

# An Algorithm for the Exact Computation of the Centroid of Higher Dimensional Polyhedra and its Application to Kernel Machines

Frederic Maire  
Smart Devices Laboratory,  
School of SEDC, IT Faculty,  
Queensland University of Technology,  
2 George Street, GPO Box 2434,  
Brisbane Q 4001, Australia.

## Abstract

The Support Vector Machine (SVM) solution corresponds to the centre of the largest sphere inscribed in version space. Alternative approaches like Bayesian Point Machines (BPM) and Analytic Centre Machines have suggested that the generalization performance can be further enhanced by considering other possible centres of version space like the centroid (centre of mass) or the analytic centre. We present an algorithm to compute exactly the centroid of higher dimensional polyhedra, then derive approximation algorithms to build a new learning machine whose performance is comparable to BPM. We also show that for regular kernel matrices (Gaussian kernels for example), the SVM solution can be obtained by solving a linear system of equalities.

## 1. Introduction

In the Kernel Machine framework [5, 8], a feature mapping  $x \mapsto \phi(x)$  from an input space to a feature space is given (generally, implicitly via a kernel function), as well as a training set  $T$  of pattern vectors and their class labels  $\{(x^1, y_1), \dots, (x^m, y_m)\}$  where the class labels are in  $\{-1, +1\}$ . The learning problem is formulated as a search problem for a linear classifier (a weight vector  $w$ ) in the feature space. Because only the direction of  $w$  matters for classification purpose, without loss of generality, we can restrict the search for  $w$  to the unit sphere. The set of weight vectors  $w$  that classify correctly the training set is called *version space* and denoted by  $\mathcal{V}(T)$ . Version space is the region of feature space defined as the intersection of the unit sphere and the polyhedral cone of the feature space  $\{w | \forall i \in [1, m], \langle w, y_i \phi(x^i) \rangle \geq 0\}$ .

The training algorithm of a Support Vector Machine

(SVM) returns the central direction  $w_{\text{svm}}$  (a unit vector) of the largest spheric cone contained in the polyhedral cone  $\{w | \forall i \in [1, m], \langle w, y_i \phi(x^i) \rangle \geq 0\}$ . The weight vector  $w_{\text{svm}}$  can be expressed as a linear combination of the vectors  $y_i \phi(x^i)$ 's. That is, there exist  $(\alpha_1, \dots, \alpha_m)$  such that  $w_{\text{svm}} = \sum_{i=1}^m \alpha_i y_i \phi(x^i)$ .

The Kernel trick is that for some feature spaces and mappings  $\phi$ , there exist easily computable kernel functions  $k$  defined on the input space such that  $k(x, y) = \langle \phi(x), \phi(y) \rangle$ . A new input vector  $x$  is classified with the sign of

$$\langle w_{\text{svm}}, x \rangle = \sum_{i=1}^m \alpha_i y_i \langle \phi(x^i), \phi(x) \rangle = \sum_{i=1}^m \alpha_i y_i k(x^i, x)$$

With a kernel function  $k$ , the computation of inner products  $\langle \phi(x), \phi(y) \rangle$  does not require the explicit knowledge of  $\phi$ . In fact for a given kernel function  $k$ , there may exist many suitable mappings  $\phi$ .

Bayes Point Machines (BPM) are a well-founded improvement over to SVM which approximate the Bayes-optimal decision by the centroid (also known as the centre of mass or barycentre) of version space. It happens that the Bayes point is very close to the centroid of version space in high dimensional spaces. The Bayes point achieves better generalization performance in comparison to SVM [6, 9, 1, 10].

An intuitive way to see why the centroid is a good choice is to view version space as a (infinite) committee of experts who all are consistent with the training set. A new and unlabelled input vector corresponds to a hyperplane in feature space that may split version space in two. It is reasonable to use the opinion of the majority of the experts that were consistent with the training set to predict the class label of a new pattern. The expert that agrees the most with the majority vote on new inputs is precisely the Bayesian point. In a standard committee machine, for each new input we seek the opinions of a finite number of experts then take a

majority vote, whereas with a BPM, the expert that most often agrees with the majority vote of the infinite committee (version space) is delegated the task of classifying the new inputs.

Following Rujan [7], Herbrich and Graepel [2] introduced two algorithms to stochastically approximate the centroid of version space; a billiard sampling algorithm and a sampling algorithm based on the well known perceptron algorithm.

