio de entrenamiento, es decir aquellas cuya información es más relevante para la modelización. Mediante la representación en histogramas de las posiciones de los vectores soporte para una determinada imagen y para cada método podremos comprender los diferentes resultados que hemos obtenido.

Por ello, a continuación vamos a presentar los histogramas obtenidos para Barbara a 0.5 bpp en las configuraciónes de tres y nueve bits de cuantización. Para esta prueba se ha escogido Barbara por dos razones; por ser la que mayores componentes de alta frecuencia presenta, y también por haber sido aquella en la que la SVM adaptativa perceptual obtuvo peores resultados en la comparación de RMSE.

La figura 5.19 es muy significativa del distinto funcionamiento de cada uno de los algoritmos que hemos usado. Por ello un análisis exhaustivo de estas gráficas nos mostrará claramente porqué el algoritmo basado en SVM adaptativa perceptual es el que consigue mejores resultados visuales.

- **SVM con perfil uniforme.** De la observación de la figura 5.19 (a), concluimos que el perfil uniforme tiende a tomar muchos vectores soporte en muy bajas frecuencias para presentar una rápida caída exponencial. Por lo que en frecuencias bajas, medias y altas escoge menor cantidad de vectores soporte. Como era de esperar, en la configuración de tres bits 5.19 (b) esta caída no es tan abrupta, y se selecciona un mayor número de SVs en todo el rango

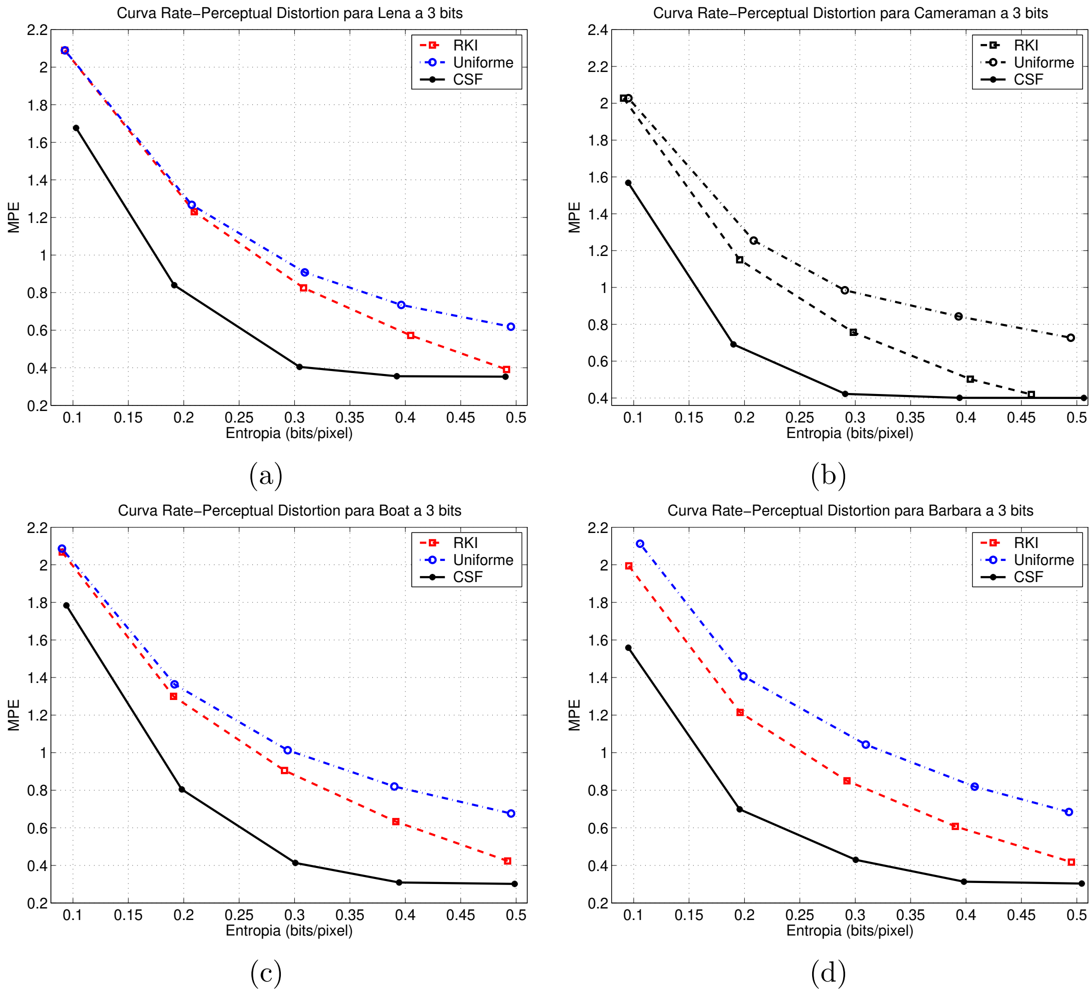


Figura 5.18: Curvas Rate-Perceptual Distortion para (a) Lena, (b) Cameraman, (c) Boat y (d) Barbara. Todas las curvas han sido obtenidas para la configuración de tres bits de cuantización. Se puede observar claramente como en todas las gráficas se reproduce la secuencia que obteníamos en la comparación visual; la implementación SVM-CSF es la que menor distorsión subjetiva presenta, seguida de la RKi. La que peores resultados obtiene es la SVM de perfil uniforme.

de frecuencias, dado que el algoritmo ajusta ε a un valor menor que el caso anterior.

