Convex Hull in feature space for Support Vector Machines

Edgar Osuna[1] and Osberth De Castro[2]

[1] Departamento de Computación
Universidad Simón Bolívar
Aptdo 89000, Caracas 1080-A
Venezuela
eosuna@bancomercantil.com
[2] Departamento de Electrnica y Circuitos
Universidad Simón Bolívar
Aptdo 89000, Caracas 1080-A
Venezuela
odcastro@usb.ve

Abstract. Some important geometric properties of Support Vector Machines (SVM) have been studied in the last few years, allowing researchers to develop several algorithmic aproaches to the SVM formulation for binary pattern recognition. One important property is the relationship between support vectors and the Convex Hulls of the subsets containing the classes, in the separable case. We propose an algorithm for finding the extreme points of the Convex Hull of the data points in feature space. The key of the method is the construction of the Convex Hull in feature space using an incremental procedure that works using kernel functions and with large datasets. We show some experimental results.

1 Introduction

In the formulation of a **SVM** [1, 2], we find that in feature space the decision surface is always an hyperplane, and the classifier is always written in terms of data instances that belongs to the outside of the *boundaries* of the classes. More specifically, in the separable case, the boundaries of the classes *contain* the instances of solution (support vectors), therefore we only need the points on those boundaries. The boundaries of the data can be obtained from the Convex Hull of each class. In particular, we only need the extreme points (vertices) of the Convex Hull. We show a particular aproach to find these extreme points in feature space, using the so called kernel functions, a key part of **SVM**s formulation. The application area for our method includes incremental training [3, 4], parallel training, and reduction of the run time complexity of **SVM**s [5–7]. Related work on convex geometry for **SVM**s geometry has been developed recently [8–10].

1.1 Support Vector Machines for Pattern Recognition

A Support Vector Machine (**SVM**) is a general optimal model for learning from examples that has become practical in the last five years. This model for pat-

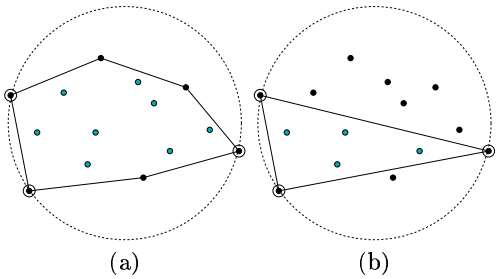


Fig. 1. (a) Relationship between the Convex Hull of a set of points and the smallest hypersphere containing the same points. (b) The first Convex Hull obtained by finding the smallest hypersphere.

tern recognition is based on the Structural Risk Minimization Principle and VC theory, focused on finding the optimal decision surface in terms of the linear function

$$f(\mathbf{x}) = \operatorname{sgn} \sum_{i=1}^{\mathcal{S}} \alpha_i \mathcal{K}(\mathbf{x}, s_i) + b \tag{1}$$

Where $\mathcal{K}$ maps the decision function into a high dimensional *feature space* in which it becomes linear. For example, $\mathcal{K}$ can convert $f(\mathbf{x})$ into a polynomial classifier using $\mathcal{K}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$, or a Radial Basis Learning Machine using a gaussian form, or a Multilayer Neural Network if we use a sigmoidal $\mathcal{K}(\mathbf{x}, \mathbf{y})$ [1]. These kernel functions can be used under certain conditions. The more intuitive condition is that $\mathcal{K}$ must satisty $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$ where Φ is a non linear map to some inner product space in which the linear function lives. When the hyperplane is on input space, $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$.

The usual aproach to train **SVM**s is to solve a quadratic optimization problem with linear constrains, starting from the problem of finding the hyperplane in feature space which maximizes the distance between the class boundary in the separable case. The non separable case is solved by including error penalty variables which transtate in the formulation by creating an upper bound on the QP variables.

2 Finding the Extreme Points of a Convex Hull in feature space

The problem of finding the Convex Hull of a set of points in feature space (also called kernel space) is manageable only if the choosen method is able to make all the calculations using the kernel function instead of mapping the points explicitly in feature space. The kernel functions can be used by writing all the formulations in terms of inner products of data points, which can later be replaced by kernel function evaluations to obtain the final feature space formulation.