In this paper, we present an algorithm to compute exactly the centroid of a polyhedron in high dimensional spaces. From this exact algorithm, we derive an algorithm to approximate a centroid position in a polyhedral cone. We show empirically that the corresponding machine presents better generalization capability than SVMs on a number of benchmark data sets.

In Section 2, we introduce an algorithm to compute exactly the centroid of higher dimensional polyhedra. In Section 3, we show a simple algorithm to compute the SVM solution of regular kernels. In Section 4, we sketch the idea of Balancing Board Machines. In Section 5, some implementation issues are considered and some experimental results are presented.

## 2. Exact Computation of the Centroid of a Higher Dimensional Polyhedron

A polyhedron  $P$  is the intersection of a finite number of half-spaces. It is best represented by a system of non redundant linear inequalities  $P = \{x \mid Ax \leq b\}$ . Recall that the 1-volume is the length, the 2-volume is the surface and the 3-volume is the every-day-life volume. The algorithm that we introduce for computing the centroid of an  $n$ -dimensional polyhedron is an extension of the work by Lasserre [4] who showed that the  $n$ -dimensional volume  $V(n, A, b)$  of a polyhedron  $P$  is related to the  $(n - 1)$ -dimensional volumes of its facets and the row vectors of its matrix  $A$  by the following formula;

$$V(n, A, b) = \frac{1}{n} \sum_i \frac{b_i}{\|a_i\|} \times V_i(n - 1, A, b)$$

where  $a_i$  denotes the  $i^{th}$  row of  $A$  and  $V_i(n - 1, A, b)$  denotes the  $(n - 1)$ -dimensional volume of the  $i^{th}$  facet  $P \cap \{x \mid a_i^T x = b_i\}$ . We obtain the centroid and the  $(n - 1)$ -volume of a facet by variable elimination. Geometrically, this amounts to projecting the facet onto an axis parallel hyperplane, then computing the volume and the centroid of this projection recursively in a lower dimensional space. From the volume and centroid of the projected facet, we can derive the centroid and volume of the original facet.

The  $n$ -volume  $V$  and the centroid  $G$  of a cone rooted at 0 are related to the  $(n - 1)$ -volume  $V_h$  and the centroid

$G_h$  of its intersection with the hyperplane  $x_n = h$  by the following equalities;

$$\begin{aligned} V &= \int_0^h V_x dx = \int_0^h V_h \times \left(\frac{x}{h}\right)^{n-1} dx = \frac{h}{n} \times V_h \\ V \times \overrightarrow{OG} &= \int_0^h \overrightarrow{OG_x} \times V_x \times dx \\ \overrightarrow{OG} &= \frac{n}{n+1} \times \overrightarrow{OG_h} \end{aligned}$$

The above formulae were derived by considering the  $n$ -fold integral defining the  $n$ -dimensional volume. These formulae allow a recursive computation of the centroid of a polyhedron  $P$  by partitioning  $P$  into polyhedral cones generated by its facets.

It is useful to observe that the computation of the volume and the centroid of a  $(n - 1)$ -dimensional polyhedron in a  $n$ -dimensional space is identical to the computation of the volume and the centroid of a facet of a  $n$ -dimensional polyhedron.

---

**Algorithm 1**  $[G, V] = \text{measurePolyhedron}(P)$

---

**Require:**  $P = \{x \mid Ax \leq b\}$  non-empty and irredundant

**Ensure:**  $G$  is the centroid of  $P$ , and  $V$  its volume

{ $m$  is the number of rows of  $A$ ,  $n$  is its number of columns}

**for**  $i = 1$  to  $m$  **do**

{Compute recursively the centroids  $G_{F_i}$  and the  $(n - 1)$ -volumes  $V_{F_i}$  of each facet  $F_i$  of  $P$ }

$[G_{F_i}, V_{F_i}] = \text{measurePolyhedron}(F_i)$

**end for**

{Compute  $G_E$ , the centroid of the envelope of  $P$ }

$G_E = \sum_i \frac{V_{F_i}}{\sum_j V_{F_j}} \times G_{F_i}$

**for**  $i = 1$  to  $m$  **do**

{Compute recursively the centroids  $G_{C_i}$  and the  $(n - 1)$ -volumes  $V_{C_i}$  of each cone  $C_i = \text{cone}(G_E, F_i)$  rooted at  $G_E$  and generated by  $F_i$ }

Compute  $h_i$ , the distance from  $G_E$  to the hyperplane containing  $F_i$

$V_{C_i} = \frac{h_i}{n} \times V_{F_i}$

$\overrightarrow{G_E G_{C_i}} = \frac{n}{n+1} \times \overrightarrow{G_E G_{F_i}}$

