TESIS DE LICENCIATURA

DEPARTAMENTO DE COMPUTACIÓN
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
UNIVERSIDAD DE BUENOS AIRES

Título: Reconocimiento de Dígitos Manuscritos Aplicando Transformadas Wavelet sin Submuestreo y Máquinas de Soporte Vectorial

Tesista: Mauro Krikorian
Directora: Dra. Ana Ruedin
Codirectora: Lic. Leticia Seijas

Índice

Resumen

En esta tesis se trata el problema del reconocimiento de dígitos manuscritos, aplicando diferentes transformadas wavelet para la extracción de características, y utilizando máquinas de soporte vectorial para la clasificación de los dígitos, método ampliamente reconocido por su alto rendimiento en el reconocimiento de dígitos manuscritos. Dicho problema tiene aplicaciones tales como el reconocimiento automático de códigos postales, reconocimiento de importes en cheques bancarios, entre otros. Una de las principales dificultades que caracterizan a este problema es que la varianza entre los representantes de cada clase es alta, esto se debe a las diferentes formas que puede tener un mismo patrón debido al estilo particular de escritura de cada individuo. Empleamos en el análisis variantes de la transformada wavelet discreta sin submuestreo (à trous) en dos dimensiones utilizando una serie de transformadas ampliamente conocidas: Daubechies 9/7 [8], Mallat [9] y Symmlet [8], siendo las primeras dos biortogonales y simétricas, y la restante ortogonal. Esto nos aporta distintos esquemas de extracción de características basados en estas transformadas que serán aplicados a los dígitos, a manera de preprocesamiento, para luego entrenar y testear clasificadores basados en máquinas de soporte vectorial consiguiendo una medida del desempeño del esquema aplicado. Por último aplicamos técnicas de combinación de estos clasificadores utilizando diferentes modos de votación. Las pruebas están realizadas sobre bases que son un referente en la literatura, CENPARMI y MNIST, pudiendo lograr resultados que son comparables con muchos de los trabajos ya publicados.

Abstract

This thesis addresses the problem of recognizing handwritten digits, applying different wavelets transforms to extract features, and using support vector machines to classify the digits, widely known method recognized for its high efficacy in the focused problem. This topic as several practical applications like automatic recognition of postal codes, values on banking checks, and others. One of the main difficulties that characterize this problem is that the variance between the representatives of each class is high, this is due to the different ways that you can have the same pattern due to the particular style of writing for each individual. We apply to the analysis different types of discrete wavelet transforms without sub sampling (à trous) in two dimensions using the following widely known wavelet functions: Daubechies 9/7 [8], Mallat [9] and Symmlet [8], being the first two ones biorthogonal and symmetric, and the remaining one orthogonal. This gives us different feature extraction schemes based on these transforms to be applied to the digits, as a preprocessing, and then train and test classifiers based on support vector machines getting a measure of performance of the scheme applied. Finally we apply techniques of combining these classifiers using different modes of voting. The tests are conducted on bases that are a benchmark in the literature, CERPARMI and MNIST, achieving results that are comparable with many published works.

Capítulo 1 – Introducción

El reconocimiento automático de dígitos manuscritos es un tema difícil de abordar por la diversidad de formas, estilos, orientaciones que los mismos presentan. Este tema presenta muchas aplicaciones prácticas como el reconocimiento automático de códigos postales, montos en cheques bancarios, procesamiento automático de formularios, entre otros. Buenos resultados en estos tópicos se han logrado utilizando redes neuronales y máquinas de soporte vectorial, estas últimas serán utilizadas en el presente trabajo, y la performance de los clasificadores varía en dependencia de cuan bueno sea el preprocesamiento aplicado. La elección de 'buenas características' de las muestras es un desafío interesante, no sabiendo a priori cuales de ellas representan una caracterización eficiente de las clases a las cuales cada uno de los patrones pertenece, así como la reducción en la cantidad de valores utilizados para reducir tiempo de cálculos.

Así, por ejemplo, para dos dígitos de la misma clase podemos encontrar diferencias en el trazo, en la inclinación y en el tamaño. Hasta la actualidad no se ha presentado un modelo matemático capaz de cuantificar las variaciones entre los patrones [1]. Muchos modelos han sido presentados para lidiar con este problema pero ninguno se compara con el poder de clasificación que puede tener un ser humano.

La transformada wavelet ha probado ser una herramienta idónea para muchas aplicaciones relacionadas con el procesamiento de imágenes dando muy buenos resultados en algunas áreas como son por ejemplo el reconocimiento de bordes [6] y en la clasificación de texturas [7]. En [1] se utilizó, como método de preprocesamiento una transformada wavelet unidimensional, discreta y diádica aplicada al contorno previamente extraído de los dígitos. En [7] se utilizó como método de preprocesamiento, la aplicación de una transformada unidimensional multiwavelet discreta a los contornos de los dígitos previamente extraídos. La transformada wavelet discreta (DWT) provee una descomposición de la imagen en detalles a diferentes resolución y orientación; esto es, una biyección entre el espacio de la imagen y el espacio de sus coeficientes [8][9]. La misma ha sido ampliamente utilizada en la compresión de imágenes, pero presenta una desventaja: no es invariante frente a las traslaciones, pero el modo en que será utilizada, como se detallará a continuación (sin submuestreo), si resulta invariante frente a traslaciones.

En el presente trabajo utilizamos la teoría de wavelets para extraer características presentes en cada uno de los dígitos representantes de cada clase, y luego estamos utilizando máquinas de soporte vectorial para realizar la clasificación de los mismos. Nuestros experimentos fueron realizados sobre 2 bases de dígitos manuscritos ampliamente difundidas: CENPARMI y MNIST.

La aplicación de estas metodologías dará como resultado la generación de un vector (descriptor), que será la entrada de entrenamiento y clasificación para el sistema. Dicho descriptor contendrá dos secciones, ambas basadas en la transformada wavelet sin submuestreo. La primera consistirá en los coeficientes de aproximación y la segunda en características surgidas de los coeficientes de detalle.

En los últimos años el uso de máquinas de soporte vectorial [2][3], herramienta que utilizaremos en este trabajo, ha ganado mucho terreno debido a su buen rendimiento [4][5]. La eficacia, en cuanto a la buena clasificación, de un sistema de reconocimiento depende fuertemente de como se representa cada dígito, a través de características extraídas en la etapa de preprocesamiento.

Se trabajará con dos bases de datos de dígitos manuscritos: La del CENPARMI [11] (Centre for Pattern Recognition and Machine Intelligence at Concordia University, Canada) con dígitos de tamaño de 16x16 píxeles con valores binarios representando blancos y negros, y otra base conocida como es la MNIST [12] con dígitos de tamaño 28x28 píxeles y valores en escala de grises de 256 tonos. Se espera que el porcentaje de reconocimiento de los dígitos, con el preproceso basado en la transformada, sea notablemente superior al que se obtendrá sin ningún preprocesamiento y además que el mismo sea comparable al obtenido por otros autores trabajando con las mismas bases de datos pero con diferentes metodologías.

Además se harán pruebas utilizando técnicas basadas en la combinación de varios clasificadores [13] entrenados con parametrizaciones diferentes y en nuestro caso sobre distintos preprocesamientos de la entrada. Se espera que la utilización de esta técnica mejore la eficacia de reconocimiento y se analizará el beneficio obtenido en contraste con el tiempo de preprocesamiento y clasificación/votación necesitado para implementar estas técnicas.

Como paquetes de software para este análisis se utilizará el WaveLab [14] en MatLab para aplicar las transformadas y para realizar la generación de descriptores que nutrirán al clasificador basado en máquinas de soporte vectorial, y el Torch [15] que es una implementación de máquinas de soporte vectorial soportando distintas parametrizaciones. Este último será modificado para proveer datos a un software propietario que realizará la combinación de clasificadores en la fase final.

El trabajo se encuentra organizado de la siguiente manera: En el capítulo 2 se hará una breve reseña sobre teoría de wavelets y se introducirán las wavelets que se utilizaron en este trabajo. Son 3 muy conocidas en la literatura (Daubechies 9/7, Symmlet y Mallat). En el capítulo 3 se introduce al tema de las máquinas de soporte vectorial, su teoría y aplicaciones, así como los kernels utilizados durante los experimentos. En el capítulo 4 se describen las bases utilizadas. El capítulo 5 muestra los diferentes abordajes encarados para realizar el preprocesamiento de los patrones de entrenamiento y testeo. En el mismo se observan los distintos análisis basados en las características que podrían interesarnos como determinantes de cada una de las clases tratadas. En el capítulo 6 presentamos las diferentes formas de entrenar y clasificar con la teoría de máquinas de soporte vectorial que estamos utilizando, así como técnicas de combinación de estos clasificadores luego. En el capítulo 7 se presentan los resultados obtenidos como un resumen de lo realizado durante el presente trabajo y principalmente sobre lo descripto en los capítulos 5 y 6. El capítulo 8 presenta las conclusiones de los distintos experimentos realizados así como los posibles futuros enfoques y trabajos.

Capítulo 2 – Wavelets

Introducción

Las wavelets surgieron en los años 80. Son utilizadas en diversos campos de investigación y aplicadas para el análisis de señales gráficas, de audio, médicas, geológicas, económicas, entre otras. También son muy efectivas en el uso de técnicas de compresión, detección de singularidades, entre otras, presentando un amplio espectro de aplicabilidad.

Como hemos dicho estas funciones son ampliamente utilizadas para el análisis de señales. Cumplen ciertos requisitos: la integral de una wavelet es nula. Además generalmente tiene soporte compacto o decaimiento en el infinito, por esta razón tiene buena localización temporal. Se las ha llamado wavelets, onditas u ondelettes. La transformada wavelet es el producto escalar de la señal con dilataciones y traslaciones de una misma wavelet. Se obtiene información sobre los detalles de la señal en tiempos o lugares determinados, en varias escalas.

Las wavelets tienen buena localización frecuencial y excelente localización temporal o espacial, como hemos aclarado en el párrafo anterior, lo que nos posibilita el análisis de fragmentos de la señal en diferentes escalas dependiendo del detalle con que se requiera trabajar (escalas finas nos brindarán mayor detalle en altas frecuencias, y escalas más gruesas nos servirán perfectamente para zonas donde predominen las bajas frecuencias).

En resumen, para marcar el porqué del uso de wavelets para análisis/extracción de características de señales (en este caso imágenes), tenemos que las wavelets son localizadas en espacio/tiempo y dominios de escala/frecuencia, y de aquí el porqué, que estas funciones, pueden fácilmente detectar características 'locales' en una señal.

Definiciones previas

Sobremuestreo

Dada una señal S, la aplicación de un sobremuestreo a la misma, por un factor p, se realiza de la siguiente manera:

$$(S \uparrow p)_k = \begin{cases} S_r & si\ k = pr \\ 0 & en\ otro\ caso \end{cases}$$

Convolución

Dada una señal S $(S_n, 0 \leq n \leq N)$, y un filtro definido por h $(h_m, 0 \leq m \leq M)$, la aplicación del filtro a dicha señal en la forma particular que definimos a continuación, y se nota como $S * h$, es llamada convolución:

$$(S * h)_m = \sum_{i=0}^{M-1} S_{m-i} h_i$$

Usaremos el símbolo $*$ para representar este tipo de convolución ($S * h$).

Observación: En las wavelets biortogonales (Daubechies 9/7), en los cuales los filtros son simétricos o antisimétricos, se emplea una extensión simétrica de la señal para realizar la convolución.

Convolución circular

Dada una señal S $(S_n, 0 \leq n \leq N)$, y un filtro definido por h $(h_m, 0 \leq m \leq M)$, la aplicación del filtro a dicha señal en la forma particular que definimos a continuación es llamada convolución circular:

$$(S \circledast h)_n = \sum_{i=0}^{M-1} S_{(n-i) mod\ N}\, h_i$$

Usaremos el símbolo $\circledast$ para representar este tipo de convolución ($S \circledast h$).

Observación: En las wavelet ortogonales (Symmlet 8) y en la wavelet de Mallat, se emplea una extensión periódica de la señal para realizar la convolución (convolución circular).

Nota: la convolución circular genera la misma cantidad de coeficientes de salida.

Filtro reverso

Dado un filtro h, definimos su filtro reverso, notado como h' de la siguiente manera:

$$h'_k = h_{-k}$$

Diferentes clases de wavelets

Existen diferentes clases de wavelets con aplicaciones en diferentes dominios. Estas 'clases' determinarán las propiedades particulares con las que la wavelet debe contar y de aquí es que tenemos la generación de distintas funciones wavelet, con lo que podemos distinguir, generalizando, una división entre los siguientes grupos:

1. La transformada wavelet continua
2. La transformada wavelet discreta
3. La transformada wavelet sin submuestreo o 'à trous'

Describiremos el detalle de cada una de ellas y las propiedades con las que debe contar en las siguientes secciones.

La transformada wavelet continua

Una wavelet continua es una función $\psi \in L^2(\mathbb{R})$ tal que:

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \quad (1)$$

A esta función se la normaliza para tener $\| \psi \| = 1$, y también es centrada alrededor de $t = 0$. Las dos operaciones que se realizan sobre las mismas son dilatación y translación, y su aplicación sobre una wavelet ψ genera toda una familia de wavelets derivada de esta notada como:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi(\frac{t-u}{s}) \quad (2)$$

Los subíndices u y s indican que la wavelet generada por dicha especificación sobre la familia se encuentra trasladada y escalada en cada uno de estos dos subíndices respectivamente. La aplicación de estas operaciones sobre dicha función genera los diferentes espacios de detalle.

Dicha wavelet no tiene asociada una función de escala (generalmente notada como $\phi(x)$), que si veremos existe en el caso de estar trabajando con wavelets discretas.

A partir de estas la transformada wavelet continua sobre una función $f \in L^2(\mathbb{R})$ se define como:

$$Wf(u,s) = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right)dt \quad (3)$$

La transformada wavelet discreta

La transformada wavelet discreta surge de una discretización en el tiempo y en la escala. Habitualmente tienen filtros discretos y finitos (FIR – finite impulse response) que permiten calcular transformadas de señales discretas utilizando convoluciones de manera eficiente.

Dicha wavelet tiene asociada una función de escala notada como $\phi(x)$ y son las traslaciones de versiones dilatadas de esta función las que generan los espacios de aproximación.

Estas funciones presentan las siguientes características:

- Están basadas en el análisis multiresolución. La descomposición en wavelets permite el análisis de una señal a diferentes niveles de resolución (o escalas).
- Son suaves y pueden ser caracterizadas por la cantidad de momentos nulos que tengan. Una función tiene n momentos nulos si se cumple que

$$\int f(x)x^i dx = 0 \; \forall i = 0,..n-1 \tag{4}$$

Y mientras mayor la cantidad de momentos nulos, mejor será la aproximación de la señal utilizando una base de wavelets.

Además de estos puntos también contamos con un algoritmo de $O(n)$ que es estable para calcular la transformada wavelet discreta y su inversa.

Las mismas se pueden aplicar en forma recursiva a la señal analizada transformando los coeficientes que la componen a diferentes escalas para trabajar con aproximaciones y/o detalles según el tipo de análisis que se pretenda realizar.

El análisis de multiresolución

Si consideramos el espacio L^2, el espacio de funciones cuadráticas integrables en $\mathbb{R}$:

$$L^2 = \left\{f: \int_{-\infty}^{+\infty} f^2(x)dx < \infty\right\} \tag{5}$$

Dada una función $\phi(x)$, llamada función de escala, que cumple con la ecuación de dilatación o refinamiento:

$$\phi(x) = \sqrt{2}\sum_k h_k \phi(2x-k) \tag{6}$$

y a partir de las traslaciones enteras de $\phi(x)$ a diferentes escalas se generan los subespacios de aproximación definidos por:

$$V_j = \overline{gen\{\phi(2^J x-k)\}_k} \tag{7}$$

Como consecuencia de la ecuación (6) estos espacios se encuentran anidados de la siguiente manera:

$$\dots V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \dots \quad (8)$$

y se cumple que la clausura de la unión de los mismos es L^2, y su intersección contiene solo la función 0

$$\overline{\cup_{j=-\infty}^{+\infty} V_j} = L^2, \ \cap_{j=-\infty}^{+\infty}\{0\} \quad (9)$$

Para simplificar la notación es que definimos:

$$\phi_{j,k}(x) \equiv 2^{\frac{j}{2}}\phi(2^j x - k) \quad (10)$$

Dada otra función $\psi(x)$, llamada wavelet, que es ortogonal a $\phi(x)$, y que cumple con:

$$\psi(x) = \sqrt{2}\sum_k g_k \phi(2x - k) \quad (11)$$

y a partir de las traslaciones enteras de $\psi(x)$ a diferentes escalas se generan los subespacios de detalle definidos por:

$$W_j = \overline{gen\{\psi(2^j x - k)\}_k}$$

Como consecuencia de la ecuación (11) tenemos la siguiente relación entre W_j y V_{j+1}:

$$W_j \subset V_{j+1}$$

Ahora podemos tomar la descomposición de V_{j+1} en V_j y W_j (porque $V_j \subset V_{j+1}$), y tenemos entonces que W_j es el complemento ortogonal de V_j en V_{j+1}:

$$V_j \oplus W_j = V_{j+1} \quad (12)$$
$$V_j \perp W_j \quad (13)$$

Teniendo entonces que la suma directa de los subespacios W_j es igual a L^2:

$$\overline{\cup_{-\infty}^{+\infty} V_j} = \overline{\bigoplus_{-\infty}^{+\infty} W_j} = L^2 \quad (14)$$

Para simplificar la notación tenemos:

$$\psi_{j,k}(x) \equiv 2^{\frac{j}{2}}\psi(2^j x - k) \quad (15)$$

Vemos, a partir del análisis mostrado que, entre los subespacios de aproximación, V_j es de menor resolución que V_{j+1}, mientras que los subespacios de detalle W_j capturan los detalles que están en V_{j+1} y no están en V_j.

Diremos entonces que el análisis de multiresolución consiste en la generación de una secuencia de espacios anidados definidos por las dilataciones y traslaciones de la función wavelet y su función de escala asociada. Estas dilataciones y translaciones son las que generan las diferentes resoluciones, y las bases, de cada uno de los diferentes espacios respectivamente [16]. Y dicho análisis es la proyección de una señal de entrada en los distintos espacios de detalle a diferentes resoluciones. La señal se expresa como suma de detalles a diferentes escalas, más una aproximación burda de la señal.

La transformada wavelet discreta rápida (FWT)

De acuerdo al análisis anterior tenemos que:

Los espacios V son generados por la función de escala siendo dilatada y trasladada sucesivamente, y el hecho de que $V_0 \subset V_1$ nos indica que podemos escribir a la función de escala que es base de V_0 como una combinación de las bases en el siguiente espacio (ecuación 6).

Lo mismo ocurre con los subespacios W que son generados por la wavelet también a través de dilataciones y traslaciones sucesivas, y al suceder lo mismo con $W_0 \subset V_1$ nos permite escribir la wavelet (base de W_0) como una combinación de las bases del espacio siguiente (ecuación 11).

De acuerdo al análisis multiresolución realizado, podemos decir que estas relaciones son también válidas para V_{j+1}, V_j y W_j con un j arbitrario, esto sería expresado en fórmulas de la siguiente manera:

$$\phi_{j,0}(x) = \sum_k h_k \phi_{j+1,k}\,(x) \quad (16)$$

$$\psi_{j,0}(x) = \sum_k g_k \phi_{j+1,k}(x) \quad (17)$$

Los coeficientes utilizados como factores de las bases en el siguiente espacio, en la combinación de bases que define el espacio previo, son las componentes del vector que generan los dos filtros para la transformada (los llamaremos h y g), y tendremos luego también dos filtros para la antitransformada ($\tilde{h}$ y $\tilde{g}$). Los h son los filtros 'pasa bajos' y los g los filtros 'pasa altos'.

Como se cumple que $V_{j+1} = V_j \oplus W_j$, entonces podemos expresar una función $f(x)$ que es escrita en términos de una base de funciones de V_{j+1} en términos de una base de funciones de V_j y W_j. Para ejemplificar esto consideremos que si llamamos $c^{(0)}$ a la señal original, y $f_0(x)$ es su función asociada, entonces $f_0 \in V_0$; luego si descomponemos a $f_0(x)$ en la suma de sus proyecciones sobre V_{-1} y W_{-1} se tiene lo siguiente:

$$f_0(x) = \sum_k c_k^{(0)} \phi(x-k)$$

$$= f_{-1}(x) + r_{-1}(x)$$

$$= \sum_k c_k^{(-1)} \frac{1}{\sqrt{2}} \phi(2^{-1}x - k) + \sum_k d_k^{(-1)} \frac{1}{\sqrt{2}} \psi(2^{-1}x - k) \quad (18)$$

Donde la función $f_{-1}(x) \in V_{-1}$ es una versión más 'suave' de la función original, tiene menor resolución y sus coeficientes $c_k^{(-1)}$ son llamados coeficientes de aproximación, y la función $r_{-1}(x) \in W_{-1}$ contiene los detalles de la función original que no se encuentran en $f_{-1}(x)$, y sus coeficientes $d_k^{(-1)}$ son llamados coeficientes de detalle.

Esto constituye un paso de la transformada wavelet que se realiza como se encuentra detallado en el siguiente esquema (figura 1):

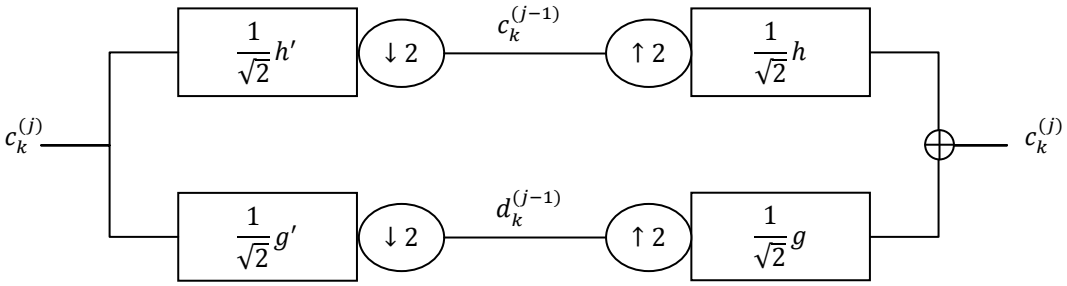


Figura 1. Esquema de análisis-síntesis para 1 dimensión

Si generalizamos lo explicado anteriormente, mediante una aplicación recursiva del método, tendremos la descomposición de la señal original en tantos niveles como queramos de la siguiente manera:

$$f = f_0 = f_{-1} + r_{-1} = f_{-2} + r_{-2} + r_{-1} = \cdots = f_{-L} + \sum_{j=-L+1}^{-1} r_j,$$

$$f(x) = \sum_{k \in Z} c_k^{(-L)} \phi_{-L,k}(x) + \sum_{j=-L}^{-1} \sum_{k \in Z} d_k^{(j)} \psi_{j,k}(x), \quad (19)$$

donde las funciones ϕ y ψ son las definidas en las ecuaciones (6) y (15) respectivamente.

Y podemos ver que, a partir del esquema y la aplicación de cada paso, los coeficientes de la transformada $c_k^{(j)}$ y $d_k^{(j)}$ están definidos por:

$$c_k^{(j)} = \frac{1}{\sqrt{2}} \sum_i h_{i-2k} c_i^{(j+1)} \quad (20)$$

$$d_k^{(j)} = \frac{1}{\sqrt{2}} \sum_i g_{i-2k} c_i^{(j+1)} \quad (21)$$

Las ecuaciones (20) y (21) presentas aquí son convoluciones seguidas de submuestreos, y radican básicamente en tomar como entrada una señal y aplicarle el filtro correspondiente en sentido inverso (lo que se puede observar en las mismas para cada uno de los subíndices k de la señal original). La transformada wavelet inversa se puede obtener según el esquema de la figura 1. Este algoritmo es conocido como el algoritmo de banco de filtros.

Procesamiento de imágenes

La aplicación más simple para procesar imágenes de la teoría wavelet, para el caso discreto presentado en la sección anterior, se basa en aplicar un paso de la transformada wavelet unidimensional tomando las señales que representan las filas de la imagen, y luego tomar cada una de las columnas como señal a procesar. Esto da origen a las siguientes funciones de escala y wavelets:

- $\phi(x)\phi(y)$: extraerá la representación burda o aproximada de la imagen
- $\psi(x)\phi(y)$: extraerá los detalles verticales de la imagen
- $\phi(x)\psi(y)$: extraerá los detalles horizontales de la imagen
- $\psi(x)\psi(y)$: extraerá los detalles diagonales de la imagen

La relación de los subespacios generados a partir de la aplicación de esta manera de la transformada wavelet a una imagen determinada es la siguiente:

$$\begin{aligned} V_0 = V_0^{\{x\}} \otimes V_0^{\{y\}} &= gen\{\phi(x-i)\phi(y-j)\} \\ &= \left[V_{-1}^{\{x\}} \oplus W_{-1}^{\{x\}}\right] \otimes \left[V_{-1}^{\{y\}} \oplus W_{-1}^{\{y\}}\right] \\ &= V_{-1} \oplus W_{-1} \end{aligned}$$

en donde

$$V_{-1} = \left[V_{-1}^{\{x\}} \otimes V_{-1}^{\{y\}}\right]$$

$$W_{-1} = \left[\left\{V_{-1}^{\{x\}} \otimes W_{-1}^{\{y\}}\right\} \oplus \left\{W_{-1}^{\{x\}} \otimes V_{-1}^{\{y\}}\right\} \oplus \left\{W_{-1}^{\{x\}} \otimes W_{-1}^{\{y\}}\right\}\right]$$

Ahora, si tenemos la imagen original I de NxN, y la función asociada $f(x,y) \in V_0$, y tomamos como señales horizontales las determinadas por las filas a través de la coordenada y de la matriz subyacente, y las señales verticales las determinadas por las columnas definidas a través de la coordenada x, entonces, las fórmulas asociadas para la aplicación de un paso de la transformada wavelet de esta manera son las siguientes:

$$f(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I_{ij}\phi(x-i)\phi(y-j)$$

$$= \sum_{i=0}^{N-1} \left\{ \sum_{j=0}^{N-1} I_{ij}\, \phi(x-j) \right\} \phi(x-i)$$

Entonces, cada fila de la imagen es tomada como una señal para aplicar la transformada wavelet unidimensional. Cada una de estas filas están determinadas por el índice i de la imagen y definida por I_{i*}. De la aplicación de esta transformada se obtienen los coeficientes de aproximación $c_{i,*}^{(-1)}$ y los coeficientes de detalle $d_{i,*}^{(-1)}$ que surgen de la misma.

