DISEÑO Y EVALUACIÓN DE UN CLASIFICADOR DE TEXTURAS BASADO EN LS-SVM

Beitmantt Cárdenas Quintero
Magister en Ciencias de la Información y las Comunicaciones
Docente planta de la Universidad Distrital Francisco José de Caldas
bgcardenasq@udistrital.edu.co
Bogotá, Colombia

Nelson Enrique Vera Parra
Magister en Ciencias de la Información y las Comunicaciones
Docente planta de la Universidad Distrital Francisco José de Caldas
neverap@udistrital.edu.co
Bogotá, Colombia

Pablo Emilio Rozo García
Magister en Ciencias de la Información y las Comunicaciones
Docente planta de la Universidad Distrital Francisco José de Caldas
perozog@udistrital.edu.co
Bogotá, Colombia

Tipo: Artículo de investigación

Fecha de Recepción: Marzo 2 de 2013
Fecha de Aceptación: Mayo 2 de 2013

DESIGN AND EVALUATION OF A TEXTURE CLASSIFIER BASED ON LS-SVM

ABSTRACT
To Evaluate the performance and the computational cost of different Least Square Support Vector Machine (LS-SVM) architectures and methodologies to image segmentation by texture and and based on these results apply a texture classifier model LS-SVM methodology; facing a binary classification problem segmentation represented by 32 images, organized into 4 groups formed by pairs of typical textures (granite / crust, brick / upholstery, wood / marble, fabric / fur), the performance and the computational cost of two types of core(radial/Polynominal) , two optimization functions (local mínimum/exhaustive search) and and two cost functions (random cross validation / Cross Validation leaving at least one) are measured and compared in a LS-SVM that takes as input the pixels that make up the neighborhood pixel cross-evaluate (there is no feature extraction). LS-SVM texture classifier presents better performance and requires less computational cost when a radial basis kernel and an optimization function based on a search algorithm to local minima accompanied by a cost function that uses random cross-validation are used.

Key words: kernel, least square, optimization, textures, segmentation, support vector machine,

RESUMEN
Evaluar el desempeño y el costo computacional de diferentes arquitecturas y metodologías Least Square Support Vector Machine (LS-SVM) ante la segmentación de imágenes por textura y a partir de dichos resultados postular un modelo de un clasificador de texturas LS-SVM. Metodología: Ante un problema de clasificación binaria representado por la segmentación de 32 imágenes, organizadas en 4 grupos y formadas por pares de texturas típicas (granito/corteza, ladrillo/tapicería, madera/mármol, tejido/pelaje), se mide y compara el desempeño y el costo computacional de dos tipos de núcleo (Radial / Polinomial), dos funciones de optimización (mínimo local / búsqueda exhaustiva) y dos funciones de costo (validación cruzada aleatoria / Validación cruzada dejando al menos uno) en una LS-SVM que toma como entrada los pixeles que conforman la vecindad cruz del pixel a evaluar (no se hace extracción de características). Resultados: LS-SVM como clasificador de texturas, pre-

senta mejor desempeño y exige menor costo computacional cuando utiliza un kernel de base radial y una función de optimización basada en un algoritmo de búsqueda de mínimos locales acompañado de una función de costo que use validación cruzada aleatoria.

Palabras claves: máquinas de vectores de soporte, núcleo, mínimos cuadrados, segmentación, optimización, texturas.

1. INTRODUCCIÓN

Support Vector Machine (SVM) en un método de clasificación y regresión proveniente de la teoría de aprendizaje estadístico postulada por Vapnik y Chervonenkis [1][2]. La metodología base de SVM se puede resumir de la siguiente forma: Si se requiere clasificar un conjunto de datos (representados en un plano n-dimensional) no separables linealmente, se toma dicho conjunto de datos y se mapea a un espacio de mayor dimensión donde si sea posible la separación lineal (esto se realiza mediante funciones llamadas Kernel). En este nuevo plano se busca un hiperplano que sea capaz de separar en 2 clases los datos de entrada; el plano debe tener la mayor distancia posible a los puntos de ambas clases (los puntos más cercanos a este hiperplano de separación son los vectores de soporte). Si lo que se requiere es hacer una regresión, se toma el conjunto de datos y se transforma a un espacio de mayor dimensión (donde sí se pueda hacer una regresión lineal) y en este nuevo espacio se realiza la regresión lineal pero sin penalizar errores pequeños.