- **RKi.** Como ya vimos en el cuarto capítulo el algoritmo RKi elimina ad-hoc los 191 últimos coeficientes de la DCT, por ello en el histograma sólo aparecen vectores soporte en las 64 primeras posiciones. Como vemos, este algoritmo elimina las componentes frecuenciales que se encuentren más allá de dicho coeficiente. La comparación entre 5.19 (a) y (b) muestra que la caída con el algoritmo RKi no es tan abrupta como en el perceptual, y la cantidad de vectores soporte que se seleccionan a bajas frecuencias es mayor, lo cual irá asociado a una zona ε permitida menor. La conclusión que obtenemos es que el algoritmo RKi tiene la ventaja frente al de perfil uniforme de que, al solo considerar los primeros coeficientes de la DCT, puede permitirse una modelización mejor para un mismo nivel de entropía, es decir, como tiene menos datos a modelar, puede hacerlo mejor que el de perfil constante. Por ello precisa una ε menor para obtener buenos resultados y obtiene un número mayor de vectores soporte.

- **SVM adaptativa perceptual.** En comparación con los otros dos algoritmos, éste es el que presenta una mayor concentración de coeficientes en las frecuencias bajas. Este comportamiento es consecuencia de haber usado como perfil de error la CSF. Por otro lado, la caída no es tan abrupta como lo era en los dos casos anteriores. También, hay que destacar que en 5.19 (c) vemos que se han seleccionado vectores soporte en frecuencias medias y altas. Estos SVs estarán representando componentes reales de alta frecuencia existentes en la imagen los cuales nuestro algoritmo ha sido capaz de detectar y modelar, al contrario que ocurría en RKi, donde esos coeficientes han sido eliminados después del entrenamiento.

También resulta útil conocer el número total de vectores soporte que ha empleado cada uno de los algoritmos para la modelización de la imagen. Estos datos los podemos ver en la tabla 5.1.

	Uniforme	**RKi**	**SVM Adaptativa Perceptual**
9 bits	3425	3373	3768
3 bits	11457	7664	11302

Cuadro 5.1: Tabla comparativa del número de vectores soporte selecionados por cada uno de los algoritmos.

Se comprueba que RKi es el algoritmo que menos vectores soporte selecciona, lo cúal dado que se lleva a cabo una eliminación de coeficientes, era algo de esperar. El algoritmo basado en SVM insensitiva y el que usa SVM adaptativa obtienen valores similares, lo que ocurre es que, tal y como se ve en los histogramas, la colocación es muy distinta.

Después de estas observaciones, llegamos a la conclusión de que el algoritmo SVM adaptativo perceptual es que el que mejores resultados perceptuales obtiene porque es el que realiza una selección de vectores soporte más acorde con la percepción visual. Esto era lo que queriamos conseguir mediante el uso de la CSF como perfil de error variable en el entrenamiento de la SVM.