Si escribimos la sumatoria que tenemos entre llaves en la expresión anterior, que se encuentra definida en $V_0^{(y)}$, en las bases correspondientes al siguiente subespacio $V_{-1}^{(y)}$ y $W_{-1}^{(y)}$ tenemos lo siguiente:

$$f(x,y) = \sum_{i=0}^{N-1} \left\{ \sum_{j=0}^{\frac{N}{2}-1} c_{i,j}^{(-1)} \phi_j^{(-1)}(y) + \sum_{j=0}^{\frac{N}{2}-1} d_{i,j}^{(-1)} \psi_j^{(-1)}(y) \right\} \phi(x-i)$$

$$= \sum_{j=0}^{\frac{N}{2}-1} \left\{ \sum_{i=0}^{N-1} c_{i,j}^{(-1)} \phi(x-i) \right\} \phi_j^{(-1)}(y) + \sum_{j=0}^{\frac{N}{2}-1} \left\{ \sum_{i=0}^{N-1} d_{i,j}^{(-1)} \phi(x-i) \right\} \psi_j^{(-1)}(y)$$

Por último operamos fijando el índice j y se aplica la transformada wavelet a los coeficientes $c_{*,j}^{(-1)}$ y $d_{*,j}^{(-1)}$. Entonces es que podemos reescribir los términos entre llaves en la fórmula anterior, que pertenecen a $V_0^{(x)}$, en las bases correspondientes a los subespacios $V_{-1}^{(x)}$ y $W_{-1}^{(x)}$:

$$f(x,y) = \sum_{j=0}^{\frac{N}{2}-1} \left\{ \sum_{i=0}^{\frac{N}{2}-1} LL_{i,j} \phi_i^{(-1)}(x) + \sum_{j=0}^{\frac{N}{2}-1} LH_{i,j} \psi_i^{(-1)}(x) \right\} \phi_j^{(-1)}(y)$$

$$+ \sum_{j=0}^{\frac{N}{2}-1} \left\{ \sum_{i=0}^{\frac{N}{2}-1} HL_{i,j} \phi_i^{(-1)}(x) + \sum_{j=0}^{\frac{N}{2}-1} HH_{i,j} \psi_i^{(-1)}(x) \right\} \psi_j^{(-1)}(y)$$

dándonos por resultado la siguiente ecuación:

$$f(x,y) = \sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{\frac{N}{2}-1} LL_{i,j}\phi_i^{(-1)}(x)\,\phi_j^{(-1)}(y) + \sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{\frac{N}{2}-1} LH_{i,j}\psi_i^{(-1)}(x)\,\phi_j^{(-1)}(y)$$
$$+ \sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{\frac{N}{2}-1} HL_{i,j}\phi_i^{(-1)}(x)\,\psi_j^{(-1)}(y) + \sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{\frac{N}{2}-1} HH_{i,j}\psi_i^{(-1)}(x)\,\psi_j^{(-1)}(y)$$

La transformada wavelet sin submuestreo o 'à trous'

Como mencionamos anteriormente, en la transformada discreta se calculan las escalas que son potencias de dos, de aquí que sea llamada diádica. El uso de estas escalas presenta básicamente estas ventajas:

- Las escalas sucesivas son obtenidas aplicando un algoritmo recursivo sobre la escala anterior por medio de convoluciones discretas.
- Las transformadas sucesivas no pierden ni descartan información y la señal original puede reconstruirse a partir de los datos retenidos (también el algoritmo de reconstrucción es muy similar al de descomposición).

La transformada 'à trous' es una transformada discreta que presenta redundancia de información, pero es mejor para el análisis de singularidades ya que es invariante frente a translaciones (la salida es desplazada proporcionalmente al desplazamiento aplicado en la entrada).

Para aplicar este tipo de transformada se sobremuestrean los filtros utilizados mediante la introducción de ceros (técnica que le dio el nombre al método 'à trous' que significa 'con agujeros' en francés). El filtro es sobremuestreado con factor 2, y luego la transformada y antitransformada 'à trous' de una señal c_0 (de N componentes con $N = 2^k$) se define de la siguiente manera (figura 2):

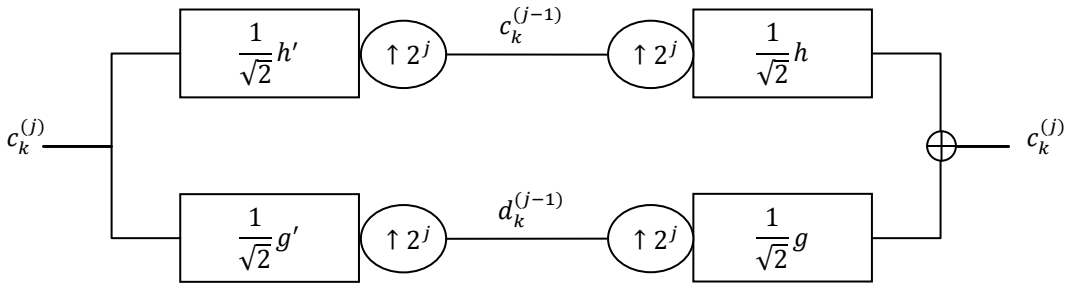


Figura 2. Aplicación de la transformada wavelet 'à trous' de 1 nivel y reconstrucción mediante la antitransformada

De esta manera la transformada wavelet 'à trous' de $c_k^{(0)}$, una señal original de tamaño N, se define como la aproximación en la escala J de la misma, más el conjunto de coeficientes de detalle $\{d_k^{(j)}\}$ hasta la escala 2^J:

$$\left[c_k^{-J}, \{d_k^i\}_{-1 \le i \le -J}\right] \qquad \text{con} \quad J = \log_2 N$$

Donde:

$$c_k^{(j-1)} = c_k^{(j)} \circledast (h' \uparrow 2^j)$$
$$d_k^{(j-1)} = c_k^{(j)} \circledast (g' \uparrow 2^j)$$

Luego el resultado es centrado de manera tal que la morfología de la señal sea lo más parecida entre una escala y la siguiente [6] (que no se presenten desplazamientos que son intrínsecos del proceso de filtrado).

La recuperación de la señal original en este tipo de transformada se realiza de la siguiente manera:

$$c_k^{(j)} = \frac{1}{2}(c_k^{(j-1)} \circledast \left(h \uparrow 2^j\right) + d_k^{(j-1)} \circledast (g \uparrow 2^j))$$

Para que esta valga se debe cumplir que:

$$h * h' + g * g' = 2\delta_0$$

Siendo δ_0 la delta de Kroneckler.

La aplicación de esta transformada a imágenes es teóricamente similar al caso presentado para la transformada discreta pero, utilizando las convoluciones circulares, los tamaños de las matrices resultantes tendrán el mismo tamaño, en cada cambio de escala, que la matriz de la cual se parte, a diferencias del caso discreto presentado anteriormente donde el submuestreo hace que cada una de las matrices obtenidas sean ¼ del tamaño de la matriz origen. En el presente trabajo no estaremos utilizando la información que reside en la matriz diagonal (al aplicar la transformada en dos dimensiones).

En el primer paso, luego de aplicar la transformada, se obtiene una matriz de coeficientes de aproximación y otras dos de coeficientes de detalle fino (vertical y horizontal, no es utilizada la matriz diagonal). En el segundo paso se opera sobre los coeficientes de aproximación del primer paso, y se obtiene una matriz de coeficientes de aproximación más borrosa y otras dos de coeficientes de detalle menos fino (vertical y horizontal). Y así se procede sucesivamente descomponiendo la imagen original en estas sucesivas resoluciones.

Wavelets utilizadas en este trabajo

Daubechies 9/7

Esta wavelet fue introducida por Cohen/Daubechies/Feauveau, también conocida como CDF 9/7. Pertenece históricamente a la primera familia de wavelets biortogonales, que se hicieron populares por medio de Ingrid Daubechies (de ahí su nombre). Las wavelets pertenecientes a esta familia no son las mismas que las wavelets Daubechies ortogonales, y tampoco son similares ni en forma ni en propiedades, pero de todos modos la idea de su construcción es similar.

El estándar de compresión JPEG 2000 utiliza una wavelet que pertenece a la familia que esta misma pertenece para aplicar compresión sin pérdida y es la definida por los filtros 5/3 conocida como CDF 5/3 (también llamada LeGall 5/3), y aplica esta wavelet, la CDF 9/7 para compresión con pérdida.

A continuación presentamos las funciones de escala y wavelet, más los filtros asociados a esta wavelet:

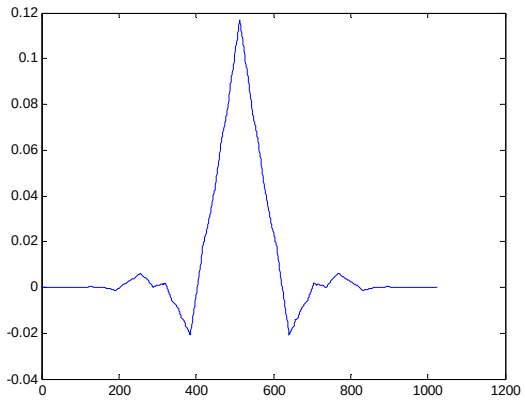


Figura 3. Función de escala de la Daubechies 9/7

Filtro asociado:
[0.0378 -0.0238 -0.1106 0.3774 0.8527
0.3774 -0.1106 -0.0238 0.0378]

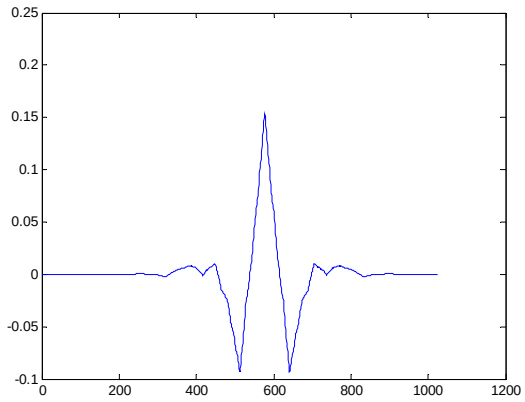


Figura 4. Función wavelet de la Daubechies 9/7

Filtro asociado:
[0 0.0645 -0.0407 -0.4181 0.7885
-0.4181 -0.0407 0.0645 0]

Symmlet

Esta wavelet tiene 8 momentos nulos, y soporte mínimo; es ortogonal y aproximadamente simétrica (lo podemos observar en las figuras 5 y 6).

A continuación presentamos las funciones de escala y wavelet, más los filtros asociados a esta wavelet:

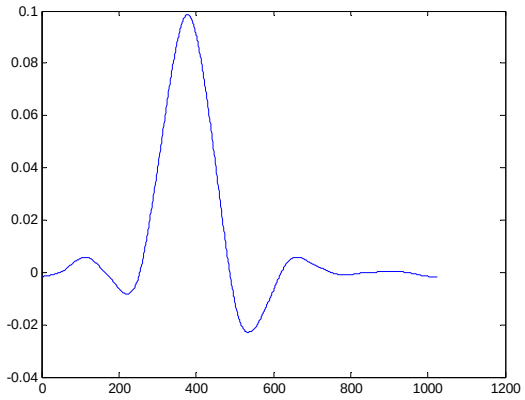


Figura 5. Función de escala de la Symmlet 8

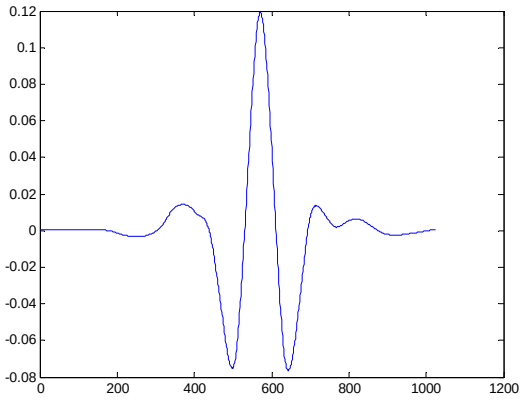


Figura 6. Función wavelet de la Symmlet 8

Filtro asociado:

[-0.0034 -0.0005 0.0317 0.0076 -0.1433 -0.0613
0.4814 0.7772 0.3644 -0.0519 -0.0272 0.0491
0.0038 -0.0150 -0.0003 0.0019]

Filtro asociado:

[-0.0019 -0.0003 0.0150 0.0038 -0.0491 -0.0272
0.0519 0.3644 -0.7772 0.4814 0.0613 -0.1433
-0.0076 0.0317 0.0005 -0.0034]

Mallat

A continuación presentamos las funciones de escala y wavelet, más los filtros asociados a esta wavelet:

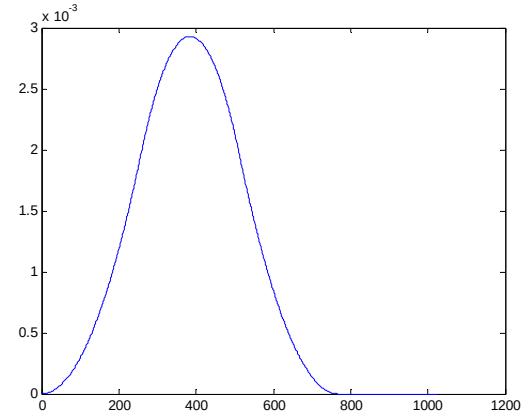


Figura 7. Función de escala de Mallat

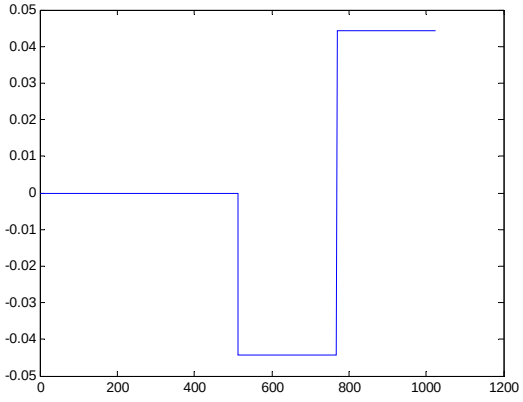


Figura 8. Función wavelet de Mallat

Filtro asociado:

[0.1768 0.5303 0.5303 0.1768]

Filtro asociado:

[0.7071 -0.7071]

Capítulo 3 – Máquinas de soporte vectorial

Introducción

Las máquinas de soporte vectorial son un método relativamente nuevo para clasificación binaria. La idea básica radica en encontrar un hiperplano que separa los datos perfectamente en dos clases. Y luego la clasificación consiste en observar de qué lado del hiperplano queda el vector que representa una nueva muestra no conocida previamente.

En casos donde las clases no son separables, se introducen cotas de error para permitir que muestras de entrenamiento queden identificadas como pertenecientes a la clase incorrecta, y en muchos casos para poder realizar la clasificación de muestras que no son linealmente separables se introduce la utilización de 'kernels' con la idea de transformar el conjunto de datos a un espacio de dimensión superior donde este si es perfectamente separable, o bajo una cota de error aceptable. Este tipo de transformación normalmente podría traer complejidades computacionales difíciles de resolver, pero la clave en la utilización de este tipo de metodologías radica en que el espacio nuevo de dimensión no tiene que ser directamente tratado sino que solo es necesario conocer cómo resolver el producto escalar en el mismo.

Se introduce además un concepto llamado la dimensión VC. Esta es una medida como para tener una idea del buen comportamiento del sistema con nuevos conjuntos de datos de la máquina vectorial, y puede ser explícitamente calculado, no como otros métodos del estilo redes neuronales donde no se encuentran estas nociones. En general las máquinas de soporte vectorial son muy intuitivas, poseen buen fundamento teórico y han demostrado en la práctica su buen desempeño.

Un poco de historia

Las máquinas de soporte vectorial fueron introducidas a finales de los 70 (Vapnik, 1979 [5]) y en los últimos años se produjo un creciente aumento de atención. Las mismas se utilizan para aplicar aprendizaje estadísticos sobre muestras pudiendo realizar clasificación y/o regresión sobre el conjunto tratado. En la mayoría de los casos la generalización de performance (es decir las tasas de error en muestras de pruebas) es equivalente o significativamente mejor a métodos similares, pero pueden ser bastante lentas en la fase de pruebas.

Introducción a las cotas de riesgo

Supongamos que tenemos un conjunto de *l* muestras donde cada una de ellas consiste en un par conteniendo un vector x_i (de dimensión $\mathbb{R}^n$) y un valor de verdad y_i que representa un positivo o negativo sobre la observación tratada (por ejemplo en nuestro caso x_i puede ser un vector de los píxeles de una imagen de un dígito e y_i indicar que corresponde a cierta clase, como podría ser '2', con un 1 o -1 en caso contrario). Asumiendo que existe cierta distribución de probabilidad no conocida $P(x, y)$ (P representa la función de probabilidad acumulada y p su densidad) de donde la data fue obtenida (independiente e idénticamente distribuida).

Si tenemos una máquina cuya tarea es aprender la relación entre x_i e y_i (para todos los elementos de la muestra) esta estará definida por un conjunto de mapeos posibles del tipo $x \rightarrow f(x, \alpha)$. Estas funciones f son determinísticas y definidas por ciertos parámetros α, donde una determinada elección de parámetros, en forma adecuada, definirá lo que llamaremos una 'máquina entrenada'.

En dicha formulación el error de prueba esperado estará definido por:

$$R(\alpha) = \int \frac{1}{2}|y - f(x, \alpha)|dP(x, y)$$

Nota: cuando la densidad p existe se puede escribir $dP(x, y)$ como $p(x, y)dxdy$, pero al menos que tengamos un estimado de $P(x, y)$ esto no tiene sentido práctico.

$R(\alpha)$ es el riesgo esperado o simplemente el riesgo que llamaremos 'riesgo actual' para indicar que es la cantidad en la que estamos interesados en acotar.

El 'riesgo empírico' ($R_{emp}(\alpha)$) estará definido como la tasa de error promedio en el conjunto de entrenamiento utilizado:

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i, \alpha)|$$

Nota: acá no aparece ninguna distribución de probabilidad, el resultado está determinado por los parámetros α utilizados y el conjunto de muestras de entrenamiento.

El valor $\frac{1}{2}|y_i - f(x_i, \alpha)|$ es llamado la *pérdida* y para el caso definido acá solo puede tomar valores 0 y 1. Ahora si tomamos un n en el intervalo [0, 1], para pérdidas, tomando esos valores con probabilidad $1 - n$ tenemos la siguiente cota (Vapnik, 1995 [5]):

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log\left(\frac{2l}{h}\right) + 1) - \log(\frac{n}{4})}{l}}$$

Donde la parte derecha de la formula presentada es la 'cota de riesgo' (por más detalle ver [5]) y de la misma podemos observar o tener las siguientes propiedades:

1) Es independiente de $P(x, y)$
2) Usualmente no es posible calcular la parte izquierda ('riesgo actual')
3) Conociendo h (la dimensión VC) podemos calcular fácilmente la cota de riesgo

Las mismas nos servirán para determinar entre una familia de máquinas la que mejor se comporte.

Dimensión VC

La dimensión VC es una propiedad sobre un conjunto de funciones $\{f(\alpha)\}$ que llevada particularmente al caso de reconocimiento de dos clases tal que $f(\alpha)$ es 1 o -1, se define por el máximo número de elementos de la muestra tomada, tal que teniendo una cantidad l de los mismos, éstos pueden ser etiquetados de 2^l maneras y para cada una de éstas un miembro del conjunto $\{f(\alpha)\}$ puede hacer correctamente la asignación de ese mapeo.

Por ejemplo en $\mathbb{R}^2$ y utilizando un conjunto de funciones $\{f(\alpha)\}$ definidos por líneas orientadas, la dimensión VC es 3 ya que no más puntos pueden ser 'separados' en distintas clases (de todas las posibles maneras) por dichas líneas:

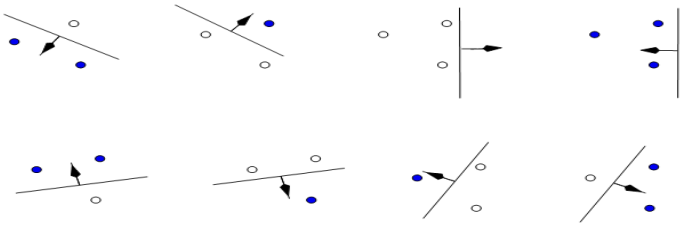


Figura 9. Se puede separar en 8 clases 3 puntos, pero no 4.

Para el caso particular que tratamos y la formulación de las máquinas de soporte vectorial donde tenemos un hiperplano que será quien separe las muestras tenemos los siguientes teoremas y corolarios:

Teorema1. Consideremos un conjunto de m puntos en $\mathbb{R}^n$. Tomemos alguno de ellos como origen. Entonces los m puntos pueden ser separados por un hiperplano orientado si y solo si los vectores de posición de los puntos remanentes son linealmente independientes [5].

Colorario1. La dimensión VC de un conjunto de hiperplanos orientados en $\mathbb{R}^n$ es $n + 1$, ya que siempre podemos elegir $n + 1$ vectores tal que tomando uno de ellos como origen los n restantes son linealmente independientes, pero esto no sucede con $n + 2$ vectores [5].

Definición de la máquina de soporte vectorial

Ahora trataremos la definición y armado de la máquina de soporte vectorial para el caso más simple donde la muestra es separable (el análisis para el caso general de máquinas no lineales en muestras no separables es similar). Suponiendo que tenemos un conjunto de datos de entrenamiento $\{x_i, y_i\}$ con $i = 1, \dots l$, $y_i \in \{-1,1\}$ y $x_i \in \mathbb{R}^d$, y que tenemos un hiperplano que separa los ejemplos negativos de los positivos. Los puntos x que caen en dicho hiperplano satisfacen la siguiente condición $w \cdot x + b = 0$, donde w es la normal al hiperplano, $|b|/\| w \|$ es la distancia perpendicular del hiperplano al origen y $\| w \|$ es la norma euclidiana de w. Llamaremos a d_+ (d_-) como la distancia más corta que separa al hiperplano del más cercano valor positivo (negativo) entre los ejemplos. Definimos al 'margen' del hiperplano que separa las clases al valor arrojado por $d_+ + d_-$. Para el caso linealmente separable el algoritmo de soporte vectorial simplemente busca el hiperplano que separe con mayor margen, esto puede ser formulado como sigue:

$$x_i \cdot w + b \geq +1 \qquad \text{para } y_i = +1 \tag{22}$$

$$x_i \cdot w + b \leq -1 \qquad \text{para } y_i = -1 \tag{23}$$

Que pueden ser combinadas en la siguiente:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall i \tag{24}$$

Ahora si consideramos los puntos para los cuales la igualdad de la ecuación (22) se cumple. Dichos puntos pertenecen al hiperplano H_1, determinado por $x_i \cdot w + b = 1$ con normal w y distancia perpendicular al origen determinada por $|1 - b|/\| w \|$. En forma similar para los puntos que cumplen con la igualdad de la ecuación (23) pertenecen al hiperplano H_2 que está determinado por $x_i \cdot w + b = -1$ otra vez con normal w y distancia perpendicular al origen, esta vez, determinada por $|-1 - b|/\| w \|$. De esto obtenemos que $d_+ = d_- = 1/\| w \|$ y el margen es simplemente $2/\| w \|$ (nótese que H_1 y H_2 son paralelos y no existen puntos entre ellos). Entonces podemos encontrar este par de hiperplanos que determinan el máximo margen minimizando $\| w \|^2$ sujeto a respetar la condición de la ecuación (24), y esperamos entonces que la solución a este caso típico de dos dimensiones sea como mostramos en la siguiente figura (10):

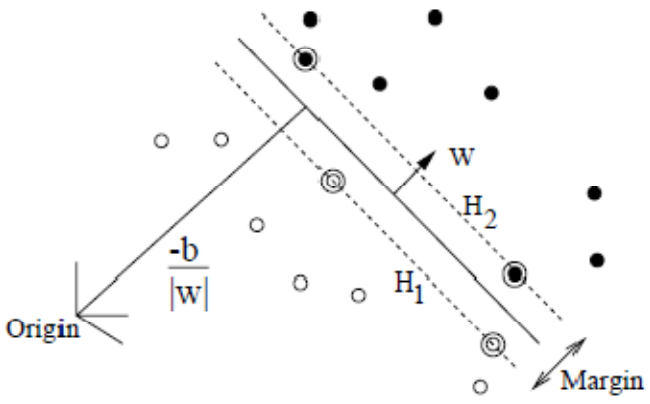


Figura 10. Hiperplanos lineales para el caso separable (marcados con un círculo se encuentran los vectores de soporte)

Todos aquellos puntos para los cuales la igualdad de la condición de la ecuación (24) se cumple serán llamados 'vectores de soporte' y si se los removiera la solución cambiaría.

La formulación lagrangiana del problema nos genera las dos siguientes razones por las cuales el manejo de este problema se nos facilita: la primera es que la condición de la ecuación (24) será reescrita como condicionantes sobre los multiplicadores lagrangianos y la segunda es que los datos (ya sean de entrenamiento o testeo) aparecerán como productos escalares en los algoritmos de resolución. Esta última propiedad es crucial y será la que permita luego generalizar este procedimiento para el caso no lineal mediante la utilización de diferentes 'kernels'. Por consiguiente introduciremos multiplicadores lagrangianos positivos α_i con $i = 1, \dots l$, uno por cada uno de las condiciones de desigualdad. Considerando que la regla para el uso de condicionantes de la forma $c_i \geq 0$, las ecuaciones condicionantes son multiplicadas por multiplicadores lagrangianos positivos y sustraídos de la función objetiva para formar el lagrangiano, lo que resulta en la siguiente formulación:

$$L_P \equiv \frac{1}{2} \| w \|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i \quad (25)$$

Ahora debemos minimizar L_P con respecto a w, b, y simultáneamente se requiere que las derivadas de L_P con respecto a todos los α_i desaparezcan, todo sujeto a las condiciones que $\alpha_i \geq 0$ (llamaremos a este conjunto de condicionantes C_1). Este es un problema cuadrático convexo dado que la función objetiva es convexa y los puntos que satisfacen las condiciones es un conjunto convexo (cualquier condición lineal define un conjunto convexo, y las N condiciones lineales simultáneas definen la intersección de N conjuntos convexos que es convexo). Esto significa que podemos resolver el siguiente problema dual: maximizar L_P, sujeto a las condiciones de que el gradiente de L_P con respecto a w y b desaparezca, y también sujeto a las condiciones que $\alpha_i \geq 0$ (llamaremos a este conjunto de condicionantes C_2). Esta formulación particular del problema es conocida como el "Wolfe dual" (Fletcher, 1987) y tiene la propiedad de que el máximo L_P, sujeto a las condiciones C_2, ocurre sobre los mismos valores de w, b y α, como el mínimo L_P, sujeto a las condiciones C_1.

Que el gradiente de L_P con respecto a w y b desaparezca nos da las condiciones:

$$w = \sum_i \alpha_i y_i x_i \quad (26)$$
$$\sum_i \alpha_i y_i = 0 \quad (27)$$

Como estas son condicionantes de igualdad en la formulación dual, podemos sustituirlas en la ecuación (25):

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \quad (28)$$

Se le dió al lagrangiano diferentes letras como subíndices ($_P$ primal, $_D$ dual) para hacer énfasis es que sus formulaciones son diferentes, si bien L_D y L_P surgen de la misma función objetiva tienen diferentes condicionantes, y la solución será encontrada ya sea minimizando L_P o maximizando L_D.