La forma como se soluciona el problema de encontrar un hiperplano maximizando las márgenes entre éste y los puntos de ambas clases (del conjunto de datos de entrada), define si la SVM es tradicional o es LS-SVM. SVM soluciona dicho problema mediante el principio de structural risk minimization mientras que LS-SVM lo soluciona mediante un conjunto de ecuaciones lineales [3]. Mientras que en SVM tradicional muchos valores de soporte son cero (valores diferentes a cero corresponden a los vectores de soporte), en LS-SVM los valores de soporte son proporcionales a los errores.

LS-SVM ha sido propuesto y usado en el procesamiento de imágenes en diferentes campos. A continuación se nombran algunos ejemplos: Análisis de imágenes médicas diagnósticas, tales como, magnectic resonance imaging y magnetic resonance spectroscopy, [4][5]; Interpretación de imágenes provenientes de datos genéticos, tales como los microarray [6]; Tratamiento de imágenes en el ámbito de la seguridad, como la detección de objetos en imágenes infrarojas [7], la detección de rostros [8] y el reconocimiento de matrículas de vehículos [9]; Tratamiento de imágenes espaciales, como Remote sensing Image [10].

En este artículo se mide el desempeño de LS-SVM, teniendo como variable: la función kernel, la funcion de optimización y la función de costo, ante un proceso muy útil en el tratamiento de imágenes, como lo es la clasificación de texturas. Con ésta evaluación se busca obtener criterios de diseño de LS-SVM para desarrollar clasificadores de texturas óptimos que se pueden aplicar a cualquiera de los campos mencionados anteriormente. Este artículo está organizado de la siguiente forma: En la sección 2 se presentan los fundamentos teóricos de LS-SVM, en la sección 3 se describe la metodología desarrollada, en la sección 4 se exponen los resultados obtenidos y por último en la sección 5 se analizan los resultados y se obtienen conclusiones.

2. FUNDAMENTOS DE LS-SVM [3],[11]

Dado un conjunto de entrenamiento (ecuación (1)).

$$\{y_k, x_k\}_{k=1}^{N} \; Entrada x_k \in R^n, Salida y_k \in R^n \quad (1)$$

SVM tiene como objetivo construir un clasificador de la forma de la ecuación (2).

$$y(x) = sign\left[\sum_{k=1}^{N} \alpha_x y_x \psi(x, x_k) + b\right] \quad (2)$$

Donde σ y b son constantes y Ψ representa el kernel y puede ser cualquiera de las funciones descritas mediante la ecuación (3), (4) y (5).

$$\psi(x, x_k) = x_k^T x. \;\; Lineal \quad (3)$$

$$\psi(x, x_k) = (x_k^T x + 1). \;\; Polinomial \quad (4)$$

$$\psi(x, x_k) = e^{\left\{\frac{-\| x - x_k \|_2^2}{\sigma^2}\right\}} \quad (5)$$

El clasificador se construye mediante la ecuación (6).

$$\begin{cases} \omega^T \varphi(x_k) + b \geq 1 \;\; if \;\; y_k = +1 \\ \omega^T \varphi(x_k) + b \leq 1 \;\; if \;\; y_k = -1 \end{cases} \quad (6)$$

Que se puede expresar de forma compacta como lo muestra la ecuación (7).

$$y_k\left[\omega^T \varphi(x_k) + b\right] \geq 1 \;\; , \;\; k = 1, ..., N \quad (7)$$

Donde φ es la función que mapea los datos de entrada a un nuevo espacio de mayor dimensión donde exista el hiperplano separador. Si no se consigue obtener el hiperplano en el espacio de mayor dimensión se introduce un tipo de bias (ecuación (8)).