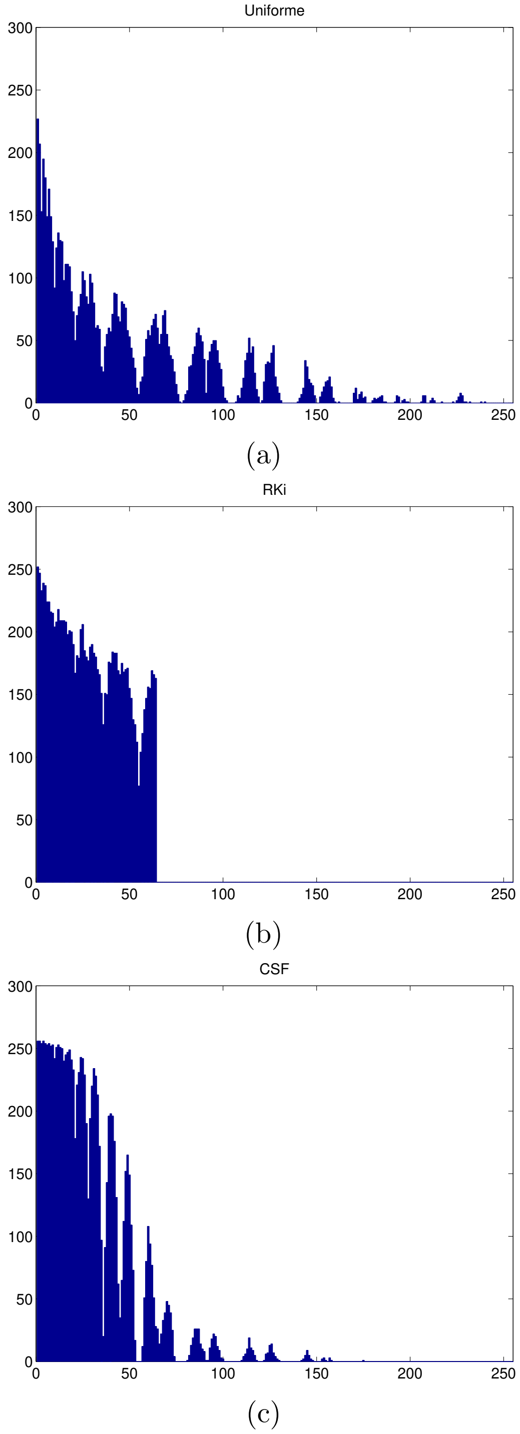


Figura 5.19: Histogramas de distribución de vectores soporte para Barbara. En (a) tenemos el histograma para una SVM insensitiva, en (b) para el algoritmo RKi y en (c) la SVM adaptativa perceptual. En el eje horizontal esta representada la posición en el dominio zigzagueado de la DCT, mientras que en el vertical se muestra la cantidad de vectores soporte en una determinada posición para toda la imagen. Como se puede ver la distribución es distinta según el método de compresión que hayamos usado.

5.3.3. Parámetros óptimos: zona de insesibilidad ε

El principal parámetro que ajusta nuestro algoritmo es la anchura de la zona de insensibilidad ε. En el caso de la SVM adaptativa, como ya hemos comentado en más de una ocasión, esta zona depende de la muestra que estemos modelizando. Por ello, la anchura de esta zona varía a lo largo del dominio de la DCT. De este modo conseguimos que las zonas con menos importancia perceptual sean poco consideradas en beneficio de las partes con más información. Los algoritmos con perfiles uniformes no pueden hacer esto, por lo que no consideran necesariamente la información más relevante. En la figura 5.20 podemos ver un zoom de los 64 primeros coeficientes para un bloque promedio de Barbara en el dominio de la DCT.

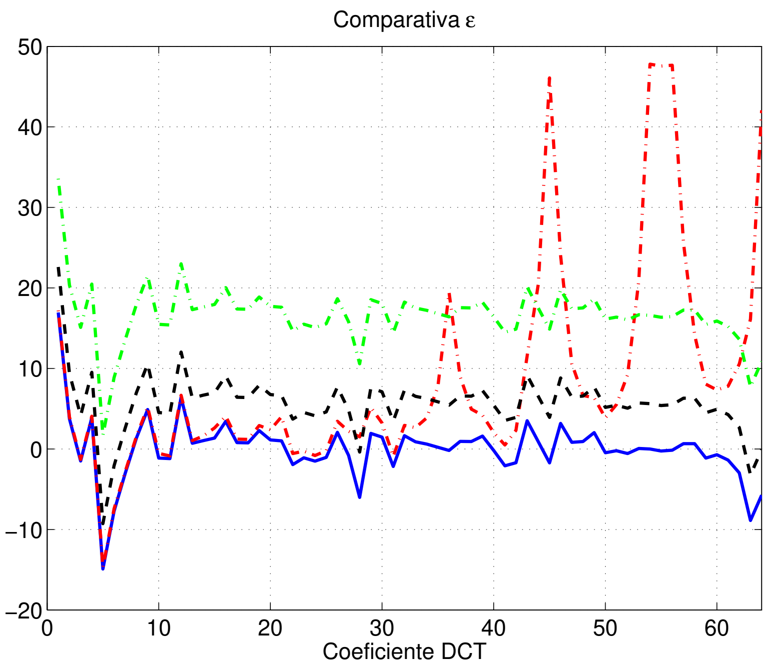


Figura 5.20: Comparativa de las zonas de error permitido para los distintos algoritmos. La línea continua azul representa el valor promedio de la DCT de todos los bloques de Barbara. Las lineas discontinuas representan zonas ε, la roja representa la CSF, la negra RKi y la verde el perfl uniforme. Para esta gráfica se han usado los valores de la configuración de tres bits ya que es la configuración que mejores resultados ha mostrado.

Como se puede ver en la figura 5.20, las zonas de insensibilidad son distintas para cada implementación. Estas zonas, como hemos comentado anteriormente, las escoge el algoritmo según sea el valor de la entropía (compresión) deseada por el usuario.

A menor entropía deseada (mayor compresión) la zona de insensibilidad tendrá que ser más ancha, y viceversa.

Gracias a la SVM adaptativa y a la forma de la CSF, podemos obtener una zona de insenbilidad muy baja (línea roja de la gráfica 5.20) en la modelización en los coeficientes de la DCT más significativos, a costa de permitir mucho error en el resto de coeficientes. Los otros algoritmos al no variar su zona de insensibilidad pierden eficacia en los coeficientes relevantes a cambio de ganarla en frecuencias poco relevantes.