Entonces el armado de la máquina de soporte vectorial (para el caso linealmente separable) se obtiene maximizando L_D con respecto a los α_i, sujeto a las condiciones dadas por la ecuación (27) y $\alpha_i \geq 0$, con la solución dada por la ecuación (26). Nótese que hay un multiplicador lagrangiano por cada

punto de entrenamiento y, en la solución, aquellos puntos para los cuales $\alpha_i \geq 0$ son los que llamaremos vectores de soporte y pertenecen a los hiperplanos H_1 y H_2; estos últimos vectores son los elementos críticos en el armado de las máquinas de soporte vectorial, y yacen lo mas cercanamente posible al límite de decisión, tanto que si los otros puntos restantes fueran removidos, y el entrenamiento se repitiera, la máquina de soporte vectorial sería entrenada de la misma manera.

Análisis del caso no linealmente separable

En el caso no linealmente separable no encontraremos una solución factible utilizando todo lo anterior [5] (se evidencia a partir de la función objetiva [en el lagrangiano dual] quien crecerá arbitrariamente). Entonces para extender estas ideas a este caso tendremos que relajar las condiciones de las ecuaciones (22) y (23), pero solamente cuando fuera necesario, con la introducción de un 'costo' (un incremento en la función objetiva primal) para lograr esto. Esto puede ser realizado con la introducción de variables positivas $\xi_i, i = 1, .. l$ en las condiciones con lo que nos quedaría:

$$x_i \cdot w + b \geq +1 - \xi_i \qquad \text{para } y_i = +1 \qquad (29)$$

$$x_i \cdot w + b \leq -1 + \xi_i \qquad \text{para } y_i = -1 \qquad (30)$$

$$\xi_i \geq 0 \ \ \forall i \qquad (31)$$

Donde, para que ocurra un error, el correspondiente ξ_i debe exceder la unidad, entonces $\sum_i \xi_i$ es una cota superior en el numero de errores de entrenamiento. Luego una manera natural de asignar un costo extra para errores es cambiando la función objetiva para ser minimizada desde $\frac{\|w\|^2}{2}$ hasta $\frac{\|w\|^2}{2} + C(\sum_i \xi_i)^k$, donde C es un parámetro a ser elegido por el usuario, y un C grande correspondería a asignar una alta penalidad a errores. Tal como está, este es un problema de programación convexo que para cada entero positivo $k \in \{1,2\}$ es también un problema de programación cuadrático, y la elección de $k = 1$ presenta la mayor ventaja de que ninguno de los ξ_i, tampoco los multiplicadores de Lagrange, aparecen en el problema "Wolfe dual", que se transforma en:

Maximizar:

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \qquad (32)$$

Sujeto a:

$$0 \leq \alpha_i \leq C, \qquad (33)$$

$$\sum_i \alpha_i y_i = 0 \qquad (34)$$

La solución está dada otra vez por:

$$w = \sum_{i=1}^{N_S} \alpha_i y_i x_i \qquad (35)$$

donde N_S es el número de vectores de soporte. Entonces la única diferencia contra el hiperplano óptimo es que ahora los α_i están acotados superiormente por C. La idea queda presentada en la siguiente figura (11):

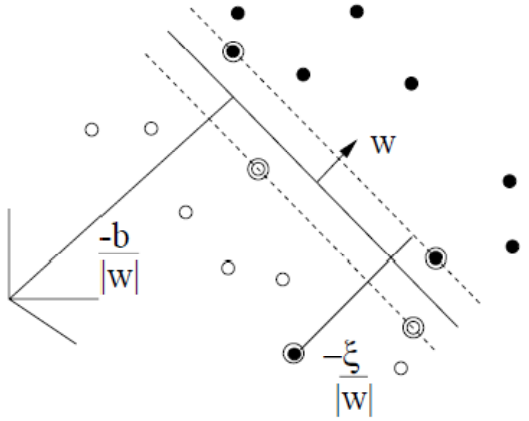


Figura 11. Hiperplanos lineales para el caso no separable

Vamos a necesitar las condiciones de Karush-Kuhn-Tucker [5] para el problema primal. El lagrangiano primal es el siguiente:

$$L_P = \frac{1}{2} \| w \|^2 + C \sum_i \xi_i - \sum_i \alpha_i \{y_i (x_i \cdot w + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i \quad (36)$$

donde los μ_i son los multiplicadores de Lagrange introducidos para reforzar la positividad de los ξ_i. Las condiciones KKT para el problema primal son entonces (donde i va desde 1 hasta el número de puntos de entrenamiento, y v desde 1 hasta la dimensión de los datos)

$$\frac{\partial L_P}{\partial w_v} = w_v - \sum_i \alpha_i y_i x_{iv} = 0 \quad (37)$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i = 0 \quad (38)$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad (39)$$

$$y_i (x_i \cdot w + b) - 1 + \xi_i \geq 0 \quad (40)$$

$$\xi_i \geq 0, \alpha_i \geq 0, \mu_i \geq 0 \quad (41)$$

$$\alpha_i \{y_i (x_i \cdot w + b) - 1 + \xi_i\} = 0 \quad (42)$$

$$\mu_i \xi_i = 0 \quad (43)$$

Y como antes, podemos utilizar las condiciones KKT complementarias (ecuaciones (55) y (56) en [5]) para determinar el umbral b. Nótese que la ecuación (39) combinada con la ecuación (43) muestran que $\xi_i = 0$ si se cumple que $\alpha_i < C$. Por lo que podemos simplemente tomar cualquier punto de entrenamiento para el cual $0 < \alpha_i < C$ y usar la ecuación (42) para calcular b.

Introducción de hiperplanos de separación no lineales

Ahora nos preguntamos cómo podemos generalizar aun más todo esto para el caso donde la función de decisión no es lineal a los datos de entrenamiento. Para esto debemos notar que los datos aparecen en el problema de entrenamiento, en las ecuaciones (32 - 34), en la forma de productos escalares $x_i \cdot x_j$. Entonces supongamos que primero mapeamos los datos a otro (posiblemente de dimensión infinita) espacio euclidiano H, utilizando un mapeo que llamaremos ϕ, tal que $\phi: \mathbb{R}^d \mapsto H$.

Entonces el algoritmo de entrenamiento solo dependería de los datos a través de su uso como productos escalares en H, es decir en funciones de la forma $\phi(x_i) \cdot \phi(x_j)$. Ahora si hubiera una función de 'kernel' K tal que $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, entonces necesitaríamos solamente usar esta función en el algoritmo de entrenamiento, y nunca tendríamos que lidiar explícitamente o siquiera conocer como la función ϕ está definida. Para que esta función represente legítimamente un producto escalar en el espacio mapeado, debe satisfacer las siguientes condiciones de Mercer:

$$K(x_i, x_j) = \sum_k^\infty m_k \phi_k(x_i)\phi_k(x_j), \quad m_k \geq 0 \tag{44}$$

$$\iint K(x_i, x_j) g(x_i) g(x_j) dx_i dx_j > 0, \quad g \in L^2 \tag{45}$$

Un posible ejemplo sería:

$$K(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2} \tag{46}$$

En este ejemplo en particular la dimensión de H sería infinita, con lo que no nos resultaría fácil trabajar con ϕ en forma explícita. En cambio si utilizamos la función de 'kernel' en el remplazo de $x_i \cdot x_j$ en todos los lados del algoritmo de entrenamiento donde se necesite, entonces el algoritmo producirá los vectores de soporte que viven en un espacio infinito, y también lo hará en el mismo tiempo en que le tomaría realizar un entrenamiento en los datos no mapeados. Todas las consideraciones que hicimos anteriormente se mantienen, dado que aun estamos haciendo una separación lineal, pero en un diferente espacio.

No hemos ahondado demasiado en los detalles para el caso no separable, ni en el uso de 'kernels', pero como se intentó resumir, las ideas son similares a las presentadas para el primer caso donde los datos son linealmente separables. La introducción de un 'kernel', como vimos, es básicamente el definir una función que transforme el espacio de dimensiones en otro (donde haya más elementos para determinar una mejor separación de clases) y en la ecuación (7) sería abstraernos a considerar que $x_i \cdot x_j$ está definido por una función $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$; luego esta función 'kernel' puede ser reemplazada para definir operaciones de 'producto escalar' sobre espacios de dimensionalidad mas compleja (véase para mayor detalle [5]).

Existen varios tipos de funciones 'permitidas' (que cumplen las condiciones necesarias) para ser utilizadas como 'kernels'. Entre ellas encontramos algunas polinomiales, radiales guassianas, radiales exponenciales, perceptrones multicapa, series de Fourier, splines, bases de splines, de 'kernels' aditivos, de productos tensoriales sobre 'kernels' [3][19].

De estos posibles kernels estaremos usando para este trabajo el polinomial, y el gaussiano, que se encuentran implementados en la herramienta que hemos optado por utilizar (Torch [15]) para

realizar la clasificación utilizando esta teoría, teniendo preferencia por el kernel gaussiano por ser el que mejores resultados nos brinda para el problema tratado [4][34].

Aplicación de la teoría

En resumen podemos decir que aplicar los conceptos de máquinas de soporte vectorial, para clasificar un conjunto binario en la forma más popular que se deriva de su teoría, consiste en:

- Elegir un parámetro C que presente un balance entre la minimización de la cota de error del conjunto de entrenamiento y a su vez maximice el margen de separación donde descansa el hiperplano generado
- Elegir una función de kernel adecuada (si este fuera el caso) y los parámetros que serán utilizados en la aplicación de la misma [20]
- Resolver el problema cuadrático dual (ecuación 28) o una formulación alternativa que represente lo mismo usando programación cuadrática o un algoritmo de programación lineal
- Recuperar el umbral representado por la variable b usando los vectores de soporte
- Clasificar un nuevo punto en base a observar el signo en la siguiente función (lo que determina de qué lado del hiperplano, generado por el conjunto de entrenamiento, se encuentra el vector analizado):

$$f(x) = sign(\sum_i y_i \alpha_i K(x, x_i) - b)$$

Generalmente los parámetros necesarios para el entrenamiento de la máquina de soporte vectorial se obtienen, o determinan, haciendo validaciones cruzadas si se dispone de la suficiente cantidad de muestras a tal efecto, aunque recientemente nuevas estrategias para la selección de este modelo nos pueden brindar una estimación razonable para los parámetros del kernel sin la utilización de este tipo de validaciones [18].

Capítulo 4 – Bases de dígitos manuscritos utilizadas

Las bases utilizadas, CENPARMI y MNIST, son ampliamente aceptadas como estándares en la medición y comparación de distintos métodos de reconocimiento y clasificación de patrones. Cada una de ellas está dividida en un conjunto de dígitos de entrenamiento y un conjunto de dígitos de testeo así, como se dijo anteriormente, los resultados de la aplicación de diversos algoritmos sobre los mismos puede ser comparada entre sí.

Base CENPARMI

La base CENPARMI [11] (siglas que en inglés corresponden Centre for Pattern Recognition and Machine Intelligence) fue generada por el Centro para el reconocimiento de patrones e inteligencia de máquinas de la universidad de Concordia Canadá. Contiene dígitos sin restricciones representados por pixeles binarios. El conjunto de entrenamiento dispone de 4000 muestras (400 por clase) y el conjunto de testeo de unas 2000 (200 por clase). Las imágenes han sido escaladas a un tamaño de 16x16 de tal manera que el aspecto de cada una de las muestras fue preservado. Algunas muestras de dígitos de cada una de las clases pertenecientes a esta base se presentan a continuación:

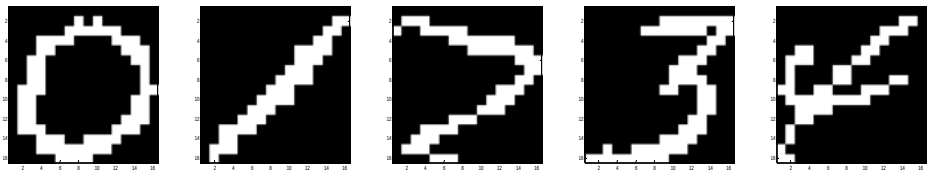


Figura 12. Las muestras pertenecen (de izquierda a derecha) a dígitos de las clases 0, 1, 2, 3 y 4 respectivamente del conjunto de entrenamiento

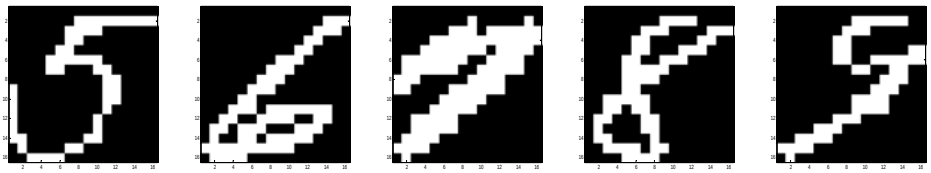


Figura 13. Estas muestras (de izquierda a derecha) pertenecen a dígitos de las clases 5, 6, 7, 8 y 9 respectivamente del conjunto de entrenamiento

Como se puede observar a continuación, mostraremos algunos ejemplos de cómo los trazos entre dos representantes de la misma clase puede variar significativamente (esto se verá mejor más adelante donde presentemos un análisis de varianzas y es lo que hace tan complicado el reconocimiento sin preprocesos):

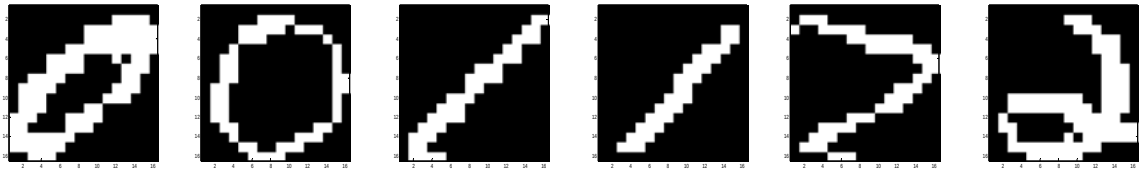


Figura 14. En este caso vemos diferencias muy marcadas entre dos representantes de la clase 0, 1 y 2 del conjunto de entrenamiento (sobre la clase 0, el 1er con forma elipsoidal, el 2do con trazo mas circular)

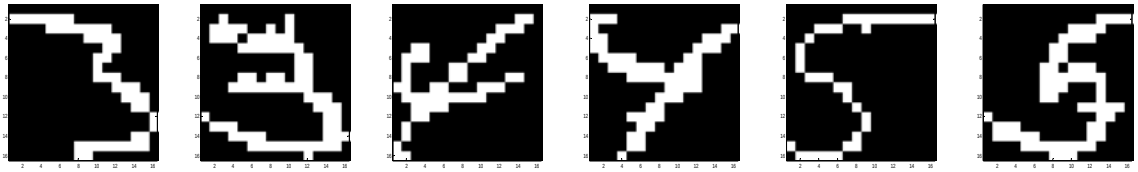


Figura 15. En este caso vemos diferencias muy marcadas entre dos representantes de las clases 3, 4 y 5 del conjunto de entrenamiento

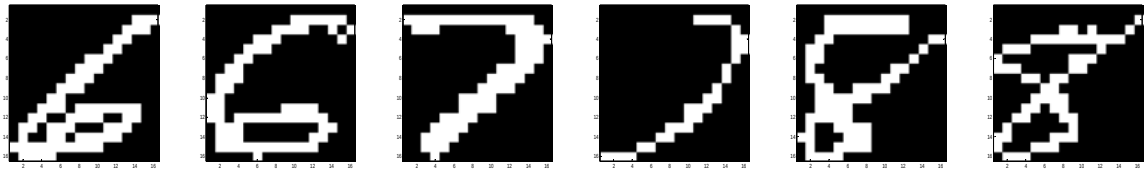


Figura 16. En este caso vemos diferencias muy marcadas entre dos representantes de las clases 6, 7 y 8 del conjunto de entrenamiento

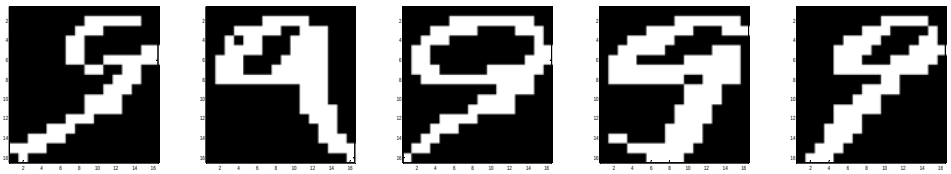


Figura 17. En este caso vemos diferencias muy marcadas entre varios representantes de la clase 9 del conjunto de entrenamiento y la facilidad de encontrar un grupo tan diferente (seleccionados en forma uniforme)

Base MNIST

La base MNIST [12] es una versión modificada de la base NIST que originalmente fue presentada por el grupo AT&T. Dicha base contiene un conjunto de 60000 muestras para el entrenamiento de los clasificadores y unas 10000 muestras para el testeo, dichos patrones no están divididos de manera uniforme, es decir que cada una de las 10 clases de dígitos que contiene no presenta igual cantidad de elementos para su tratamiento. Las imágenes están generadas en escala de grises (256 valores). Algunas muestras de dígitos de cada una de las clases pertenecientes a esta base se presentan a continuación:

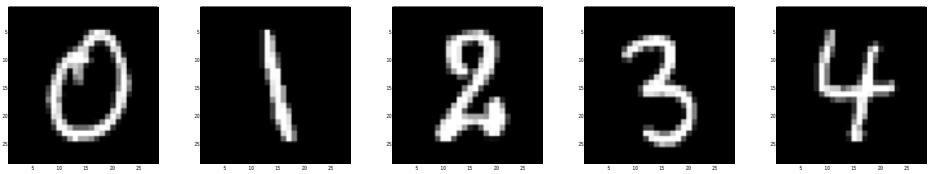


Figura 18. Las muestras pertenecen (de izquierda a derecha) a dígitos de las clases 0, 1, 2, 3 y 4 respectivamente del conjunto de entrenamiento

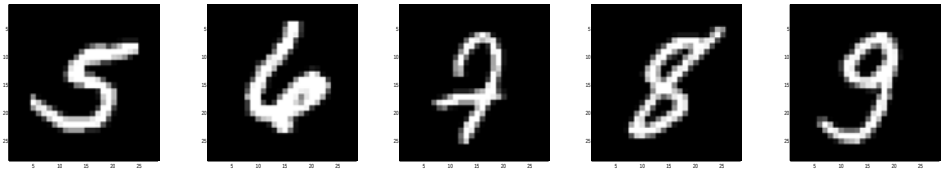


Figura 19. Estas muestras (de izquierda a derecha) pertenecen a dígitos
de las clases 5, 6, 7, 8 y 9 respectivamente del conjunto de entrenamiento

Como se puede observar a continuación, mostraremos algunos ejemplos de cómo los trazos entre dos representantes de la misma clase puede variar significativamente (esto se verá mejor más adelante donde presentemos un análisis de varianzas y es lo que hace tan complicado el reconocimiento sin preprocesos):

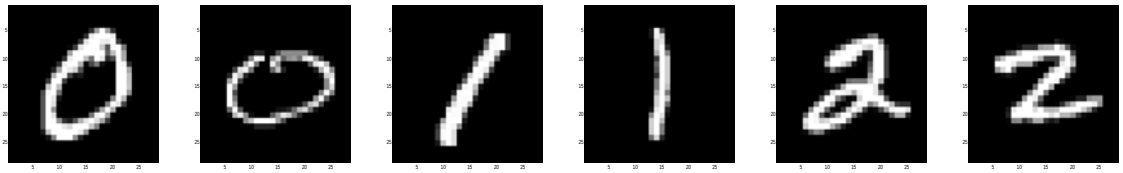


Figura 20. En este caso vemos diferencias muy marcadas entre dos representantes de la clase 0, 1 y 2
del conjunto de entrenamiento (sobre la clase 0, el 1er con forma elipsoidal vertical, el 2do con forma elipsoidal horizontal)

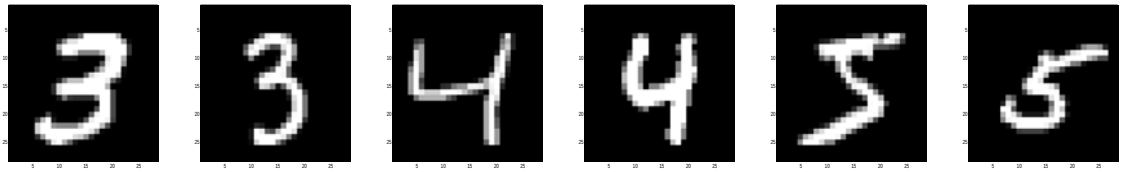


Figura 21. En este caso vemos diferencias muy marcadas entre dos representantes
de las clases 3, 4 y 5 del conjunto de entrenamiento

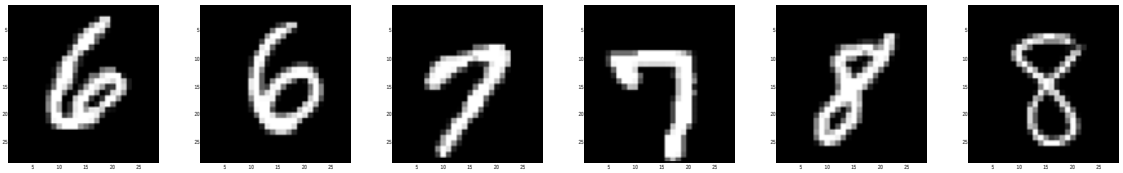


Figura 22. En este caso vemos diferencias muy marcadas entre dos representantes
de las clases 6, 7 y 8 del conjunto de entrenamiento

Figura 23. En este caso vemos diferencias muy marcadas entre varios representantes de la clase 9
del conjunto de entrenamiento y la facilidad de encontrar un grupo tan diferente (seleccionados en forma uniforme)

En general como los dígitos de la base CENPARMI poseen menor resolución que los pertenecientes a la base MNIST (esto se puede observar a simple vista sobre las muestras presentadas

como ejemplos), como así también contienen menor cantidad de muestras para realizar el entrenamiento, éstos son considerados mucho más difíciles de clasificar que los pertenecientes a la base MNIST.

Capítulo 5 – Preprocesamiento y Análisis

Introducción

Utilizaremos la teoría de wavelets introducida para la detección y análisis de características de las imágenes que representan los patrones de las bases seleccionadas, es decir que las estaremos procesando, en un intento de destacar las características más sobresalientes que hacen que un dígito pertenezca a una determinada clase. Para esto nos valdremos del uso de las 3 wavelets que nos hemos propuesto utilizar, a diferentes resoluciones, y utilizaremos mapeos de los resultados de la transformada a otros espacios (por ejemplo el mapeo de los detalles, tanto horizontales como verticales que surgen de aplicar la transformada wavelet a imágenes, a un espacio de ángulos y módulos) en un intento de normalizar las características que surgen de este análisis multiresolución y diferenciarlas de forma tal que pudieran determinar de la mejor manera posible la clase a que corresponde cada dígito analizado.

Luego a partir de este análisis preliminar, donde se realizan las transformadas correspondientes usando las distintas wavelets que introdujimos, es que compondremos al vector de características (descriptor) que será la entrada con la que estarán trabajando las máquinas de soporte vectorial que estemos entrenando bajo distintas parametrizaciones. En este análisis, y luego de hacer uso de las distintas wavelets bajo una misma configuración establecida para la extracción de éstas características, también haremos uso de los distintos kernels (con parametrización uniforme – con lo que queremos decir que para las 3 wavelets seleccionadas estaremos 'configurando' de igual manera a los clasificadores a fin de poder compararlos entre sí) en el entrenamiento de cada una de las máquinas de soporte vectorial que estaremos armando. Entonces es en este punto inicial donde realizaremos pruebas con distintas parametrizaciones sobre las transformadas, distintas composiciones del descriptor y hasta distintas parametrizaciones de kernels, y a pesar que dichas variables podrían ser combinadas en muchas posibles maneras, presentaremos solo algunos de estos casos y nos quedaremos con aquellos que mejores resultados nos hayan otorgado, para luego continuar un paso más en el tratamiento de este problema, utilizando técnicas actualmente muy aplicadas al mismo, con el objetivo de conseguir una mayor eficacia en el reconocimiento.

Entonces, como hemos contado, una vez que hayamos encontrado una composición, de las características extraídas en forma aislada o separada, para generar descriptores cuyos resultados nos satisfagan en el entrenamiento y la clasificación utilizando clasificadores individuales (basados en cada una de las wavelets que hemos seleccionado para este trabajo) procederemos a utilizar técnicas para la combinación [13] de estos clasificadores con el fin de incrementar la eficacia en el reconocimiento, pero sin degradar en demasía la performance en cuanto al tiempo computacional requerido a tales efectos.

Comenzaremos en la siguiente sección presentando un análisis espacial de la varianza de cada una de las clases de dígitos para que el lector se haga una idea sobre la dificultad del problema que surge a partir de las diferentes formas de representación de los datos que existen.

Luego analizaremos por separado características que surgen del análisis multiresolución en la aplicación de la transformada wavelet para tomarlas en forma individual y luego componerlas como parte del vector de características que será entrada de la máquina de soporte vectorial.

En última instancia presentaremos en forma detallada como está compuesto el vector que utilizaremos para entrenar las máquinas de soporte vectorial con las que estaremos haciendo pruebas, y cuáles son las características de este análisis que hemos seleccionado a partir de los resultados individuales que cada una de estas nos pudieran ofrecer. Todos estos detalles nos servirán para entender los resultados que mostramos en el próximo capítulo en las pruebas que hemos realizado.

Análisis de la varianza espacial de dígitos

Como hemos dicho, comenzamos por realizar un análisis de las varianzas sobre los dígitos de cada clase coordenada a coordenada (varianza espacial o localizada por píxel), para poder tener alguna idea aproximada sobre la similitud entre los mismos y hacernos una idea de la dificultad para un reconocimiento simple sin aplicar ningún preproceso de extracción o análisis minucioso, y también para la determinación de buenas características para el entrenamiento. Podemos ver, para las distintas clases, valores de varianza altos (blancos) donde las curvas que representan cada uno de los dígitos no siguen el mismo trazado en cada una de las diferentes muestras de la clase que tenemos, y valores bajos (negros) que serían los más esperados ya que indicarían coincidencias entre los trazos que representan a cada muestra perteneciente a una clase de dígito. Esto nos indicaría que la clase que presente su imagen, en este análisis, con más puntos apagados (negros) tendrá sus características más parecidas y podría ser, ya que no lo podemos asegurar bajo ningún aspecto, la más fácil de identificar (al menos la decisión de si dos dígitos tienen las mismas características es fácil de ver, y como dijimos podrían ser de dicha clase o pudiera ser un dígito de otra clase que por alguna deformidad presentara características 'desfasadas' por decirlo de alguna manera, pero hay una alta probabilidad de que lo que suceda sea lo primero).