$$y_k\left[\omega^T \varphi(x_k) + b\right] \geq 1 - \varepsilon_k \;\; , \;\; \varepsilon_k \geq 0 \quad k = 1, ..., N \quad (8)$$

SVM tradicional soluciona el problema de encontrar el mejor hiperplano clasificador utilizando el principio structural risk minimization (ecuación (9)).

$$\min_{\omega, \varepsilon_k} f(\omega, b, e) = \frac{1}{2}\omega^T \omega + c\sum_{k=1}^{N} \varepsilon_k \quad (9)$$

LS-SVM plantea una modificación a la ecuación (9) para ser resuelta mediante un conjunto de ecuaciones lineales. La ecuación (10) es propuesta por LS-SVM.

$$\min_{\omega, b, e} f(\omega, b, e) = \frac{1}{2}\omega^T \omega + \gamma \frac{1}{2}\sum_{k=1}^{N} e_k^2 \quad (10)$$

De acuerdo a la ecuación (8) se construye el lagrangiano descrito por la ecuación (11).

$$\mathcal{L}(\omega, b, e; a) = f(\omega, b, e) - \sum_{k=1}^{N} \alpha_k \{y_k[\omega^T \varphi(x_k) + b] - 1 + e\} \quad (11)$$

Donde αk son los multiplicadores de Lagrange. Se optimiza a partir de la ecuación (12).

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial \omega} = 0 \rightarrow \omega = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \\ &\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \omega = \sum_{k=1}^{N} \alpha_k y_k = 0 \\ &\frac{\partial \mathcal{L}}{\partial e_k} = 0 \rightarrow \alpha_k = \gamma e_k \;\; , \;\; k = 1, ..., N \\ &\frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 \rightarrow \alpha_k = y_k[\omega^T \varphi(x_k) + b] - 1 + e_k \\ &\alpha_k = 0 \;\; , \;\; k = 1, .., N \end{aligned} \quad (12)$$

Las condiciones para optimizar se escriben como la solución a un conjunto de ecuaciones lineales (ecuación (13)).

$$\begin{bmatrix} I & 0 & 0 & -Z^Y \\ 0 & 0 & 0 & -Y^T \\ 0 & 0 & \gamma^I & -I \\ Z & Y & I & 0 \end{bmatrix} \begin{bmatrix} \omega \\ b \\ e \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vec{1} \end{bmatrix}$$
$$Z = [\varphi(x_1)^T y_1; ...; \varphi(x_N)^T y_N]$$
$$Y = [y_1; ...; y_N]$$
$$\vec{1} = [1; ...; 1]$$
$$e = [e_1; ...; e_N]$$
$$\alpha = [\alpha_1; ...; \alpha_N] \quad (13)$$

La solución se expresa también mediante la ecuación (14).

$$\begin{bmatrix} 0 & -Y^T \\ Y & ZZ^T + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix} \quad (14)$$

Y aplicando la condición de Mercer (ecuación (15)).

$$\Omega = ZZ^T$$
$$\Omega_{kl} = y_k y_k \varphi(x_k)^T \varphi(x_1)$$
$$\Omega_{kl} = y_k y_1 \psi(x_k, x_1)$$

Por último el clasificador descrito en la ecuación (2) es encontrado solucionando el conjunto de ecuaciones lineales (14) y (15). Los valores de soporte αx son proporcionales a los errores en la ecuación (12).

3. METODOLOGÍA

3.1. El problema

El problema utilizado para evaluar el desempeño de LS-SVM es un problema de clasificación binaria: clasificar cada uno de los píxeles de una imagen entre dos categorías correspondientes a dos tipos de textura. La clasificación se hace sin extracción de características, los datos de entrada al clasificador son el píxel a clasificar y sus 4 vecinos.

3.2. Data set

Se seleccionaron 8 categorías de texturas de una base de datos [12] de 1000 imágenes (25 categorías de texturas y 40 ejemplos por textura) construida mediante un trabajo en conjunto con: the National Science Foundation, European project LAVA, the UIUC-CNRS Research Collaboration Agreement, the UIUC Campus Research Board, and the Beckman Institute.