5.3.4. Errores de modelización

Otra medida que vamos a llevar a cabo es el RMSE de las modelizaciones que llevan a cabo los tres algoritmos, es decir el error sobre la señal reconstruida, los propios coeficientes de la DCT. Esta medida también nos proporcionará otro enfoque para comprobar cual de los tres métodos que hemos usado es el mejor. Como el proceso de cuantización es igual para los tres algoritmos (cuantización uniforme), es de suponer que este proceso introducirá un error muy similar en los tres. Por ello la medida del RMSE antes de llevar a cabo la cuantización nos mostrará que algoritmo es el que modela mejor los coeficientes de la DCT.

Por ello, lo que hacemos es medir el RMSE a unas determinadas entropías en función de la ε que ha ajustado el algoritmo, según el número de niveles de cuantización, para cumplir las especificaciones del usuario sobre la entropía deseada. Esta prueba la realizamos sobre la imagen Barbara a los niveles de entropía de 0.3 bpp y 0.5 bpp.

	9 bits	**6 bits**	**3 bits**
Uniforme	6.35	6.06	4.86
RKi	6.45	6.10	4.71
SVM Adap. Per.	6.13	5.67	4.47

Cuadro 5.2: Tabla comparativa de RMSE para los distintos algoritmos para Barbara a 0.5 bpp. Se observa como simpre se obtiene un RMSE menor con la SVM adaptativa perceptual.

Como se puede observar en las tablas 5.2 y 5.3, los valores menores de RMSE se obtienen siempre con la SVM adaptativa perceptual. Sin embargo, las diferencias entre los valores son tan pequeñas que podrían no ser más que una desviación estadística. En cambio sí que hemos comprobado que visualmente la diferencia entre los tres algoritmo si que era destacable.

Una posible explicación para este hecho la encontramos si recordamos que la

	9 bits	6 bits	3 bits
Uniforme	7.11	6.96	6.16
RKi	7.10	6.96	6.17
SVM Adap. Per.	6.76	6.56	5.69

Cuadro 5.3: Tabla comparativa de RMSE para los distintos algoritmos para Barbara a 0.3 bpp. De nuevo nuestro algoritmo es que mejores resultados consigue.

percepción final de una imagen se encuentra en el dominio de la respuesta, y no en el de la frecuencia local. La transformación que conecta estos dos dominios, según los últimos estudios en el campo, es no lineal, por lo que pequeños errores en el dominio de frecuencia local, pueden no serlo tanto en el dominio de la respuesta. De este modo, se explicaría la apreciables diferencia visual entre imágenes, aunque sus errores en el dominio de la DCT estén muy parejos. También hay que considerar que será mas relevante que el error se haya cometido en bajas frecuencias que en altas.

5.3.5. Valores usados para los parámetros libres del algoritmo

Tal y como comentamos en la sección 4.3.1, disponemos principalmente de cuatro parámetros libres (C, ε_0, σ y el número de niveles de cuantización) a los cuales hemos de dar unos valores apropiados. Para la constante de penalización C ya argumentamos en la misma sección que un valor apropiado es $C = 1000$, que es el que hemos empleado en todo el proyecto.

Para el parámetro σ, que representa la anchura de la Gaussiana de la función *kernel*, se ha de escoger un valor de compromiso entre valores altos, que captarán más relaciones entre coeficientes vecinos, y valores bajos que modelizarán mejor cada coeficiente por separado. El valor típico que hemos usado es $\sigma = 0{,}1$.

El número de niveles de cuantización óptimo, según hemos observado en las curvas de Rate-Distortion, ha resultado ser ocho niveles, que corresponde a un esquema de cuantización de tres bits. Era con esta configuración con la que el RMSE era menor para todas las imágenes.

Finalmente, la constante inicial ε_0, la selecciona el propio algoritmo para cumplir la especificación de entropía dada por el usuario. En las tablas 5.4 y 5.5 podemos ver los valores de ε_0 que escogen los tres algoritmos para las imágenes Lena y Barbara respectivamente.

Entropía	Perfil Uniforme	RKi	SMV-CSF
0.1	79	79	9000
0.2	39	37	2298
0.3	24	20	401
0.4	16	10	98
0.5	11	2	22

Cuadro 5.4: Tabla comparativa de los valores que selecciona cada uno de los algoritmos para ε. Los resultados de esta tabla se refieren a Lena con ocho niveles de cuantización y con $C = 1000$ y $\sigma = 0{,}1$.

Entropía	Perfil Uniforme	RKi	SMV-CSF
0.1	85	79	10000
0.2	50	40	1700
0.3	33	24	400
0.4	23	12	130
0.5	17	4	48

Cuadro 5.5: Tabla comparativa de los valores que selecciona cada uno de los algoritmos para ε. Los resultados de esta tabla se refieren a Barbara con ocho niveles de cuantización y con $C = 1000$ y $\sigma = 0{,}1$.