Let $\mathcal{L} \in \mathbb{R}^N$ be the set of points. The convex hull of a set of points $\mathcal{L}$ is defined by the set $\mathcal{C} = conv(\mathcal{L})$ that satisfy

$$conv(\mathcal{L}) = \{\mathbf{x} \in \mathcal{L} \mid \mathbf{x} = \sum_{i=1}^{\ell} \lambda_i \mathbf{x}_i\} \text{ where } \sum_{i=1}^{v} \lambda_i = 1, \quad \text{and} \quad \lambda_i \geq 0 \quad (2)$$

Thus, $\mathcal{C}$ are the set of points of $\mathcal{L}$ that can be generated by convex combinations of some subset of elements $\mathcal{V} \in \mathcal{L}$. This subset $\mathcal{V}$ is the set of extreme points of $\mathcal{L}$, and the vertices of the smallest convex polyhedron containing $\mathcal{L}$. The method we show in this paper finds the subset $\mathcal{V}$.

2.1 Finding the extreme points $\mathcal{V}$

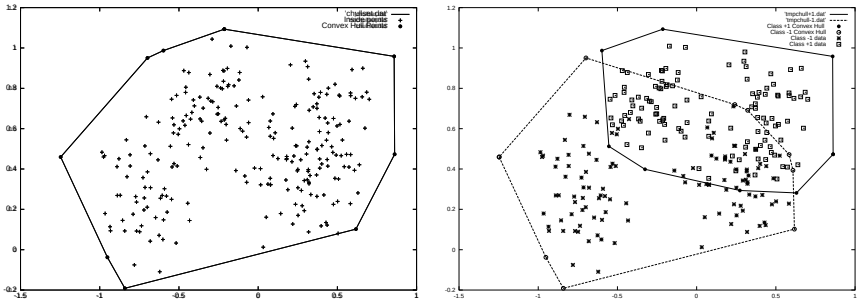
In order to find $\mathcal{V}$, we use an incremental algorithm that uses the following ideas:

1. Checking the containment of a point in a Convex Hull can be done by: **1.** Solving a convex quadratic optimization program formulated in terms of inner products, so it can be solved in feature space, or **2.** Solving a linear program that tries of find a separating hyperplane between a point and the rest of the data set.
2. Using a measure of the distance to the center of the smallest hypersphere containing $\mathcal{L}$ gives us admissible spatial knowledge in order to use heuristic procedures to find $\mathcal{V}$.

These ideas take us to an incremental algorithm that constructs the set $\mathcal{V}$ based on the iterative inclusion of points in a candidate set of extreme points $\mathcal{V}_0$, based on whether it can be written as a convex combination of the points on $\mathcal{V}_0$. We use a *from outside to inside* inclusion order defined by the distance of the point to the center of the smallest hypersphere containing $\mathcal{L}$. We do this until we have checked all the points. At the end, the algorithm has a candidate $\mathcal{V}_0$ containing the solution $\mathcal{V}$ and some *extra* interior points near the boundaries of the Convex Hull defined by $\mathcal{V}$. De final step is a *one against all* check, to discard the *extra* interior points.

Initial Condition The most important condition is choosing the first $\mathcal{V}_0$. It can be shown that the points lying on the surface of the smallest hypersphere containing $\mathcal{L}$ are also in $\mathcal{V}$ (see figure 1(b)), then our first $\mathcal{V}_0$ are these points. The calculation of this hypersphere was done using a large scale incremental implementation in feature space previously used with other aplications using **SVM**s [11, 6]. In what follows, given a set $\mathcal{L}$, we will call the points on the surface of the smallest hypersphere containing $\mathcal{L}$, $SphereSet(\mathcal{L})$.