Se tomaron 8 ejemplos de cada una de las 8 categorías seleccionadas y se agruparon por pares (granito/corteza, ladrillo/tapicería, madera/mármol y tejido/pelaje) conformando 32 imágenes de 40x40 pixels agrupadas en 4 clases de acuerdo a la combinación de texturas. En la figura 1 se muestra un ejemplo de cada una de las clases de imágenes.

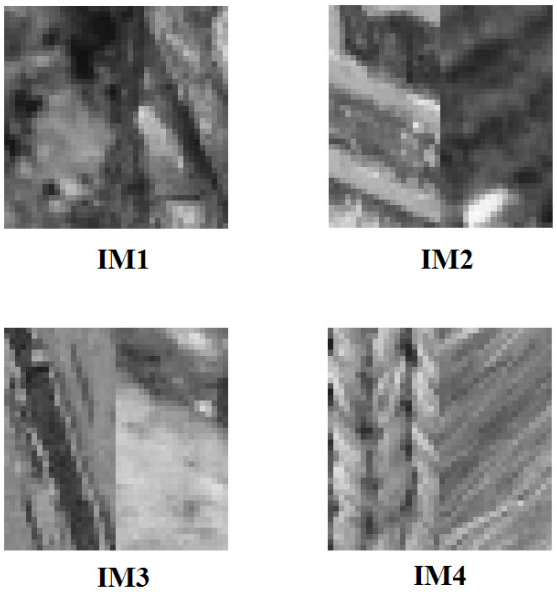

Figura 1. Imágenes de ejemplo de cada una de las 4 clases. IM1: granito/corteza. IM2: ladrillo/tapicería. IM3: madera/mármol. IM4: tejido/pelaje.

3.3. Modelos

Kernel: Es una función que transforma los datos de entrada a un espacio de mayor dimensión donde sea posible su separación lineal. En este artículo se evalúan dos funciones Kernel: Polynomial, que utiliza una ecuación polynomial no homogénea (4) y RBF (Radial Base Funtion) que es una ecuación cuyos valores dependen solo de la distancia a un punto tomado como referencia central. En este caso se utiliza una RBF gausiana (5).

Función de optimización: Función utilizada para hallar de forma óptima los parámetros tuning (gam y sig2) de la LS-SVM. Ésta función toma unos valores tuning iniciales obtenidos mediante CSA (Coupled Simulated Annealing) y hace una búsqueda, que para este articulo puede ser Simplex (optimización multidimensional no lineal sin restricciones que encuentra mínimos locales mediante el algoritmo de Nelder-Mead [13]) o Gridsearch (optimización no lineal que utiliza búsqueda exhaustiva) con el objetivo de encontrar los parámetros tuning mejores posibles de acuerdo a la medida dada por una función de costo.

Función de costo: Es la función que da un estimado del desempeño del modelo ante un conjunto de entrenamiento determinado. Esta función utiliza una validación cruzada, que para efectos de la presente evaluación se ha empleado la random crossvalidation (validación donde se divide aleatoriamente los datos que van a servir de entrenamiento y los que van a servir de prueba) y la leaveoneout (validación que separa los datos de forma que para cada iteración se tenga una sola muestra para los datos de prueba y todo el resto conformando los datos de entrenamiento).

En la tabla 1 se encuentran los modelos a evaluar.

Tabla 1. Modelos a evaluar

MODELO	KERNEL	FUNCIÓN DE OPTIMIZACIÓN	FUNCIÓN DE COSTO
1	Polynomial	Simplex	Leaveoneout
2	Polynomial	Simplex	Random
3	RBF	Simplex	Leaveoneout
4	RBF	Simplex	Random
5	RBF	Gridsearch	Leaveoneout
6	RBF	Gridsearch	Random

3.4. Métricas

Para evaluar una clasificación binaría, que en este caso es separar una textura B (textura del lado derecho de cada una de las imágenes del data set) de una textura A (textura del lado izquierdo de cada una de las imágenes del data set), se han utilizado 3 métricas:

Sensibilidad: Indica la capacidad del clasificador de identificar los píxeles que realmente pertenecen a la textura B. Se determina a partir de la ecuación (16).

$$Sensibilidad = \frac{VP}{(VP + VN)} * 100 \qquad (16)$$

VP: Verdadero positivo.
VN: Verdadero negativo.