5.4. Comparación con JPEG

Una vez llevada a cabo la comparación entre las distintas familias de algoritmos que usan las SVM para compresión de imágenes, vamos a pasar a comparar la SVM adaptativa perpceptual, con el estándar de compresión de imágenes JPEG que describimos en la introducción. En esta sección presentaremos las curvas comparativas de Rate-Distortion para Barbara, así como las imágenes obtenidas en los niveles de 0.5 y 0.3 bpp.

Las curvas del RMSE que se observan en la figura 5.21 muestran que se obtienen mejores resultados mediente el uso de la SVM adaptativa perceptual para todo el rango de entropías representadas. Sin embargo, esta diferencia relativa entre las curvas es más patente a entropías altas, puesto que conforme disminuye la compresión ambos errores van tomando valores más cercanos. Para Barbara a niveles altos de entropía el RMSE obtenido para JPEG llega ser el doble que el obtenido mediante la SVM adaptativa perceptual.

Una de las principales conclusiones que sacamos de estas curvas es que nuestro algoritmo no presenta tanta robustez como JPEG cuando exigimos mucha compresión.

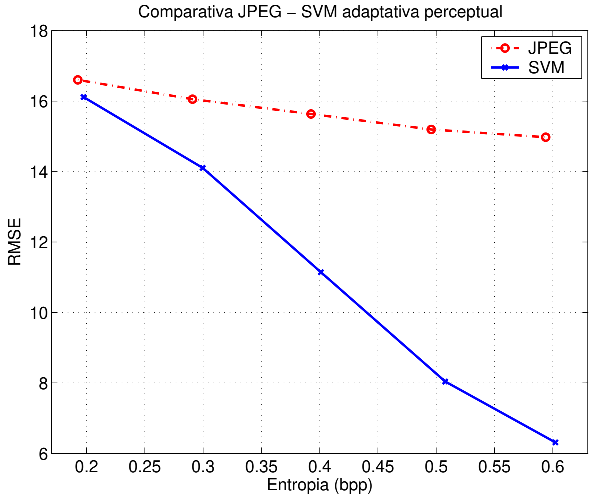


Figura 5.21: El menor RMSE para todas la entropías se obtiene con la SVM adaptativa. Para esta imagen (Barbara) en concreto la diferencia entre JPEG y SVM adaptativa es bastante apreciable.

En cualquier caso, tal y como ya se comentó en secciones anteriores, debido al comportamiento del sistema visual humano el error en el dominio espacial (RMSE) no es muy relevante, por lo que vamos a comparar subjetivamente las imágenes obtenidas mediante ambos métodos. Los resultados obtenidos para Barbara en la figura 5.22.

También aquí vamos a aplicar la medida perceptual de la distancia de Nill [30] a la imagen obtenida mediante nuestro algoritmo y a la imagen que nos ofrece JPEG. Los resultados a los que llegamos se pueden ver en la figura 5.23. La principal conclusión a la que se llega mediante la observación de la figura nombrada anteriormente es que la SVM perceptual obtiene mejores resultados en todo el rango de entropías que se estudia.

(a) JPEG 0.3 bpp

(b) JPEG 0.5 bpp

(a) SVM 0.3 bpp

(b) SVM 0.5 bpp

Figura 5.22: En este caso observamos que en ambos niveles las imágenes obtenidas mediante nuestro algoritmo son subjetivamente mejores, puesto que reproducen mejor las altas frecuencias, y no están tan afectadas por el bloqueado que queda más al descubierto en las imágenes obtenidas mediante JPEG.

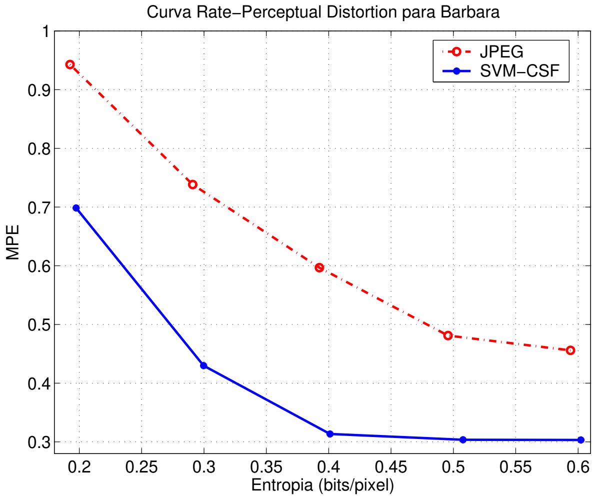


Figura 5.23: Como vemos en la gráfica la SVM-CSF obtiene menores distorsiones perceptuales en todo el rango de entropías.