**end for**

$V = \sum_i V_{C_i}$

$G = \sum_i \frac{V_{C_i}}{V} \times G_{C_i}$

---

The Matlab code for this algorithm is available at <http://www.fit.qut.edu.au/~maire/G>

### 3. Computing the Spheric Centre of a Polyhedral Cone Derived from a Non Singular Mercer Kernel Matrix

Let  $P = \{x | Ax \leq 0\}$  be the non-empty polyhedral cone derived from a non-singular kernel matrix. The matrix  $A$  is square ( $m = n$ ). Without loss of generality, we assume that its rows are normalized. Recall that the centre of the largest spheric cone contained in the polyhedral cone  $P$  is the SVM solution  $w_{\text{svm}}$ . Because  $A$  is square and non-singular, each facet of the polyhedral cone touches the spheric cone. If each facet is moved by a distance of one in the direction of its normal vector, the new cone obtained is a translation of the original cone in the direction of  $w_{\text{svm}}$ . That is  $w_{\text{svm}}$  can be obtained by simply solving  $Ax = -\mathbf{1}$  instead of solving the traditional constrained quadratic optimization problem.

### 4. Balancing Board Machines

The point of contact of a board posed in equilibrium on a sphere (assumed to be the only source of gravity) is the centroid of the board. This observation is the basis of our *balancing board algorithm*. In the rest of this paper, the term *board* will refer to the intersection of the polyhedral cone of version space with a hyperplane normal to a unit vector of version space. This definition implies that if the polyhedral cone is  $n$ -dimensional then a board will be a  $(n - 1)$ -dimensional polyhedron tangent to the unit sphere. In the algorithm we propose, the approximation of the centroid direction of the polyhedral cone of  $\mathcal{V}(T)$  is refined by computing the centroid of the board normal to a current estimate  $w$ , and then rotating  $w$  towards the centroid of the board (stopping at a local minimum of the volume of the board in a line search).

Once we know an orthonormal basis  $U$  of  $V$  the vector subspace generated by  $\{\phi(x^1), \dots, \phi(x^m)\}$  (the orthonormality is with respect to the inner product in feature space corresponding to the kernel function in the input space), we can express the polyhedral cone inequalities with respect to this orthonormal basis. Then we can apply the formulae of Section 2 to compute the centroid of any polyhedron expressed in this orthonormal basis. The kernel PCA basis is a suitable orthonormal basis  $U$  of  $V$ . The vectors of the kernel PCA basis  $U$  are the eigenvectors of the kernel matrix  $K = (k(x^i, x^j))_{i,j}$ . By expressing the polyhedral cone defined by the training examples in the orthonormal basis  $U$ , we will be able to approximate a centroid direction with the board balancing algorithm sketched above.

The complexity of the algorithm of Section 2 to compute exactly the centroid is unfortunately exponential. The computational cost of the exact calculation of the centroid is too high even for medium size data sets. However, the recursive

formulae allow us to derive an approximation of the volume and the centroid of a polyhedron once we have approximations for the volumes and the centroids of its facets.

Because the balancing board algorithm requires several board centroid estimations, it is desirable to recycle intermediate results as much as possible to achieve a significant reduction in computation time. Because the intersection of a hyperplane and a spheric cone is an ellipsoid, we estimate the volume and the centroid of the intersection of the board and a facet of the polyhedral cone (this intersection is  $(n-2)$ -dimensional) with the volume and the centroid of the intersection of the board and the largest spheric cone contained in the facet (this spheric cone is  $(n-1)$ -dimensional). The computation of the largest spheric cones contained in each facet of the polyhedral cone is done only once. The centre of the ellipsoid and its quadratic matrix is easily derived from the centre and radius of the spheric cone.

To simplify the computations, we have restricted our study to non-singular kernel matrices (like those obtained from Gaussian kernels). This way, we were able to use the geometric trick of Section 3 to compute the centres of the spheric cones.