Especificidad: Indica la capacidad del clasificador de identificar los píxeles que realmente pertenecen a la textura A. Se determina a partir de la ecuación (17).

$$Especificidad = \frac{VN}{(VN + FP)} * 100 \qquad (17)$$

FP: Falso positivo.
VN: Verdadero negativo.

Precisión: indica la capacidad del clasificador de dar el mismo resultado en mediciones diferentes realizadas en las mismas condiciones. Se determina a partir de la ecuación (18).

$$\Pr ecision = \frac{VP + VN}{(VP + VN + FP + FN)} * 100 \qquad (18)$$

FP: Falso positivo.
VN: Verdadero negativo.

Costo computacional: tiempo de entrenamiento bajo las mismas condiciones computacionales.

4. RESULTADOS

A continuación en la tabla 2 se presentan los valores de sensibilidad, especificidad, precisión y tiempo de cómputo para los 6 modelos evaluados ante las 4 clases de imágenes.

Ademas se presenta el promedio de cada métrica por modelo. En esta tabla los modelos se identifican por números (ver tabla 1 para equivalencia) y las imágenes por abreviaciones (ver figura 1 para equivalencia).

En la figura 2 se ilustra el patrón de salida deseado para todas las imágenes el proceso de clasificación, cero (negro) para la textura A y uno (blanco) para la textura B. En las figura 3, 4, 5 y 6 se ilustran las imágenes resultantes para cada uno de los 6 modelos y cada una de las 4 imágenes.

Tabla 2. Resultados generales

MODELO	SENSIBILIDAD		ESPECIFICIDAD		PRECISIÓN		TIEMPO	
1	IM1	70,75	IM1	72,75	IM1	71,75	IM1	1,14
	IM2	93,37	IM2	82,50	IM2	87,94	IM2	1,01
	IM3	89,12	IM3	95,25	IM3	92,19	IM3	1,17
	IM4	93,25	IM4	81,62	IM4	87,44	IM4	1,15
	AV	86,62	AV	83,03	AV	84,83	AV	1,12
2	IM1	70,75	IM1	72,87	IM1	71,81	IM1	1,12
	IM2	93,50	IM2	82,50	IM2	88,00	IM2	1,05
	IM3	89,12	IM3	95,50	IM3	92,31	IM3	1,04
	IM4	93,25	IM4	81,37	IM4	87,31	IM4	0,95
	AV	86,66	AV	83,06	AV	84,86	AV	1,04
3	IM1	79,50	IM1	79,62	IM1	79,56	IM1	0,87
	IM2	95,12	IM2	85,87	IM2	90,50	IM2	1,39
	IM3	90,12	IM3	95,25	IM3	92,69	IM3	0,97
	IM4	96,37	IM4	88,37	IM4	92,37	IM4	1,25
	AV	90,28	AV	87,28	AV	88,78	AV	1,12
4	IM1	87,78	IM1	88,37	IM1	88,12	IM1	0,92
	IM2	95,12	IM2	86,50	IM2	90,81	IM2	0,95
	IM3	91,25	IM3	95,50	IM3	93,37	IM3	1,03
	IM4	96,00	IM4	87,75	IM4	91,87	IM4	1.05
	AV	92,54	AV	89,53	AV	91,04	AV	0,97
5	IM1	81,87	IM1	80,12	IM1	81,00	IM1	1,09
	IM2	95,12	IM2	85,50	IM2	90,31	IM2	1,02
	IM3	90,25	IM3	95,25	IM3	92,75	IM3	1,07
	IM4	96,00	IM4	88,25	IM4	92,12	IM4	0,99
	AV	90,81	AV	87,28	AV	89,05	AV	1,04
6	IM1	82,75	IM1	84,25	IM1	83,50	IM1	0,94
	IM2	95,12	IM2	85,75	IM2	90,44	IM2	1,02
	IM3	90,87	IM3	95,50	IM3	93,19	IM3	0,90
	IM4	94,62	IM4	86,00	IM4	90,31	IM4	1,34
	AV	90,84	AV	87,88	AV	89,36	AV	1,05