Capítulo 6

Conclusiones

6.1. Conclusiones finales

Los resultados que hemos obtenido mediante el uso de la SVM adaptativa y perceptual, han superado claramente a los obtenidos en [37], tanto en el test objetivo (RMSE y SNR) como en el subjetivo. Como hemos podido ver en la comparación visual directa entre las imágenes obtenidas, en todos los casos las que presentaban una distorsión perceptual menor eran las que se obtenían mediante nuestro algoritmo. Esto deja patente que las aproximaciones que se realizan en el algoritmo RKI no sólo no son muy justificables, sino que tampoco resultan demasiado eficientes.

Tal y como comentamos en el capítulo tercero, la reducción en la distorsión subjetiva de una imagen comprimida dependía de tener en cuenta la distinta importancia perceptual de los coeficientes del dominio transformado. Esto lo hemos conseguido mediante el uso de la función de sensibilidad al contraste, obteniendo así una ganancia en la calidad de la imagen respecto al algoritmo RKi. Por otro lado, también hemos demostrado que la eliminación de coeficientes que hacen Robinson y Kecman, no tiene efectos sobre la disminución de la entropía de los datos a almacenar.

También hemos obtenido mejores resultados que el estándar JPEG, tanto en el RMSE como en la distancia de Nill. Dado que ambos algoritmos tienen en cuenta la misma información perceptual (la CSF), la ganancia obtenida en la calidad es debida a la capacidad de aprendizaje de las SVMs. Esto nos muestra que la combinación de entrenamiento de SVMs con información perceptual es lo que constituye una buena herramienta para la compresión de imágenes.

6.2. Trabajo futuro

De las conclusiones obtenidas anteriormente llegamos a que se abren principalmente tres nuevas vías de ampliación del algoritmo aquí presentado tanto desde el punto de vista teórico como práctico.

La mejora del algoritmo se deberá centrar en:

- **Usar la transformada Wavelet en lugar de la DCT.** La transformada Wavelet lleva a cabo un mapeado a un dominio transformado de frecuencia local, en lugar de dominio frecuencial únicamente como hace la DCT. Según los últimos estudios sobre el sistema visual humano parecer ser que las neuronas llevan acabo una transformción del estímulo incidente de modo similar a como lo hace la transformada Wavelet. Este cambio de orientación ya se ha llevado a cabo con éxito en JPEG2000.

- **Uso de los modelos perceptuales no lineales.** La respuesta final de nuestro sistema visual es no lineal con los coeficientes del dominio transformado. En el algoritmo que hemos propuesto, este hecho no lo hemos considerado, quedándonos sólo con él término lineal de la primera aproximación. Estos modelos no lineales ya se han usado con éxito en [26] y [25], por lo que su uso en la SVM adaptativa perceptual se espera que mejore el comportamiento de ésta última.

- **Función kernel adaptativa.** Según se muestra en [25] los coeficientes de la DCT están muy poco correlacionados en las bajas frecuencias y mucho en las altas. Por ello el uso de funciones kernel RBF que pudieran adaptar su anchura de Gaussiana en función de la frecuencia, mejoraría la modelización de la DCT.

Por otro lado las aplicaciones de este PFC pueden extenderse a un campo más general de la codificación de imágenes:

- **Extensión del algoritmo a compresión de imágenes en color**. Todo el estudio que hemos realizado en este PFC ha tenido lugar en el plano de la lumninancia. Las imágenes en color constan además de planos crómaticos, que también deberán ser codificados. La extensión a este problema resulta inmediato desde el punto de vista de la formulación ya que se trata, básicamente, de un problema de multiregresión (varias señales a modelar simultaneamente). En este campo, se han dado importantes avances recientemente mediante la formulación de SVR de multi-regresión que aprovechan la información de unas modelizaciones para mejorar las otras [35].

- **Aplicación a codificación de video**. Los algoritmos de compresión de video constan de un codificador de imágenes y de un sistema de predicción de movimiento. Dado que nosotros hemos desarrollado un nuevo sistema de compresión de imágenes, deberemos probarlo en la codificación de video. La experiencia en este campo resulta importante [19], por lo que la extensión a este campo mediante la inclusión de la CSF-SVR puede resultar relativamente sencilla.

- **Aplicación a imágenes hiperespectrales.** La teledetección consiste en la extracción de información de una área mediante un sistema de sensores que captan la radiación emitida o reflejada en diferentes longitudes de onda y en su posterior tratamiento. Las imágenes obtenidas mediante estos procedimientos contienen un gran volumen de información, puesto que los sensores actuales son capaces de medir en un amplio rango del espectro electromagnético. La aplicación de algoritmos basados en SVM podría suponer un nuevo avance en la compresión de las imágenes hiperespectrales. Por este motivo, la compresión es un aspecto clave para el tratamiento y gestión de este tipo de imágenes. Este hecho se ve agravado más si cabe por el incesante aumento de la resolución espectral y espacial de los sensores, así como por el interés de monitorización temporal de determinadas zonas geográficas, lo que provoca un almacenamiento masivo de datos para su posterior análisis, visualización, y tratamiento. Por tanto, se hace necesaria la introducción de herramientas avanzadas y automáticas para el tratamiento de imágenes hiperespectrales que exploten las regularidades espaciales y espectrales de la imagen y que conduzcan a dos resultados inmediatos: (1) reducción del tamaño ocupado por la información, y (2) facilidad de manejo de la información relevante. Estos dos problemas implican etapas de compresión y clasificación, respectivamente. La experiencia previa [6, 5, 7] sobre algoritmos de clasificación, selección y extracción de características mediante SVM para la clasificación de imágenes hiperespectrales, permite la extensión al campo de la compresión.