Las imágenes presentadas nos permiten hacer un análisis visual y ver en que pueden coincidir, y en que no, las muestras de una clase como factor de aporte en la decisión de las características a tomar para formar los vectores de clasificación. Los resultados son los siguientes:

Sobre las muestras de entrenamiento de la base CENPARMI:

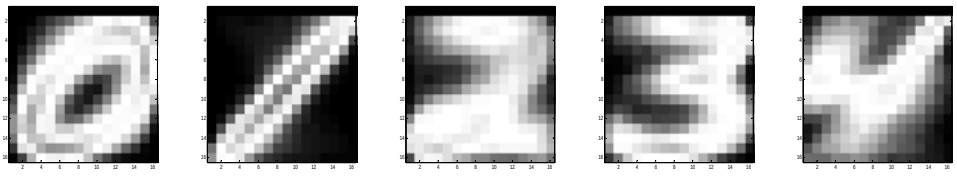


Figura 24. De izquierda a derecha los resultados de la varianza espacial para las muestras de las clases 0, 1, 2, 3 y 4 respectivamente del conjunto de entrenamiento

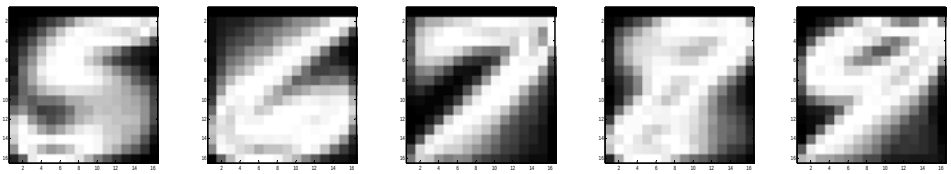


Figura 25. De izquierda a derecha los resultados de la varianza espacial para las muestras de las clases 5, 6, 7, 8 y 9 respectivamente del conjunto de entrenamiento

Sobre las muestras de entrenamiento de la base MNIST:

Figura 26. De izquierda a derecha los resultados de la varianza espacial para las muestras de las clases 0, 1, 2, 3 y 4 respectivamente del conjunto de entrenamiento

Figura 27. De izquierda a derecha los resultados de la varianza espacial para las muestras de las clases 5, 6, 7, 8 y 9 respectivamente del conjunto de entrenamiento

Como se puede observar en estas imágenes (a simple vista), los dígitos de la base CENPARMI presentan más diferencias de trazos (son mucho más blancas), así como de orientación y ubicación, dando como resultado final una menor similitud entre ellos para cada una de las clases, por lo que, como se mencionó anteriormente, su clasificación es mucho más difícil y requiere análisis y tratamiento más minucioso que las muestras de la base MNIST.

Uso de la aproximada (como similitudes) de la imagen

Algo que nos permite obtener el análisis multiresolución es una representación más suave de la señal original. Entre más niveles de análisis apliquemos, tendremos una versión más 'burda' de la imagen original, es decir una aproximación cada vez con menor detalle (pero más suavizada) de la imagen original que estemos tratando. Debido a los diferentes rangos de valores que componen las imágenes originales (binario para imágenes CENPARMI y de 256 valores para representar dígitos de la base MNIST) sucede que en la aplicación de filtros (convoluciones con valores reales) el rango de valores resultado de la transformada en las matrices resultado puede variar significativamente del conjunto de partida original. Mientras más amplio sea el rango de valores posibles más difícil será 'normalizar' características del mismo en un intento de reconocer características principales que identifiquen la pertenencia a cada una de las clases que estamos analizando.

Es nuestra intención el hacer uso de la aproximación resultante de aplicar la transformada wavelet a la imagen original, pero también intentaremos minimizar las diferencias entre los distintos dígitos de cada clase que se podrían generar a partir de la aplicación de las mismas. Para lograr este punto, en algunas de las pruebas que hemos desarrollado, es que estaremos binarizando los resultados a partir de un umbral que generalmente será el promedio de los valores que se encuentren en la matriz. De esta manera estaremos reduciendo un rango del tipo $[-n, +m]$ a un rango que pertenecerá al conjunto {0, 1}. Pero como hemos mencionado esto sería solo una de las posibles parametrizaciones que estemos usando y comparando, pudiendo utilizar otras sin aplicar este binarizado.

También aprovecharemos para reducir la dimensión de los datos de entrada para incrementar la performance de la máquina de soporte vectorial que hará uso de esta componente y buscaremos que la dimensión final del descriptor (una vez que este compuesto con todas las características que hayamos localizado) sea significativamente menor a la dimensión del descriptor que utilizaría a la imagen original como dato.

Por los motivos que hemos expuesto es que haremos un submuestreo de la matriz de aproximación resultante de la aplicación de la transformada wavelet. Como hemos mencionado anteriormente el modo de aplicación de dicha transformada no reduce el tamaño de las matrices de aproximación y detalles, sino que mantiene las mismas dimensiones. Es decir que partiendo de una imagen original de tamaño NxN la matriz de aproximación resultante de una y/o varias aplicaciones de la transformada wavelet seguirá conservando el mismo tamaño NxN. La aplicación de un submuestreo a esta matriz (aproximada) una vez que se haya aplicado la transformada wavelet correspondiente nos reducirá significativamente la dimensión de dicha matriz a un tamaño de $\frac{NxN}{4}$.

Como mostraremos a continuación la aplicación de todo esto reducirá el error o diferencias entre dos dígitos representantes de una misma clase respecto a la imagen original y respecto a la aproximada de la transformada wavelet sin haber realizado ningún tipo de procesamiento como aquí mencionamos. Veremos a continuación ejemplos de dígitos representantes, utilizando patrones de la base MNIST, de una misma clase bastante disímiles y los errores o diferencias que tienen:

Representantes de la clase 2 (base MNIST - imágenes originales):

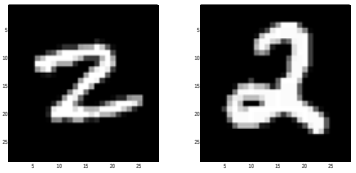


Figura 28. Dos dígitos del conjunto de entrenamiento

Diferencia cuadrática entre ambos: 9.3789e+003
Dimensión de los datos: 28x28

Aplicación de un nivel de transformada Mallat:

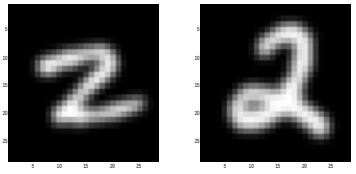


Figura 29. La aproximada de los dígitos originales (figura 28) transformados

Diferencia cuadrática entre ambos: 2.0762e+004
Dimensión de los datos: 28x28

Binarizacion de las matrices de aproximación generadas:

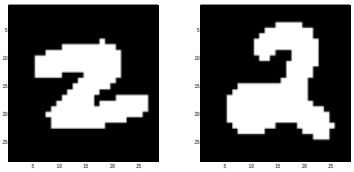


Figura 30. El binarizado de los dígitos transformados (figura 29)

Diferencia cuadrática entre ambos: 0.2117
Dimensión de los datos: 28x28

Submuestreo de las matrices de aproximación binarizadas:

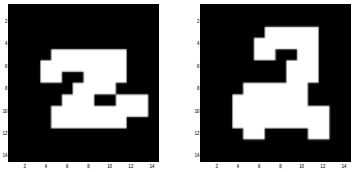


Figura 31. El submuestreo de los dígitos binarizados (figura 30)

Diferencia cuadrática entre ambos: 0.1990
Dimensión de los datos: 14x14

Podemos observar que, luego de un paso de transformada, la imagen de los dígitos analizados se encuentran suavizadas, no obstante el error y/o diferencias entre los mismos es mayor, y estamos manteniendo las mismas dimensiones sin lograr ninguna mejora. En el caso donde hemos binarizado los valores que representan la imagen, vemos como el error entre ambas imágenes disminuye considerablemente, pero aun mantenemos la misma dimensión, con lo que el costo de entrenar máquinas de soporte vectorial y su utilización para la clasificación de otros dígitos desconocidos presentará una mayor complejidad y costo de cálculo.

En el último paso presentado, donde hemos binarizado y submuestreado a la matriz de aproximación resultante de aplicar la transformada wavelet a la imagen original, vemos como logramos los objetivos que nos interesan para el uso de esta componente en el descriptor con el que trabajará la máquina de soporte vectorial. Estos factores de interés son la reducción del error entre dos representantes de una misma clase, y la reducción de la dimensión de esta componente, lo cual resultará en una reducción de la dimensión final del descriptor que utilizaremos en el entrenamiento y clasificación de estos patrones.

Uso de los detalles (como diferencias) de la imagen

Basados en lo explicado en el capítulo 3 sobre el procesamiento de imágenes, veremos cómo serán utilizadas las matrices de detalles (solo vertical y horizontal) para la generación de características del descriptor que usaremos para la clasificación.

Como hemos mencionamos antes, respecto al uso de la aproximada que, debido a los diferentes rangos de valores que componen las imágenes originales (binario para imágenes CENPARMI y de 256 valores para representar dígitos de la base MNIST) sucede que en la aplicación de filtros (convoluciones con valores reales) el rango de valores resultado de la transformada en las matrices resultado puede variar significativamente del conjunto de partida original. En la generación de estas matrices de detalles también observamos que mientras más amplio es el rango de valores posibles más difícil nos resulta el 'normalizar' características del mismo en un intento de determinar características principales que identifiquen la pertenencia a cada una de las clases que estamos analizando, y que a su vez nos diferencie las distintas clases que tenemos.

Otro objetivo que se logrará en las transformaciones, en cuanto al cambio de dominio de los valores contenidos en cada una de las matrices de detalle, será la reducción de la dimensión de los datos de entrada también con la intención de incrementar la performance de la máquina de soporte vectorial que hará uso de esta componente buscando que la dimensión final del descriptor (una vez que este compuesto con todas las características que hayamos localizado) sea significativamente menor a la dimensión del descriptor que utilizaría a la imagen original como dato (veremos como la transformación a otro espacio sumado al uso de histogramas nos aportará a conseguir estos objetivos en las próximas secciones que presentaremos).

Mostraremos a continuación el resultado de aplicar un paso de la transformada wavelet a dos dígitos representantes de una misma clase, y veremos la diferencia a priori que se presenta entre las matrices de detalle generadas, tanto horizontal como vertical, para cada uno de ellos:

<u>Representantes de la clase 3 (base MNIST - imágenes originales):</u>

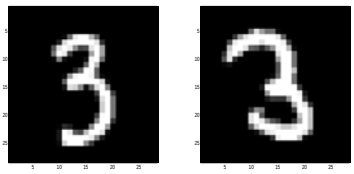


Figura 32. Dos dígitos del conjunto de entrenamiento

Diferencia cuadrática entre ambos: 5.2453e+003
Dimensión de los datos: 28x28

Detalles horizontales de la aplicación de un paso de la transformada Mallat:

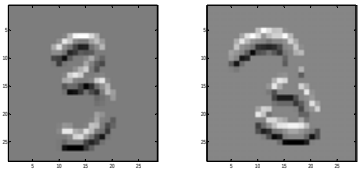


Figura 33. Los detalles horizontales de los dígitos originales (figura 32) transformados

Diferencia cuadrática entre ambos: 2.1659e+003
Dimensión de los datos: 28x28

Detalles verticales de la aplicación de un paso de la transformada Mallat:

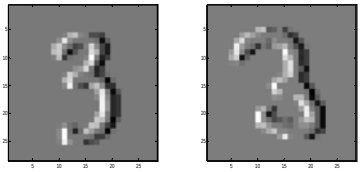


Figura 34. Los detalles verticales de los dígitos originales (figura 32) transformados

Diferencia cuadrática entre ambos: 1.6057e+003
Dimensión de los datos: 28x28

Podemos observar que luego de un paso de transformada los detalles de la imagen de los dígitos el error y/o diferencias entre los mismos es menor, pero no lo suficiente, y estamos manteniendo las mismas dimensiones sin lograr ninguna mejora. Es por ello que no nos dispondremos a utilizar estas matrices tal cual han sido transformadas sino que aplicaremos el proceso que describiremos a continuación para poder 'normalizarlas' y lograr extraer características que nos sean favorables al proceso que estamos desarrollando.

Utilización de histogramas de la cuantización de ángulos

Haremos a continuación un análisis para ver cómo utilizar las matrices de detalle resultantes de la aplicación de la transformada wavelet en la composición del descriptor con que entrenaremos y haremos clasificación utilizando la máquina de soporte vectorial. Para normalizar o generar una representación más fácil de manipular sobre estos detalles es que haremos un mapeo de los mismos (detalles horizontales y detalles verticales) a un espacio en donde trabajaremos con los ángulos y módulos de los vectores que se generen a partir de la representación coordenada a coordenada de cada uno de los valores de dichas matrices.

La transformación del espacio resultante a este nuevo espacio nos permitirá realizar un análisis sobre la agrupación de los ángulos y módulos para cada uno de los dígitos, como siempre en un intento de determinar características que nos representen cada una de las clases y que también provean las diferencias entre los representantes de las distintas clases. A este momento no hemos tratado aun el tema de la dimensión de los datos que utilizaremos, ni tampoco como normalizaremos los mismos o

como se intentará generar características a partir de los mismos, pero es un tema siempre tenido en cuenta.

A continuación mostramos como quedan las matrices de detalle horizontal y vertical, presentadas en las figuras 33 y 34 respectivamente, cuando son transformadas de esta manera para que representen matrices de módulos y ángulos:

Matriz de módulos generada a partir de la transformación utilizando las matrices de detalles (horizontal y vertical):

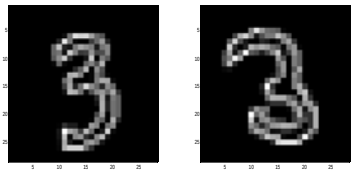


Figura 35. Los módulos generados a partir de las matrices de detalle (de las figuras 33 y 34)

Diferencia cuadrática entre ambos: 2.4368e+003
Dimensión de los datos: 28x28

Matriz de ángulos generada a partir de la transformación utilizando las matrices de detalles (horizontal y vertical):

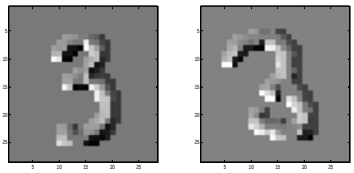


Figura 36. Los ángulos generados a partir de las matrices de detalle (de las figuras 33 y 34)

Diferencia cuadrática entre ambos: 0.8658
Dimensión de los datos: 28x28

Para reducir la dimensión de los datos que compondrán al descriptor, tal como hicimos en el caso donde consideramos el utilizar la aproximación de la señal original, es que aplicaremos cuantización en diferentes intervalos sobre los resultados de estas matrices transformadas (es decir, llevadas a otro espacio) y luego generaremos un histograma para conocer la acumulación de valores en cada uno de estos intervalos como características determinantes que presentan cada una de las clases. Este proceso nos brindará una de las principales características que se utilizará para componer la componente de alta del vector de clasificación, y a la vez, estaremos reduciendo la cantidad de valores que componen a esta característica utilizando solo el histograma generado por los representantes de cada uno de los intervalos (16 valores por ejemplo, contra 784 que contiene cada una de las matrices de detalle).

El detalle del proceso que estaremos aplicando entonces es el siguiente:

1. Se aplica la transformada wavelet a la imagen original generando las correspondientes matrices de detalles horizontales y verticales.

2. Dichas matrices son convertidas a matrices que representan módulos y ángulos, a través de interpretar los valores de las matrices de detalle como coordenadas (x, y) de un vector de $\mathbb{R}^2$ (para los valores de las matrices de detalle horizontal y vertical respectivamente).
3. La matriz de ángulos es cuantizada en una cantidad N de intervalos generados dentro de los posibles rangos $[-\pi, +\pi]$ o $[0, \pi]$ según el caso que estemos tratando (en este último caso los valores en el intervalo $[-\pi, 0]$ son incrementados en $+\pi$).
4. Se genera el histograma de los representantes de cada uno de los intervalos de cuantización.

La generación de dicho histograma presenta una suma de características de las clases y cuanto mayor el tamaño de los intervalos en que se cuantice mayor será la cantidad de elementos que sumen a dicha característica, pero habrá que tener en cuenta un balance entre cantidad y performance no perdiendo generalidad en dicho proceso. Los intervalos que estaremos utilizando para componer al vector de características generalmente serán con 8, 16 y 32 intervalos.

A continuación presentamos el histograma circular generado a partir del análisis de dos dígitos representantes de una misma clase (los mismos que hemos seleccionado en la sección anterior para poder comparar la diferencia entre los mismos y que se pueden observar en la figura 32) utilizando este proceso. Se ha utilizado el intervalo $[-\pi, +\pi]$ para el tratamiento de los ángulos, se ha tomado un umbral del 50% del promedio generado por la suma de los índices positivos de la matriz de módulos, y por último se ha cuantizado en 16 intervalos.

Histogramas de dichos dígitos para una cuantización de 16 intervalos:

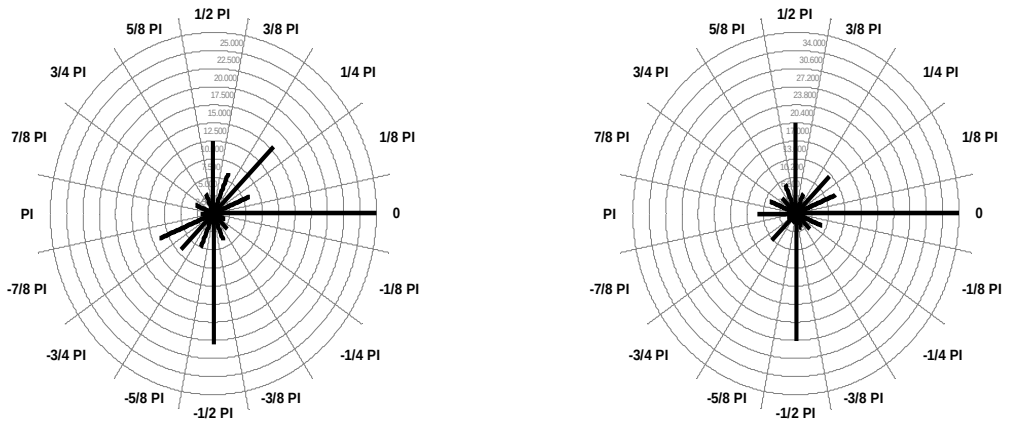


Figura 37. Los histogramas de ángulos luego de aplicar el proceso descripto a los dígitos originales (figura 32)

Diferencia cuadrática entre ambos: 17.1523
Dimensión de los datos: 16

En el caso presentado vemos como la clase presentada tiene, bajo este proceso que hemos aplicado, una gran cantidad de ángulos 0, y en menor cantidad, pero que también podría ser considerado como determinante de los dígitos de esta clase, que hemos utilizado como ejemplo, el valor $^{-\pi}/_{2}$. También vemos como la cantidad de datos que componen esta característica que formará parte del vector de clasificación se reduce drásticamente.

Para comprender con mayor detalle, como haremos uso de esto, y como dicha característica aportaría a determinar la clase a la que un dígito pudiera corresponder, se presentan a continuación algunas imágenes de 1500 histogramas de ángulos cuantizados en el intervalo $[-\pi, +\pi]$ tomando 16 intervalos de cuantización (tomando como umbral la mitad del promedio de los módulos mayores a 0). Se puede distinguir, nos referimos a la observación que se puede realizar de manera visual, el patrón vertical que se genera a partir de cada uno de los valores del histograma de ángulos y presenta las características necesarias para distinguir dicha clase. Se debe considerar que las características, en este caso extraídas a partir del análisis de los ángulos generados en el proceso, más representativos estarían fuertemente definidos en las columnas donde los valores son más altos (blancos). Como hemos hecho en el análisis sobre los dígitos anteriores para este caso, la wavelet aplicada es Mallat, y se transforma con un solo nivel de aplicación sobre muestras pertenecientes a la base MNIST.

Realizando una simple comparación visual de las imágenes generadas (a modo de patrones por líneas) de los histogramas para dos clases bastante similares (1 y 9), tenemos lo siguiente que puede ser observado en las figuras 38 y 39:

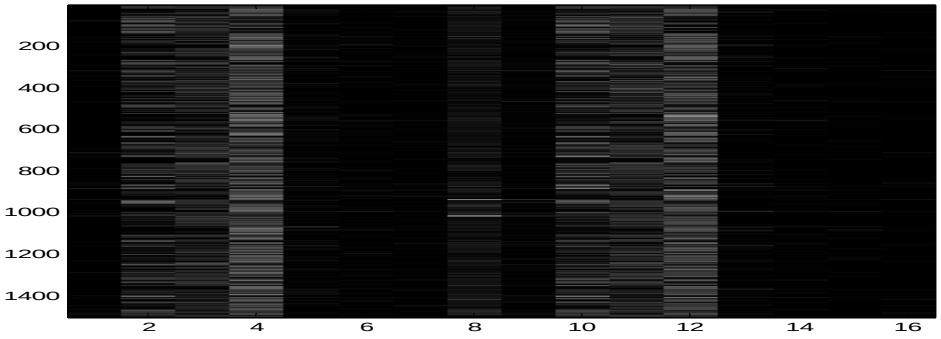


Figura 38. Mapa de histogramas de 1500 muestras de la clase 1 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

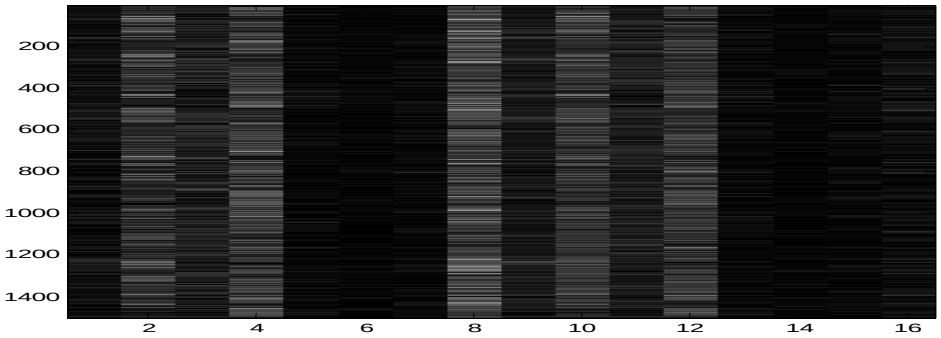


Figura 39. Mapa de histogramas de 1500 muestras de la clase 9 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

Si ahora realizamos la comparación de los patrones de los histogramas para dos clases no tan similares (2 y 5) a simple vista vemos que presentan ángulos muy similares (por la generación de los patrones en cada conjunto de histogramas) luego del proceso descripto anteriormente y aplicado para obtener los histogramas de dichas clases, el resultado del armado de estas imágenes para comparar se puede observar en las siguientes figuras 40 y 41:

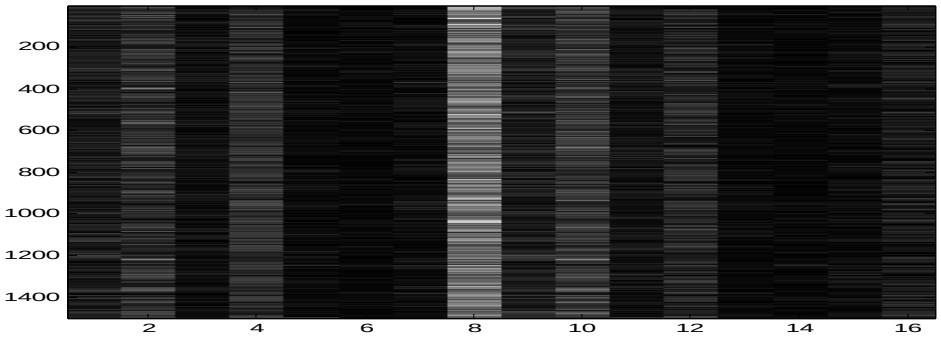


Figura 40. Mapa de histogramas de 1500 muestras de la clase 2 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

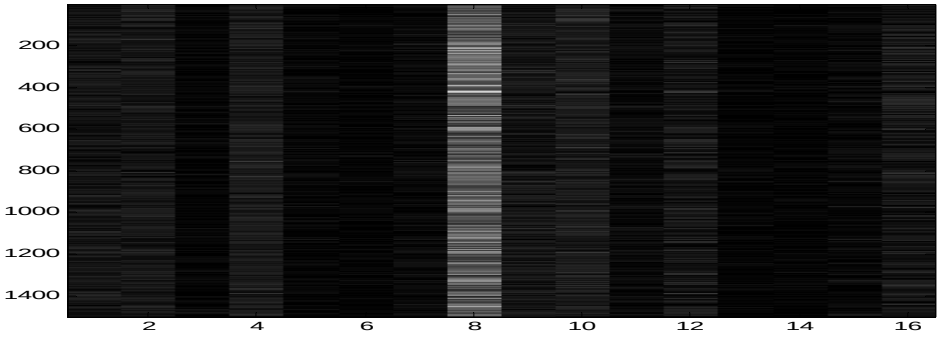


Figura 41. Mapa de histogramas de 1500 muestras de la clase 5 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

Ahora si comparamos la imagen de los histogramas de clases no muy similares (como por ej. 1 y 2 en las figuras anteriores, la 38 y 40 respectivamente) observamos que se determinan fuertemente patrones muy distintos y a través de ellos sería fácilmente reconocible para cualquier observador saber si el histograma obtenido representa un dígito de la clase 1 o de la clase 2, y vemos que la clase 2 tiene más altos valores en el 8vo intervalo mientras que la clase correspondiente a los dígitos 1 distribuye sus valores más altos entre los intervalos 4 y 12; de esta manera y con una simple mirada de estos valores como características (es decir el histograma obtenido luego de las transformaciones y preprocesamiento) se puede determinar su pertenencia. Este mismo análisis se puede aplicar entre la clase 1 y la clase 5 con una simple observación, y vemos que en la última el concentrado de valores en el 8vo intervalo es muchísimo mayor que en la clase 2, es decir que además de compararlo fácilmente con la clase 1, también podríamos compararlo de manera más minuciosa contra un histograma de un dígito de la clase 2 y poder identificar si correspondiera a la 5 o a esta última.

Como realizando este análisis observamos que sería muy simple para un humano el poder determinar si un patrón nuevo pertenece a una determinada clase (clasificación) teniendo uno o varios

patrones que identifican a dígitos de dicha clase (entrenamiento como por ejemplo la 5), entonces esta característica genera un aporte muy fuerte al entrenamiento y la clasificación de estos dígitos, que realizado por un sistema suficientemente complejo como es el resultante de entrenar a una máquina de soporte vectorial a tal efecto, dará muy buenos valores de efectividad.

Utilización de las entropías de cuadrantes de las matrices transformadas

Otra característica que estaremos utilizando estará generada a partir de las matrices resultantes de aplicar la transformada wavelet (aproximada), mas la conversión a módulos/ángulos de los detalles (detalle horizontal y detalle vertical), y la aplicación de una función que nos retorne cuanta información en la composición del dígito aporta cada sección, y esta característica será la que presentaremos a continuación.

Para armar esta característica estaremos dividiendo el dígito en 4 secciones (o cuadrantes) generadas a partir de la división por la mitad, tanto horizontal como vertical, de la matriz transformada que estaremos tomando, ya sea para la matriz de aproximación o para algunas de las matrices de detalles, para la obtención de los valores que compondrán a dicha característica. Teniendo esta separación de cada matriz en 4 partes, estaremos aportando 1 valor por cada una de estas secciones a la característica. Es decir que cada componente que estemos agregando al vector descriptor contendrá 4 valores que aportarán al entrenamiento de la máquina de soporte vectorial (tanto como a la clasificación).