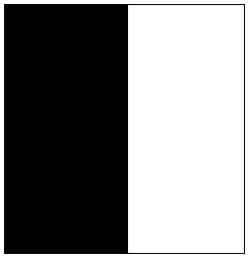

Figura 2. Patrón de salida.

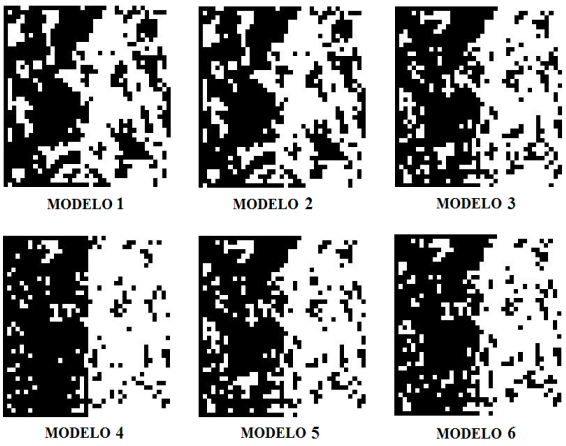

Figura 3. Resultado de la segmentación de una imagen de la clase IM1: granito/corteza para los 6 modelos.

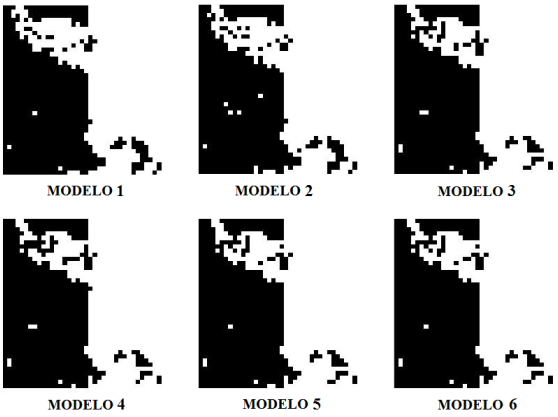

Figura 4. Resultado de la segmentación de una imagen de la clase IM2: ladrillo/tapicería para los 6 modelos.

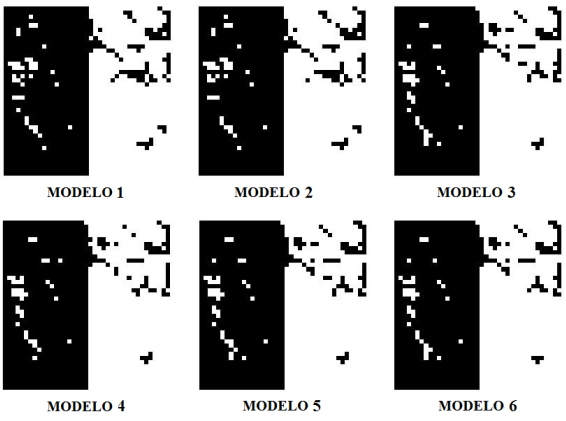

Figura 5. Resultado de la segmentación de una imagen de la clase IM3: madera/mármol para los 6 modelos.

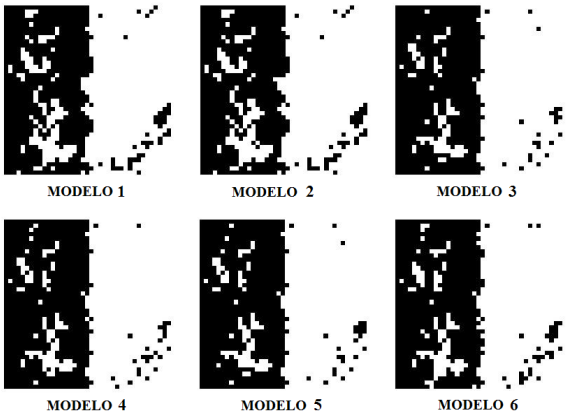

Figura 6. Resultado de la segmentación de una imagen de la clase IM4: tejido/pelaje para los 6 modelos.