Por último, cabe destacar que el presente trabajo ha sido resumido en dos trabajos que se encuentran en estos momentos bajo evaluación:

1. "Compresión de imágenes mediante SVM adaptativa perceptual" Gabriel Gómez Pérez, 2004. Resumen del Proyecto Final de Carrera de la titulación de Ingeniería Electrónica (Universitat de València – Estudi General) que concurre a los VI Premios Fundación 3M a la Innovación 2004 en la categoría de *Aplicaciones en Industria*. Mayo, 2004

2. Gómez-Pérez, G., Camps Valls, G., J. Guitérrez, Malo-López, J.: Perceptual Adaptive Insensitivity for SupportVector Machine Image Coding. IEEE Transactions on Neural Networks, 2004. (Enviado).

Bibliografía

[1] A. Ben-Hur, D. Horn, H. Siegelsmann, and V. Vapnik. A support vector method for clustering. *Neural Information Processing Systems, NIPS2000*, 2000.

[2] F.W. Campbell and J.G. Robson. Application of Fourier analysis to the visibility of gratings. *Journal of Physiology*, 197:551–566, 1968.

[3] G. Camps, E. Soria, J. Pérez, A. Artés, F. Pérez, and A. Figueiras. A profile-dependent kernel-based regression for cyclosporine concentration prediction. In *NIPS 2001 – Workshop on New Directions in Kernel-Based Learning Methods*, Vancouver, Canada, December 2001.

[4] G. Camps Valls, A. Chalk, A. Serrano, J. Martín-Guerrero, and E. Sonnhammer. Profiled svr for antisense oligonucleotide efficacy. *BMC Bioinformatics*, 2004.

[5] G. Camps Valls, L. Gómez Chova, J. Calpe, J.D. Martín, E. Soria, and J. Moreno. Kernel methods for hymap imagery knowledge discovery. 2003.

[6] G. Camps Valls, L. Gómez Chova, J. Calpe, J.D. Martín, E. Soria, and J. Moreno. Support vector machines for crop classification using hyperspectral data. 2003.

[7] G. Camps Valls, L. Gómez Chova, J. Calpe, E. Soria, J. Martín, L. Alonso, and J. Moreno. Robust support vector method for hyperspectral data classification and knowledge discovery. *IEEE Transactions on Geoscience and Remote Sensing (2004)*, 2004.

[8] G. Camps-Valls, E. Soria-Olivas, J. Pérez-Ruixo, A. Artés-Rodríguez, F. Pérez-Cruz, and A. Figueiras-Vidal. Cyclosporine concentration prediction using clustering and Support Vector Regression methods. *IEE Electronics Letters*, 38(12):568–570, June 2002.

[9] R.J. Clarke. Relation between the Karhunen-Loeve transform and cosine transforms. *Proceedings IEE, F*, 128(6):359–360, 1981.

[10] C. Cortes and V.Ñ. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[11] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience Publications. John Wiley and Sons, New York, U.S.A., 1953.

[12] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with application in pattern recognition. *IEEE Transaction on Electronic Computers*, 14:304–314, 1965.

[13] B. Dubrovin, S.Ñovikov, and A. Fomenko. *Modern Geometry: Methods and Applications*, chapter 3: *Algebraic Tensor Theory*. Springer Verlag, New York, 1982.

[14] I. Epifanio, J. Gutiérrez, and J. Malo. Linear transform for simultaneous diagonalization of covariance and perceptual metric matrix in image coding. *Pattern Recognition*, 36:1799–1811, 2003.

[15] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press, Boston, 1992.

[16] Steve R. Gunn. Support vector machines for classification and regression. matlab support vector machine toolbox. Technical report, Image Speech and Intelligent Systems Research Group, University of Southampton, http://www.isis.ecs.soton.ac.uk/resources/svminfo/, 1997.

[17] I. Guyon, B. Boser, and V.Ñ. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.

[18] D. J. Heeger. Normalization of cell responses in cat striate cortex. *Visual Neuroscience*, 9:181–198, 1992.

[19] J.Malo, J.Gutierrez, I.Epifanio, F.Ferri, and J.M.Artigas. Perceptual feed-back in multigrid motion estimation using an improved DCT quantization. *IEEE Transactions on Image Processing*, 10(10):1411–1427, 2001.

[20] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

[21] G.E Legge. A power law for contrast discrimination. *Vision Research*, 18:68–91, 1981.

[22] H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, National Taiwan University, Department of Computer Science and Information Engineering, 2003. Available at http://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf.