Mostraremos a continuación las 4 secciones generadas, a partir de la división que hemos descripto, aplicada a un dígito transformado con la wavelet Mallat en 1 nivel para la matriz de aproximación:

Dígito original perteneciente a la clase 3:

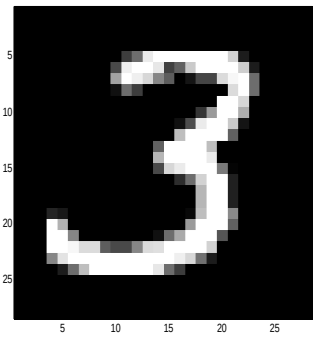


Figura 42. Dígito de la clase 3 del conjunto de entrenamiento

Matriz de aproximación resultante de aplicar 1 paso de la transformada wavelet:

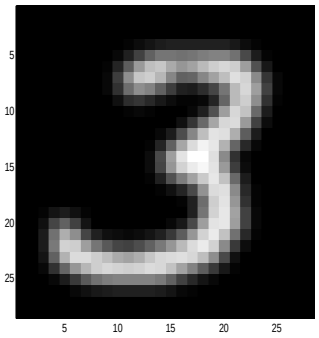
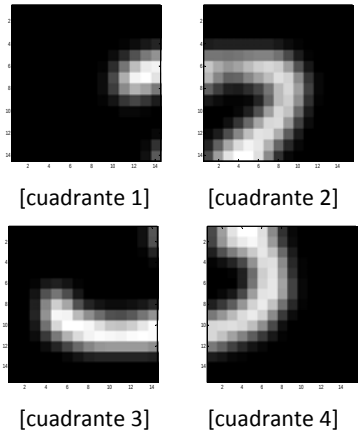


Figura 43. Aproximada del dígito original (figura 42) transformado

Las 4 secciones en las que podemos dividir la matriz aproximada que hemos obtenido al transformar:



[cuadrante 1] [cuadrante 2]

[cuadrante 3] [cuadrante 4]

Figura 44. Las 4 sub-matrices que se originan a partir de dividir la aproximada (figura 43)

Finalmente, para obtener el valor de cada una de las partes que hemos dividido aplicaremos a las mismas la función de entropía con la intención de que nos aporte información acerca de la información (valga la redundancia) que cada una de estas 4 secciones del dígito contienen luego de haber sido transformadas.

Los valores obtenidos para esta componente a partir de aplicar la función que hemos mencionado a la matriz subdividida, en la forma que se muestra en la figura 44, son los siguientes:

0.8303 1.1239 1.1104 1.0358

Como hicimos cuando realizamos el análisis sobre los histogramas resultantes de la cuantización aplicada a la matriz de ángulos, mostraremos a continuación gráficos similares acerca de cómo el aporte de esta característica (generada a partir de las matrices de aproximada de transformar 1500 dígitos para cada clase) nos servirá para ayudar a discriminar la pertenencia de un dígito a la clase a la cual pertenece con un poco mas de certeza. Tanto como para los casos anteriores, para este caso, la wavelet aplicada también es Mallat, y se transforma con un solo nivel de aplicación sobre muestras pertenecientes a la base MNIST (figuras 45 y 46):

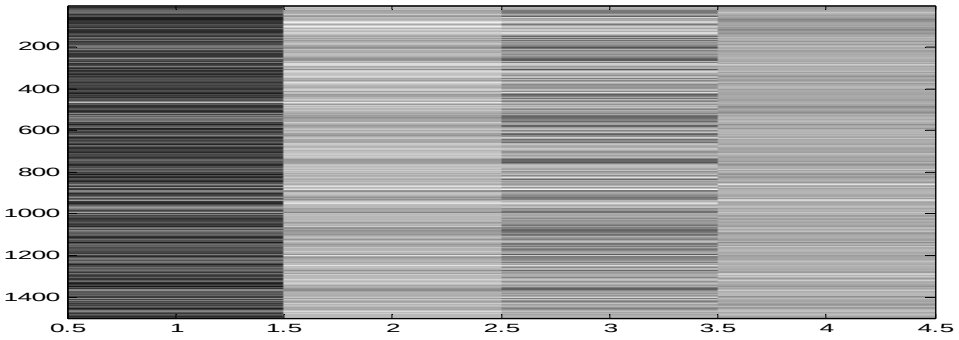


Figura 45. Mapa de entropías para cada uno de los cuadrantes (de la matriz de aproximación) de 1500 dígitos de la clase 1 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

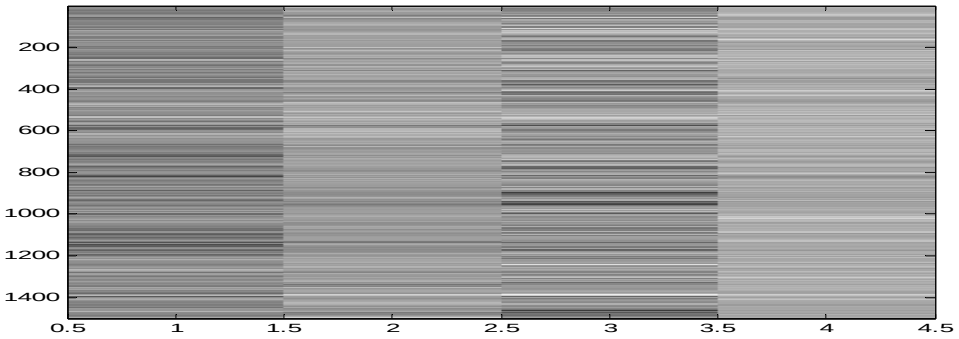


Figura 46. Mapa de entropías para cada uno de los cuadrantes (de la matriz aproximación) de 1500 dígitos de la clase 9 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

Como hicimos en la comparación de la característica generada a partir de los histogramas para las matrices de módulos y ángulos, en la sección anterior, podemos en este caso ver también a simple vista como fácilmente podríamos diferenciar un digito perteneciente a la clase 1 o a la clase 9 en base a los valores de entropía para cada uno de los cuadrantes de la matriz aproximada resultado luego de aplicar la transformada wavelet.

Si observamos los patrones generados a partir de aplicar el mismo análisis a las matrices de módulos para cada uno de estos dígitos que hemos tomado para realizar las muestras anteriores tenemos un resultado similar, donde fácilmente podemos identificar a que clase correspondería cada dígito tomado al azar (figuras 47 y 48):

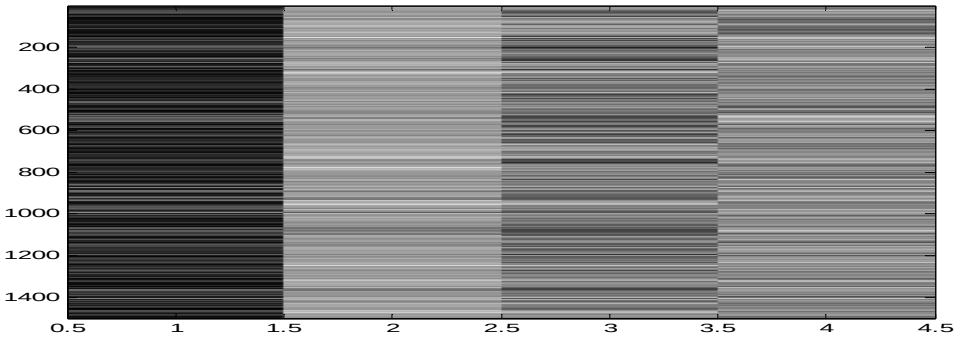


Figura 47. Mapa de entropías para cada uno de los cuadrantes (de la matriz de módulos) de 1500 dígitos de la clase 1 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

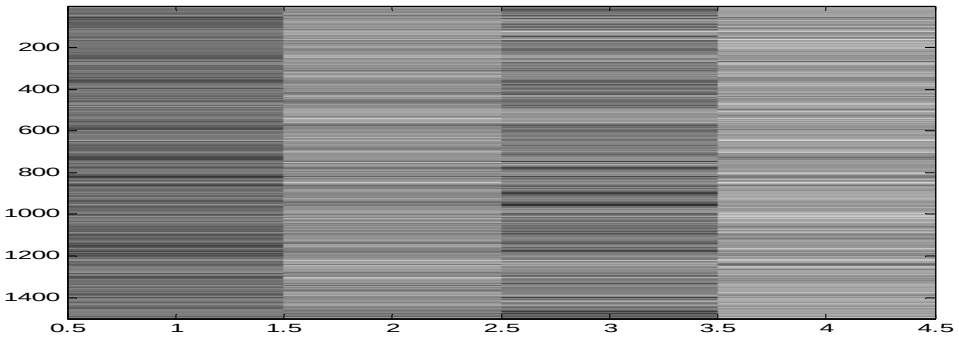


Figura 48. Mapa de entropías para cada uno de los cuadrantes (de la matriz de módulos) de 1500 dígitos de la clase 9 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

Y a continuación podemos también observas este tratamiento aplicado a las matrices de ángulos para los mismos conjuntos de dígitos (figuras 49 y 50):

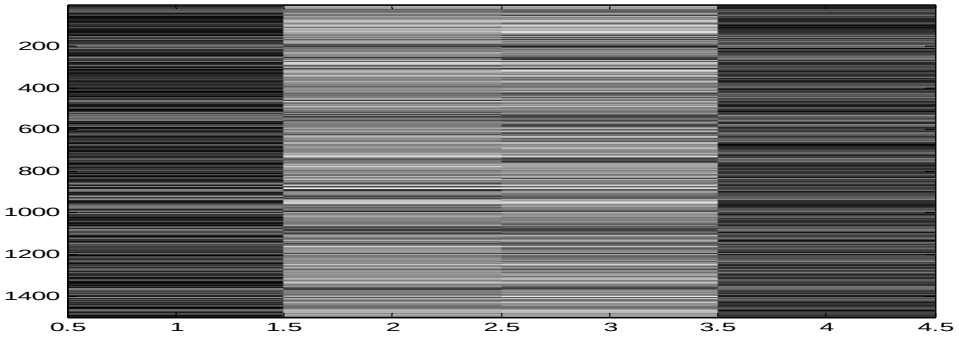


Figura 49. Mapa de entropías para cada uno de los cuadrantes (de la matriz de ángulos) de 1500 dígitos de la clase 1 del conjunto de entrenamiento luego de haber aplicado 1 paso de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

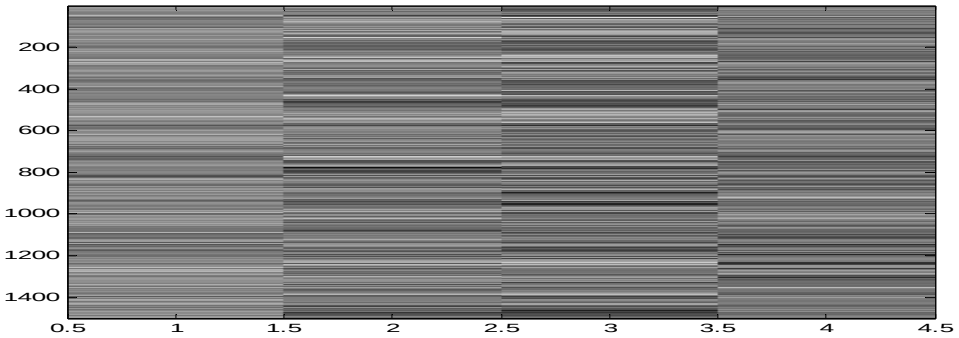


Figura 50. Mapa de entropías para cada uno de los cuadrantes (de la matriz de ángulos)
de 1500 dígitos de la clase 9 del conjunto de entrenamiento luego de haber aplicado 1 paso
de la trasformada wavelet Mallat. Color blanco: valores más grandes, Color negro: valores más pequeños

Se puede observar en los mapas de entropías presentados arriba (figuras 49 y 50) que se presentan diferencias marcadas entre los representantes de estas dos clases pudiendo, en base a ver sus coeficientes de entropía, si un dígito preprocesado de esta manera pertenece a una clase u otra.

Entonces el aporte de estas 3 características (entropía de los cuadrantes de la matriz aproximada, entropías de los cuadrantes de la matriz de módulos y entropías de los cuadrantes de la matriz de ángulos), que total agregarán 12 valores al vector descriptor que estaremos utilizando, nos posibilitarán, junto a las otras características que hemos tratado en este trabajo, determinar con mayor certeza la correcta clasificación de un dígito.

Composición del vector descriptor para las máquinas de soporte vectorial

Como se puede deducir, de todo el análisis que hemos contado anteriormente para cada miembro de los subconjuntos pertenecientes a las diferentes clases de números, los dígitos originales son preprocesados en un intento de magnificar las características que pudieran representar cada clase como componentes de entrenamiento que compondrán al vector con el que trabajará luego la máquina de soporte vectorial. Para el armado de los descriptores se realizaron varias pruebas y comparaciones, buscando obtener la mayor eficacia en el reconocimiento y reducir la dimensión de los mismos para acortar los tiempos de entrenamiento/clasificación.

Presentamos en ésta sección la composición general del vector descriptor que estaremos utilizando en lo que resta del trabajo (mostrando cómo será la estructura general de las componentes que tendrán los vectores descriptores) para entrenamiento y clasificación usando máquinas de soporte vectorial y el armado de diferentes parametrizaciones a las características utilizadas para lograr así una variación sobre el preprocesamiento que estaremos aplicando.

El vector que se utiliza como entrada de las máquinas de soporte vectorial que estemos construyendo, tanto para entrenamiento de la misma como para la posterior clasificación de los dígitos de testeo, está compuesto principalmente por dos partes (o secciones) que serán generadas a partir de diversos procesos que se apliquen a la imagen original. Estas secciones contienen características obtenidas a partir de la aplicación de transformadas wavelets a la imagen original, por lo que dispondrá de una componente de baja resolución y una de alta resolución (correspondiente a cada uno de los

filtros, bajos y altos, que la transformada wavelet aplica en cada paso de cambio de dimensión). El mismo entonces se puede representar con la siguiente figura (para proveer mayor comprensión):

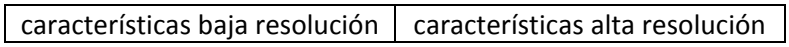


Figura 51. Composición del vector utilizado en la máquina de soporte vectorial

Más allá de la idea de intentar introducir al sistema de clasificación menor y mejor información (para tener buena performance en velocidad y reconocimiento), la transformada wavelet utilizada, como se vio anteriormente, nos genera mayor información intermedia a partir de cada muestra original que luego será utilizada en distintos tipos de parametrizaciones y análisis que se explican durante el desarrollo de este trabajo.

En particular es que al utilizar la transformada wavelet 'à trous' (o sin decimación) obtenemos mas información de la muestra original dado que partimos de una imagen que representa al dígito de tamaño NxN, y la transformación aplicada con cada función wavelet en particular, nos retorna 3 matrices del mismo tamaño donde tenemos información sobre la aproximada (componente de baja resolución obtenida a partir del filtro pasa bajos), los detalles horizontales y los detalles verticales (datos obtenidos a partir del filtro pasa altos).

La matriz con la información de la aproximada resultante de la función de escala la denominaremos como la componente 'aproximada' y será generalmente utilizada sin realizar más tratamiento sobre la misma (aunque podría ser empleada en forma binarizada en algunos de los análisis realizados, y donde así fuera, estará indicado). De esta componente se extraerán las características utilizadas en la primera sección del vector de clasificación (por lo que primera sección del mismo será denominada componente de baja). Y como se menciona en la sección en la que se extrae la entropía de cada uno de los cuadrantes de la matriz de aproximada, también estaremos agregando éstas características en la composición de esta componente.

Las otras dos matrices resultantes son convertidas, en la mayoría de los análisis y parametrizaciones realizadas, a matrices de módulos y ángulos para luego ser tratadas generando histogramas sobre cuantizaciones de las mismas con una cantidad de representantes que varía en las diversas pruebas realizadas en un intento de resumir y acentuar las propiedades de la imagen (precisamente intentando generalizar y caracterizar a la clase de dígitos que la mismas representan). Realizando estos y otros análisis es que se extraerán datos que conformarán la segunda sección del vector de clasificación (por lo que la segunda sección del mismo será denominada componente de alta). También estaremos utilizando para la composición de esta componente las características generadas a partir de tomar la entropía de los cuadrantes de las matrices resultantes.

De modo que en base a todo lo detallado anteriormente el vector utilizado en la máquina de soporte vectorial lo podemos pensar y definir de la siguiente manera:

COMPONENTE DE BAJA	COMPONENTE DE ALTA

Figura 52. Componentes del vector utilizado en la máquina de soporte vectorial

donde a mayor detalle contamos con lo siguiente:

COMPONENTE DE BAJA		COMPONENTE DE ALTA			
MATRIZ AP	ENT QUAD AP	HIST MOD	HIST ANG	ENT QUAD MOD	ENT QUAD ANG

Figura 53. Detalle de las características utilizadas en cada componente del descriptor de la máquina de soporte vectorial

siendo cada una de las referencias anteriores las siguientes:

- MATRIZ AP: la matriz de aproximada resultante de aplicar la transformada wavelet
- ENT QUAD AP: los 4 valores de las entropías de cada uno de los cuadrantes generados a partir de dividir la matriz aproximada (MATRIZ AP)
- HIST MOD: el histograma generado a partir de cuantizar en N intervalos la matriz de módulos (generada a partir de transformar a esta representación los detalles horizontales y verticales generados luego de aplicar la transformada wavelet)
- HIST ANG: el histograma generado a partir de cuantizar en N intervalos la matriz de ángulos (generada a partir de transformar a esta representación los detalles horizontales y verticales generados luego de aplicar la transformada wavelet)
- ENT QUAD MOD: los 4 valores de las entropías de cada uno de los cuadrantes generados a partir de dividir la matriz de módulos
- ENT QUAD ANG: los 4 valores de las entropías de cada uno de los cuadrantes generados a partir de dividir la matriz de ángulos

Capítulo 6 – Clasificación con Máquinas de Soporte Vectorial

Entrenamiento de las máquinas de soporte vectorial (kernels y parámetros)

Presentamos a continuación, a modo de referencia, los dos kernels con los que mayormente estuvimos realizando pruebas, así como sus respectivas parametrizaciones (las que más hemos estado utilizando). Sobre el kernel polinomial, si bien hicimos diversas pruebas con polinomios de grados mayores (hasta 11/13), no hemos logrado obtener resultados tan buenos como los que nos brindaba el kernel radial con parametrizaciones sencillas (es decir realizando algunas pocas variaciones de sus parámetros), y es por esto que hemos puesto el esfuerzo en realizar las pruebas con este último ya que lo consideramos más adecuado para el tratamiento de esta serie de datos con la que estuvimos trabajando.

Kernel polinomial

El kernel polinomial que usamos en este trabajo para transformar el espacio original de datos correspondiente a los descriptores se encuentra definido por la siguiente función:

$$K(a,b) = (s\ a.b + r)^d$$

Y salvo que indiquemos lo contrario los parámetros con los que aplicaremos el mismo serán:

$$s = 1, r = 1, d = 2$$

es decir que estaremos utilizando básicamente polinomios de grado 2.

Kernel radial (o gaussiano)

El kernel radial que usamos en este trabajo para transformar el espacio original de datos correspondiente a los descriptores se encuentra definido por la siguiente función:

$$K(a,b) = e^{-|a-b|^2/\sigma^2}$$

Y salvo que indiquemos lo contrario los parámetros con los que aplicaremos el mismo serán:

$$\sigma = 10$$

Observación: también, y salvo que indiquemos lo contrario, el parámetro de trade-off *C* (33), entre los márgenes del hiperplano generado por la máquina de soporte vectorial, será valuado en 100.

Armado de clasificadores

En nuestro caso diremos que un clasificador es una función C que tomando un dato dígito d como entrada nos retorna a que clase c pertenece el mismo y, en nuestro caso en particular, será una función que tome un dígito cualquiera del conjunto de testeo (y/o provistos por un agente externo) y devuelva, de manera determinante, a que clase pertenece el mismo. En base a esta definición que hemos planteado, y al caso particular que estaremos usando en este trabajo, entonces podemos definirlo de la siguiente manera:

$$C(d) = c, \qquad d \in \{digitos\_testing\}, c \in \{0,..,9\}$$

La función de clasificación es definida a través del análisis de un conjunto de dígitos, llamados de entrenamiento, los que a través de la recopilación de las diferentes características que presentan, nos sirven para categorizar a otros representantes de cada una de las clases. Luego, una vez que se ha armado el clasificador podemos aplicarlo sobre cualquier dígito perteneciente al conjunto de dígitos de testeo (no conocido a priori) para que determine a que clase corresponde. No sabemos a ciencia cierta si la decisión es acertada o no.

Los clasificadores que armaremos en el presente trabajo estarán basados en máquinas de soporte vectorial, apoyados en la teoría al respecto que se mostró en el capítulo 3 sobre las mismas.

Clasificadores basados en máquinas de soporte vectorial para más de dos clases

La aplicación del concepto introducido en la teoría de máquinas de soporte vectorial permite la clasificación binaria, es decir que solo nos permite decidir si una muestra pertenece a la clase que nos interesa o no. En la definición de clasificador que introdujimos en la sección anterior, estamos considerando que un clasificador (como máquina de clasificación que toma un *input* (la muestra) y retorna un *output* (la clase) que es la clasificación de dicho *input*) puede retornar la clase, entre todas las existentes, a la cual el dígito introducido al mismo pertenece. En este caso particular que estamos tratando necesitamos que el dominio de salida de nuestro clasificador se encuentre definido en el conjunto de dígitos enteros que tenemos entre 0 y 9 inclusive, por lo que tendríamos:

El rango de la entrada: $d \in \{digitos_{testing}\}$,
y el rango de la salida: $c \in \{0,..,9\}$

Para poder trabajar con clasificadores binarios, en este caso en particular los generados a partir de la utilización de máquinas de soporte vectorial, de forma de componer o utilizar varios de ellos para tener un clasificador de mayor nivel que en base a la muestra introducida pueda determinar entre N clases a cual pudiera pertenecer, se usan generalmente dos técnicas (entre una variedad de posibilidades) ampliamente conocidas. Estas dos técnicas más utilizadas se comportan como se describe a continuación:

- La aplicación de uno contra todos: en esta técnica se entrenan N clasificadores basados en el concepto de máquinas de soporte vectorial, y luego se procede a clasificar la muestra en todos ellos para determinar a qué clase pudiera pertenecer. En la clasificación de cada uno de estos

clasificadores (que se encuentran entrenados utilizando la clase de interés, contra todo el resto) se obtiene un valor que sirve para ponderar la pertenencia a dicha clase. Este 'peso' es utilizado para maximizar esta noción de pertenencia respecto a la clase y poder quedarnos con la que es considerada la más adecuada (si observamos las ecuaciones referentes a la teoría de máquinas de soporte vectorial (ecuaciones 32 - 35), este valor se encuentra definido por el mayor valor posible para c/u de los α respetando la cota superior definida por C, es decir los vectores de soporte que maximicen el margen entre diferentes máquinas). Podemos ver en la siguiente figura los hiperplanos que se generarían (líneas punteadas en colores) utilizando esta técnica en la clasificación de muestras pertenecientes a 3 clases:

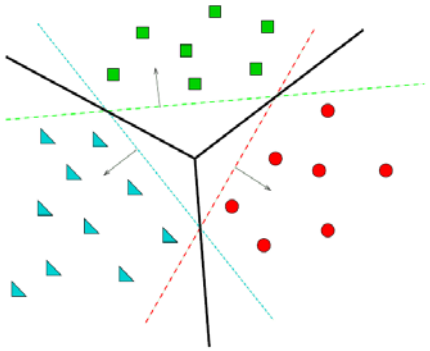


Figura 54. Técnica de clasificación multiclase de uno contra todos

En particular el software que estaremos utilizando para realizar la clasificación multiclase (Torch [15]) de las muestras de las bases de datos seleccionadas aplica esta técnica (armado de clasificadores utilizando una clase contra el resto). Entre las posibles técnicas que se encuentran en la literatura, la seleccionada nos proporciona los mejores resultados en cuanto a la eficacia de clasificación (valores significativos respecto al resto), y el tiempo de clasificación de la muestra, es decir la performance para determinar la clase a la que pertenece un dígito, se encuentra entre valores encima de la media (encontrándose solo la performance del entrenamiento entre las más pobres de las posibles técnicas) [37].

- La aplicación de uno contra uno: en esta técnica se entrenan $\binom{N}{2}$ clasificadores basados en el concepto de máquinas de soporte vectorial, y luego se procede a clasificar la muestra en todos ellos para determinar a qué clase pudiera pertenecer. En la clasificación de cada uno de estos clasificadores (que se encuentran entrenados utilizando la clase de interés, contra alguna de las clases restantes, de manera de cubrir todas contra todas) se toma como clase aquella que mayor cantidad de votos tiene [19][37]. Podemos ver en la siguiente figura los hiperplanos que se generarían (líneas punteadas en negro) utilizando esta técnica en la clasificación de muestras pertenecientes a 3 clases:

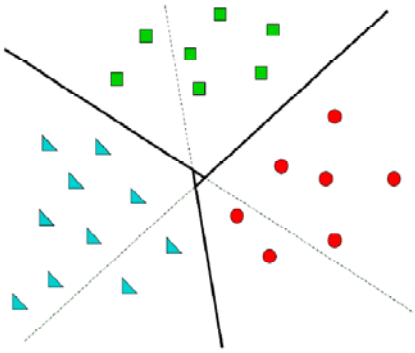


Figura 55. Técnica de clasificación multiclase de uno contra uno

Combinación de múltiples clasificadores

En los últimos años las estrategias basadas en la combinación de múltiples clasificadores crearon gran interés en el tópico de reconocimiento de caracteres. Un largo número de estrategias de combinación han sido exploradas, pero el sistema más simple de votación uniforme logra una robusta y comparable performance frente a sistemas mucho más complejos. La utilización de estas técnicas mejora el porcentaje de dígitos reconocidos eficientemente siendo que el perfecto reconocimiento aun es un problema no resuelto. La perfecta clasificación de estas bases es un problema que no se puede resolver, debido al análisis humano realizado donde la tasa de error de Bayes no es cero [21], nos indica que no es posible lograr certeza absoluta sobre todas las muestras.

En este trabajo exploraremos varias técnicas de combinación de clasificadores [13] que presentamos a continuación:

- Votación uniforme: en éste caso cada clasificador presenta su voto, sin tener preferencia en particular por ninguno, y la clase mayormente votada es la elegida.
- Votación con pesos: cada clasificador tiene un peso para cada una de las clases entrenadas de acuerdo al desempeño del mismo en la clasificación de dígitos de dicha clase, los resultados de la votación están ponderados de acuerdo a estos pesos y la clase que mayor valor tenga será la ganadora.
- Votación uniforme por ranking: en este caso cada clasificador presenta votos priorizados (es decir una lista ordenada de clases a las que cada dígito podría pertenecer) y la clase mayormente elegida de acuerdo a estas prioridades es la ganadora.
- Votación con pesos por ranking: en este caso cada clasificador presenta sus votos priorizados (como en el caso anterior), y además cada uno de ellos tiene un peso para cada una de las clases entrenadas de acuerdo al desempeño del mismo en la clasificación de dígitos de dicha clase, luego la clase que mayor valor tenga (teniendo en cuenta la priorización definida y los pesos aplicados) es la ganadora.

En el presente trabajo estaremos utilizando, para combinar, clasificadores basados en diferentes wavelets, así como distintos niveles de aplicación de las mismas. Como hemos mencionado todos estos clasificadores estarán basados en máquinas de soporte vectorial, lo que podría variar entre ellos, además del descriptor con el que se entrenará cada uno de ellos, podrá ser el kernel del mismo así como diversos parámetros que pudieran aplicárseles.