5. ANÁLISIS DE RESULTADOS

5.1. Kernel

El kernel Polynomial se evaluó con la función de optimización simplex (la gridsearch no aplica para este tipo de kernel) utilizando 2 funciones de costo (leaveoneout y random). Se obtuvo un desempeño levemente superior, con un leve menor costo computacional cuando se utilizó la función de costo random validate.

El kernel RBF se evaluó con 2 funciones de optimización (simplex y gridsearch) utilizando 2 funciones de costo (leaveoneout y random). Se obtuvo un mejor desempeño con menor costo computacional utilizando una función de optimización simplex y una función de costo random.

Al comparar los dos Kerrnel utilizando la misma función de optimización y de costo (simplex y random) se observó un mejor desempeño por parte del kernel RBF. En cuanto a costo computacional no se marcó una diferencia. En la figura 7 se muestra la comparación de kernels.

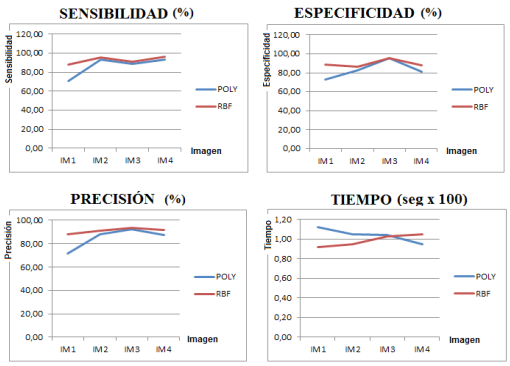

Figura 7. Desempeño de los dos tipos de kernel con función de costo y optimización constantes.

5.1. Función de optimización

La función de optimización simplex se evaluó con ambos kernel y ambas funciones de costo. Se obtuvo un mejor desempeño y un menor costo computacional para el kernel RBF con una función de costo random.

La función de optimización gridsearch se evaluó para el kernel RBF usando ambas funciones de costo. Se obtuvo mejor desempeño para la función de costo random aunque con un leve mayor costo computacional.

Al comparar las 2 funciones de optimización con el mismo kernel y la misma función de costo (RBF y random) se observó un mayor desempeño y un menor costo computacional para la función de optimización gridsearch. En la figura 8 se muestra la comparación de las funciones de optimización.

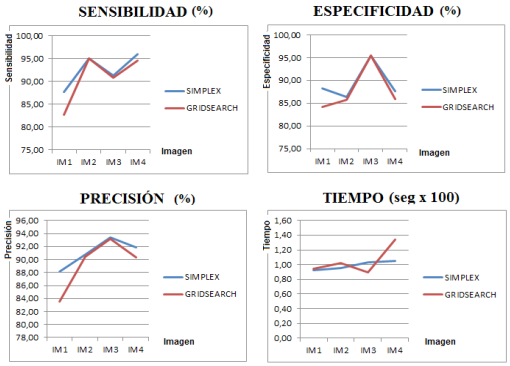

Figura 8. Desempeño de las dos funciones de optimización con kernel y función de costo constantes.

5.3. Función de costo

La función de costo leaveoneout se evaluó con ambos kernel y ambas funciones de costo (la gridsearch únicamente para el kernel RBF). Se obtuvo un mejor desempeño y un menor costo computacional para el kernel RBF con una función de optimización gridsearch.

La función de costo random se evaluó con ambos kernel y ambas funciones de costo (la gridsearch únicamente para el kernel RBF). Se obtuvo un mejor desempeño y un menor costo computacional para el kernel RBF con una función de optimización simplex.