The Algorithm Let $\mathcal{V}$ be the set of extreme points of $\mathcal{L}$, and $\mathbf{x} \in \mathcal{L}$. Suppose that we have the functions $SphereSet(\mathcal{L})$ and $SphereSort(\mathcal{L})$. The first returns



(a) (b)

Fig. 2. (a) The Convex Hull of the Ripley dataset.(b) The Convex Hulls for classes +1 and -1 on Ripley dataset.

the subset of $\mathcal{L}$ lying on the surface of the smallest sphere containing $\mathcal{L}$, and the second returns a descending sort list of $\mathcal{L}$ by the distance to the center of the same sphere. We have also the function $CheckPoint(\mathbf{x}, \mathcal{V})$ returning $TRUE$ if $\mathbf{x}$ belongs the interior of the Convex Hull defined by $\mathcal{V}$. In what follows, $\mathcal{V}_0$ is a set of candidate extreme points.

$ExtremePoints(\mathcal{L})$

1. Initialize $\mathcal{V}_0 = \{Sphere(\mathcal{L})\}$, and $\mathcal{V} = \emptyset$
2. Create the sorted list $\mathcal{L}^* = \{SphereSort(\mathcal{L})\} - \mathcal{V}_0$
3. Until $\mathcal{L}^*$ is empty
   - Get first $\mathbf{x} \in \mathcal{L}^*$, update $\mathcal{L}^* = \mathcal{L}^* - \{\mathbf{x}\}$
   - If $(CheckPoint(\mathbf{x}, \mathcal{V}_0) = FALSE)$ then $\mathcal{V}_0 = \mathcal{V}_0 \cup \{\mathbf{x}\}$
4. Until $\mathcal{V}_0$ is empty
   - Get next $\mathbf{x} \in \mathcal{V}_0$
   - If $(CheckPoint(\mathbf{x}, \mathcal{V}_0 - \{\mathbf{x}\}) = FALSE)$ then $\mathcal{V} = \mathcal{V} \cup \{\mathbf{x}\}$

At step 4, $\mathcal{V}_0 = \mathcal{V} + \mathcal{A}$, where $\mathcal{A}$ is a set of extra points near the surface of the boundaries of the Convex Hull, and $\mathcal{V}$ the set of extreme points. $\mathcal{A}$ is eliminated in 4. The Algorithm passes once through all $\mathcal{L}$, and twice on $\mathcal{V}_0$. The advantage of this algorithm is that it never uses all the points on a $CheckPoint()$ operation, and the biggest set used is $\mathcal{V}_0$. In most cases, the *from outside to inside* incremental procedure allows us to obtain a small set $\mathcal{A}$. In the following sections we give some remarks on the mathematical formulation in feature space for the functions used by the algorithm.

3 Feature space Mathematics

In this section we present the mathematical formulation that allows us to use the previous algorithmic aproach in feature space. Thus, we analize the functions used in the section 2.1 on input and feature space.

3.1 Finding the Hypersphere of $\mathcal{L}$

We use the same formulation used in [11, 6]. The problem of finding the radius of the smallest sphere containing $\mathcal{L}$ is solved by minimizing the largest distance R between a variable center point $\mathbf{a}$ and every point $\mathbf{x}$.

$$\min_{\mathbf{a}} \max_{i=1,\ldots,\ell} R(\mathbf{a}, \mathbf{x}_i) \tag{3}$$

Which can be written in dual form as:

$$\max_{\mathbf{\Lambda}} \textstyle\sum_{i=1}^{\ell} \lambda_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{\Lambda}^T Q \mathbf{\Lambda} \text{ where } Q_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \tag{4}$$
$$\text{with constrains} \quad \mathbf{\Lambda}^T \mathbf{1} = 1, \ -\mathbf{\Lambda} \leq 0$$

Where $\mathcal{K}(\mathbf{x}, \mathbf{y})$ is the kernel functions making the implicit mapping to feature space for $\mathbf{x}$ and $\mathbf{y}$ and computing their dot product. The solution of this QP yields the sphere radius R, the set of points lying on the surface (points whose coefficient are $\lambda > 0$), and a representation for the center $\mathbf{a}$ (as a linear combination of the points on the surface). The distance from any point $\mathbf{y}$ to the center a can be obtained by:

$$d(\mathbf{a}, \mathbf{y}) = \sqrt{\textstyle\sum_{i,j}^{s} \alpha_i \alpha_j \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) + \mathcal{K}(\mathbf{y}, \mathbf{y}) - 2 \sum_i \alpha_i \mathcal{K}(\mathbf{x}_i, \mathbf{y})} \tag{5}$$

Where $\mathcal{S}$ is the set of surface points, $\mathbf{x} \in \mathcal{S}$, and α are the variables in (4).