The change of basis is performed as follows. Let  $w_B$  be the coordinates of  $w$  with respect to  $B$ . Recall that the Kernel PCA basis  $U$  is made of the eigenvectors of the symmetric matrix  $K$ . Let  $w_U$  be the coordinates of  $w$  with respect to the basis  $U$ . We have  $w_B = U w_U$ . We are looking for  $w_B$  such that

$$\begin{cases} -\text{diag}(y)K w_B \leq 0 \\ \langle w, w \rangle = 1 \end{cases}$$

and  $w$  near the centroid direction of the polyhedral cone. Here,  $\text{diag}(y)$  represents the diagonal matrix made with the entries of vector  $y$ . As we have  $\langle w, w \rangle = (w_U)^T w_U$ , in practice, we look for the centroid direction of

$$\begin{cases} -\text{diag}(y)K U w_U \leq 0 \\ (w_U)^T w_U = 1 \end{cases}$$

### 5. Implementation Issues and Experimental Results

We have implemented the exact computation of the centroid and the volume in Matlab. A direct recursive implementation of Lasserre formula would be very inefficient as faces of dimension  $k$  share faces of dimension  $k - 1$ . Our implementation caches the volumes and centroids of the lower dimensional faces in a hash-table.

Our algorithm has been validated by comparing the values returned with a Monte-Carlo method.

The kernel matrix of a Gaussian kernel can only be singular when identical input vectors occur more than once the training set. We remove repeated occurrences of the same

input vector and assign the most common label for this input vector to the occurrence that we leave in the training set.

The table that follows summarises generalization performance (percentage of correct predictions on test sets) of the Balancing Board Machine (BBM) on 6 standard benchmarking data sets from the UCI Repository, comparing results for illustrative purposes with equivalent hard margin support vector machines. In each case the data was randomly partitioned into 20 training and test sets in the ratio 60% 40%.

Data set	SVM	BBM
heart disease	58.36	58.40
thyroid	94.34	95.23
diabetes	66.89	67.68
waveform	83.50	83.50
sonar	85.06	85.78
ionsphere	86.79	86.86

**Table 1. UCI benchmark Data Sets**

The results obtained with a BBM are comparable to those obtained with a BPM, but the improvement is not always as dramatic as those reported in [3]. We observed that the improvement was generally better for smaller data sets. We suspect that this is due to the fact the volumes considered become very small in high dimensional spaces. In fact, on a PC, unit spheres “vanish” when their dimension exceed 340. The volume of a unit sphere of dimension 340 is  $4.5 \cdot 10^{-223}$ . This is why we consider the logarithm of the volume in our programs.

## 6. Conclusions

The exact computation algorithm can be useful for benchmarking to people developing new centroid approximation algorithms. We do not claim that our BBM approach is superior to any other given that the computational cost is in the order of times the cost of a SVM computation (where is the number of training examples).

Replacing the ellipsoids with a more accurate estimation would probably give better results, but deriving the volume and the centroid of the intersection of a facet and a board from the volume and the centroid of the intersection of the same facet with another board seems to be a hard problem.

The computation of the SVM point presented in Section 3 provides an efficient learning algorithm for Gaussian kernels.

## Acknowledgement

I would like to thank Professor Tom Downs and Professor Peter Bartlett for their valuable comments on a previous

version of the BBM algorithm. This work was partially supported by an ATN grant.

## References

- [1] T. Graepel, R. Herbrich, and C. Campbell. Bayes point machines: Estimating the bayes point in kernel space. In *in Proc. of IJCAI Workshop Support Vector Machines*, pages 23–27, 1999.
- [2] R. Herbrich and T. Graepel. Large scale bayes point machines. In *in Advances in Neural Information System Processing 13*, pages 528–534, 2001.
- [3] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1, pages 245–279, 2001.
- [4] J. Lasserre. An analytical expression and an algorithm for the volume of a convex polyhedron in  $r^n$ . *Journal of Optimization Theory and Applications*, Vol 39, No 3, pages 363–377, 1983.
- [5] K. Muller, S. Mika, G. Ratch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. on NN*, vol 12, no 2, 2001.
- [6] M. Opper and D. Haussler. Generalization performance of bayes optimal classification algorithm for learning a perceptron. *Phys. Rev. Lett.* vol. 66, pages 26–77, 1991.
- [7] P. Rujan. Playing billiard in version space. *Neural Comput.*, vol. 9, pages 197–238, 1996.
- [8] B. Scholkopf and A. Smola. *Learning with Kernels*. The MIT Press, Cambridge, Massachusetts, London, England, 2002.
- [9] J. Shawe-Taylor and R. C. Williamson. A pac analysis of a bayesian estimator. Technical report, Royal Holloway, Univ. London., 1997. R Tech. Rep. NC2-TR-1997-013.
- [10] T. Watkin. Optimal learning with a neural network. *Europhys. Lett.*, vol. 21, pages 871–877, 1993.