Al comparar las 2 funciones de costo utilizando el mismo kernel y la misma función de optimización (RBF y simplex), se observó un mejor desempeño y un menor costo computacional para la función de costo random. En la figura 9 se muestra la comparación de las funciones de costo.

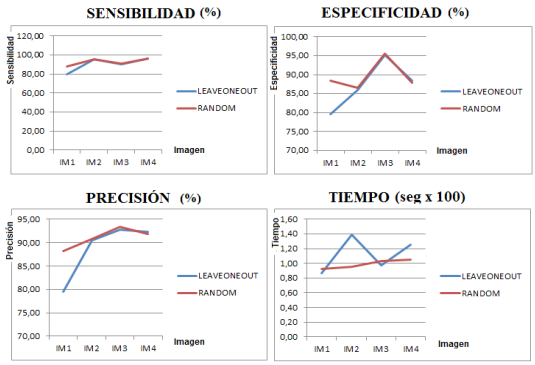

Figura 9. Desempeño de las dos funciones de costo con kernel y función de optimización constantes.

6. CONCLUSIONES

Todos los modelos de LS-SVM evaluados presentaron una sensibilidad superior al 86%, una especificidad superior al 83% y una precisión superior al 84%, lo que valida a LS-SVM como una eficiente herramienta de clasificación de texturas.

Un clasificador de texturas basado en LS-SVM presenta mejor desempeño y menos costo computacional cuando utiliza un kernel RBF

junto con una función de optimización simplex (basada en la búsqueda de mínimos locales mediante el algoritmo de Nelder-Mead) y una función de costo basada en validación cruzada aleatoria.

Referencias Bibliográficas

[1] V. Vapnik and A. Chervonenkis; On the uniform convergence of relative frequencies of events to their probabilities, Theory of Probability and its Applications, 16(2):264–280, 1971.

[2] V. Vapnik; Statistical Learning Theory, Wiley, 1998.

[3] Suykens J.A.K., Vandewalle J; Least squares support vector machine classifiers, Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293-300.

[4] Luts J; Classification of brain tumors based on magnetic resonance spectroscopy, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Jan. 2010, 256 p.

[5] Luts J., Laudadio T., Martinez-Bisbal M.C., Van Cauter S., Molla E., Piquer J., Suykens J.A.K., Himmelreich U., Celda B., Van Huffel S; Differentiation between brain metastases and glioblastoma multiforme based on MRI, MRS and MRSI, in Proc. of the 22nd IEEE International Symposium on Computer-Based Medical Systems (CBMS), Albuquerque, New Mexico, Aug. 2009, pp. 1-8.

[6] Ojeda F; Kernel based Methods for Microarray and Mass Spectrometry Data Analysis, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), May 2011, 150 p.

[7] Danyan Yin, Yiquan Wu; Detection of Small Target in Infrared Image Based on KFCM and LS-SVM, Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2010 2nd International Conference on Volume:1. Publication Year: 2010 , Page(s): 309 - 312.

[8] Guangying Ge, Xinzong Bao, Jing Ge; Study on automatic detection and recognition algorithms for vehicles and license plates using LS-SVM, Intelligent Control and Automation, 2008, WCICA 2008, Publication Year: 2008, Page(s): 3760 - 3765.

[9] Xinming Zhang, Jian Zou; Face Recognition Based on Sub-image Feature Extraction and LS-SVM. Computer Science and Software Engineering, 2008 International Conference on Volume: 1, Publication Year: 2008 , Page(s): 772 - 775.

[10] Sheng Zheng, Wen-zhong Shi, Jian Liu, and Jinwen Tian; Remote Sensing Image Fusion Using Multiscale Mapped LS-SVM. IEEE Transactions on geoscience and remote sensing, VOL. 46, NO. 5, MAY 2008.

[11] Nello Cristianini, John Shawe-Taylor; An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2003.

[12] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce; A Sparse Texture Representation Using Local Affine Regions, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1265-1278, August 2005.

[13] Nelder J. A. and Meayd R; A simplex method for function minimization, Computer. Journal, 7, 308-313, 1965.