3.2 Checking a point y in a Convex Hull $\mathcal{C}$ defined by $\mathcal{V}$

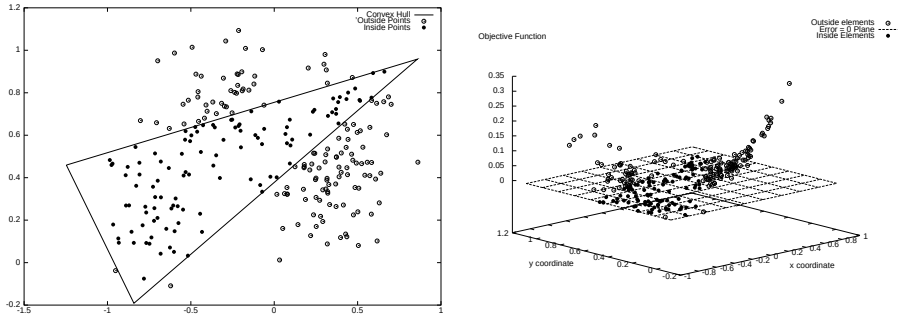
In this section we show a couple of formulations that allow us to check in feature space the containment of a point in the interior of the Convex Hull $\mathcal{C}$ defined by the vertices $\mathcal{V}$.

Writing a point as a Convex Combination in feature space We have formulated this problem as a linearly constrained convex quadratic optimization program that minimizes an error measure between $\mathbf{y}$ itself and the aproximation of convex combinations of points in $\mathcal{V}$. If we can minimize this measure to cero, $\mathbf{y}$ can be written as a convex combination of $\mathbf{x} \in \mathcal{V}$. Formally,

$$\min_{\lambda_i} f(\mathbf{y}, \lambda) = \left(\mathbf{y} - \textstyle\sum_{i=1}^{v} \lambda_i \mathbf{x}_i \right)^2 \quad \text{with} \quad \sum_{i=1}^{v} \lambda_i = 1, \quad \text{and} \quad \lambda_i \geq 0, \quad \mathbf{x}_i \in \mathcal{V} \tag{6}$$

Which can be reduced to an expression in terms of inner products in input space as:

$$\min_{\mathbf{\Lambda}} f(\mathbf{y}, \mathbf{\Lambda}) = \mathbf{y}^T \mathbf{y} + \mathbf{\Lambda}^T \mathbf{Q} \mathbf{\Lambda} - 2 \textstyle\sum_{i=1}^{v} \lambda_i \mathbf{x}_i^T \mathbf{y} \qquad \text{where } Q_{ij} = \mathbf{x}_i \mathbf{x}_j \tag{7}$$
$$\text{with constrains } \mathbf{\Lambda}^T \mathbf{1} = 1, \ -\mathbf{\Lambda} \leq 0, \ \mathbf{x}_i \in \mathcal{V}$$



(a) (b)

Fig. 3. (a) *Checkpoint* Test on all the Ripley data set for a first Convex hull obtained using the hypersphere calculation from section 3.1. (b) Values of the objective function when $ChecPoint(\mathbf{x}, \mathcal{V}_0)$ is formulated as a QP in section 3.2 the test.

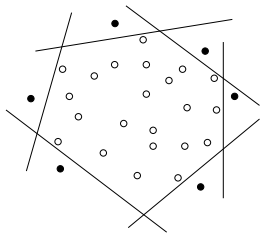


Fig. 4. Separating hyperplanes and the extreme points. We can see that for the interior points the hyperplane can't be found.