Votación uniforme

De acuerdo a la definición de cada clasificador como una función que mapea dígitos con clases, podemos decir que la votación uniforme, utilizando la combinación de votos de N clasificadores previamente entrenados, se encuentra definida por una función $CC(d)$ tal que $CC(d) \mapsto c, c \in \{0,..9\}$ para todo dígito d que se desee clasificar, de la siguiente manera:

$$CC(d) = \max_i\{V_i\} \ \forall i \in \{0,..9\} \tag{47}$$

Donde $V_i = \sum_{j=1}^{N} B(d,i,j)$,

Siendo $B(d,i,j) = \begin{cases} 1 & si\ C_j(d) = i \\ 0 & otro\ caso \end{cases}$,

y el conjunto de clasificadores definido por $\{C_j\},\ j \in \{1,..N\}$

Es posible que tengamos $V_i = V_j, i \neq j$ y estos sean máximos, es decir que no podríamos determinar con exactitud a que clase corresponde el dígito analizado; otra técnica debería ser utilizada. No obstante esta técnica es igualmente efectiva en comparación con otras más elaboradas como mencionamos anteriormente. Luego de la enumeración y descripción de las distintas combinaciones de clasificadores realizadas presentaremos, en el próximo capítulo, los resultados que cada uno de ellos nos retorna sobre los conjuntos de testeo de las bases que hemos utilizado.

Votación con pesos

La definición de la función que combina estos clasificadores es muy similar a la anterior, pero además de tener como entradas para el armado de la función de clasificación combinada al conjunto de N clasificadores que queremos utilizar, también conocemos los pesos que los mismos presentan para reconocer cada una de las clases con eficiencia; es decir que tenemos una función $W(C,c) = w$ que para cada clasificador C nos indica cual es su peso y/o eficacia w en el reconocimiento de la clase c.
De esta manera podemos definir esta combinación de clasificadores de la siguiente manera:

$$CC(d) = \max_i\{V_i\}\ \forall i \in \{0,..9\} \quad (48)$$

Donde $V_i = \sum_{j=1}^{N} B(d,i,j)$,

Siendo $B(d,i,j) = \begin{cases} W(C_j,i) & si\ C_j(d) = i \\ 0 & en\ otro\ caso \end{cases}$,

y el conjunto de clasificadores definido por $\{C_j\},\ j \in \{1,..N\}$

En este caso estamos sumando los pesos (o eficiencia) de cada clasificador sobre la clase que eligió, de esta manera estamos teniendo en cuenta la 'buena' performance que el mismo tiene sobre cada una de las clases para tomar su decisión.

Como en el caso anterior también es factible que tengamos $V_i = V_j, i \neq j$ y estos sean máximos, es decir que no podríamos determinar con exactitud a que clase corresponde el dígito analizado; otra técnica debería ser utilizada, pero la probabilidad de que esto suceda disminuye con respecto al ejemplo anterior ya que los pesos en este caso no son uniformes como sucedía anteriormente.

Votación uniforme por ranking

Para poder definir este tipo de combinación necesitamos primero modificar la definición de clasificador que habíamos presentado anteriormente. En este caso debemos 'extender' el funcionamiento de cada clasificador diciendo que éste nos retorna a partir de un dígito d tomado como entrada, un conjunto de vectores de 2 dimensiones donde la primer componente nos representa la clase a la que podría pertenecer, y la segunda componente nos indica la prioridad (en orden descendente, como si fuera una especie de escala de confiabilidad que va del 10 al 1) con que el dígito pertenecería a dicha clase. Es decir que si tomamos el n-ésimo dígito del conjunto de testeo d_n y este perteneciera con mayor probabilidad a la clase 7 que a cualquier otra clase, entonces el vector (7, 10) estaría en el conjunto de resultados de evaluar d_n y el resto de los vectores no tendría ninguno de estos valores ni en la primera ni en la segunda componente respectivamente. Luego podemos definir para cada clasificador Cx a una función $P(Cx, d, c) = p,\ d \in \{digitos_{testing}\}, c \in \{0,..9\}, p \in \{1,..10\}$ que dependiendo del clasificador, del dígito seleccionado y de la clase que se quiera evaluar, nos determina la prioridad p, en forma unívoca, con que dicho dígito d podría pertenecer a la clase c si utilizáramos el clasificador Cx (es decir arma una lista ordenada de clases a las que podría pertenecer el dígito de entrada de acuerdo a su entrenamiento). De acuerdo a esto, la nueva definición que necesitamos para extender estos clasificadores que utilizamos en este caso, es la siguiente:

$$Cx(d) = \{v_0,..v_9\}, \qquad d \in \{digitos\_testing\}$$

Siendo cada $v_i = (i, P(Cx, d, i))$ donde P es biyectiva

Con la nueva definición que acabamos de introducir para estos 'clasificadores extendidos' podemos definir ahora a la función $CCx(d)$ que clasifica realizando la combinación de los mismos en la forma que presentamos a continuación:

$$CCx(d) = \max_i\{V_i\}\ \forall i \in \{0,..9\} \tag{49}$$

Donde $V_i = \sum_{j=1}^{N} P(Cx_j, d, i)$,

y el conjunto de clasificadores definido por $\{Cx_j\},\ j \in \{1,..N\}$

Como en el caso anterior también es factible que tengamos $V_i = V_j, i \neq j$ y estos sean máximos, es decir que no podríamos determinar con exactitud a que clase corresponde el dígito analizado; otra técnica debería ser utilizada, pero la probabilidad de que esto suceda disminuye al aumentar la complejidad del clasificador con la introducción del ranking que determina el orden en que un clasificador determina la pertenencia de un dígito con respecto a las clases evaluadas.

Votación con pesos por ranking

La definición de la función que combina estos clasificadores es muy similar a la anterior, pero además de tener como entradas para el armado de la misma al conjunto de N clasificadores que queremos utilizar y la función que determina las prioridades con que cada uno de los dígitos pertenecen a cada una de las clases, también conocemos los pesos que los mismos presentan para reconocer cada una de las clases con eficiencia; es decir que tenemos una función $W(C, c) = w$ que para cada clasificador C nos indica cual es su peso y/o eficacia w en el reconocimiento de la clase c.

Ahora, utilizando la definición introducida previamente sobre los 'clasificadores extendidos' podemos definir a la función $CCx(d)$ que clasifica realizando la combinación de estos y considerando también la función de pesos como dato adicional de la siguiente manera:

$$CCx(d) = \max_i\{V_i\} \ \forall i \in \{0,..9\} \tag{50}$$

Donde $V_i = \sum_{j=1}^{N} P(Cx_j, d, i)W(Cx_j, i)$,

y el conjunto de clasificadores definido por $\{Cx_j\},\ j \in \{1,..N\}$

Como en el caso anterior también es factible que tengamos $V_i = V_j, i \neq j$ y estos sean máximos, es decir que no podríamos determinar con exactitud a que clase corresponde el dígito analizado; otra técnica debería ser utilizada, pero la probabilidad de que esto suceda disminuye al aumentar la complejidad del clasificador con la introducción del ranking que determina el orden en que un clasificador determina la pertenencia de un dígito con respecto a las clases evaluadas y la utilización de pesos basados en la eficacia del clasificador para el reconocimiento de cada una de las clases. Si bien la probabilidad de que suceda que tengamos para un dígito dos o más posibilidades disminuye a medida que fuimos aumentando la complejidad de combinación de clasificadores, esto no quiere decir automáticamente que los últimos son mucho más eficaces que los primeros (hecho que se verá reflejado cuando presentemos luego los resultados para cada una de estas técnicas sobre los conjuntos de testeo de las bases utilizadas).

Capítulo 7 – Resultados

Introducción

En este capítulo mostramos resultados concretos aplicando los métodos expuestos previamente al set de datos de la base CENPARMI y MNIST. La utilización y resultados arrojados al realizar este preprocesamiento del conjunto inicial se compara respecto a la no utilización de algún tipo de preprocesamiento (es decir a la utilización del conjunto de datos inicial tal cual es suministrado). Mostramos las distintas eficacias de los clasificadores individuales armados con el preprocesamiento que hemos seleccionado, y luego las eficacias que logramos con las distintas técnicas de combinación de estos clasificadores entrenados anteriormente para obtener los mejores resultados de este trabajo.

No quedándonos solo con esta posible técnica de combinación de clasificadores y/o resultados individuales, también comparamos dicha técnica, en cuanto a eficacia y tiempo computacional, contra una posible técnica que sería la de trabajar con un descriptor que es compuesto por los descriptores utilizados para entrenar cada uno de los clasificadores anteriores.

Por último estamos presentando nuestros mejores resultados y comparándolos contra los resultados encontrados en otras publicaciones describiendo brevemente las técnicas utilizadas en cada caso de mayor complejidad para obtener índices de eficacia de a lo sumo 1% más efectivos.

Clasificación (y entrenamiento) sin preprocesamiento

En las siguientes tablas (1 y 2) mostramos los porcentajes de eficacia en la clasificación sin preprocesar el conjunto de datos de entrada (refiriéndonos a ambos subconjuntos: entrenamiento y testeo), tanto para patrones de la base CENPARMI como para patrones de la base MNIST. En este caso en particular se aplica un kernel radial (con parámetros $\sigma = 10, C = 100$) para tener una idea general sobre los resultados obtenidos aplicando directamente este kernel sobre los patrones sin preprocesar y poder ver luego la eficacia lograda aplicando el mismo tipo de kernel luego de realizar preprocesamiento de los datos originales. En este caso, que tomaremos como ejemplo comparativo (ya que los otros kernels y parametrizaciones posibles se comportan de manera similar), tenemos los siguientes resultados:

BASE DE PATRONES	EFICACIA EN RECONOCIMIENTO
CENPARMI	90.80%
MNIST	N/A (varios días training)
MNIST [normalizado]	98.43%

Tabla 1. Eficacia en el reconocimiento del kernel radial sin preprocesar las muestras de los conjuntos de testeo analizados

Nota: observar que los patrones de la base MNIST fueron 'normalizados' [36] (utilizando una función basada en transformaciones lineales y traslaciones) para poder obtener esta eficacia con el kernel utilizado (sino los tiempos requeridos eran extremadamente alto no consiguiendo resultados adecuados). En lo que resta del trabajo utilizaremos esta base MNIST 'normalizada' como punto de partida para el entrenamiento y la clasificación.

Clasificación (y entrenamiento) aplicando un preprocesamiento particular

Para analizar los resultados obtenidos aplicando el método de preprocesamiento que definimos en los capítulos anteriores es que parametrizamos en forma particular cada uno de los diferentes componentes para obtener el descriptor con los que entrenamos y testeamos el sistema de clasificación. Si bien cada una de estas componentes puede ser parametrizada de distintas maneras, por ejemplo binarizando los valores que la componen, cuantizando en más o menos intervalos en los casos que usemos histogramas, agregando o quitando componentes, entre otras, nos hemos quedado con la parametrización que nos ha arrojado los mejores resultados en las pruebas que realizamos para mostrar los conceptos y técnicas aquí planteados.

El descriptor que utilizaremos a partir de este momento está definido por la siguiente parametrización particular:

- Se utiliza la aproximada de la imagen en forma submuestreada (una vez realizada la transformada wavelet). Esta componente aportará ¼ de elementos al descriptor preprocesado, en comparación al descriptor generado a partir de la imagen original.
- La matriz anterior es a su vez dividida en 4 matrices iguales y se calcula la entropía de cada una de ellas como subcomponente adicional que suma a la componente de baja.
- Se utiliza el histograma de ángulos previamente cuantizados en 16 intervalos (esta componente aportará solo 16 elementos al descriptor preprocesado), con los siguientes detalles particulares:
    - Se cuantiza en el intervalo $[-\pi, +\pi]$.
    - Se utiliza un umbral para tomar en cuenta cada ángulo a partir de un módulo mínimo definido por el promedio de los módulos cuyo valor absoluto es distinto de 0, se toma como cota inferior la mitad de este valor (es decir un 50%).
    - Se normaliza el histograma de ángulos generado utilizando el representante de valor máximo del mismo.
- Se utiliza el histograma de módulos previamente cuantizados en 16 intervalos (esta componente aportará solo 16 elementos al descriptor preprocesado), con los siguientes detalles particulares:
    - Se normaliza el histograma de módulos generado utilizando el representante de valor máximo del mismo.
- Las dos matrices anteriores son a su vez divididas en 4 matrices iguales para calcular la entropía de cada una de ellas como subcomponentes adicionales que sumarán a la llamada componente de alta del vector descriptor.

Descriptores CENPARMI

En base a lo descripto previamente la composición del descriptor preprocesado para los patrones de la base CENPARMI presenta la siguiente estructura:

COMPONENTE DE BAJA		COMPONENTE DE ALTA			
MATRIZ AP	ENT QUAD AP	HIST MOD	HIST ANG	ENT QUAD MOD	ENT QUAD ANG
{64 elementos}	*{4 elementos}*	*{16 elementos}*	*{16 elementos}*	*{4 elementos}*	*{4 elementos}*

Figura 56. Detalle del descriptor generado para muestras de la base CENPARMI

Por lo tanto el descriptor preprocesado para los dígitos de la base CENPARMI tiene una dimensión de 108 elementos (significativamente menor que la cantidad de elementos [dimensión] que

contiene el descriptor generado a partir de los datos originales sin aplicar ningún tipo de preprocesamiento: 256 elementos).

Descriptores MNIST

En base a lo descripto previamente la composición del descriptor preprocesado para los patrones de la base MNIST presenta la siguiente estructura:

COMPONENTE DE BAJA			COMPONENTE DE ALTA			
MATRIZ AP	ENT QUAD AP		HIST MOD	HIST ANG	ENT QUAD MOD	ENT QUAD ANG
{196 elementos}	*{4 elementos}*		*{16 elementos}*	*{16 elementos}*	*{4 elementos}*	*{4 elementos}*

Figura 57. Detalle del descriptor generado para muestras de la base MNIST

Por lo tanto el descriptor preprocesado para los dígitos de la base MNIST tiene una dimensión de 240 elementos (significativamente menor que la cantidad de elementos [dimensión] que contiene el descriptor generado a partir de los datos originales sin aplicar ningún tipo de preprocesamiento: 784 elementos).

Armado de clasificadores individuales

En esta sección mostramos los resultados que nos arrojan el armado de máquinas de soporte vectorial en forma individuales utilizando distintas wavelets (para realizar el preprocesamiento) y diferentes kernels (para realizar el entrenamiento). Nos limitaremos a mostrar solo algunas simples parametrizaciones (es decir no mostraremos muchas de las posibles combinaciones que se pudieran generar a partir de las infinitas parametrizaciones posibles) ya que, como dijimos previamente, el objetivo de este trabajo no es encontrar o descubrir la mejor combinación de kernel/parámetros para cada una de las bases utilizadas, sino el presentar técnicas y enfoques de preprocesamiento, transformaciones del espacio de datos, combinación de resultados individuales, etc. como métodos para obtener mejores resultados en forma simple y sencilla sin tener un alto impacto o penalidad en performance a partir de un mismo set de datos iniciales.

Para el armado de estos clasificadores, para cada una de las bases utilizadas, estamos aplicando las siguientes técnicas en cada uno de los casos:

- En el caso de las muestras de la base CENPARMI, estamos tomando el 100% del conjunto de entrenamiento, y estamos generando (a partir del mismo) un subconjunto de 'nuevos' dígitos utilizando una técnica de suma de dígitos consecutivos (de a 20, con lo que estamos obteniendo un 5% de una 'nueva' cantidad de muestras) y cada dígito 'nuevo' se está umbralando para lograr que tenga una apariencia parecida a los dígitos originales desde los cuales se partió [31][33]. Este 5% generado nos servirá para obtener los pesos en base a la clasificación que realiza el clasificador ya entrenado sobre este conjunto de datos desconocido. Estos pesos son los que utilizaremos luego en la combinación de clasificadores donde introduzcamos los conceptos de pesos en la votación de cada uno de ellos.
- En el caso de las muestras de la base MNIST, estamos tomando el 95% del conjunto de entrenamiento, reservando el restante 5% para obtener los pesos del clasificador respecto a cada una de las clases que debe reconocer, para poder utilizarlos luego cuando realicemos la

combinación de clasificadores. En particular los tipos de combinaciones que se basan en estos pesos para ponderar su votación.

Si bien para ambas bases de datos probamos estas dos posibles formas de entrenar los clasificadores (y también otras parametrizaciones con diferentes porcentajes de generación y extracción de subconjuntos, umbrales en los casos donde aplicara, binarizado, etc.) nos hemos quedado para cada uno de los casos con la mejor forma de realizar el entrenamiento en base a los resultados que nos han producido.

En todos los casos la composición del vector generado, así como su dimensión, no cambia y la estructura es generada como fue definido en la sección previa. Es decir que para cada una de las bases que estaremos utilizando la cantidad de elementos que componen a cada descriptor es de 108 para patrones de la base CENPARMI y de 240 para patrones de la base MNIST.

Kernel polinomial

En las siguientes tablas (2 y 3) mostramos los resultados que obtuvimos aplicando un kernel polinomial de distintos grados, tanto para los patrones de la base CENPARMI como para los patrones de la base MNIST. Como hemos mencionado hemos focalizado en trabajar con el kernel radial al obtener los mejores resultados sobre las distintas wavelets y preprocesamientos utilizados, no obstante, ponemos aquí a modo de ejemplo algunos de los resultados obtenidos aplicando este tipo de kernel en distintos grados sobre el conjunto de patrones obtenidos a partir de aplicar el preprocesamiento que hemos definido pero de una misma wavelet (la que mejor resultado nos da utilizando un kernel radial) para tener un punto de referencia respecto al kernel que estaremos utilizando en la próxima sección.

Patrones de la base CENPARMI

GRADO DEL POLINOMIO	1 NIVEL DE TRANSF MALLAT
GRADO 2	93.60%
GRADO 3	93.75%
GRADO 4	*93.90%*
GRADO 5	93.75%
GRADO 6	87.20%
GRADO 7	20.20%

Tabla 2. Eficacia en el reconocimiento aplicando kernel polinomial al conjunto CENPARMI preprocesado aplicando un paso de la transformada wavelet Mallat

Patrones de la base MNIST

GRADO DEL POLINOMIO	1 NIVEL DE TRANSF MALLAT
GRADO 2	98.55%
GRADO 3	*98.60%*
GRADO 4	51.02%
GRADO 5	10.10%
GRADO 6	9.80%
GRADO 7	9.80%

Tabla 3. Eficacia en el reconocimiento aplicando kernel polinomial al conjunto MNIST preprocesado aplicando un paso de la transformada wavelet Mallat

Para polinomios de grado mayor no obtenemos buenos resultados por eso es que hemos realizado pruebas solo hasta el uso de polinomios de grado 7.

Como veremos a continuación los resultados utilizando un kernel radial, con la parametrización definida anteriormente, son en todos los casos superiores a los resultados de eficacia que tenemos aquí, más la dificultad de parametrizar el grado del polinomio y el coeficiente *s* (definido anteriormente donde mostramos las fórmulas de los dos kernels que estamos utilizando), y es por esta razón que hemos decidido continuar con la mayoría de pruebas que hemos realizado basándonos en el kernel radial que estamos presentando a continuación.

Kernel radial

En las siguientes tablas (4 y 5) mostramos los resultados (valores de eficacia en el reconocimiento de los patrones) que obtuvimos aplicando un kernel radial, con la parametrización definida previamente, tanto para los patrones de la base CENPARMI como para los patrones de la base MNIST. Este kernel, al devolver los mejores resultados [4][34], fue el más utilizado durante el resto de las pruebas, y luego de haber realizado varias pruebas de cross-cutting (variando los parámetros que el mismo tiene), y logrando una eficacia suficientemente satisfactoria a nuestros objetivos, hemos luego puesto el foco en las parametrizaciones de procesamiento. Presentamos los índices de eficacia que este kernel nos proporcionó sobre el preprocesamiento que hemos seleccionado para los patrones de ambas bases de dígitos elegidas:

Patrones de la base CENPARMI

TRANSFORMADA UTILIZADA	1 NIVEL DE TRANSF	2 NIVELES DE TRANSF	3 NIVELES DE TRANSF
DAUBECHIES 9/7	*94.60%*	*93.65%*	*89.20%*
MALLAT	94.20%	92.75%	87.00%
SYMMLET	94.15%	94.30%	88.10%

Tabla 4. Eficacia en el reconocimiento de patrones de la base CENPARMI utilizando un preprocesamiento con distintas wavelets y distintos niveles para el kernel radial

Patrones de la base MNIST

TRANSFORMADA UTILIZADA	1 NIVEL DE TRANSF	2 NIVELES DE TRANSF	3 NIVELES DE TRANSF
DAUBECHIES 9/7	98.62%	98.49%	97.66%
MALLAT	*98.84%*	*98.66%*	*97.36%*
SYMMLET	98.83%	98.68%	97.20%

Tabla 5. Eficacia en el reconocimiento de patrones de la base MNIST utilizando un preprocesamiento con distintas wavelets y distintos niveles para el kernel radial

Armado de pruebas combinando clasificadores

Para generar los siguientes resultados a partir de las técnicas de combinación de clasificadores que presentamos anteriormente, utilizaremos los datos obtenidos por cada clasificador individual en la parametrización definida al inicio del capítulo 6 realizada sobre el kernel radial (es decir que se podrán tomar a modo de comparación los resultados mostrados en la sección anterior para este kernel y cada uno de los clasificadores individuales armados). Cada uno de los resultados arrojados por las máquinas de soporte vectorial entrenadas en la sección anterior es combinado en diversas formas para mejorar la eficacia de reconocimiento en la utilización de preprocesamientos similares (o no) y las técnicas de votación que hemos presentado. Los resultados que obtenemos a partir de las distintas técnicas de combinación más la elección de que clasificadores combinamos para cada uno de los casos serán los que estaremos mostrando a continuación.

Pondremos nombres a los clasificadores que hemos armado (a modo de códigos) y serán los que estaremos utilizando en las tablas de combinaciones para ver con cuáles de ellos se armo cada una de las combinaciones que se probaron y los resultados arrojados. Entonces los clasificadores generados en la sección anterior, tanto para CENPARMI como para MNIST, serán llamados en forma abreviada de la siguiente manera:

- DAUB_1: clasificador generado con la wavelet Daubechies 9/7 aplicando un nivel de transformada wavelet
- DAUB_2: clasificador generado con la wavelet Daubechies 9/7 aplicando dos niveles de transformada wavelet
- DAUB_3: clasificador generado con la wavelet Daubechies 9/7 aplicando tres niveles de transformada wavelet
- MALLAT_1: clasificador generado con la wavelet Mallat aplicando un nivel de transformada wavelet
- MALLAT_2: clasificador generado con la wavelet Mallat aplicando dos niveles de transformada wavelet
- MALLAT_3: clasificador generado con la wavelet Mallat aplicando tres niveles de transformada wavelet
- SYMM_1: clasificador generado con la wavelet Symmlet aplicando un nivel de transformada wavelet
- SYMM_2: clasificador generado con la wavelet Symmlet aplicando dos niveles de transformada wavelet
- SYMM_3: clasificador generado con la wavelet Symmlet aplicando tres niveles de transformada wavelet

Y agregamos los siguientes códigos para combinaciones de los mismos:

- ALL_1: DAUB_1 © MALLAT_1 © SYMM_1
- ALL_2: DAUB_1 © DAUB_2 © MALLAT_1 © MALLAT_2 © SYMM_1 © SYMM_2
- ALL_3: DAUB_1 © DAUB_2 © DAUB_3 © MALLAT_1 © MALLAT_2 © MALLAT_3 © SYMM_1 © SYMM_2 © SYMM_3

Nota: usamos el símbolo © para indicar la combinación de clasificadores.

Hay que tener presente que estos clasificadores son los entrenados utilizando el kernel radial para las máquinas de soporte vectorial que fueron generadas.

Implementación de combinación de clasificadores

Para poder realizar la combinación de clasificadores según lo descripto en el capítulo anterior, desarrollamos un software a tal efecto que se encuentra construido en el ejecutable '*svmmulticlass.exe*'. Este software fue implementado en C# y a pesar de que la implementación es propietaria, las técnicas en las que se encuentra basado son las que fueron presentadas. En cada uno de los modos de combinación aclararemos si fuera necesario detalles particulares de su implementación para tener una idea de su funcionamiento.

En el mismo se han implementado las 4 técnicas de combinación que presentamos en este trabajo, y ejecutando el mismo sin parámetros se obtiene ayuda sobre la utilización del mismo:

```
===>> SVM Multi Classifiers' Combinator -  mk 2009 - v 2.1 <<===
usage: svmmulticlass <voting mode> <path to files> <filenames, ...>
* voting mode:
      .0 uniform
      .1 weights
      .2 ranking
      .3 ranking and weights
* path to files: directory (path) where files are
* filenames: 1 or more file names separated by blank space
```

También para poder realizar estas pruebas, el software utilizado para entrenar y clasificar basado en máquinas de soporte vectorial (Torch [15]) fue modificado para poder extraer información que fue utilizada para armar los esquemas de pesos y rankings utilizados en las técnicas de combinación que hemos aplicado. En particular se agregaron los siguientes parámetros, particularmente al ejecutable '*svmtest.exe*' que fue reconstruido a tal efecto:

```
.
.
* Output options:
*   -oa <file>   -> write in ASCII the SVM output into <file>
*   -ob <file>   -> write in binary the SVM output into <file>
*   -rxd <file>  -> write CSV file with results per descriptor into <file>
*   -rxc <file>  -> write CSV file with results per class into <file>
*   -rxr <file>  -> write CSV multirank [3 classes] file values <file>
*
============>> SVMTorch is (c) Ronan Collobert 2001 (IDIAP & DIRO U. de Montreal) <<============
```

Combinación uniforme de clasificadores

La combinación de clasificadores de esta manera (ecuación 47) es la más simple de todas, y es considerando que todos los clasificadores involucrados en la decisión tienen el mismo peso al realizar la votación. De esta manera estamos tomando el conjunto de muestras de testeo y estamos, para cada uno de los dígitos que pertenecen a la misma, viendo cual fue la clase que más votos tuvo según lo que cada clasificador dice y nos estaremos quedando con dicha clase.