[23] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. Department of Statistics TR 1016, University of Wisconsin-Madison, 2000. Submitted to Machine Learning (www.kernel-machines.org/papers/upload_683_tr1016.ps).

[24] J. Malo. *Eliminación de Información en el Sistema Visual Humano: Nueva Formulación y Aplicaciones en Codificación.* PhD thesis, Dpt. d'Òptica, Universitat de València, June 1999.

[25] J. Malo. Normalized image representation for efficient coding. *Proc. IEEE 37th Asilomar Conf. Signals Systems & Comp.*, 2:1408–1412, 2003.

[26] J. Malo, F. Ferri, R.Ñavarro, and R. Valerio. Perceptually and statistically decorrelated features for image representation: Application to transform coding. *Proc. XV Intl. Conf. Patt. Recog.*, 3:242–245, 2000.

[27] J. Malo, A.M. Pons, A. Felipe, and A. Artigas. Characterization of human visual system threshold performance by a weighting function in the Gabor domain. *Journal of Modern Optics*, 44(1):127–148, 1997.

[28] S. Marcelja. Mathematical description of the response of simple cortical cells. *Journal of the Optical Society of America*, 70(11):1297–1300, 1980.

[29] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using Support Vector Machines. *Proceedings of IEEE NNSP'97. Amelia Island, FL.*, pages 511–519, 1997.

[30] N.B. Nill and B.R. Bouzas. Objective image quality measure derived from digital image power spectra. *Optical Engineering*, 32(4):813–825, 1992.

[31] H. J. C. Pasika. Neural network sensor fusion engines for remote sensing. Master's thesis, McMaster University, 1999.

[32] Fernando Pérez Cruz. *Máquina de Vectores Soporte Adaptativa y Compacta.* PhD thesis, Universidad Politécnica de Madrid, December 2000.

[33] Massimiliano Pontil, Sayan Mukherjee, and Federico Girosi. On the noise model of support vector machines regression. Technical report, A.I. Memo 1651 CBCL Paper 168, 1998.

[34] W.K. Pratt. *Digital Image Processing*, chapter 8: *Two-Dimensional Unitary Transforms*. John Wiley & Sons, New York, 1991.

[35] F. Pérez-Cruz, G. Camps-Valls, E. Soria-Olivas, J. J. Pérez-Ruixo, A. R. Figueiras-Vidal, and A. Artés-Rodríguez. Multi-dimensional function approximation and regression estimation. In *International Conference on Artificial Neural Networks, ICANN2002*, Madrid, Spain., Aug 2002. Lecture Notes in Computer Science. Springer–Verlag.

[36] J. S. Robinson and V. Kecman. The use of support vector machines in image compression. *Proc. 2nd Intl. Conf. Eng. Intell. Systems*, 36(7):93–96, 2000.

[37] J. S. Robinson and V. Kecman. Combining support vector machines learning with the discrete cosine transform in image compression. *IEEE Transaction on neural networks*, 14(4):950–958, 2003.

[38] B. Schölkopf and A. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press Series, 2001.

[39] Bernhard Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, 1997.

[40] Bernhard Schölkopf, Kah-Kay Sung, Chirs J.C. Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and VladimirÑ. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transaction on Signal Processing*, 45(11):2758–2765, November 1997.

[41] E.P. Simoncelli and E.H. Adelson. *Subband Image Coding*, chapter Subband Transforms, pages 143–192. Kluwer Academic Publishers, Norwell, MA, 1990.

[42] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998. ftp://www.neurocolt.com/pub/ neurocolt/tech_reports/1998/98030.ps.Z.

[43] P.C. Teo and D.J. Heeger. Perceptual image distortion. *Proc. of the First IEEE Intl. Conf. Im. Proc.*, 2:982–986, 1994.

[44] V.Ñ. Vapnik. Estimation of dependences based on empirical data [en ruso]. *Nauka*, 1979. Versión en inglés disponible en Springer-Verlag, New York, 1982.

[45] VladimirÑ. Vapnik. *The Nature of Statistical Learning Theory*. Springer–Verlag, New York, 1995.

[46] VladimirÑ. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[47] G.K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):31–43, 1991.

[48] B.A. Wandell. *Foundations of Vision*. Sinauer Assoc. Publish., Massachusetts, 1995.

[49] A.B. Watson. The cortex transform: Rapid computation of simulated neural images. *Computer Vision, Graphics and Image Processing*, 39:311–327, 1987.

[50] A.B. Watson. DCT quantization matrices visually optimized for individual images. In B.E. Rogowitz, editor, *Human Vision, Visual Processing and Digital Display IV*, volume 1913, 1993.

[51] A.B. Watson and J.A. Solomon. A model of visual contrast gain control and pattern masking. *Journal of the Optical Society of America A*, 14:2379–2391, 1997.

Compresion de imágenes mediante SVM adaptativa perceptual

Gabriel Gómez Pérez, 2004.