We can traslate this QP to the feature space form replacing the inner products by kernel functions, obtaining:

$$\min_{\mathbf{\Lambda}} \; f(\mathbf{y}, \mathbf{\Lambda}) = \mathcal{K}(\mathbf{y}, \mathbf{y}) + \mathbf{\Lambda}^T \mathbf{Q} \mathbf{\Lambda} - 2 \sum_{i=1}^{v} \lambda_i \mathcal{K}(\mathbf{x}_i, \mathbf{y}) \quad \text{where } Q_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \tag{8}$$

$$\text{with constrains } \mathbf{\Lambda}^T \mathbf{1} = 1, \; -\mathbf{\Lambda} \leq 0, \; \mathbf{x}_i \in \mathcal{V}$$

When we evaluate a point $\mathbf{y}$ outside $\mathcal{C}$, the final objective function is $f^*(\mathbf{y}, \mathbf{\Lambda}) > 0$. If $\mathbf{y}$ is inside the polyhedron defined by $\mathcal{C}$ then $f^*(\mathbf{y}, \mathbf{\Lambda}) = 0$. Figure 3 shows an experimental demonstration in the 2d dataset Ripley[3]. Figure 2(b) shows the Convex Hull for each class of points (Ripley is a binary classification dataset).

Finding an extreme point using a Separating hyperplane The hyperplane separating a point $\mathbf{x}_i \in \mathcal{L}$ from the rest $\{\mathcal{L}\} - \mathbf{x}_i$, must satisty

$$\mathbf{w} \cdot \mathbf{x}_i + b <= 0 \tag{9}$$

[3] Available on ftp://markov.stats.ox.ac.uk/pub/neural/papers

$$\mathbf{w} \cdot \mathbf{x}_j + b >= 0 \; \forall j \neq i$$

In order to formulate the problem of finding this hyperplane for any point in $\mathcal{L}$, not only the extreme points, we introduce penalty variables P for each point in $\mathcal{L}$. Therefore, we can formulate a problem wich tries to minimize P_i for every point. Formally,

$$\begin{aligned} &\min_{\lambda,b,P_i} \; \sum_{i=1}^{\ell} P_i \qquad (10) \\ &\text{with constrains } \sum_{j=1}^{\ell} \lambda_j \mathbf{x}_j \cdot \mathbf{x}_i + b - P_i <= 0 \\ &\qquad\qquad \sum_{j=1}^{\ell} \lambda_j \mathbf{x}_j \cdot \mathbf{x}_z + b + P_z >= 0 \; \forall z \neq i, \; P_z, P_i >= 0 \end{aligned}$$

Since the problem has the form of dot products, we can replace all of these operations with kernel functions, obtaining the feature space formulation

$$\begin{aligned} &\min_{\lambda,b,P_i} \; \sum_{i=1}^{\ell} P_i \qquad (11) \\ &\text{with constrains } \sum_{j=1}^{\ell} \lambda_j \mathcal{K}(\mathbf{x}_j, \mathbf{x}_i) + b - P_i <= 0 \\ &\qquad\qquad \sum_{j=1}^{\ell} \lambda_j \mathcal{K}(\mathbf{x}_j, \mathbf{x}_z) + b + P_z >= 0 \; \forall z \neq i, \; P_z, P_i >= 0 \end{aligned}$$

In this formulation the variables b and λ_j are free $\forall j$. It's easy to see that eq. (11) solves the problem for the point $\mathbf{x}_i$ in $\mathcal{L}$. At the end of the minimizing process, if we can minimize all the penalties P_i to cero, the hyperplane separating $\mathbf{x}_i$ from the rest has been found, and we can say that $\mathbf{x}_i$ is an extreme point of $\mathcal{L}$.

4 Conclusions and Final Remarks

In this paper we have shown a procedure to compute the set of extreme points defining the Convex Hull of a data set in feature space. The difference with previous convex hull computation algorithms used in the **SVM** arena [3, 12] is that our aproach doesn't need explicit knowledge of dimensionality or explicitly mapping the data points in feature space. A first application area for our method is incremental and parallel training speedup, where work reported in [3, 4] could be extended to feature space very easily, although some extensions would need to be worked out to include non-separable data. A second possible application would be reduction **SVM** run time complexity [5–7], since, for example, some misclassified support vectors could in principle be rewritten as convex combination of extreme points of the data that are already support vectors. Another interesting property of the method is its dimensionality independence as a general geometric Convex Hull extreme points algorithm, when compared with other algorithms like Quickhull [13], and divide and conquer methods.