A modo de ejemplo, si tenemos un dígito del conjunto de testeo y usamos tres clasificadores en la combinación, podemos tener las siguientes opciones: los 3 clasificadores votaron por la misma clase, dos clasificadores votaron por la misma clase o ninguno de los clasificadores votó por la misma clase (en este caso nos estamos quedando en forma determinística con la votación que arroja el 1er clasificador). Los resultados de combinar los clasificadores armados para el kernel radial con la parametrización ya explicada son los que se presentan en las siguientes tablas (6 y 7):

Patrones de la base CENPARMI

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	94.25%	2	1
DAUB_1 © DAUB_2 © DAUB_3	94.50%	3	1
MALLAT_1 © MALLAT_2	93.20%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	94.00%	3	1
SYMM_1 © SYMM_2	94.45%	2	1
SYMM_1 © SYMM_2 © SYMM_3	94.60%	3	1
ALL_1	94.55%	3	3
ALL_2	94.50%	6	3
ALL_3	*94.90%*	*9*	*3*

Tabla 6. Resultados de combinar los clasificadores individuales CENPARMI (tabla 4) para la votación uniforme

Patrones de la base MNIST

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	98.54%	2	1
DAUB_1 © DAUB_2 © DAUB_3	98.60%	3	1
MALLAT_1 © MALLAT_2	98.71%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	98.84%	3	1
SYMM_1 © SYMM_2	98.78%	2	1
SYMM_1 © SYMM_2 © SYMM_3	98.71%	3	1
ALL_1	98.79%	3	3
ALL_2	*98.86%*	*6*	*3*
ALL_3	98.81%	9	3

Tabla 7. Resultados de combinar los clasificadores individuales MNIST (tabla 5) para la votación uniforme

Combinación de clasificadores teniendo en cuenta la habilidad para reconocer una clase

La combinación de clasificadores realizada de esta manera (ecuación 48) es similar a la uniforme, pero contamos con los pesos obtenidos al realizar clasificación sobre un subconjunto generado a partir del conjunto de entrenamiento total y otro sobre un subconjunto extraído del conjunto de entrenamiento total para cada uno de los casos CENPARMI y MNIST respectivamente (como se contó en el armado de clasificadores individuales). Luego consideramos que cada clasificador involucrado en la decisión vota por la clase a la que considera que pertenece el dígito evaluado, pero con el peso que le corresponde (los calculados anteriormente teniendo en cuenta que tan bueno es el clasificador para reconocer esa clase respecto al resto de las otras). De esta manera estamos tomando el conjunto de muestras de testeo y estamos, para cada uno de los dígitos que pertenecen a la misma, viendo cual fue la clase que más peso tuvo según lo que cada clasificador dice y nos estamos quedando con dicha clase.

A modo de ejemplo, si tenemos un dígito del conjunto de testeo y usamos tres clasificadores, podemos tener las siguientes opciones: los 3 clasificadores votaron por la misma clase, por lo que esa clase contendrá un peso distinto de cero y el resto de las clases serán 0, en cambio si dos clasificadores votaron por la misma clase, tendremos que esa clase tendrá más peso que la que haya votado el clasificador restante, salvo que los primeros clasificadores sean muy malos decidiendo sobre la clase que eligieron y el otro clasificador sea muy bueno decidiendo sobre la clase que eligió él, en este caso muy particular podría ser que la votación de este último sea más acertada y estaríamos tomando ésa, y por último ninguno de los clasificadores votó por la misma clase (en este caso nos estamos quedando en forma determinística con la votación que arroja el 1er clasificador, si los pesos fueren iguales, sino el clasificador que mayor habilidad, de acuerdo a su peso en la selección de dicha clase, tenga será el que determine la elección de la clase correspondiente).

Los resultados de combinar los clasificadores armados para el kernel radial con la parametrización ya explicada son los que se presentan en las siguientes tablas (8 y 9):

Patrones de la base CENPARMI

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	94.70%	2	1
DAUB_1 © DAUB_2 © DAUB_3	94.75%	3	1
MALLAT_1 © MALLAT_2	94.15%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	94.50%	3	1
SYMM_1 © SYMM_2	94.15%	2	1
SYMM_1 © SYMM_2 © SYMM_3	*95.00%*	*3*	*1*
ALL_1	94.60%	3	3
ALL_2	94.55%	6	3
ALL_3	94.90%	9	3

Tabla 8. Resultados de combinar los clasificadores individuales CENPARMI (tabla 4) para la votación con pesos por clase

Patrones de la base MNIST

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	98.60%	2	1
DAUB_1 © DAUB_2 © DAUB_3	98.60%	3	1
MALLAT_1 © MALLAT_2	98.69%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	98.80%	3	1
SYMM_1 © SYMM_2	98.70%	2	1
SYMM_1 © SYMM_2 © SYMM_3	98.76%	3	1
ALL_1	98.82%	3	3
ALL_2	98.78%	6	3
ALL_3	*98.83%*	*9*	*3*

Tabla 9. Resultados de combinar los clasificadores individuales MNIST (tabla 5) para la votación con pesos por clase

Combinación de clasificadores usando conceptos de ranking

La combinación de clasificadores para este caso (ecuación 49) se realiza permitiendo que cada clasificador nos retorne por cada dígito que analiza una lista donde prioriza las clases a las que supone que el mismo pertenece. En la implementación que hemos realizado, estamos permitiendo que esta lista contenga como máximo 3 clases, y luego sería tan simple como asignar pesos (fijos, no generados a partir de ninguna clasificación como en el caso donde se tienen en cuentan los pesos pero aplicados a la eficacia del clasificador sobre cada una de las clases) según el orden en que se encuentre la clase que nos ha retornado en la lista de ellas. De esta manera estamos tomando el conjunto de muestras de testeo y estamos, para cada uno de los dígitos que pertenecen a la misma, viendo cual fue la clase que más 'peso' tuvo (la 'preferida') según lo que cada clasificador dice y nos estamos quedando con dicha clase.

Entonces para realizar esta combinación hemos decidido tomar como 'pesos', según el orden en que el clasificador nos retorna la lista de clases de acuerdo a la que mayor preferencia tiene hasta la menor. Al ser esta última un ajuste de parámetros realizado de manera 'manual', por decirlo de alguna manera, es un tema sobre el que se podría seguir investigando y utilizando diferentes criterios de ajuste para lograr distintos resultados.

A modo de ejemplo, si tenemos un dígito del conjunto de testeo y usamos tres clasificadores, podemos tener múltiples opciones, y no contaremos o describiremos cada una de ellas aquí, pero diremos brevemente lo siguiente: si los 3 clasificadores votaron por la misma clase en primer lugar, esta será la elegida, en cambio si dos clasificadores votaron por la misma clase en primer lugar, tendremos que esa será la clase elegida, y por último si ninguno de los clasificadores votó por la misma clase en primer lugar, pero 3 o 2 de ellos votaron por la misma clase en segundo lugar, entonces esa clase sumará 6 o 4, respectivamente, por lo que será la elegida, y así.

Los resultados de combinar los clasificadores armados para el kernel radial con la parametrización ya explicada son los que se presentan en las siguientes tablas (10 y 11):

Patrones de la base CENPARMI

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	94.20%	2	1
DAUB_1 © DAUB_2 © DAUB_3	94.50%	3	1
MALLAT_1 © MALLAT_2	93.45%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	94.25%	3	1
SYMM_1 © SYMM_2	94.50%	2	1
SYMM_1 © SYMM_2 © SYMM_3	94.60%	3	1
ALL_1	94.35%	3	3
ALL_2	94.80%	6	3
ALL_3	*95.00%*	*9*	*3*

Tabla 10. Resultados de combinar los clasificadores individuales CENPARMI (tabla 4) para la votación con ranking

Patrones de la base MNIST

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	98.56%	2	1
DAUB_1 © DAUB_2 © DAUB_3	98.56%	3	1
MALLAT_1 © MALLAT_2	98.77%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	98.80%	3	1
SYMM_1 © SYMM_2	98.77%	2	1
SYMM_1 © SYMM_2 © SYMM_3	98.68%	3	1
ALL_1	98.82%	3	3
ALL_2	*98.87%*	*6*	*3*
ALL_3	98.84%	9	3

Tabla 11. Resultados de combinar los clasificadores individuales MNIST (tabla 5) para la votación con ranking

Combinación de clasificadores usando conceptos de ranking y teniendo cuenta la habilidad para reconocer una clase

La combinación de clasificadores utilizando esta técnica (ecuación 50) se realiza permitiendo que cada clasificador nos retorne por cada dígito que analiza una lista donde prioriza las clases a las que supone que el mismo pertenece. En la implementación que hemos realizado, estamos permitiendo que esta lista contenga como máximo 3 clases, y luego sería tan simple como asignar pesos (fijos, no generados a partir de ninguna clasificación como en el caso donde se tienen en cuentan los pesos pero aplicados a la eficacia del clasificador sobre cada una de las clases) según el orden en que se encuentre la clase que nos ha retornado en la lista de ellas. Luego, además de esta lista de preferencias que el clasificador nos retorna para el dígito, también estaremos teniendo en cuenta la habilidad del clasificador para reconocer dígitos de dicha clase; es decir que estaremos utilizando los pesos (al igual que en la decisión por pesos que hemos contado antes) que cada uno de los clasificadores tiene para cada una de las clases. De esta manera estamos tomando el conjunto de muestras de testeo y estamos,

para cada uno de los dígitos que pertenecen a la misma, viendo cual fue la clase que más peso tuvo usando como factor para aumentar el valor de dicho peso la preferencia del clasificador y nos estamos quedando con dicha clase.

Los factores de multiplicación para componer el resultado de la votación son los mismos que los tomados en la sección anterior según el orden en que el clasificador nos retorna la lista de clases de acuerdo a la que mayor preferencia tiene hasta la menor. Y como también mencionamos anteriormente: Al ser esta última un ajuste de parámetros realizado de manera 'manual', por decirlo de alguna manera, es un tema sobre el que se podría seguir investigando y utilizando diferentes criterios de ajuste para lograr distintos resultados.

A modo de ejemplo, si tenemos un dígito del conjunto de testeo y usamos tres clasificadores, podemos tener múltiples opciones, y nos contaremos o describiremos cada una de ellas aquí, pero para tener una idea, tenemos que considerar las posibilidades que surgen de tener de combinar los ejemplos dados anteriormente para la combinación de clasificadores por ranking y la combinación de clasificadores por pesos, y el valor final para cada clase estará finalmente dado por la suma, por cada uno de los clasificadores que combinamos, del peso del clasificador para reconocer dicha clase respecto a las otras, multiplicado por la preferencia del clasificador por dicha clase en la lista de clases retornadas. En base a esto nos podemos imaginar todos los posibles escenarios que se podrían plantear y que no son simples o pocos para contarlos en unos párrafos.

Los resultados de combinar los clasificadores armados para el kernel radial con la parametrización ya explicada son los que se presentan en las siguientes tablas (12 y 13):

Patrones de la base CENPARMI

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	94.45%	2	1
DAUB_1 © DAUB_2 © DAUB_3	94.60%	3	1
MALLAT_1 © MALLAT_2	94.00%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	94.35%	3	1
SYMM_1 © SYMM_2	94.30%	2	1
SYMM_1 © SYMM_2 © SYMM_3	*95.05%*	*3*	*1*
ALL_1	94.45%	3	3
ALL_2	94.65%	6	3
ALL_3	94.90%	9	3

Tabla 12. Resultados de combinar los clasificadores individuales CENPARMI (tabla 4) para la votación con ranking y pesos por clase

Patrones de la base MNIST

CLASIFICADORES COMBINADOS	EFICACIA LOGRADA	CLASIFICADORES	WAVELETS
DAUB_1 © DAUB_2	98.63%	2	1
DAUB_1 © DAUB_2 © DAUB_3	98.55%	3	1
MALLAT_1 © MALLAT_2	98.69%	2	1
MALLAT_1 © MALLAT_2 © MALLAT_3	*98.83%*	*3*	*1*
SYMM_1 © SYMM_2	98.70%	2	1
SYMM_1 © SYMM_2 © SYMM_3	98.74%	3	1
ALL_1	98.80%	3	3
ALL_2	98.82%	6	3
ALL_3	98.82%	9	3

Tabla 13. Resultados de combinar los clasificadores individuales MNIST (tabla 5) para la votación con ranking y pesos por clase

Diferentes combinatorias en la combinación de clasificadores

Si bien hemos mejorado los resultados mediante la técnica de combinación de clasificadores, existen muchas otras posibilidades de cómo combinar los mismos para obtener diferentes resultados, en total 512 diferentes formas de hacerlo. En todas las tablas de combinación mostradas anteriormente (tablas 6, 7, 8, 9, 10, 11, 12 y 13) hemos mantenido siempre las mismas elecciones de clasificadores, quedándonos sin probar 503 posibles combinaciones para cada uno de las técnicas de votación propuestas en este trabajo.

En esta etapa procedimos a probar diferentes combinatorias de los mismos basándonos básicamente en variaciones de aquellas formas de combinación que mejores resultados nos retornaron. Es así que para el caso CENPARMI partimos de la combinación que mejor eficacia nos proporcionó, mostrada en la tabla 12 (SYMM_1/SYMM_2/SYMM_3), y para el caso de los clasificadores basados en la base MNIST partimos de la combinación que se muestra en las tablas 7 y 11 (ALL_2). Luego quitando y/o agregado otros clasificadores a la combinación, fuimos obteniendo diferentes resultados, y solo para el caso MNIST encontramos una parametrización más simple que la mejor que habíamos obtenido antes y cuya eficacia es mayor que la lograda en ese momento.

En el caso MNIST logramos una mejor eficacia que la que habíamos obtenido (98.87%) combinando tan solo 3 de los clasificadores que habían logrado este resultado, es decir, hemos quitado de la combinación 3 clasificadores (con lo que se reduce el tiempo computacional de tener que entrenar y clasificar con los mismos) y también hemos utilizado una técnica de votación diferente. Para lograr estos resultados hemos combinado los siguientes clasificadores MALLAT_1, MALLAT_2 y DAUB_1, y mostramos los resultados obtenidos en la siguiente tabla (14):

CLASIFICADORES COMBINADOS	COMB. UNIFORME	COMB. PESOS	COMB. RANKING	COMB. RANKING + PESOS
MALLAT_1 © MALLAT_2 © DAUB_1	98.88%	98.91%	*98.93%*	*98.93%*

Tabla 14. Resultados de combinar los clasificadores seleccionados bajo las técnicas de votación mostradas para la base MNIST

Y mostramos a continuación el detalle de las clases donde la clasificación no es correcta, para el mejor de los resultados (tomamos el último), contra la mejor cantidad de errores que podemos tomar al clasificar cada una de las clases. Lo vemos en la siguiente tabla (15):

CLASE	M1/M2/D1	M1	M2	D1	D2	S1	S2	Min(ALL_2)
0	*6*	7	6	6	7	7	6	*6*
1	*6*	6	5	6	7	5	5	*5*
2	*9*	10	10	11	13	11	10	*10*
3	*11*	14	11	14	14	11	15	*11*
4	*10*	11	13	14	19	9	16	*9*
5	*9*	10	18	12	15	11	14	*10*
6	*9*	10	13	10	13	13	11	*10*
7	*13*	15	17	22	23	16	16	*15*
8	*14*	14	18	18	15	13	12	*12*
9	*20*	19	23	25	25	21	27	*19*
ERRONEOS	*107*	116	134	138	151	117	132	*107*

Tabla 15. Detalle de la cantidad de muestras mal clasificadas para la combinación armada en la tabla 14 contra las clasificaciones individuales y el clasificador que se basaría en elegir la mejor opción para cada clase

Para simplificar las etiquetas de las celdas tomamos M1 como MALLAT_1, y así con los clasificadores basados en las otras wavelets, y como Min(ALL_2) a lo que nos daría la mínima cantidad de errores (suponiendo que asignemos, de alguna forma, a cada clasificador el peso suficiente sobre las clases que clasifica de mejor manera, cosa que es imposible saber a priori). Esto nos muestra, que aun sabiendo cómo cada clasificador clasificaría cada una de las muestras y los errores que podría cometer, las técnicas utilizadas de selección de pesos y ranking nos retornan muy buenos resultados, haciendo que la técnica de combinación de clasificadores utilizando las mismas, sea tan buena como el resultado que se obtendría tomando la mejor clasificación proporcionada por separado para una de las clases.

Armado de clasificadores basado en la composición de descriptores

Otra posible forma de entrenar la máquina de soporte vectorial, aplicando las técnicas de preprocesamiento que introdujimos en este trabajo, se presenta bajo la posibilidad de generar para cada una de las wavelets las características de cada uno de los descriptores como hemos establecido, y luego componer realizando diferentes combinaciones dichas características.

La idea utilizada para obtener todos estos diferentes descriptores estaría basada en generar los diferentes niveles de transformada (1, 2 y 3) que podemos aplicar para cada una de las wavelets seleccionadas. Esto nos permitiría generar luego un descriptor compuesto que se encontraría basado en la suma (o composición) de cada uno de estos descriptores generados por los preprocesamientos aplicados. Hay que tener en cuenta que esta técnica incrementaría la dimensión del vector descriptor, tema que siempre tratamos de reducir lo más que se pudiera, en dos o tres veces más dependiendo de los niveles de aplicación de transformada wavelet que se hayan aplicado.

A continuación presentamos los resultados obtenidos a partir de componer el vector de clasificación con los datos que tienen algunos de los clasificadores que fueron combinados en la sección anterior. A modo de ejemplo comparativo estaremos basándonos en la wavelet Mallat y generando los descriptores, parametrizados de igual manera, para la base MNIST y la aplicación de la transformada wavelet para los niveles 1 y 2, y también para la composición de los niveles 1, 2 y 3. En estos casos donde se utilizan los niveles 1 y 2, y en el caso donde se utilizan los niveles 1, 2 y 3, hay que tener en cuenta que para cada uno de estos niveles la parametrización del descriptor individual que se está utilizando para la composición del descriptor final es la detallada para el caso de la base MNIST con lo que el descriptor tiene el doble y triple de tamaño para cada uno de los casos respectivamente.

Los resultados que obtenemos en el clasificador que utiliza el descriptor compuesto por la suma de estos descriptores son los siguientes:

COMPOSICION DE DESCRIPTORES	EFICACIA LOGRADA
MALLAT_1 + MALLAT_2	98.72%
MALLAT_1 + MALLAT_2 + MALLAT_3	98.75%

Tabla 16. Resultados de la composición de descriptores para la wavelet Mallat sobre la base MNIST

Si comparamos la tabla 16 contra la siguiente (17), donde presentamos los resultados para la combinación de los clasificadores Mallat utilizando las distintas técnicas de combinación podemos ver lo siguiente:

CLASIFICADORES COMBINADOS	COMB. UNIFORME	COMB. PESOS	COMB. RANKING	COMB. RANKING + PESOS
MALLAT_1 © MALLAT_2	98.71%	98.69%	98.77%	98.69%
MALLAT_1 © MALLAT_2 © MALLAT_3	98.84%	98.80%	98.80%	98.83%

Tabla 17. Resultados obtenidos en las pruebas de combinación de clasificadores para la wavelet Mallat sobre la base MNIST (extraídos de las tablas 7, 9, 11 y 13)

Si bien los resultados obtenidos para este clasificador que utiliza el descriptor compuesto, al ser la dimensión del mismo el doble de la utilizada en los clasificadores anteriores, la performance del mismo es menor, en cuanto a que el tiempo para realizar el entrenamiento y la clasificación del conjunto de testeo, así es similar la clasificación y entrenamiento de los clasificadores individuales para Mallat nivel 1 y Mallat nivel 2, no nos da, a priori, un buen indicio que se sucederá cuando compongamos mas descriptores, y es un factor a considerar en el balance entre performance y eficacia del sistema.

También mostramos los resultados para comparar la eficacia lograda a partir de la composición de la wavelet Mallat para los siguientes niveles de aplicación de transformada: 1, 2 y 3, contra los resultados de la combinación de los clasificadores individuales, utilizando las diferentes técnicas, que generamos en las secciones anteriores. En este último caso la dimensión del descriptor utilizado en este caso es 3 veces más que para los clasificadores individuales, aunque el tiempo de entrenamiento y clasificación, si bien crece, pareciera no hacerlo en forma lineal para estos descriptores armados, sino que se produce un extraño fenómeno donde este tiempo crece en forma logarítmica, con lo que, si podemos suponer que esta tendencia se mantiene, sería buena si se tuviera un descriptor compuesto por un gran número de descriptores (los que se usarían en los clasificadores individuales). Aunque por otro lado, y como vemos en el descriptor de los 3 niveles, el porcentaje de eficacia no es mayor al que se

consigue utilizando 2 niveles (o 3 niveles) en la combinación de clasificadores, con lo que nos hace pensar que componiendo de esta forma aumentando la dimensión del descriptor final no nos llevará a conseguir mejores resultados.

Es entonces que optamos por armar el descriptor compuesto, por más de 3 clasificadores, para uno de los mejores resultados que habíamos obtenido en la técnica de combinación de clasificadores. Este resultado es 98.87% y puede ser consultado en la tabla 11 para la combinación por ranking. El resultado obtenido para el mismo no es mejor a la mejor combinación realizada en la tabla 14, es por esto que también componemos el descriptor de los clasificadores que utilizamos en dicho caso. Presentamos entonces en la siguiente tabla (18) los resultados para el descriptor compuesto de aplicar para cada una de las wavelets utilizadas en este trabajo la técnica de preprocesamiento descripta para los niveles 1 y 2 de las mismas:

DESCRIPTOR COMPUESTO	EFICACIA LOGRADA	TIEMPO DE ENTRENAMIENTO (en segundos)
DAUB_1 + MALLAt_1 + SYMM_1	98.90%	2537
DAUB_1 + MALLAT_1 + MALLAT_2	98.85%	2581
DAUB_1 + DAUB_2 + MALLAT_1 + MALLAT_2 + SYMM_1 + SYMM_2	98.45%	14907

Tabla 18. Resultados del entrenamiento y clasificación del clasificador utilizando el descriptor compuesto

En estos casos, estamos utilizando en la composición 3 y 6 descriptores individuales, con lo que la dimensión del descriptor final para estos casos es de 720 y de 1440 para cada uno respectivamente. Es de esperar que, si se cumple la suposición anterior sobre el crecimiento de tiempo en forma logarítmica, el entrenamiento de la máquina de soporte vectorial para este caso no sea mayor, sino significativamente menor, a la suma de los tiempos de los clasificadores individuales utilizados para realizar la combinación en el caso correspondiente que hemos mencionado.

Presentamos en la siguiente tabla (19) los tiempos de entrenamiento requeridos para cada uno de los clasificadores individuales que fueron utilizados en las pruebas de combinación de los mismos para poder compararlos, además de comparar la eficacia obtenida con cada una de las técnicas, contra el tiempo necesario para poder entrenar el clasificador que se encuentra basado en este descriptor compuesto:

WAVELET DE CLASIFICADOR	NIVEL 1 (tiempo en segundos)	NIVEL 2 (tiempo en segundos)
DAUBECHIES	1955	1233
MALLAT	2814	2711
SYMMLET	1373	1873

Tabla 19. Tiempos requeridos para el entrenamiento individual del clasificador basado en la wavelet que se indica a izquierda

Mientras que el tiempo necesario para el entrenamiento del clasificador basado en el descriptor compuesto utilizando los descriptores individuales para las 3 wavelets utilizadas presenta las características mencionadas respecto a ser menor que la suma de los tiempos necesarios para los clasificadores individuales (2537 para el descriptor compuesto, contra 6142 para los 3 clasificadores por separado), esta hipótesis no se cumple en el tercer ejemplo que hemos planteado aquí donde el tiempo necesario para el entrenamiento del clasificador basado en las composición de 6 descriptores es de 14907 segundos (tabla 18). Como se puede observar, en la tabla 19, la suma del tiempo necesario para

entrenar los clasificadores individuales, y poder combinarlos (el tiempo de esta operatoria es despreciable), nos arroja el valor de 11959 segundos, que es menos del tiempo necesario para el entrenamiento del clasificador basado en el descriptor compuesto. Además la eficacia de éste no es mejor que la eficacia de la combinación de los clasificadores individuales, para cualquiera de las técnicas de combinación mostradas en la tabla 17 (donde vemos que se utiliza una sola wavelet y dos niveles de la misma). Por lo que podemos ver que las técnicas basadas en combinación de buenos clasificadores es mejor que el intentar lograr armar un clasificador que sea el más eficaz (el uso de técnicas de combinación se ve en muchos trabajos relacionados con temas de clasificación).

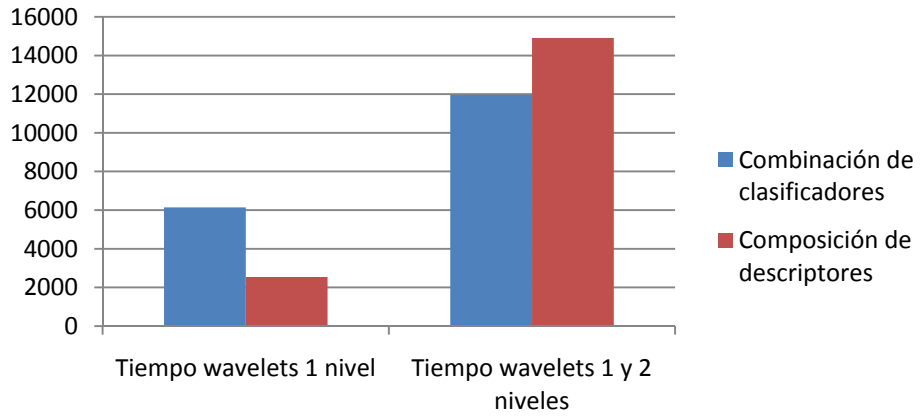


Figura 58. Tiempos necesarios para entrenar/clasificar los conjuntos de dígitos de la base MNIST con las técnicas de combinación de clasificadores vs. la composición de descriptores.

Nota: Habría que tener en cuenta también, en todo este análisis, que el entrenamiento de los clasificadores individuales para ser combinados luego podría realizarse en paralelo con lo que el tiempo total necesario para obtener el clasificador combinado estaría acotado por el tiempo necesario para entrenar al clasificador que mayor cantidad de minutos requiera.

Comparación con otros trabajos publicados

Resultados CENPARMI

Los resultados que hemos obtenido para la base CENPARMI son comparables con los que podemos encontrar en otras publicaciones, presentamos a continuación una breve tabla donde comparamos este resultado con los obtenidos por otras investigaciones:

REFERENCIA	METODO	EFICACIA LOGRADA
[35] Concordia	Multiwavelets + neural networks	92.20%
ESTE TRABAJO	*Wavelets + SVM + classifier combination*	*95.05%*
[34] IWFHR-8	Support vector machine	98.90%

Tabla 20. Resultados encontrados en publicaciones que abordan este mismo problema para la base CENPARMI [11]. La etiqueta 'ESTE TRABAJO' se refiere al presente ("Reconocimiento de Dígitos Manuscritos Aplicando Transformadas Wavelet sin Submuestreo y Máquinas de Soporte Vectorial")

Podemos observar, en la tabla 20, que encontramos mejores y peores resultados para el tratamiento de este problema sobre la base CENPARMI. También encontramos publicaciones donde los resultados logrados no son tan buenos como en este trabajo, y otras obteniendo resultados mucho mejores pero con técnicas de preprocesamiento más complejas pero no contamos con la información o el detalle del preprocesamiento aplicado para poder compararlo (tampoco pudiendo realizar benchmarks relativos a performance en cuanto a los tiempos de clasificación, al no tener la información disponible que nos pudiera facilitar este trabajo, estos resultados son difícilmente comparables).