One important drawback to be dealt with in this topic is that the complexity of the convex hull and the number of extreme points have an exponential dependence on the dimensionality of the feature space. We have used data in 2 and 3 dimensions to test the algorith, but the adaptation of this aproach to be used in large scale applications and higher dimension feature spaces is subject of further work.

Acknowledgments Edgar Osuna also works in the Risk Management Department of Banco Mercantil, Caracas, Venezuela. We would like to thank José Ramírez for his useful comments.

References

1. V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1.995.
2. C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.
3. D. Caragea, A. Silvescu, and V. Honavar. Agents that learn from distributed dynamic data sources. In *Proceedings at The Fourth International Conference on Autonomous Agents*, pages 53–60, Barcelona, Catalonia, Spain, 2000.
4. N. A. Syed, H. Liu, and K. Kay Sung. Incremental learning with support vector machines. In J. Debenham, S. Decker, R. Dieng, A. Macintosh, N. Matta, and U. Reimer, editors, *Proceedings of the Sixteenth International Joint Conference on artificial Intelligence IJCAI-99*, Stockholm, Sweden, 1.999.
5. C. Burges. Simplified support vector decision rules. In Lorenza Saitta, editor, *Proceedings or the Thirteenth International Conference on Machine Learning*, pages 71 – 77, Bari, Italia, 1.996.
6. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
7. E. Osuna and F. Girosi. Reducing the run-time complexity of support vector machines. *Advances in Kernel Methods, Support Vector Learning*, 1998.
8. K. Bennett and E. Bredensteiner. Duality and geometry in SVM classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 57–64. Morgan Kaufmann, San Francisco, CA, 2000.
9. K. Bennett and E. Bredensteiner. Geometry in learning. In C. Gorini, E. Hart, W. Meyer, and T. Phillips, editors, *Geometry at Work*, Washington, D.C., 1997. Mathematical Association of America.
10. S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE-NN*, 11(1):124, 2000.
11. E. Osuna. *Support Vector Machines: Trainig and Applications, PhD Thesis.* MIT, Cambridge, 1.998.
12. C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4), 1996.
13. C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

Título del Artículo:

Convex Hull in Feature Space for Support Vector Machines.

Autores:

Edgar Osuna
Dirección: Universidad Simón Bolívar, Edificio de Matemáticas y Sistemas, 2do piso, Departamento de Computación y Tecnología de la Información, Sartenejas, Baruta, Edo Miranda, Aptdo 89000, Caracas 1080-A, Venezuela.
Tlf. +58-212-9063231
eosuna@bancomercantil.com
Osberth De Castro Dirección: Universidad Simón Bolívar, Edificio de Física y Electrónica I, 3er piso, Departamento de Electrónica y Circuitos, Sartenejas, Baruta, Edo Miranda, Aptdo 89000, Caracas 1080-A, Venezuela.
Tlf. +58-212-9063630
odcastro@usb.ve

Resumen:

Some important geometric properties of Support Vector Machines (SVM) have been studied in the last few years, allowing researchers to develop several algorithmic aproaches to the SVM formulation for binary pattern recognition. One important property is the relationship between support vectors and the Convex Hulls of the subsets containing the classes, in the separable case. We propose an algorithm for finding the extreme points of the Convex Hull of the data points in feature space. The key of the method is the construction of the Convex Hull in feature space using an incremental procedure that works using kernel functions and with large datasets. We show some experimental results.

Palabras claves:

clustering, optimization, artificial neural models, statistical & probabilistical data minig, classification.

Tópicos:

Aprendizaje Automático, Descubrimiento de Conocimiento y Minería de Datos.

Sección a Aplicar:

Paper Track