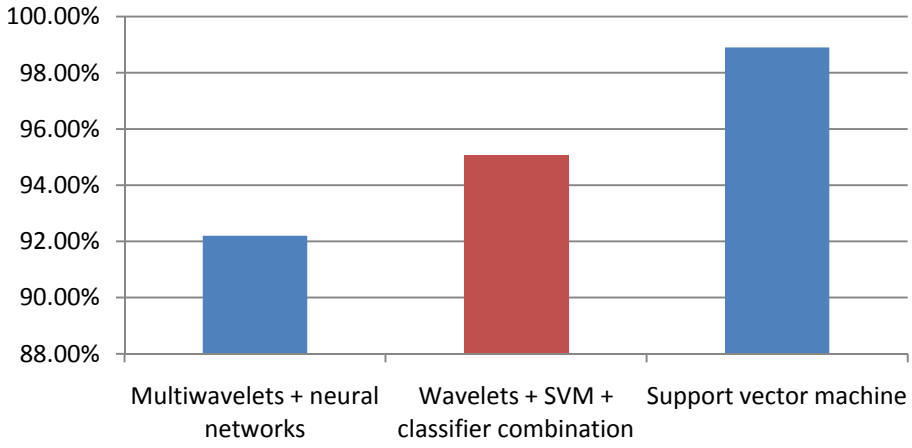


Figura 59. Comparación entre los resultados obtenidos en este trabajo (en rojo), para la base CENPARMI, contra otras publicaciones que tratan el mismo problema. Estos valores se encuentran extraídos de los datos presentados en la tabla 20 (donde podemos observar sus referencias).

Resultados MNIST

Los resultados que hemos obtenido para la base MNIST se encuentran entre los índices obtenidos por otras publicaciones y, como sobre dicha base hay muchos más trabajos realizados, podemos entonces hacer una comparación más exhaustiva pudiendo observar que este resultado es comparable con muchos otros obtenidos por grupos de investigación con mayor dedicación al problema planteado.

Presentamos entonces, a modo comparativo, resultados obtenidos en otras publicaciones, para la base MNIST, con la aplicación de técnicas diferentes:

REFERENCIA	METODO	EFICACIA LOGRADA
[22] AT&T	Euclidean nearest neighboor	96.50%
[23] AT&T	Deslant, Euclidean 3-NN	97.60%
[24] Kyushu U	Elastic matching	97.90%
[25] UC London	Products of experts	98.30%
[26] U Québec	Hyperplanes + support vector machines	98.50%
[27] TU Berlin	Support vector machine	98.60%
[28] AT&T	Neural net LeNet4	98.90%
ESTE TRABAJO	*Wavelets + SVM + classifier combination*	*98.93%*
[29] MPI, AT&T	Virtual SVM	99.20%
[30] CENPARMI	Support vector machine	99.40%
[31] Caltech, MPI	Deslant, virtual SVM (jitter,shift)	99.44%
[32] Boston U	Shape context matching	99.46%
[33] Microsoft	Neural net + virtual data	99.58%
[22] AT&T	Human performance	99.80%

Tabla 21. Resultados encontrados en publicaciones que abordan este mismo problema para la base MNIST [21]. La etiqueta 'ESTE TRABAJO' se refiere al presente ("Reconocimiento de Dígitos Manuscritos Aplicando Transformadas Wavelet sin Submuestreo y Máquinas de Soporte Vectorial")

Podemos observar, en la tabla 21, que encontramos mejores y peores resultados para el tratamiento de este problema sobre la base MNIST. También encontramos publicaciones que son bastante recientes donde los resultados logrados no son tan buenos como en este trabajo, y algunas más viejas obteniendo resultados mucho mejores pero con técnicas de preprocesamiento más complejas (no pudiendo realizar benchmarks relativos a performance en cuanto a los tiempos de clasificación, al no tener la información disponible que nos pudiera facilitar este trabajo, estos resultados son difícilmente comparables).

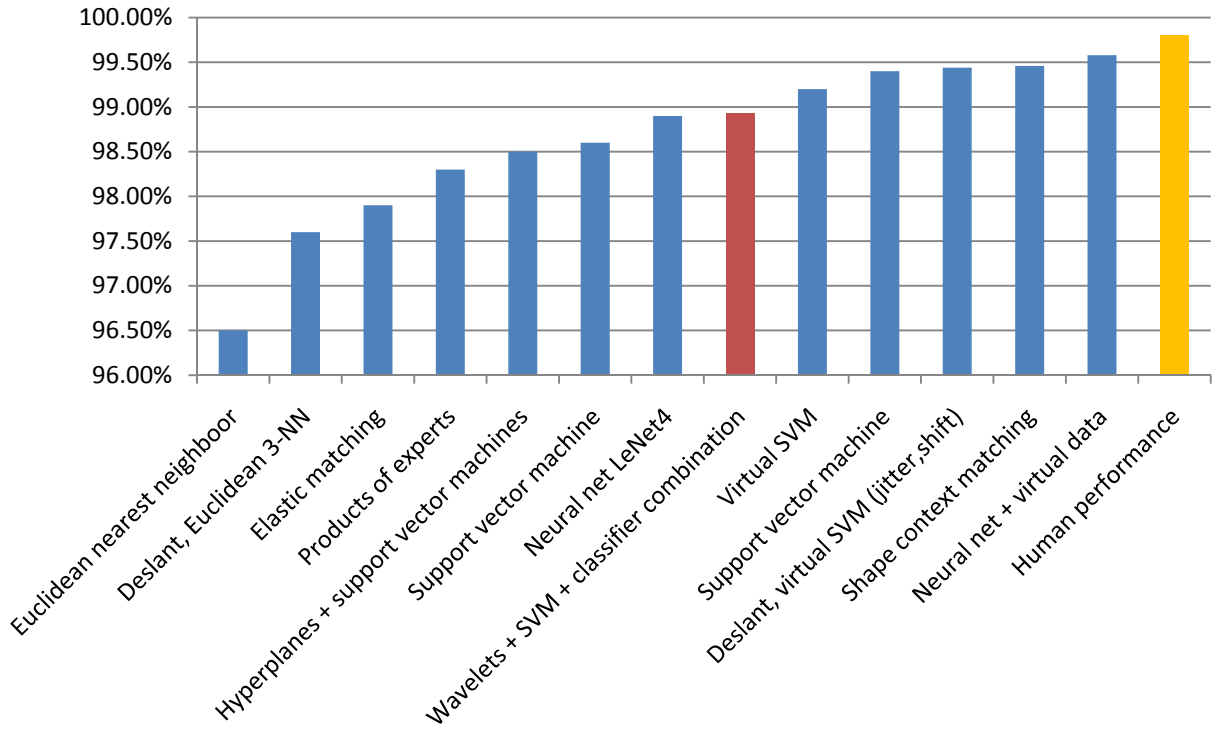


Figura 60. Comparación entre los resultados obtenidos en este trabajo (en rojo), para la base MNIST, contra otras publicaciones que tratan el mismo problema (al final, en naranja, la mejor eficacia: lograda por humanos). Estos valores se encuentran extraídos de los datos presentados en la tabla 21 (donde podemos observar sus referencias).

También cabe resaltar, que por ejemplo, en el mejor de los casos presentados [33], se está extendiendo el set de entrenamiento aplicando dos desplazamientos azarosos de la imagen que luego es suavizado con un filtro Gaussiano. Esto hace posible que se genere para cada uno de los dígitos 1000 muestras virtuales, con lo que el conjunto de entrenamiento crece significativamente, y se comenta además que estos datos son generados en forma dinámica para cada iteración de entrenamiento, con lo que, el costo de dicha generación, mas el entrenamiento sobre este set de datos inmenso no es algo para no ser tenido en cuenta (aunque no dispongamos de los tiempos necesarios para aplicar estos procesos) en lo que respecta a performance (tiempo) de dicha técnica de clasificación.

Y otra 'desventaja' sobre la base MNIST, que hace imposible la comparación directa entre estos resultados, es un problema conocido como 'entrenamiento sobre las muestras de testeo' [21]. Y en [32], además del problema sobre el conjunto en el cual se entrena, sucede algo similar en cuanto a los cálculos necesarios para entrenar el clasificador, ya que se extrae un histograma de puntos de contorno de cada imagen, y luego se utiliza el conjunto de entrenamiento y de testeo para generar un grafo bipartito (que es resuelto con el algoritmo Húngaro, cuya complejidad es conocida siendo de $O(n^4)$, o de $O(n^3)$ para versiones mas eficientes), con pesos en sus ejes de acuerdo a la similitud de los descriptores de contorno. A su vez esta asignación entre miembros de ambos conjuntos es utilizada para generar una transformación spline de dos dimensiones que mejor machea las dos imágenes; técnicas más complejas que las utilizadas en este trabajo.

En [31] por ejemplo, donde si se está utilizando máquinas de soporte vectorial como técnica de clasificación, se genera un clasificador (SVM) que se encuentra particularmente adaptado para esta tarea incorporando conocimiento previo sobre la misma. Para lograr esto se está utilizando un 'kernel especial' que básicamente radica en aplicar transformaciones de desplazamiento a cada digito. La técnica, llamada fluctuación del núcleo, consiste en lo siguiente: primero se procede a entrenar una máquina de soporte vectorial, y se utilizan los vectores de soporte de la misma como una representación compacta del problema (ya que contienen la suficiente información para ser eficientes). Como hemos mencionado la función, o kernel especial que se aplica, consiste en transformar cada uno de los dígitos que están en el soporte de la máquina (es decir que se encuentran en la zona de la frontera entre clases) realizando desplazamientos (verticales y/o horizontales de a 1 pixel con lo que tendríamos 8 'nuevas' muestras, mas desplazamientos horizontales y verticales de 2 pixeles, serian otra 4 'nuevas' muestras) lo que resulta en 13 nuevos vectores de soporte por cada vector original (técnicas similares de desplazamiento de la imagen original, aunque persiguiendo otros objetivos, son por ejemplo utilizadas también en otros papers donde se busca normalizar/centralizar cada dígito utilizando su centro de masa [21]). Luego con estos vectores virtuales dentro de su modelo de decisión una nueva máquina de soporte vectorial es entrenada (digamos que se acota mas el conjunto que representa al modelo de decisión) y se clasifica el conjunto de testeo.

Capítulo 8 – Conclusiones y Trabajos Futuros

En el presente trabajo hemos focalizado en el uso de transformadas wavelets y la teoría que existe en torno a las mismas, presentada en el capítulo 2, para la extracción de características que puedan caracterizar una imagen. A su vez hemos preferido el utilizar, entre las diversas técnicas que existen para realizar la clasificación de muestras, lo que nos ofrece la teoría de máquinas de soporte vectorial que hemos presentado en el capítulo 3, y hemos terminado obteniendo los mejores resultados aplicando técnicas de combinación de varias de estas máquinas de soporte vectorial entrenadas de diferentes formas.

Para poder obtener resultados comparables es que hemos seleccionado dos bases altamente conocidas como son la CENPARMI y MNIST, sobre las cuales existen muchas publicaciones, que presentan distintos enfoques y técnicas de clasificación, variando levemente entre sí la eficacia que logran obtener ante el desafío de clasificar los patrones que en éstas se encuentran. A la luz de los resultados obtenidos, y mostrados en el capítulo 6, podemos ver que se cumple la hipótesis planteada en el capítulo donde presentamos las mismas (capítulo 4), y en el capítulo 5 donde mostramos la varianza espacial que existe sobre cada clase para todas las muestras que contiene, y comentamos sobre la mayor dificultad para clasificar correctamente los dígitos de la base CENPARMI, viendo que los porcentajes de eficacia obtenidos para este caso, con todas las técnicas que hemos aplicado, es siempre menor que los logrados para la base MNIST.

Como podemos ver a priori, en los primeros resultados obtenidos, la aplicación de un preprocesamiento de algún tipo sobre las muestras de dichas bases nos da la posibilidad de lograr una eficacia mayor que los intentos que se podrían hacer (con diversas técnicas) sobre los patrones originales sin la aplicación de ningún tipo de proceso previo.

Luego, y como generalmente es de interés tener buena performance computacional y rendimiento del sistema, vemos que es importante realizar una reducción del espacio del vector descriptor que estaremos utilizando, algo que podemos lograr gracias al análisis multiresolución que nos brinda la teoría de wavelets y la posibilidad de convertir datos sensibles arrojados por la misma (como son los detalles) a espacios donde hacer una normalización en busca de caracterizar los representantes de cada clase nos es facilitado. La composición del vector descriptor basado en los datos de baja y alta resolución, que la aplicación de la transformada wavelet nos brinda, fue uno de los factores clave para lograr los índices de eficacia que obtuvimos y buscábamos. Pudimos comprobar que, en este caso, la composición de las características que extraemos en este análisis multiresolución se potencia sobre cualquier resultado que nos pudiera brindar cada una de estas características por si sola o la suma de ellas. Los otros factores que vemos nos han aportado significativamente a la obtención de estos índices de eficacia son la utilización de máquinas de soporte vectorial, ampliamente reconocidas por su buen rendimiento, y el uso de técnicas de combinación de clasificadores como punto final. Todos estos factores están aportando al sistema de clasificación, donde para que el mismo dé buenos resultados, tanto el preprocesamiento, como los clasificadores son importantes.

En el capítulo 5 presentamos todo el análisis realizado sobre las posibles características que podrían componer al vector descriptor que fue utilizado en las pruebas, con las diferentes parametrizaciones en cuanto a dimensión, aplicación de binarizados, submuestreos, transformación de espacios, obtención de índices acerca de la información contenida luego de transformar, entre otras, que dichas características podrían tener. Luego presentamos, en el capítulo 6, la parametrización que podríamos aplicar a las máquinas de soporte vectorial construidas, así como también los kernels que

podrían ser utilizados con sus respectivas parametrizaciones. Por último mostramos e introdujimos en este capítulo algunas de las posibles técnicas de combinación de clasificadores, que se podrían utilizar luego, al realizar las pruebas ya con una configuración más definida para el armado del descriptor. También se presentan la parametrización del entrenamiento del clasificador y combinaciones válidas de algunos de estos que utilizaremos luego.

Exploramos para la producción del capítulo 7 diversas parametrizaciones posibles basadas en todo el análisis anterior, y nos hemos quedado con la que mejores resultados nos brindaba, sin entrar en profundidad en la variación de los parámetros de entrenamiento/clasificación como hemos comentado. Hemos focalizado en querer mostrar que la combinación de clasificadores, como técnica para incrementar la eficacia obtenida, es mejor que el armado de un clasificador que esté basado en la suma de la información individual con la que dichos clasificadores se encuentran entrenados y lo hemos logrado. También hemos argumentado que el incremento de la dimensión del descriptor (para la técnica de clasificación seleccionada, en este trabajo, que estuvo basada en el uso de máquinas de soporte vectorial) nos provocará un incremento que será exponencial en la performance del clasificador, y no necesariamente nos otorgará mejores resultados, y esto también lo hemos logrado mostrar en los últimos resultados presentados en dicho capítulo.

También podemos observar que los mejores resultados los obtenemos con las técnicas de combinación de clasificadores, pero no solo por la técnica en sí, sino que también consideramos que se debe a un único factor diferente en el preprocesamiento aplicado al descriptor de estos clasificadores que los hace distintos. Este factor, que diferencia a un clasificador de otro, es la utilización de una función wavelet, de las que hemos seleccionado para trabajar, más la aplicación de un nivel particular de la misma. Basándonos en este hecho podemos arriesgar que cada una de las wavelets utilizadas posiblemente estén extrayendo diferentes características de las muestras tratadas y que, de alguna forma, la combinación donde estamos utilizando todas ellas, las 3 wavelets seleccionadas, favorece a que se complementen entre sí logrando un mejor reconocimiento (resultados de tablas 7 y 11 para todas las wavelets, y la aplicación de uno y dos niveles de transformada wavelet: ‘ALL_2’).

Como otro punto favorable a la selección de wavelets, como herramienta para el procesamiento, debemos mencionar que la aplicación de las mismas, al ser básicamente un filtro, requiere de algoritmos muy simples y rápidos, con lo que los resultados obtenidos mediante las mismas, haciendo un balance entre performance y eficacia, son más que comparables con las técnicas más complejas que podemos encontrar actualmente en la literatura.

En conclusión, con todas las técnicas utilizadas, tanto de preprocesamiento como la aplicación de teorías generales (wavelets y máquinas de soporte vectorial), estamos comparando resultados con grupos de investigación que tienen amplio conocimiento del tema desarrollando técnicas de muchísima mayor complejidad penalizando la performance (tiempo) de generación de clasificadores, más la clasificación del conjunto de testeo, y/o también no permitiendo generalizar las mismas a la aplicación sobre otros dominios de datos. Sobre este último punto mencionado debemos resaltar que la técnica que hemos utilizado no requiere conocimiento previo del tema como lo hacen mucha de las técnicas contra las cuales hemos comparados nuestros resultados donde ya sea: se conoce el conjunto sobre el que se va a trabajar y se hacen ajustes particulares para el caso, y/o se introducen en el clasificador la utilización de características asociadas a la problemática.

Con las mismas hemos obtenido los siguientes porcentajes de eficacia para estas dos bases de datos ampliamente conocidas en la literatura:

BASE DE DATOS	EFICACIA LOGRADA	TASA DE ERROR
CENPARMI	95.05%	4.95%
MNIST	98.93%	1.07%

Tabla 22. Mejores resultados obtenidos en la clasificación de los conjuntos de testeo de las bases utilizadas

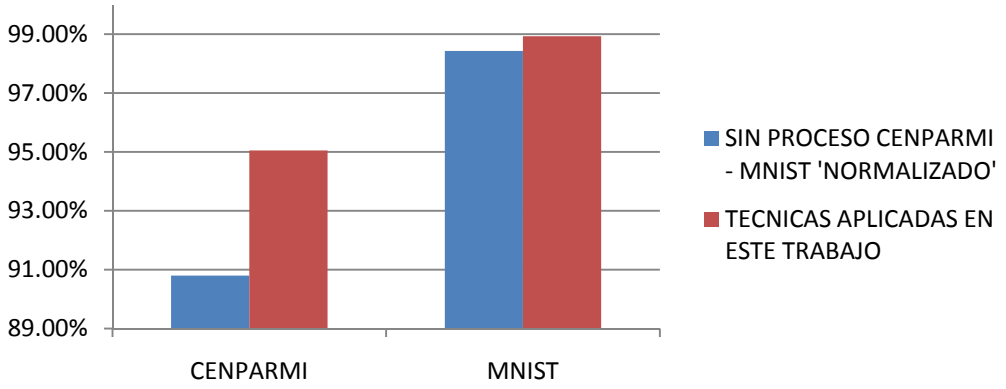


Figura 61. Comparación entre la eficacia obtenida para las bases de datos utilizadas en este trabajo, sin preprocesar (CENPARMI) y solo normalizando (MNIST), contra las técnicas aplicadas aquí

Quedan como trabajos de investigación futuros varios temas:

- La posibilidad de utilizar pesos aplicados a cada clasificador de acuerdo a su comportamiento en general y aplicado durante la combinación [13]
- La exploración (podría ser utilizando cross-cutting o técnicas más elaboradas) de distintas parametrización en los factores de ajuste para las técnicas de votación por ranking que hemos utilizado [capítulo 7]
- La exploración de nuevas y/o más elaboradas técnicas de combinación de clasificadores [13]
- La extracción de nuevas características brindadas por el análisis multiresolución provisto por la teoría de wavelets [capítulo 2][1][6][17]
- La parametrización de las diferentes opciones que nos provee la teoría de máquinas de soporte vectorial en forma más detallada (así como también la utilización y parametrización de sus kernels) [capítulo 3]
- La utilización de nuevos kernels y/o la posible construcción de un 'kernel' orientado a un preprocesamiento particular aplicado a las muestras [31]
- La utilización de esta técnica de preprocesamiento para el armado del descriptor sumado a la generación de muestras virtuales como algunos papers, que presentan mejores resultados, lo hacen para el entrenamiento [33]
- La utilización de diferentes técnicas para la clasificación multiclase con máquinas de soporte vectorial (uno contra uno, grafos dirigidos acíclicos, técnicas basadas en arboles de decisión, etc.) [37]

Bibliografía/Referencias

[1] P. Wunsch and A. Laine, “Wavelet descriptors for multiresolution recognition of handprinted characters", Pattern Recognition 28, 1237-1249 (1995).

[2] Lauer, F., Suen, C., Bloch, G.: “A trainable feature extractor for handwritten digit recognition”, Pattern Recognition 40, 1816–1824 (2007).

[3] Steve R. Gunn, “Support Vector Machines for Classification and Regression”, Faculty of Engineering and Applied Science - Department of Electronics and Computer Science (10 May 1998).

[4] Oliveira, L., Sabourin, R.: “Support vector machines for handwritten numerical string recognition”, In: 9th IEEE International Workshop on Frontiers in Handwritten Recognition, pp. 39–44. IEEE Computer Society, Washington (2004).

[5] Christopher J.C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, 2, 121–167 (1998).

[6] Joaquín Marcos Nuñez Cortés, “Estudio de los módulos máximos de la transformada wavelet 'a trous' y sus aplicaciones en la detección de bordes”, Tesis de Licenciatura del Departamento de Computación, FCEN, UBA (2005).

[7] S. Liapis and G. Tziritas, “Color and Texture Image Retrieval Using Chromaticity Histograms and Wavelet Frames", IEEE Transactions on Multimedia 6, 676-686 (2004).

[8] I. Daubechies, “Ten lectures on wavelets”, Society for Industrial and Applied Mathematics (1992).

[9] S. Mallat, in “A Wavelet Tour of Signal Processing”, A. Press, ed., (1999).

[10] J.-P. Antoine and R. Murenzi, “Two-dimensional directional wavelets and the scale-angle representation", Signal Processing 52, 256-281 (1996).

[11] http://www.cenparmi.concordia.ca/

[12] http://yann.lecun.com/exdb/mnist/

[13] A.F.R. Rahman, H. Alam, and M.C. Fairhurst, “Multiple Classifier Combination for Character Recognition: Revisiting the Majority Voting System and Its Variations”, Document Analysis Systems 2002 LNCS 2423, 167-178, Springer-Verlag Berlin, Heidelberg (2002)

[14] http://www-stat.stanford.edu/~wavelab/Wavelab_850/download.html

[15] http://www.idiap.ch/scientific-research/resources/torch/?searchterm=torch

[16] Leena-Maija REISSELL, “II: Multiresolution and Wavelets”, University of British Columbia, Siggraph ’95 Course Notes: #26 Wavelets.

[17] Diego Romero, Ana Ruedin, Leticia Seijas. “Wavelet-based Feature Extraction for Handwritten Numerals”. Departamento de Computación, FCEN/UBA.

[18] Kristin P. Bennett, Colin Campbell. “Support Vector Machines: Hype or Hallelujah?”. SIGKDD Explorations, Volume 2, Issue 2, Math Sciences Department, Rensselaer Polytechnic Institute – Department of Engineering Mathematics, Bristol University, December 2000.

[19] Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Schölkopf. “A Tutorial on *v*-Support Vector Machines”. Departament of Computer Science and Information Engineering, National Taiwan University – Max Planck Institute for Biologial Cybernetics, Tübingen, Germany.

[20] Carl Staelin. “Parameter selection for support vector machines”. HP Laboratories Israel, HPL-2002-354 (R.1), November 2003.

[21] Daniel Keysers, “Comparison and Combination of State-of-the-art Techniques for Handwritten Character Recognition: Topping the MNIST Benchmark”. Image Understanding and Pattern Recognition (IUPR) Group, German Research Center for Artificial Intelligence (DFKI), May 2006.

[22] P. Simard, Y. Le Cun, and J. Denker. “Efficient Pattern Recognition Using a New Transformation Distance”. In S. Hanson, J. Cowan, and C. Giles, editors, Advances in Neural Information Processing Systems 5, pages 50-58, San Mateo, CA, 1993. Morgan Kaufmann.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-Based Learning Applied to Document Recognition”. Proc. of the IEEE, 86(11):2278-2324, November 1998.

[24] N. Matsumoto, S. Uchida, and H. Sakoe. “Prototype Setting for Elastic Matching based Image Pattern Recognition”. In ICPR 2004, 17th Int. Conf. on Pattern Recognition, volume I, pages 224-227, Cambridge, UK, August 2004.

[25] G. Mayraz and G. Hinton. “Recognizing Handwritten Digits Using Hierarchical Products of Experts”. IEEE Trans. Pattern Analysis and Machine Intelligence, 24(2):189-197, February 2002.

[26] J. Milgram, R. Sabourin, and M. Cheriet. “Combining Model-based and Discriminative Approaches in a Modular Two-stage Classification System: Application to Isolated Handwritten Digit Recognition”. Electronic Letters on Computer Vision and Image Analysis, 5(2):1-15, 2005.

[27] B. Schölkopf. “Support Vector Learning”. Oldenbourg Verlag, Munich, 1997.

[28] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. Jackel, Y. Le Cun, U. Müller, E. Sackinger, P. Simard, and V. N. Vapnik. “Comparison of Classifier Methods: A Case Study in Handwritten Digit Recognition”. In Proc. of the Int. Conf. on Pattern Recognition, pages 77-82, Jerusalem, Israel, October 1994.

[29] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. “Prior Knowledge in Support Vector Kernels”. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, Advances in Neural Information Processing Systems 10, pages 640-646. MIT Press, June 1998.

[30] J.-X. Dong, A. Krzyzak, and C. Y. Suen. “Fast SVM Training Algorithm with Decomposition on Very Large Data Sets”. IEEE Trans. Pattern Analysis and Machine Intelligence, 27(4):603-618, April 2005. Additional results at http://www.cenparmi.concordia.ca/_people/jdong/ HeroSvm.html.

[31] D. Decoste and B. Schölkopf. “Training Invariant Support Vector Machines”. Machine Learning, 46(1-3):161-190, 2002.

[32] V. Athistos, J. Alon, and S. Sclaroff. “Efficient Nearest Neighbor Classification Using a Cascade of Approximate Similarity Measures”. In CVPR 2005, Int. Conf. on Computer Vision and Pattern Recognition, volume I, pages 486-493, San Diego, CA, June 2005.

[33] P. Simard. “Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis”. In 7th Int. Conf. Document Analysis and Recognition, pages 958-962, Edinburgh, Scotland, August 2003.

[34] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. “Handwritten digit recognition using state-of-the-art techniques”. In Proc. of 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR-8), pages 320-325, 2002.

[35] G.Y. Chen, T.D. Bui, A. Krzyzak. “Contour-based handwritten numeral recognition using multiwavelets and neural networks”. Department of Computer Science, Concordia University, 1455 De Maisonneuve West, Montreal, Quebec, Canada H3G 1 M8, August 2002.

[36] Y. LeCun, L.D. Jackel, L. Bottou, J.S. Denker, H. Drucker, I. Guyon, U.A. Müller, E. Sackinger, P. Simard, and V.N. Vapnik, “Comparison of Learning Algorithms for Handwritten Digit Recognition,” Proc. Int’l Conf. Artificial Neural Network, F. Fogelman and P. Gallinari, eds., pp. 53-60 1995.

[37] Gjorgji Madzarov, Dejan Vjorgjevikj and Ivan Chorbev, “A Multi-class SVM Classifier Utilizing Binary Decision Tree”, Department of Computer Science and Engineering, Faculty of Electrical Engineering and Information Technology, July 27, 2008.

Mauro Krikorian

Digitally signed by Mauro Krikorian
DN: cn=Mauro Krikorian, o, ou, email=maurok@gmail.com, c=AR
Date: 2010.11.17 20:23:42 -03'00'